# IEC 62325-503

Edition 1.0 2018-07

# INTERNATIONAL STANDARD

# NORME INTERNATIONALE

colour inside

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

## Framework for energy market communications – Part 503: Market data exchanges guidelines for the IEC 62325-351 profile

## Cadre pour les communications pour le marché de l'énergie – Partie 503: Lignes directrices concernant les échanges de données du marché pour le profil défini dans l'IEC 62325-351

IEC 62325-503:2018-07(en-fr)

**THIS PUBLICATION IS COPYRIGHT PROTECTED**
**Copyright © 2018 IEC, Geneva, Switzerland**

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

**About the IEC**
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

**About IEC publications**
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

**IEC Catalogue - webstore.iec.ch/catalogue**
The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

**IEC publications search - webstore.iec.ch/advsearchform**
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

**IEC Just Published - webstore.iec.ch/justpublished**
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

**Electropedia - www.electropedia.org**
The world's leading online dictionary of electronic and electrical terms containing 21 000 terms and definitions in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

**IEC Glossary - std.iec.ch/glossary**
67 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

**IEC Customer Service Centre - webstore.iec.ch/csc**
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

**A propos de l'IEC**
La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

**A propos des publications IEC**
Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

**Catalogue IEC - webstore.iec.ch/catalogue**
Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

**Recherche de publications IEC - webstore.iec.ch/advsearchform**
La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,…). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

**IEC Just Published - webstore.iec.ch/justpublished**
Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

**Electropedia - www.electropedia.org**
Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient 21 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 16 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

**Glossaire IEC - std.iec.ch/glossary**
67 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

**Service Clients - webstore.iec.ch/csc**
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: sales@iec.ch.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# IEC 62325-503

Edition 1.0 2018-07

# INTERNATIONAL STANDARD

# NORME INTERNATIONALE

colour inside

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

## Framework for energy market communications – Part 503: Market data exchanges guidelines for the IEC 62325-351 profile

## Cadre pour les communications pour le marché de l'énergie – Partie 503: Lignes directrices concernant les échanges de données du marché pour le profil défini dans l'IEC 62325-351

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 33.200

ISBN 978-2-8322-5916-0

**Warning! Make sure that you obtained this publication from an authorized distributor.**
**Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

® Registered trademark of the International Electrotechnical Commission
Marque déposée de la Commission Electrotechnique Internationale

CONTENTS



IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

# FRAMEWORK FOR ENERGY MARKET COMMUNICATIONS –

# Part 503: Market data exchanges guidelines for the IEC 62325-351 profile

## FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

International Standard IEC 62325-503 has been prepared by IEC technical committee 57: Power systems management and associated information exchange.

This edition cancels and replaces IEC TS 62325-503 published in 2014.

This edition includes the following significant technical changes with respect to the previous edition:

a) Use of ISO/IEC 19464:2014, Advanced Message Queuing Protocol (AMQP) v1.0 specification;
b) Splitting of the node described in the IEC TS 62325-503:2014 into a broker that implements the messaging function and a directory;
c) Increase of operability and resilience of the communication system with the ability for an endpoint to send and receive messages through several brokers;
d) Benefits of standardisation, performance and scalability of the AMQP protocol for transferring messages.

The text of this standard is based on the following documents:

| CDV | Report on voting |
|---|---|
| 57/1936/CDV | 57/1983/RVC |

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

In this document, the following print types are used:

*Help the visibility of information in table and diagram: in italic type*

A list of all parts in the IEC 62325 series, published under the general title *Framework for energy market communications*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

**IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours, which are considered useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.**

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# INTRODUCTION

This document is part of the IEC 62325 series for deregulated energy market communications.

The principal objective of the IEC 62325 series is to produce documents which facilitate the integration of market application software developed independently by different vendors into a market management system, between market management systems and market participant systems. This is accomplished by defining message exchanges to enable these applications or systems access to public data and exchange information independent of how such information is represented internally.

The common information model (CIM) specifies the basis for the semantics for the message exchange. The European style market profile specifications that support the European style design electricity markets are defined in IEC 62325-351. These electricity markets are based on the European regulations, and on the concepts of third party access and zonal markets. The IEC 62325-451-n International documents specify the content of the messages exchanged.

The purpose of this document is to provide the guidelines to exchange the above-mentioned messages. A European market participant (trader, distribution utilities, etc.) could benefit from a single, common, harmonised, secure platform for message exchange with the European Transmission System Operators (TSOs); thus reducing the cost of building different IT platforms to interface with all the parties involved.

This document represents an important step in facilitating parties entering into electricity markets other than their national ones; they could use the same or similar information exchange system to participate in more than one market all over Europe.

This document was originally based upon the work of the European Network of Transmission System Operators (ENTSO-E) Working Group EDI.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# FRAMEWORK FOR ENERGY MARKET COMMUNICATIONS –

# Part 503: Market data exchanges guidelines for the IEC 62325-351 profile

## 1 Scope

This part of IEC 62325 is for European electricity markets.

This document specifies a standard for a communication platform which every Transmission System Operator (TSO) in Europe can use to exchange reliably and securely documents for the energy market. Consequently a European market participant (TSO, regional supervision centre, distribution utility, power exchange, etc.) could benefit from a single, common, harmonised and secure platform for message exchange with other participants; thus, reducing the cost of building different information technology (IT) platforms to interface with all the parties involved.

“MADES” (MArket Data Exchange Standard) is the acronym to designate this standard.

MADES is a specification for a decentralised common communication platform based on international IT standards:

- From an application program perspective, MADES specifies the software interfaces to exchange electronic documents with peer applications. Such interfaces mainly provide means to send and receive documents using a so-called “MADES communication system” (or "MADES system" or simply "system"). The sender can request about the status of the delivery of a document and the recipient issues a message back, the acknowledgement, when receiving the document. This makes a MADES system usable for exchanging documents in business processes requiring a reliable delivery.
- MADES also specifies services hidden to the applications such as recipient localisation, recipient connection status, message routing and security. Services include directory, authentication, signing, encryption, message tracking, message logging and message temporary storage.

The purpose of MADES is to create a secured message exchange standard based on standard communication protocols and utilising IT best practices for exchanging data over any TCP/IP communication network, in order to facilitate business-to-business (B2B) information exchanges as described in IEC 62325-351 and the IEC 62325-451 series.

A MADES system acts as a post-office organisation: the transported object is a “message” in which the document of the sender is securely packaged in an envelope containing metadata, which is necessary information for transportation, tracking and delivery.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

## 2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC TS 61970-2, *Energy management system application program interface (EMS-API) – Part 2: Glossary*

ISO/IEC 19464:2014, *Information technology – Advanced Message Queuing Protocol (AMQP) v1.0 specification, https://www.amqp.org/* (developed by the OASIS open standards consortium)

ISO/IEC 9594-8:2017, *Information technology – Open systems interconnection – The Directory – Part 8: Public-key and attribute certificate frameworks*

## 3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC TS 61970-2 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

NOTE For general glossary definitions, see IEC 60050, *International Electrotechnical Vocabulary*.

**3.1**
**advanced message queuing protocol**
**AMQP**
open Internet protocol for business messaging, as described in IEC 19464:2014

**3.2**
**advanced message queuing protocol secured with transport layer security**
**AMQPS**
combining of the IEC 19464 business messaging protocol with transport layer security (TLS)

**3.3**
**market data exchange standard**
**MADES**
specification described in this document for the European market style market profile

**3.4**
**representational state transfer**
**REST**
method of providing interoperability between computer systems by requesting to access and manipulate textual representations of resources using predefined set of stateless operations

**3.5**
**simple authentication and security layer**
**SASL**
framework for authentication and data security in internet protocols

**3.6**
**simple object access protocol**
**SOAP**
protocol specification for exchanging structured information in the implementation of webservices

**3.7**
**transmission system operator**
**TSO**
entity involved in electric power transmission or in transmission of natural gas

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**3.8**
**transport layer security**
**TLS**
cryptographic protocol to provide privacy and data integrity between two communicating computer applications

## 4 High level concepts

### 4.1 What is the purpose of MADES?

NOTE Clause 4 describes the business context for the specification usage, introduces the main concepts of the specification and does not contain requirements.

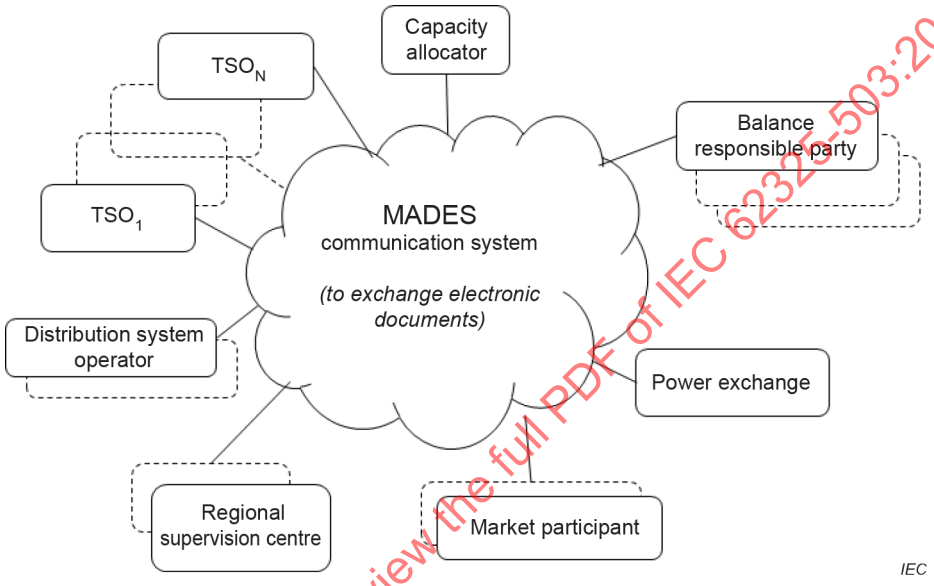



**Figure 1 – MADES overall view**

MADES' first intention is to provide transmission system operators (TSOs) with a standardised communication system for securely exchanging electronic documents between themselves and with others parties involved in the European electricity market as shown in Figure 1.

These documents are mainly those used in the energy market and described in IEC 62325-351 and the IEC 62325-451 series. Such parties include TSOs, regional supervision centres, power exchanges, distribution system operators, balance responsible parties, transmission capacity allocators, market operators, capacity traders, producers, etc.

MADES enables each party to host an access point to the communication system: applications of his information system can then use the access point to send and receive securely documents with other parties.

Technically, MADES is a data exchange standard comprised of standard communication protocols and utilising information technology standards and best practices to create a mechanism for exchanging data over any TCP/IP communication network and infrastructure, in order to facilitate business-to-business information exchanges.

New market rules induce new business processes and activities, and generally require new information exchanges between parties. Experience shows that, for the data exchanges to meet the business goals, the chosen technical solution results from an agreement of all the involved parties gathering various constraints including implementation time scale, vendors' offers, already existing communication infrastructure and links, integration capabilities of existing information systems, confidentiality of exchanged information, legal risks, performance, etc.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

Whereas business processes require that information be exchanged between multiple systems or multiple parties, solutions developed bilaterally can become complex, each interface taking time before coming in operation, taking money and resources for development and maintenance. A noticeable consequence is that some parties involved in business processes that require exchanges with many others parties have to install and operate different communication tools and solutions to interface with multiple information systems. The process towards a more integrated European electricity market creates many such situations with new coordinated processes between actors with regulated roles. Market participants acting in several countries are also looking for a harmonised interface.

MADES can support any business process whatever the transmitted document format (e.g. XML, binary) and the sequence for the exchanges.

MADES is independent of the physical underlying communication Infrastructure, which can be any internet protocol (IP) network, such as Internet, a physical private infrastructure, or a multi access-point virtual private network (VPN).

MADES relies on and only on non-proprietary IT standards for communication protocols, data integrity, peer authentication (signature), confidentiality (encryption), accessible parties' directory (e.g. HTTP, AMQP, TLS, X.509), as shown in Figure 2.

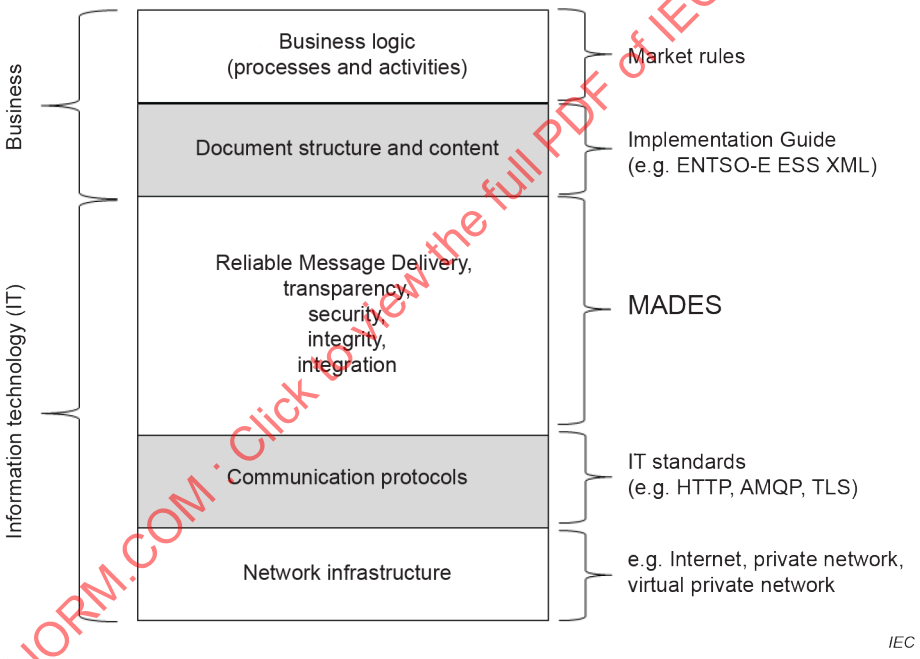



**Figure 2 – MADES scope in a layered architecture**

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

## 4.2 Overview

The MADES standard specifies a communication system that delivers messages between authorised parties. Such a system has the following key features:

1) Reliable message delivery – A (sender) party connected can send a message to another (recipient) party connected or that can connect to the system. The sender is notified when a message is not acknowledged by the recipient in due time, whatever reason.
2) Transparency – The system is not a black box: it tracks every message down to gather trustworthy information about the delivery.
3) Security – Only the recipient of the message can read the content (confidentiality) and the recipient unambiguously authenticates the sender (signature).
4) Integrity – The system guarantees that the content of a message is unaltered during the delivery process.

5) Integration – A wide variety of technologies can integrate with the services exposed for sending and receiving messages. This eases development and integration in any information system.

The first four key features (reliable message delivery, transparency, security and integrity) are capabilities of the system, while the last one (integration) is a design characteristic of the MADES components.

### 4.3 Transparent and reliable message delivery

The main feature of a MADES system is the message delivery function, as shown Figure 3.

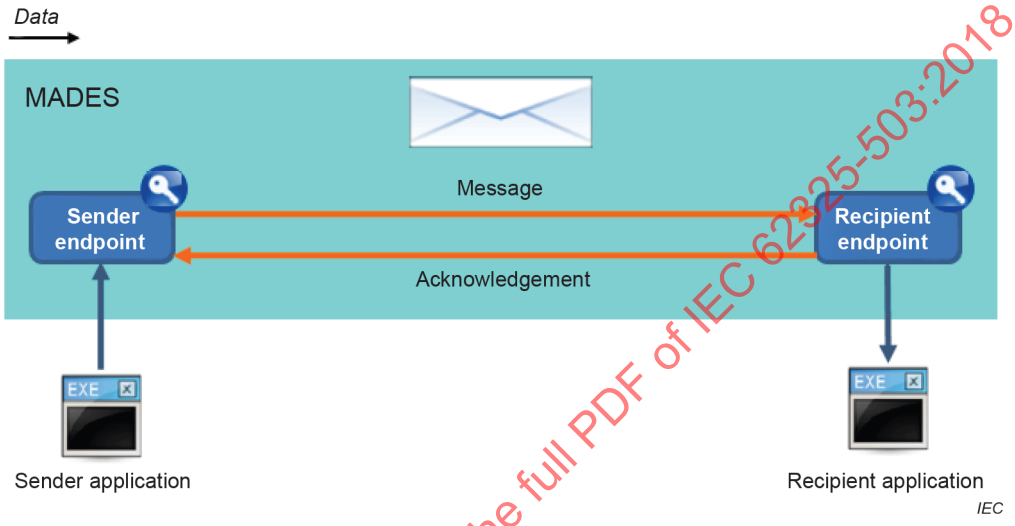



**Figure 3 – MADES message delivery**

A sender wants or needs to transfer a document to a recipient using the system. Both the sender and the recipient are applications that use an application-programming interface (API) to connect to their respective access points, named *endpoints*.

The peer applications view the system only through the defined interface. The document being transported from one to the other can be any text or binary data. Actually, the system transports a message that comprises the document with additional metadata, which includes information to route and transport the message securely such as a unique ID, the identities of the peer endpoints and cryptographic information related to the message signing and to the document encryption.

The system tracks any message that a sender endpoint accepted. The endpoint can reject a request from a business application for sending a document, e.g. if the recipient is unknown.

The sender application can check at any time the status of a sent message among ACCEPTED, DELIVERED, RECEIVED or FAILED (see 5.6.1).

The sender endpoint knows that the system has delivered a message to the recipient endpoint, when it receives back an acknowledgement issued by the recipient endpoint. The sender endpoint sets the status of the message to FAILED when the message expires, i.e. if it does not receive the acknowledgement in due time, whatever reason.

The system transports both:

- Standard message – a message composed by a sender endpoint to send an electronic document upon request from a sender application.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- Acknowledgement – a message composed by a recipient endpoint for tracking the endpoint-to-endpoint delivery[1] of a standard message. The applications do not know about acknowledgements, which remain internal to the system.

NOTE Further in the document, the single work "message" refers to "standard message", and "internal message" generically refers to either (standard) message or acknowledgement.

## 4.4 Components of a MADES system

### 4.4.1 Endpoint, broker and component-directory

A MADES system is a set of components out of three types: endpoint, broker and component-directory. Two components interact in a client / server mode as shown in Figure 4. This is the full view that the current clause explains in details.

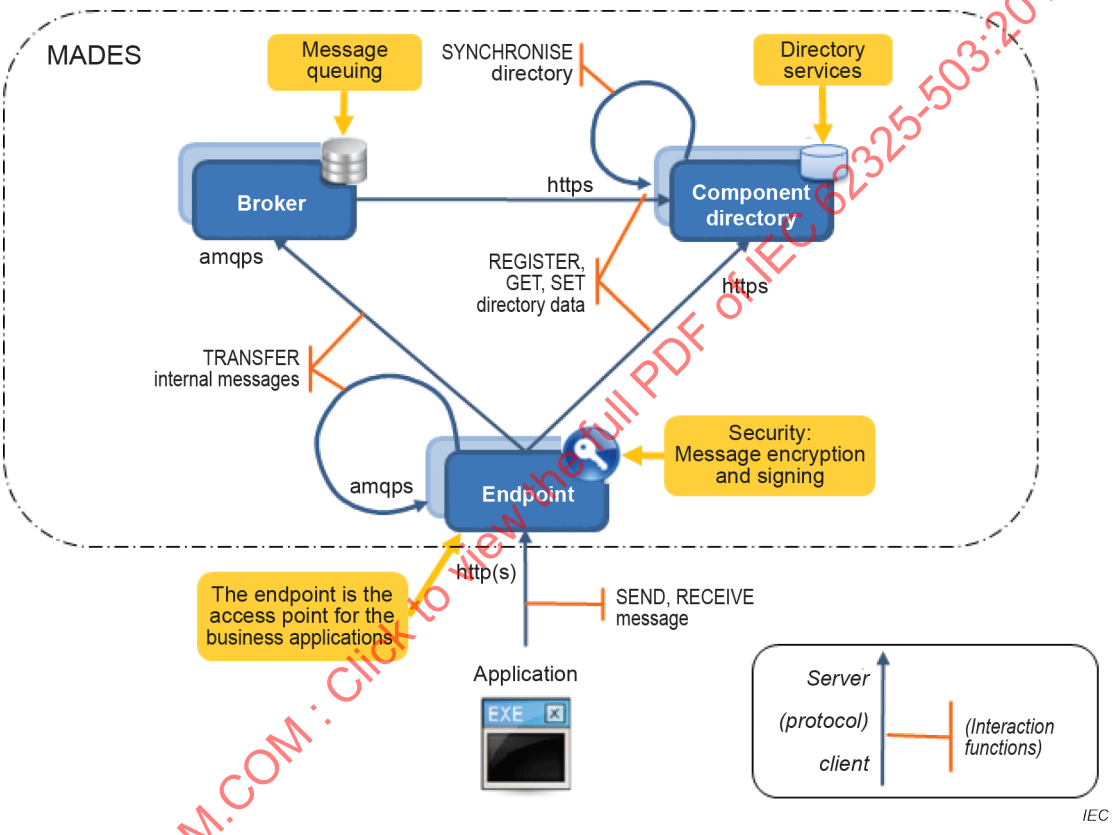



**Figure 4 – MADES components, interactions and protocols**

A party using the system hosts an endpoint. An application (*client*) uses the interface exposed by the endpoint (*server*) to send and receive messages. The interface uses the HTTP(S) protocol.

NOTE The client component opens a connection. The server component accepts the connection.

This document refers to a sender (respectively recipient) endpoint when focussing on the operations that the endpoint performs for sending (resp. receiving) a message. The endpoints perform the end-to-end security; namely, the sender endpoint signs and encrypts the message, whereas the recipient endpoint decrypts the message and verifies the sender signature.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

[1] This is a technical acknowledgement, not an acknowledgement as defined in IEC 62325-451-1.

For transferring a message, the sender endpoint (*client*) uploads it to a broker (*server*), and the recipient endpoint *(client)* downloads it from that broker: the broker temporarily stores the message in a queue; this is further referred as an *indirect* transfer. The sender endpoint (*client*) can also upload the message directly to the recipient endpoint (*server*): this is further referred as a *direct* transfer. Whatever route, the protocol to transfer the message is AMQPS, which stands for Advanced Message Queuing Protocol Secured with transport layer security (TLS).

Transferring an acknowledgement works the same way, but the peer endpoints play reverse roles, the recipient being the sender and vice versa. A broker only works with internal messages and does not distinguish between messages and acknowledgements.

The endpoints and the brokers (*all clients*) get configuration data of the system, e.g. identifications of the existing components, certificates, authorised delivery routes, from a component-directory (*a server*), which implements an address book. When the system comprises several component-directories, they synchronise each other. The directory services use the HTTPS protocol.

### 4.4.2 Delivery routes and acknowledgements

Figure 5 shows the possible routes to transfer a message:

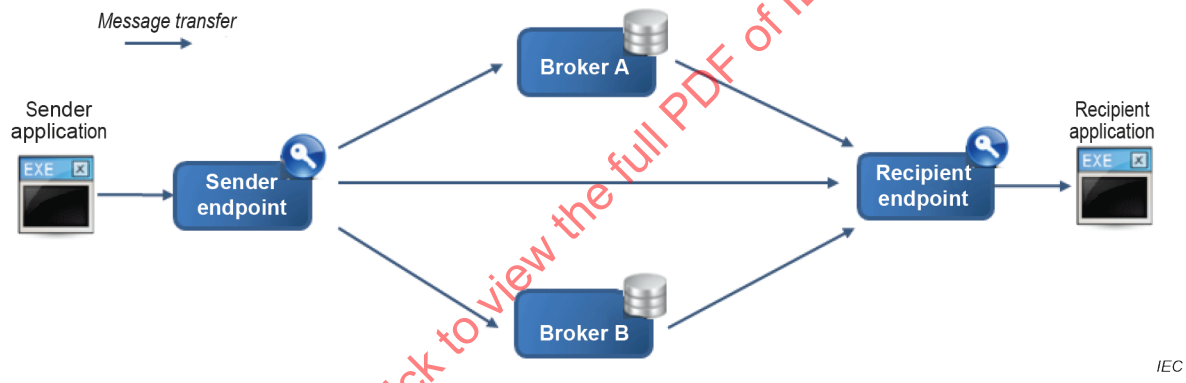



**Figure 5 – Possible routes for delivering a message**

A sender endpoint can transfer a message to a recipient endpoint either using an intermediate broker or directly. Enabling or not enabling direct transfer is a security issue (see 4.5.2).

Figure 5 shows that the system can include several brokers and that the messages between two given endpoints can use different routes. The route depends on the type of the document to transport, named after the *message-type*, which generally binds the message to some business process (see 5.2). The sender endpoint selects the route according to the message-path (see 5.3) associated with the message-type, as configured by the *recipient* endpoint, and published and accessible in the components-directories.

Figure 6 shows the communication protocols used along the message route.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

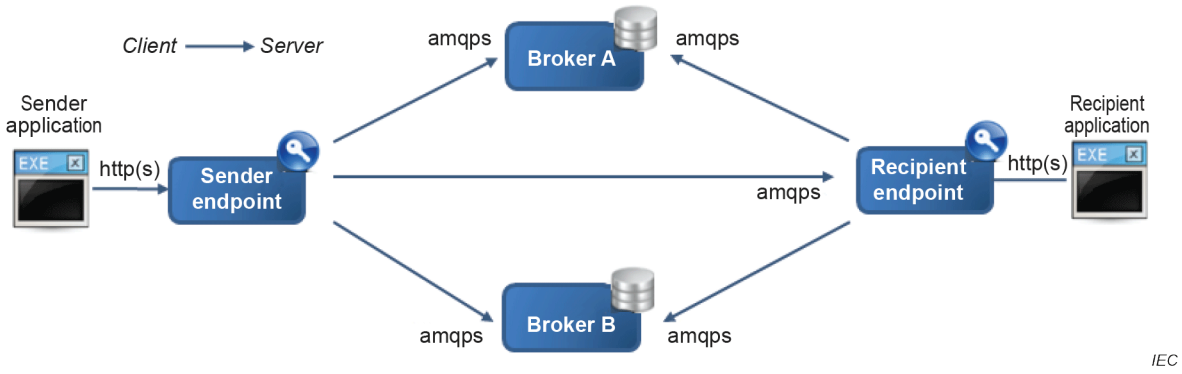



**Figure 6 – Communication protocols for delivering a message**

An acknowledgement is an internal message issued by the recipient endpoint back to the sender endpoint to notify that it received a message. The route of the acknowledgement is the *exact reverse route* of the corresponding message.

NOTE:

- The sender application sets the message-type but does not select the route. Modifying a message-path changes the route but does not affect the applications.
- Load balancing the message traffic through multiple brokers is not possible. Between two given endpoints, all messages with the same message-type use the same route. Some brokers can support high available clustered setups.
- There is at most one broker on the route from a sender endpoint to a recipient endpoint.

### 4.4.3 Sharing configuration data of the system

Implementing message routing and security transparently to applications enforces that the endpoints and the brokers access to configuration data that describes the system. This includes the following information for each component:

- unique ID,
- security certificates for authentication, signing and encryption,
- telecommunication network location to access the services that the component exposes,
- rules for message delivery: for an endpoint, the rules say what are the routes to send messages to the endpoint; for a broker the rules say which messages the broker accepts, i.e. which message-types and from which sender endpoints,
- operational contact information.

Each endpoint and each broker registers with a single component-directory, named after its *home component-directory*. A major role of the administrator of a component-directory is to accept new parties, establish trust with them, and to register their components. The role includes the reverse operations: unregister components, notably the ones potentially compromised. A component-directory and the set of its registered components forms a *subsystem*.

Figure 7 shows data flows between a component-directory "A" and its registered components:

(A) When accepting a registration request, the component-directory creates a new entry for the component in the directory. It also issues and stores the component certificates.

(B) The administrator of a registered component enters all of configuration data about the component, which is pushed for update in the home component-directory.

(C) A registered component locally caches the whole directory content and cyclically request for synchronisation. On a change, it receives a full copy of the directory, not increments.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

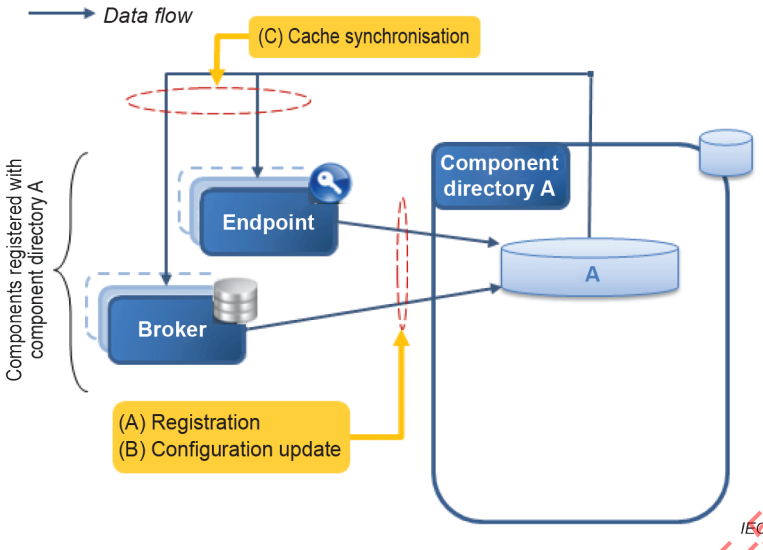



**Figure 7 – Data flows between a component-directory and its registered components**

The specification of the component-directory pursues two goals:

1) Minimise the administration tasks: data entry is limited. In addition, whereas the first registration should probably always require a manual acceptance, the certificates renewal could [2] be automatic (see 7.6) in case the request comes from a validly registered component.
2) Design a non-critical component, i.e. which does not block the message transfer when it is down. Indeed, each endpoint and each broker locally caches and refreshes directory data, which remains valid during a configurable period while the component-directory is down.

Figure 8 shows the same component-directory "A", now included in a wider system with two other subsystems, i.e. component-directories: "B" and "C".

- every component-directory couple synchronises bi-directionally, i.e. each directory requests the other for its subsystem directory content;
- an "A"-registered endpoint (respectively broker) now receives the entire system directory content ("A" + "B" + "C") when synchronising with its home component-directory;
- a "B"-registered or "C"-registered component synchronises with the "A" component-directory when its home component-directory is down. Conversely an "A"-registered component can synchronise with another component-directory (such a flow is not presented in Figure 8).

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

2 This is a system governance decision.

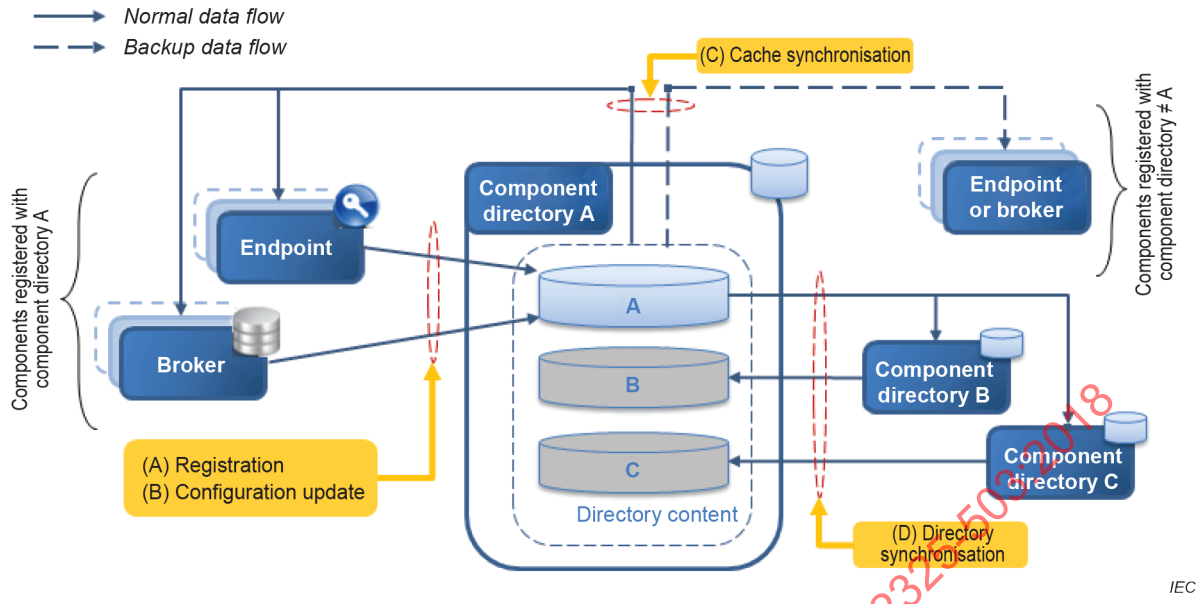



**Figure 8 – Data flows with several component-directories**

NOTE Locally caching the whole directory content is the key point for a component-directory not being critical in the system. The underlying design assumption is that, even when some hundred components compose the system, configuration data remains small and does not change frequently.

All services exposed by a component-directory are accessible using the https protocol as shown in Figure 9.

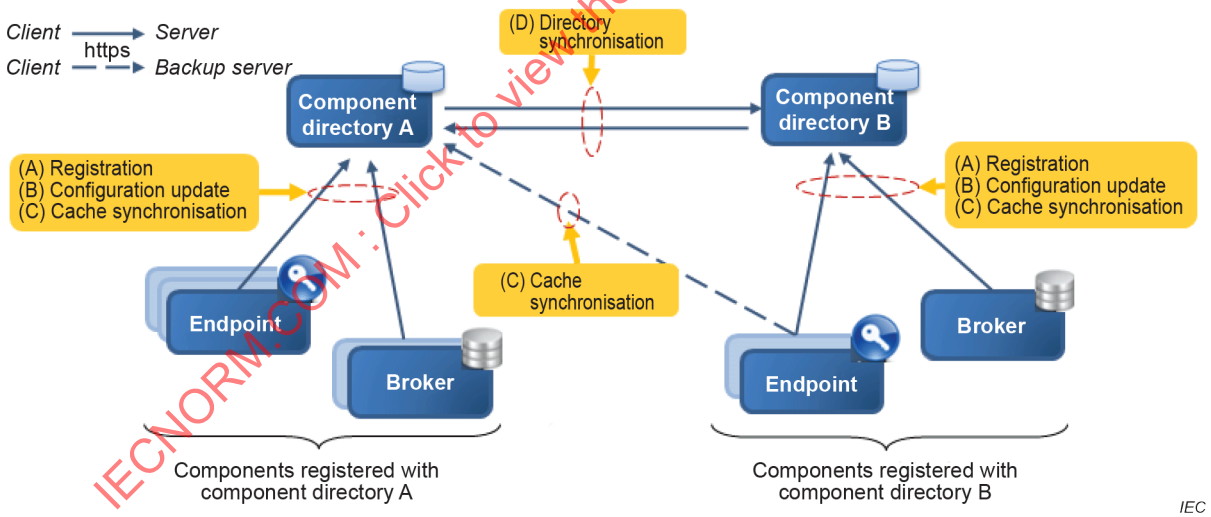



**Figure 9 – Component-directory services and protocols**

### 4.4.4 Interfaces exposed by the components

Figure 10 shows the interfaces that the specification defines in Clause 5, and that a compliant component implements, i.e. uses and exposes.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

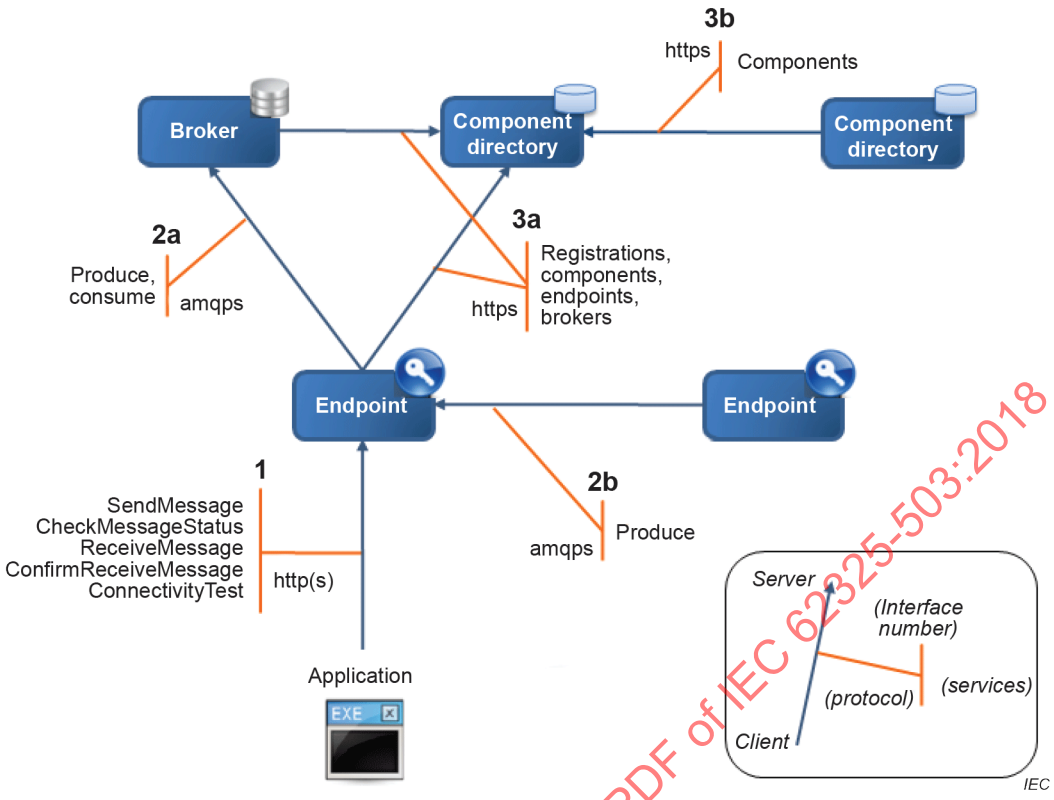



**Figure 10 – MADES Interfaces, services and protocols**

(1) Endpoint interface for applications using HTTP(S) SOAP web services – see 11.1.

- An application sends a message (*SendMessage*) through the system or requests for the delivery status of a previously sent message (*CheckMessageStatus*).
- An application requests for downloading a received message (*ReceiveMessage*) and then confirms back to the system that it has safely stored or processed the message (*ConfirmReceiveMessage*).

NOTE Safe storage is defined in 5.9.

- An application requests the system to send a test message to a peer party (*ConnectivityTest*) in order to check that the communication with that party is working correctly for a given message-type.

(2a) Broker interface for endpoints using AMQP – see 6.3.

- AMQP is a protocol intended for the interoperability between various messaging middleware: it does not describe an interface but the byte-flow structure (*frames*) to exchange between two TCP peers (*on the wire*).
- A MADES broker is as a server that manages queues of (internal) messages until the intended recipient gets them.
- When successfully authenticated to the broker, an endpoint acts as a message producer to send a message (*Produce*) and as a message consumer to receive its intended messages (*Consume*).

(2b) Endpoint interface for others endpoints using AMQP – see 6.3.

- The interface for direct transfer between endpoints is the same as (2a) without the *Consume* service. Indeed, an endpoint does not contain queues in which other endpoints can consume messages.

(3a) Component-directory interface used by endpoints and brokers – see 7.5.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- The endpoints and the brokers use the interface to register with their home directory-component into which they push configuration data. They also use the interface for keeping up-to-date a local copy of the system directory.
- The interface uses the REST principles. Requests and responses use XML format and are specified using XML schemas.
- The resources accessible to an authenticated component are:
  - *registrations*: to request for entering the system, to get and renew the certificates.
  - *components*: to get directory information about all components registered in the system in order to maintain a local copy of the directory. The answer returns either the whole directory content or nothing when the local directory is already up-to-date.
  - *endpoints* (resp. *brokers*): to get configuration data on an endpoint (resp. a broker) or to modify such information. Only a component can modify its own configuration data.

(3b) Component-directory interface used by other component-directories – see 7.5.

- A component-directory uses the interface to synchronise with another component-directory of the system.
- For synchronising, an "A" component-directory requests the *components* resources of a "B" component-directory exactly as endpoints and brokers do when using interface (3b). However, it only asks for the peer directory content restricted to the "B" subsystem components.

### 4.4.5 Architecture examples of MADES systems

A MADES system can range from a minimal architecture composed of two endpoints and one component-directory either without (Figure 11) or with a broker (Figure 12), up to a network of endpoints with several brokers and component-directories (Figure 13, Figure 14).

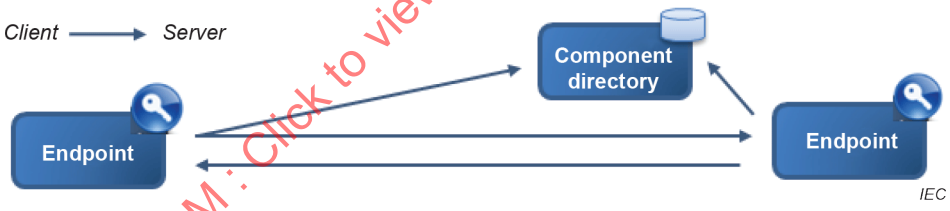



**Figure 11 – Minimal MADES system (without broker)**

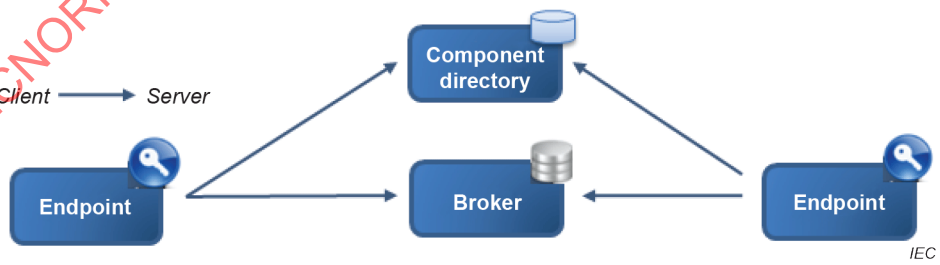



**Figure 12 – Minimal MADES system (with broker)**

Any of the two communicating parties using the architecture of Figure 12 can host either the broker or the component-directory or both.

A typical architecture is when one of the stakeholders of a business process, say "A", plays a central role (Figure 13). Therefore, he can host a broker and a component-directory. Acting as a hub in the business process logic, he also owns and hosts an endpoint for exchanging with other stakeholders.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

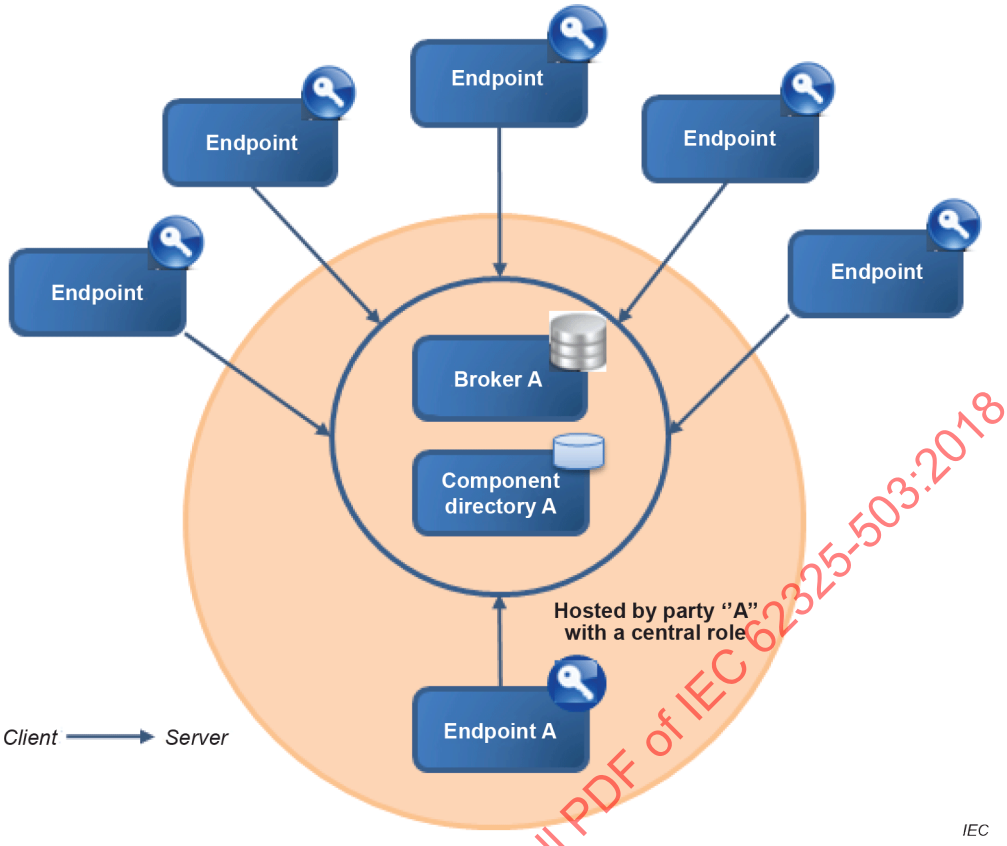



**Figure 13 – MADES system with a party in a central role**

Another business process could involve the same parties or others, or a subgroup, in which a party, say "B", now plays the central role. The parties can share the existing broker "A". But they also can use another and dedicated broker (broker "B") for reasons such as hosting localisation, business criticality, responsibility perimeter, contracted service-level agreement (SLA), project planning, etc., leading to a multi broker architecture as shown in Figure 14.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

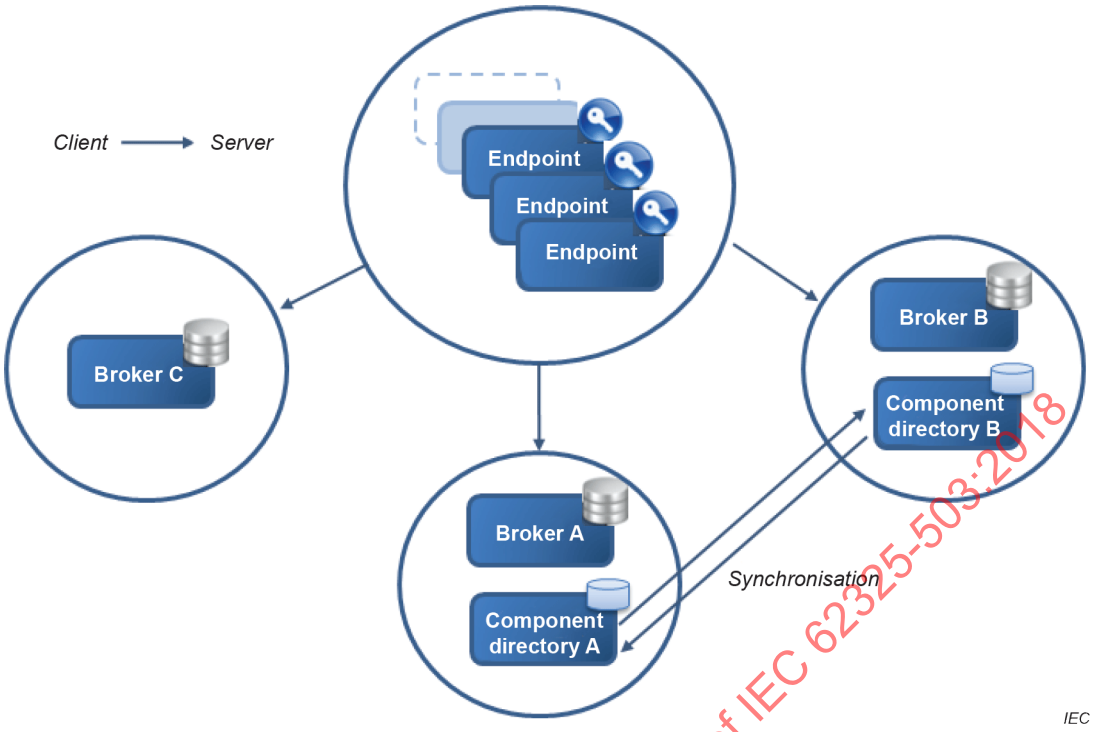



**Figure 14 – MADES system with several brokers**

Figure 14 shows two component-directories respectively hosted by parties who own broker A and broker B. There are two component-directories but possibly more. The rules are the following:

- Each component-directory synchronises with any other.
- The system has two component-directories. This is recommended for business continuity although the component-directory is not a critical component and could be down for some time without affecting the message delivery function.

In the Figure 14 example, both parties A and B host a component-directory and are registry offices each for a subsystem, i.e. a subset of endpoints and brokers of the system. However, the party C could also host a component-directory.

Figure 15 presents the same architecture viewed by a party involved in each business process. The party owns a single endpoint registered with a unique component-directory. Here the party accepts direct transfer and thus incoming connections to the endpoint. The architecture can encompass some direct bilateral exchanges between parties with no impact on the brokers.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

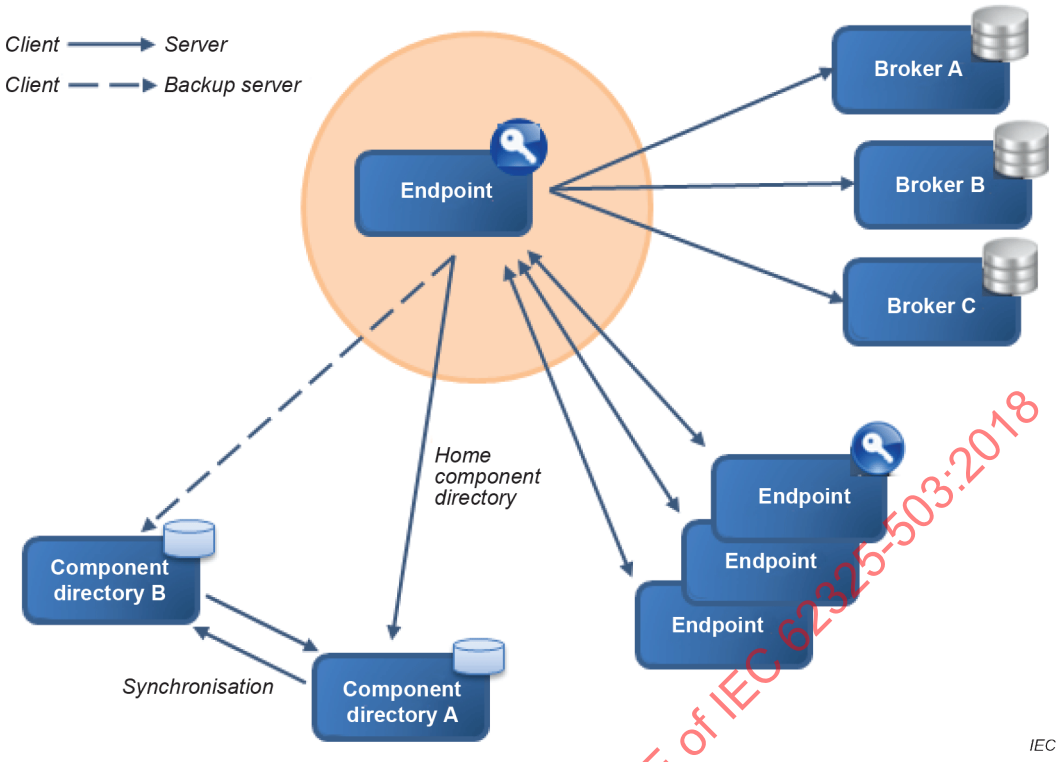



**Figure 15 – Using a single endpoint for several business processes**

The party could also choose to host multiple endpoints, each dedicated to a subset of business process or messages for reasons such as hosting localisation, business criticality, internal organisation and responsibility perimeter, contracted service-level agreement (SLA), project planning, etc. Each endpoint will then have to register with a component-directory, and the endpoints will appear in the system with different IDs as if owned by distinct parties. Such endpoints cannot behave in a master / slave mode, nor can they load balance the incoming message traffic towards the party[3].

## 4.5 Security and message integrity

### 4.5.1 Security goals and security solution

The goals of MADES security are:

- The security solution is transparent to applications – no specific implementation is required for an application to communicate securely.
- All communication channels are encrypted.
- Only the intended recipient can read a message.
- The recipient of a message can unambiguously identify the sender.
- Non-repudiation – A message together and its acknowledgement form an unambiguous proof that the sender sent the message and that the recipient received it.

The specified security solution complies with the X.509 public key infrastructure (PKI) using certificates, and it covers two security levels: a transport-layer security and a message-level security.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

3 High availability of an endpoint is an issue of the endpoint design.

- On the transport-layer, any two components communicate using an encrypted channel. In addition, both components always first unambiguously authenticate each other and check they are valid registered components of the system before exchanging data.
- On the message-level, every message is signed then encrypted.

Each component has an authentication certificate used for the transport-layer security. Each endpoint has two additional certificates used for the message-level security, one for signing the sent messages, the other for decrypting the received messages.

### 4.5.2 Transport-layer security

Security of data exchanged between two components first relies on the transport protocol layer. When communicating, two components use the "transport layer security" (TLS) protocol for encrypting the communication channel. HTTPS (resp. AMQPS) is the TLS secured version of HTTP (resp. AMQP).

Whatever the communicating peer components are, they mutually authenticate using X.509 certificates as shown in Figure 16.

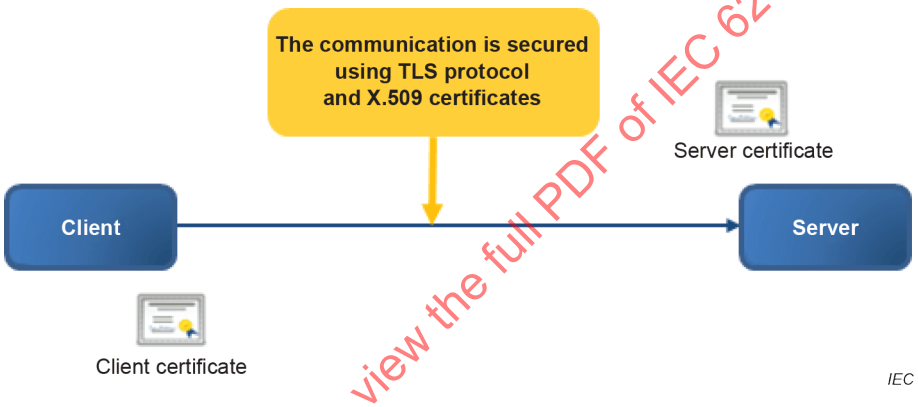



**Figure 16 – MADES transport security**

The client always initiates the communication, i.e. the IP connection. Figure 10 shows all possible peer components.

This provides a highly secured architecture when the endpoint only acts as a client, and an additional firewall can increase the protection by blocking all incoming connections as shown in Figure 17.

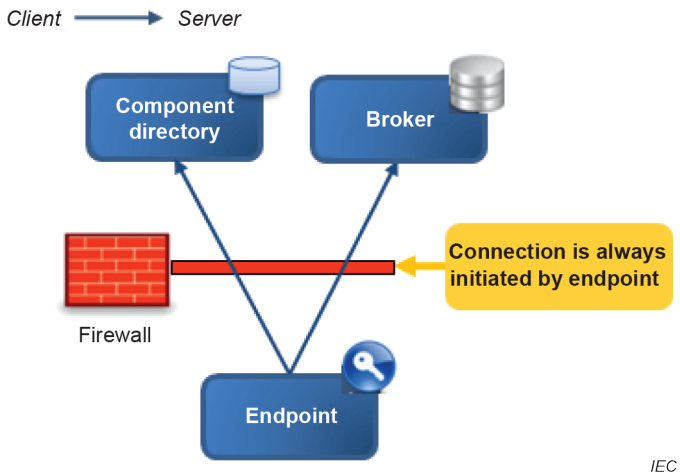



**Figure 17 – Security: protected endpoint**

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

When parties agree for transferring directly messages between endpoints, the firewall has to allow for incoming connections, possibly restricted to the expected peer endpoints as shown in Figure 18.

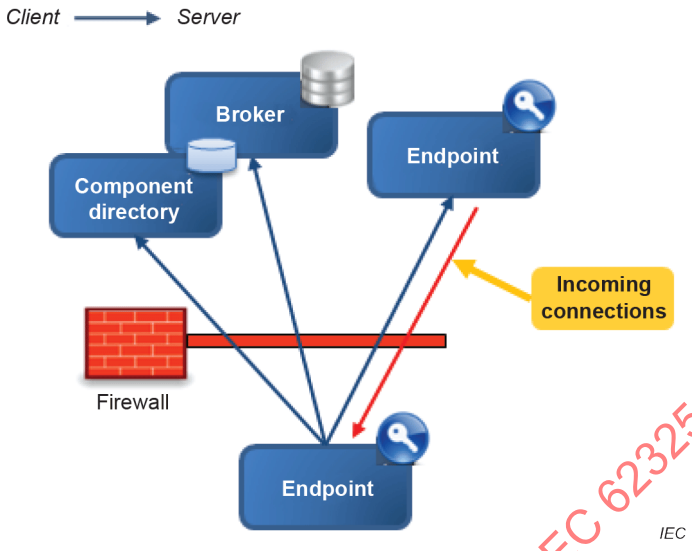



**Figure 18 – Security: exposed endpoint**

NOTE Roles are symmetric and both peer endpoints shall allow for incoming connections even when the messages flow only one way, for the acknowledgements then flow the reverse way.

### 4.5.3 Message-level security: signing and encryption

The use of digital signatures enables the unambiguous identification of the sender of each message and acknowledgement received, as shown in Figure 19.

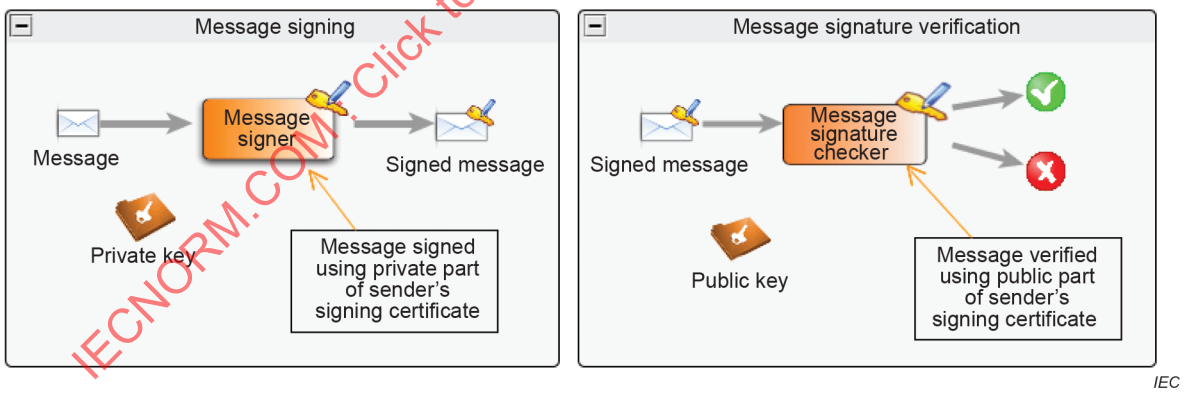



**Figure 19 – Message signing and signature verification**

When sending a message or an acknowledgement, the endpoint signs it using the private key of its own signing certificate. The endpoint that receives the message authenticates the issuer endpoint by verifying the signature, i.e. using the public key of the signing certificate of the issuer endpoint.

The sender endpoint of a message encrypts the message-content using the public key of the encryption certificate of the recipient endpoint (see 6.5.3 for algorithm). Therefore, only the recipient endpoint can read the message-content (Figure 20).

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

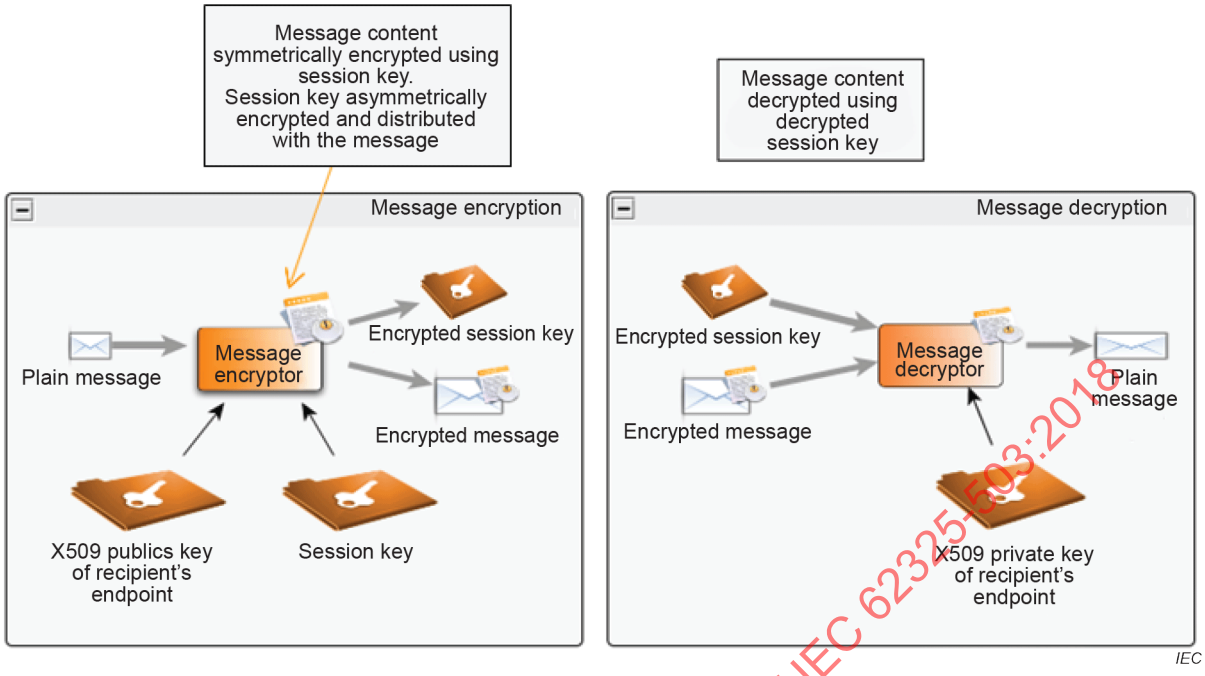



**Figure 20 – Message encryption and decryption**

The sender endpoint encrypts the application document, i.e. the content of the message, using a session key randomly generated. The session key is then encrypted with the public key of the encryption certificate of the recipient endpoint. The encrypted key is included in the message and transported to the recipient endpoint.

The recipient endpoint decrypts the key in the message with the private key of its encryption certificate, and then uses the key to decrypt the message-content.

### 4.5.4 Non-repudiation

Figure 21 illustrates how a message and its corresponding acknowledgement are signed to give each peer the ability to prove that the other peer sent either the message or the acknowledgement.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

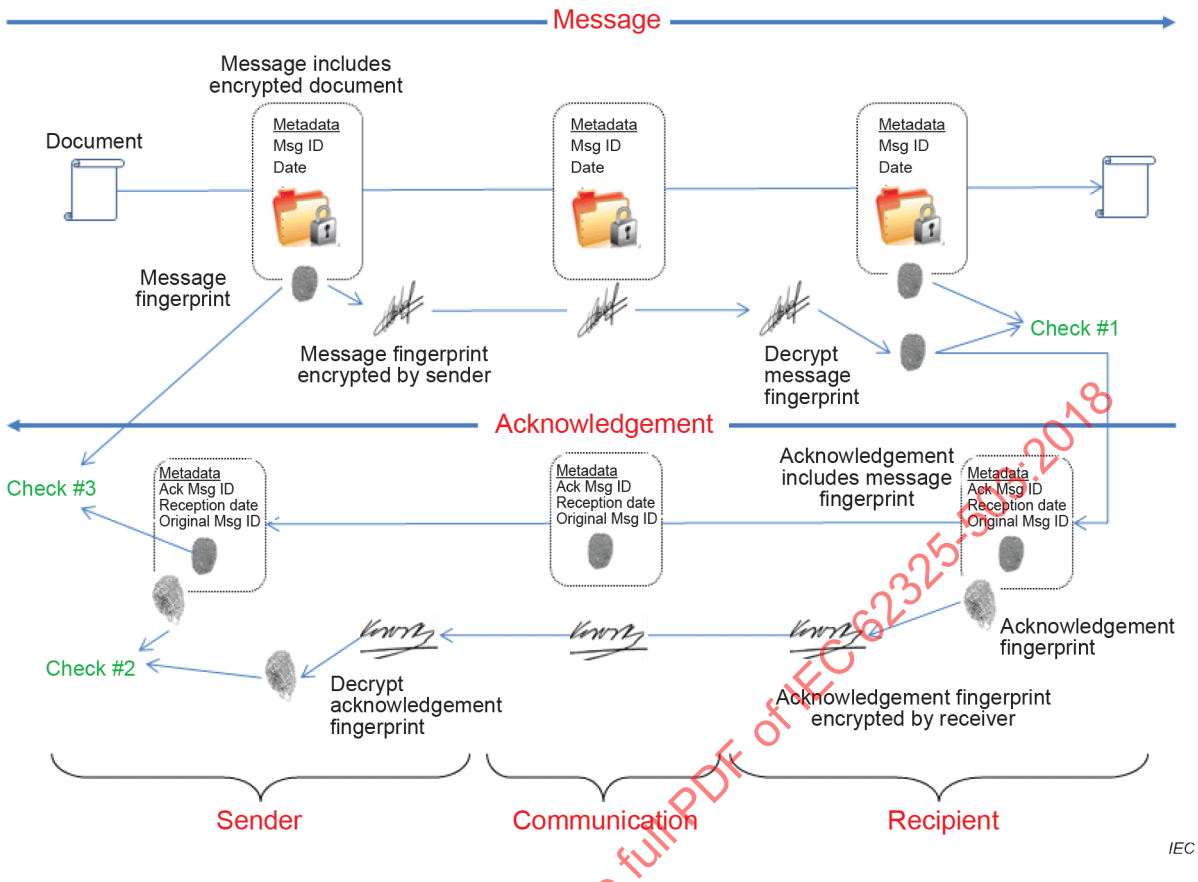



**Figure 21 – Non-repudiation**

- Message delivery:

  A message embeds the sender document. The message fingerprint uniquely identifies the document together with its metadata, such as the unique ID (*Msg ID*) of the message and sending date and time (*Date*).

  The fingerprint and the sender document are both encrypted and delivered to the recipient. The fingerprint is encrypted so that anyone can uniquely identify the sender (*signature*). The document is encrypted so that only the recipient can read it (*encryption*).

  The recipient decrypts the fingerprint and the document. He verifies (*check #1*) that the fingerprint, which he can compute from the received message, matches the transported signed fingerprint (*signature verification*). The set composed of the decrypted message, the message signature (i.e. the encrypted fingerprint) and the signing certificate of the sender proves that the sender issued the message.

- Acknowledgement:

  The recipient sends back to the sender the acknowledgement, using a similar process. The acknowledgement does not embed a document but the unique ID of the original message (*Original Msg ID*) and the fingerprint of the original message. The acknowledgement is signed and not encrypted.

  NOTE The acknowledgement referred to here is the DELIVERY_ACKNOWLEDGEMENT (see 5.6.2).

  When he receives the acknowledgement, the sender (*check #2*) identifies who sent the acknowledgement (*signature verification*). He also verifies that the acknowledged document fingerprint equals the original message fingerprint (*check #3*).

  The set composed of the original message, the signed acknowledgement and the signing certificate of the recipient proves that the recipient received the document.

NOTE For a freestanding proof set without the encryption certificate of the recipient, signing has to occur before encryption.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# 5 Delivering the messages

## 5.1 Unique identification of components and messages

Each component, either endpoint, broker or component-directory, shall have a unique identification (ID) in the system. The identification scheme is a system governance issue.

- The “component ID” (also named “component code”) is used to identify the component when exchanging with other components.
- An application uses an endpoint ID to identify the recipient when sending a document. Conversely, an application always receives a document together with the ID of the sender endpoint.
- The IDs of the sender and the recipient endpoints are included in metadata of each message.

When an endpoint composes an internal message (either message or acknowledgement), it shall identify it with a UUID (Universal Unique Identifier), as defined in IETF RFC 4122.

NOTE When delivering a document, a recipient endpoint supplies the recipient application with a guaranteed (i.e. authenticated) sender’s identity: the component ID of the sender endpoint. However the sender’s identity is often also included within the document itself, and it is up to the application that analyses the document to check that the two identities match.

## 5.2 Message-type of a message

A MADES system can support multiple and concurrent business processes.

A party “P” having an endpoint connected to the system can operate several applications, implementing functions to support internal activities and exchanges with others parties in accordance to the roles the party plays in the business processes.

The applications request the endpoint to send documents to other parties. The endpoint supports concurrent requests from the applications.

Other parties, while fulfilling their own roles in these business processes, can also send some documents to party “P”. To dispatch correctly the received documents between applications, each message contains a message-type.

The message-type is a mandatory text information provided by a sender application, included in message metadata, transported to the recipient endpoint, and used by a recipient application to retrieve the only documents it is looking for.

Each party is free to organise how he architectures activities, functions and applications in his information system, in a way transparent to the other parties. However, all parties have to agree the message-types as part of the overall information exchange design.

NOTE Message-types are comparable to port numbers: competing applications in a computer use different port numbers for routing correctly received data. Port number usage is widely shared (e.g. 21:FTP, 22:SSH, 25:SMTP) but is not part of the IP protocol, which accepts any integer.

## 5.3 Message route towards a recipient endpoint: message-paths

Each endpoint administrator shall be able to configure a list of message-paths describing how the peer endpoints send messages to his endpoint. Configuration means creation and maintenance of a set of data pushed to the home component-directory of the endpoint, and then published to all components of the system through the directory synchronisation process.

The attributes of a message-path are:

- Path – A delivery route to the endpoint for the messages:

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

"DIRECT" or "INDIRECT:<broker code>"

- Message-type – A text, possibly with a wildcard (*) at the end (e.g. ABC*), which indicates the message-types that can use the path.
- Senders – The set of the peer endpoints, i.e. a list of endpoint codes that can use the path. A wildcard (*) means any endpoint.
- From – The date time (inclusive) from which the path can be used.
- To – The date time (exclusive) until the path can be used; forever when not set.

There could exist several message-paths with the same message-type and non-overlapping usage periods. The usage period can be used to notify in advance to all the endpoints of the system about a planned change of a delivery route.

NOTE Subclause 7.3 discusses the consistency of configuration data.

To send a message with message-type $B$ to a recipient endpoint $R$ at time $T$, a sender endpoint $S$ shall look for a message-path using the following algorithm:

Let $P$ be a set of message-paths. $P$ starts as the set of all the message-paths of the recipient endpoint $R$. Each step of the algorithm excludes some elements from the $P$ set.

Step 1: only keep in $P$ the elements $p$ verifying:

- $p$.From ≤ T < $p$.To: i.e. usable at time T:=Now
- $B$ matches p.Message-type: i.e. matching the message-type of the message to send;
- If $P$ is empty then go to step 4.

Step 2:

- Let $Q$ be the subset of $P$ with elements $p$ verifying $B$=$p$.Message-type (i.e. exact matching).
- If $Q$ is not empty then $P := Q$ and go to step 4.

Step 3:

- Let $n$ be the maximum of length($p$.Message-type) for all $p$ in $P$.
- Let $Q$ be the subset of $P$ with elements $p$ verifying $n$=length($p$.Message-type).
- $P := Q$ (Note that $Q$ cannot be empty since $P$ is not empty).

Step 4: No message-path is found if one of the following conditions holds:

- $P$ is empty or contains more than one element.
- The recipient endpoint is not in the authorised peer list, namely: $R$ not in $p$.Senders where $p$ is the single element in $P$.
- $p$, the single element in $P$, describes a direct transfer to $R$, but the (current) sender endpoint $S$ does not allow for incoming connections so that no acknowledgement back is possible.

An example:

- In the context of a business process (say BP1), an endpoint exchanges messages with message-types BP1-A, BP1-B, BP1-C. However, only the endpoints E1 and E2 can send BP1-C messages and they have to use a different broker.
- A correct configuration includes two message-paths:

  #1: message-type BP1-*: any endpoint uses Broker1 from 01/01/2000 and forever

  #2: message-type BP1-C: endpoints E1, E2 use Broker2 from 01/01/2000 and forever

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- The algorithm does not find a path if E3 endpoint wants to send a BP1-C message to the endpoint. Actually, none of the apparently more widely matching message-types can apply, such as BP1-*, BP1*, BP*, B* or even *, for there is a defined BP1-C message-path.

### 5.4 Restriction on the routes by a broker

Each broker administrator shall be able to configure the list of endpoints allowed using the broker and the list of message-types allowed passing through the broker.

- Configuration means creation and maintenance of data pushed to the home component-directory, and then published to all components of the system through the directory synchronisation process.
- An empty list of endpoints (resp. message-types) means that the broker allows any endpoint (resp. message-type).
- A message-type can end with the wildcard character (*) to match a set of message-types.
- The broker rejects any connection from a non-allowed endpoint and reject any internal message (message or acknowledgement) which message-type is not allowed.

NOTES:

- For a message to pass through a broker, both the sender and the recipient endpoints have to connect to it, the former to send the message, the latter to send the acknowledgement.
- A change in the restriction lists by the broker administrator can never lead to some messages be routed towards alternative brokers. Indeed, such a change can only block or unblock some routes, for the routing algorithm (see 5.3) does not use the brokers' restrictions.

### 5.5 Message acceptance by a sender endpoint

A sender endpoint shall accept a message from a sender application only when all the following conditions are fulfilled:

- The recipient endpoint exists.
- The recipient endpoint owns a valid encryption certificate.
- A message-path for the message-type exists towards the recipient endpoint (outcome of the routing algorithm – see 5.5).
- In case the message-path is an indirect transfer through an intermediate broker, the broker exists and the broker authorises the sender endpoint, the recipient endpoint and the message-type (broker restriction – see 5.4).

### 5.6 Tracking the delivery of a message

#### 5.6.1 Message-status of a message

The system tracks the delivery process of each message. The status of the delivery of a message, named after message-status, expresses the knowledge of the sender endpoint about the delivery. A sender application can request the message-status of a recently sent message, giving the ID returned when the message entered the system.

NOTE See 5.9 concerning the past accessible period.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

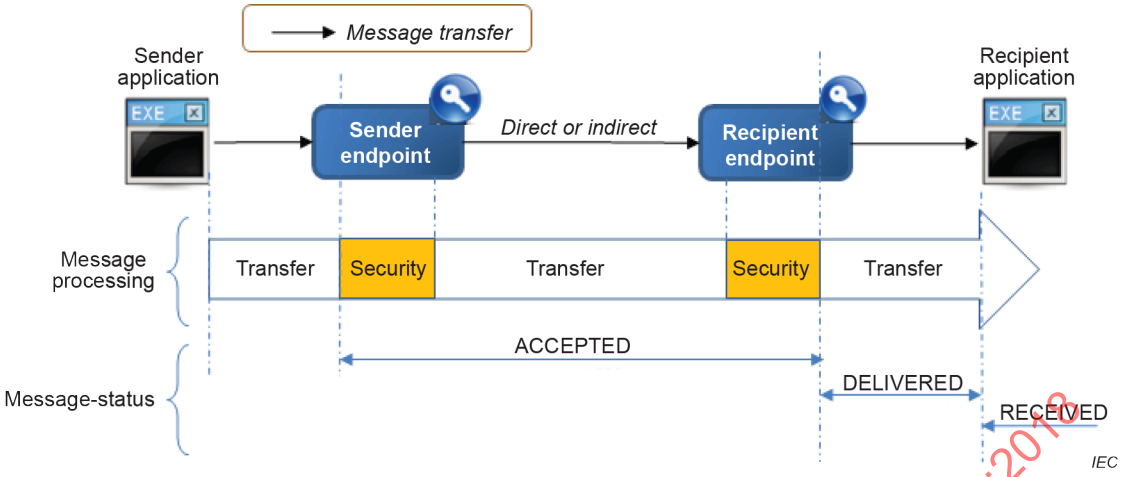



**Figure 22 – Message-status along the delivery**

Figure 22 shows the values of the message-status all along the delivery of the message. The sender endpoint performs the message-level security operations to pack the message, while the recipient endpoint unpacks it.

- ACCEPTED – The system (actually the sender endpoint) accepted the send request from the sender application, created an internal data structure for the message and safely stored it.
- DELIVERED – The recipient endpoint received the message, which passed the security checks, i.e. the content is decrypted and the signature verified, and stored it safely.
- RECEIVED – The recipient endpoint transferred the message to a recipient application, which confirmed reception.
- FAILED – The system cannot guarantee that the recipient endpoint or a recipient application received or will receive the message. In some situations, the reported error means that system cannot and will not deliver the message. E.g., the message expired.

A sender endpoint may (optional) distinguish using the DELIVERING value, part of the ACCEPTED message-status between the message successful transfer and the reception of the acknowledgement.

NOTE The RECEIVED value only means that a recipient application technically accepted the message. A further analysis could result in a "functional acknowledgement" notifying whether the peer party accepts or rejects the content of the message according to the business rules. Such a "functional acknowledgement can" then be a new message that the system will deliver to the original sender endpoint.

### 5.6.2 Delivery events and acknowledgements

The endpoints shall notify the tracking events with characteristics described in Table 1, and the sender endpoint shall update the message-status accordingly, as shown in Figure 23.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

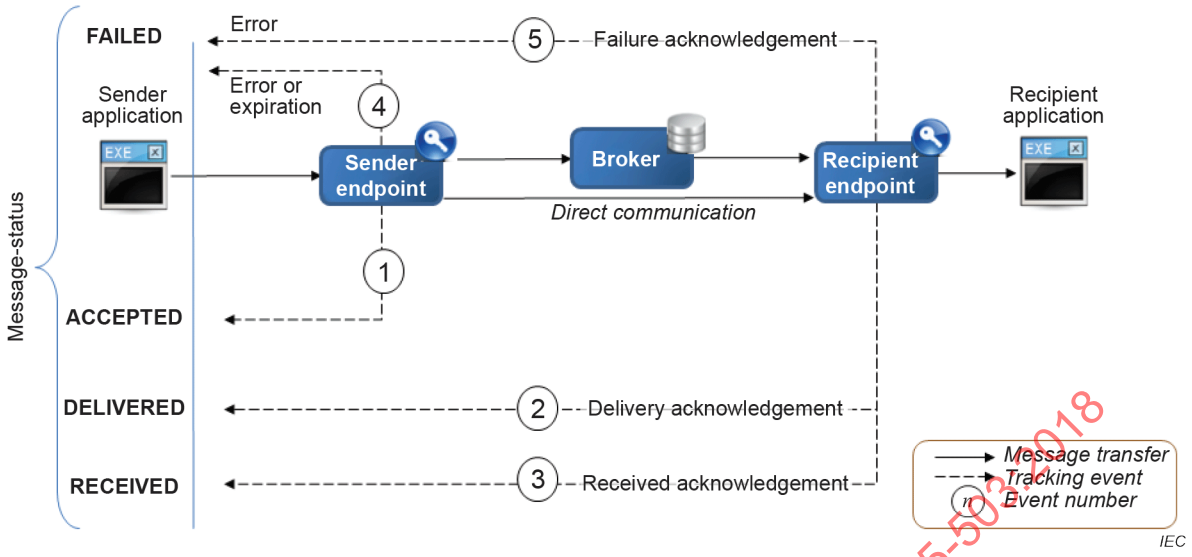



**Figure 23 – Tracking events while delivering a message**

Only the endpoints notify the events, a sender endpoint internally, a recipient endpoint by sending acknowledgements.

Errors occurring during the message transfer between an endpoint and a broker shall be logged by both components, i.e. while the sender endpoint uploads or the recipient endpoint downloads. Such errors could be either reported by the AMQP protocol (see 6.1.7) or by the broker because of restriction (see 6.3). Since routing impossibilities are checked during the message acceptance process (see 5.5), in most situations a stable software could retry transferring the pending messages. Clause 5.7 describes how to notify the message expiration.

An acknowledgement:

- notifies an event on a message named the original-message and which ID is included in acknowledgement metadata;
- has the same message-type as the original-message;
- is never encrypted;
- travels using the reverse route of the original-message, through broker X if it came through broker X, directly to the sender endpoint if it came directly.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 1 – Characteristics of the tracking events**

| Event number | Event characteristics |
|---|---|
| 1 | Message-status: ACCEPTED – see the acceptance conditions in 5.5<br>Event notification: internal to the sender endpoint |
| 2 | Message-status: DELIVERED – The sender endpoint shall set the message-status to FAILED instead of DELIVERED in case the check #3 described in 4.5.4 fails.<br>Event notification: Acknowledgement by the recipient endpoint<br>✓ Content: the non-encrypted message fingerprint of the original message.<br>✓ Internal type: DELIVERY_ACKNOWLEDGEMENT<br>✓ Signed: **Yes** / Encrypted: No |
| 3 | Message-status: RECEIVED<br>Event notification: Acknowledgement by the recipient endpoint<br>✓ Content: irrelevant but as least one character.<br>✓ Internal type: RECEIVE_ACKNOWLEDGEMENT<br>✓ Signed: No / Encrypted: No |
| 4 | Message-status: FAILED<br>Event notification: internal to the sender endpoint |
| 5 | Message-status: FAILED<br>Event notification: Acknowledgement by the recipient endpoint<br>✓ Content: an English readable description of the encountered error or the reason why the delivery of the message failed.<br>✓ Internal type: FAILURE_ACKNOWLEDGEMENT<br>✓ Signed: No / Encrypted: No |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 5.7 Message expiration

Message expiration is a mechanism to notify a sender application that the recipient endpoint did not receive a message in due time.

Every internal message shall have an expiration time set used as follows:

- An endpoint administrator can locally configure the maximum durations for the delivery of the messages that the endpoint sends. Configuration means the possibility to associate maximum durations to message-types including a default value used when no association is defined. The time count starts when the sender endpoint accepts the message (see 5.5).
- The expiration time of an acknowledgement equals the expiration time of the original-message.
- The expiration time travels with the message being included in metadata of the internal message structure (Table 11: expirationTime) and in sections of the AMQP message (Table 5: ttl; Table 6: absolute-expiry-time).
- A component does not transfer an expired internal message to another component or to an application, and it accepts but then ignores an internal message if received already expired.
- A sender endpoint sets to FAILED the message-status of an expired message which message-status is ACCEPTED, i.e. it declares that the delivery of a message failed when no acknowledgement notified that the recipient endpoint received it.

NOTE The recipient endpoint does not issue a failure acknowledgement to notify that a message expired. A default value for the maximum duration of the delivery of every message ensures that no message can be forever delivering (i.e. “zombie” message), for whatever reason.

## 5.8 Reliable transfer of a message

### 5.8.1 Rationale

As a message travels through a delivery chain of components, each component transfers to the next component the responsibility for the safe storage (see 5.9) and for the delivery of the message. The guaranteed delivery of messages between peer applications relies on reliable transfers between the successive components. It does not rely on an end-to-end completeness check, the recipient requesting the sender for resending missing messages.

Let us consider a component and a message passing through. At some moment, the component can be the only one in the delivery chain that contains the message. Therefore, it stores every transiting message on a safe storage; otherwise, the message could be lost in case of a failure.

Let us consider now one step in the delivery of a message, between a sender and a receiver. The two share an identifier of the transiting message assigned by the sender, so that two concurrently transiting messages between them never have the same identifier. The identifier can be either an existing attribute of the message such as a message identifier or an ad hoc identifier which scope is limited to the transfer between the two.

For there is no message lost, the sender settles the transfer after the receiver confirms that it safely stored the message. Settling means, from the sender side, that the message transfer is complete, that it will not transfer the message again, and that it deletes the message for the purpose of the transfer.

When the communication channel recovers from a failure, the sender transfers again the unsettled and non-expired messages because the receiver could have not received or safely stored them. For there is no duplicate message, the receiver maintains (on a safe storage) the identifiers of the recently received messages. Therefore, it can ignore an already received message. When notified that the sender has settled a message, the receiver can delete the message for the purpose of the transfer; this settles the transfer from the receiver side.

A component does not have to wait for the previous component settles a message to transfer it to the next component. For a given message, it is even possible that a component settles the outbound transfer before the inbound transfer; however, in this situation, the component minds the message identifier for stopping the still possible incoming duplicates. Actually, it cannot settle the inbound transfer until the sender confirmed it settled it.

To execute a reliable transfer, the sender and the receiver exchange information as shown in Figure 24. Both can settle independently the transfer of an expired message regardless of confirmations.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

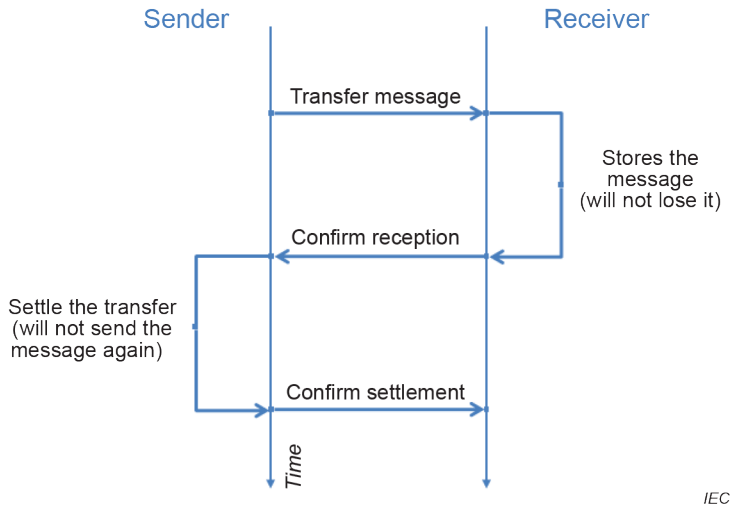



**Figure 24 – Reliable transfer**

Different implementations are possible, where the peer exchange the states of unsettled transfers, either individually or grouped, during the flow and when the communication channel recovers.

Subclauses 5.8.2 to 5.8.4 show how MADES defines reliable transfers along the delivery between applications and components, as shown in Figure 22.

### 5.8.2 Transfer between sender application and sender endpoint

The communication uses the HTTPS protocol, the sender application being the HTTP client and the receiver (i.e. the sender endpoint) being the HTTP server, as shown in Figure 25.

The *SendMessage* (see 11.1.2) request matches the *transfer message* and the response matches the *confirm reception*. The request includes a *conversationID* parameter, which is the identifier of the transiting message.

There is no *confirm settlement*. However, the receiver (i.e. the sender endpoint) shall keep the message, including *conversationID*, message ID and message-status, sometime after the delivery succeeded or failed, if only to respond to a *CheckMesssageStatus* request (see 5.9):

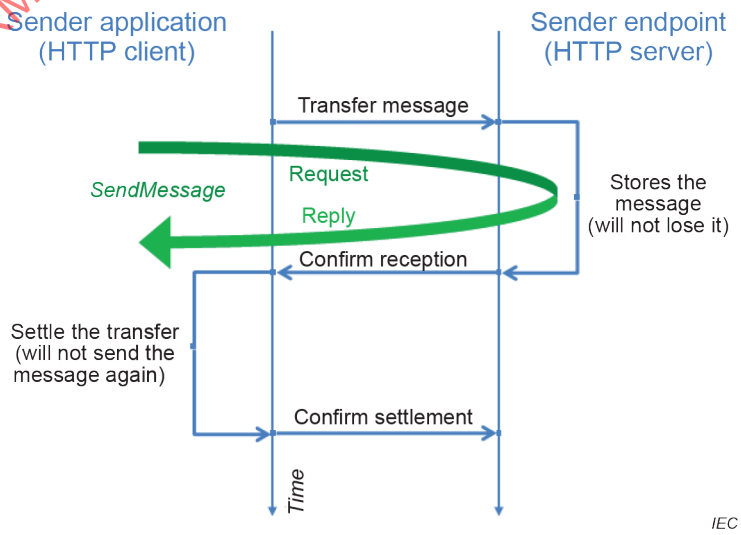



**Figure 25 –Transfer between sender application and sender endpoint**

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 5.8.3 Transfer between components using the AMQP protocol

Whether direct (in one-step) or indirect (in two steps), the message transfer between two components (endpoint, broker) is implemented using the AMQP protocol (see Clause 6).

The AMQP frames exchanged between two AMQP peers, a sender and a receiver, are TRANSFER frames to transfer the messages, and DISPOSITION frames to update the transfer state (i.e. confirmations) – Details in [AMQP 2.6.12].

The sender and the receiver configure the settling policy to *at-least-once* or *exactly-once* delivery guarantee, and either can settle a transfer due to the message expiring, regardless of the other party transfer state.

MADES requirements about AMQP implementation are in Clause 6. In addition, a recipient endpoint shall check and stop possible incoming duplicates of internal messages based on the "message ID" identifier. Indeed, some element in the delivery chain could not fully properly recover and resume from a faulty state, and therefore could prefer to create a duplicate (e.g. see 6.1 – Optional link recovery) that the recipient endpoint will not propagate.

### 5.8.4 Transfer between recipient endpoint and recipient application

The communication uses HTTPS protocol, the receiver (i.e. the recipient application) being the HTTP client and the sender (i.e. the recipient endpoint) being the HTTP server, as shown in Figure 26.

The *ReceiveMessage* (see 11.1.3) response matches the *transfer message*. The message ID (see 5.1) identifies the transiting message.

The *ConfirmReceiveMessage* (see 11.1.4) request matches the *confirm reception* and the response matches the *confirm settlement*.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

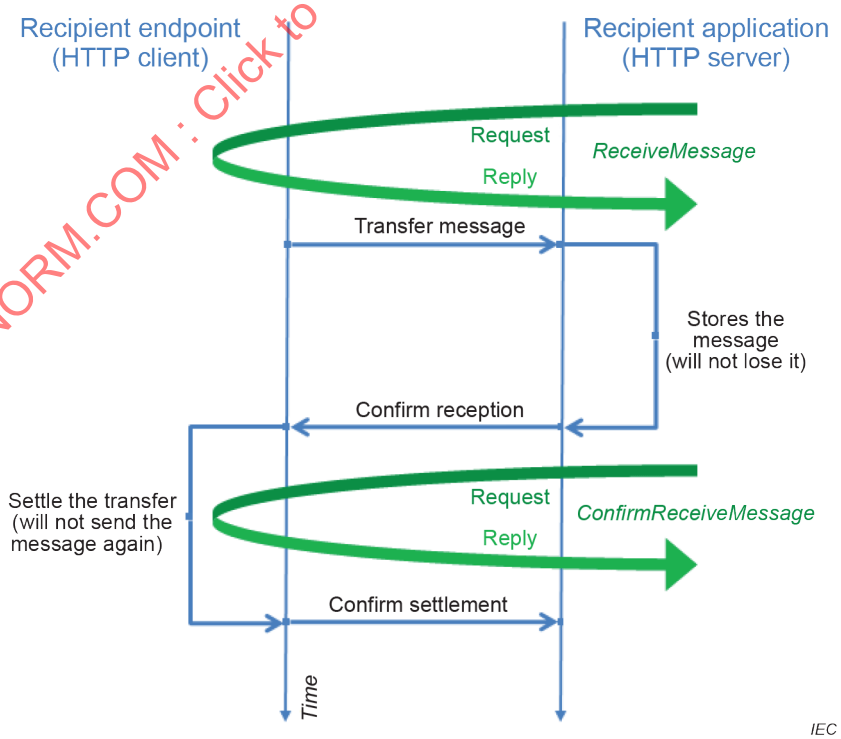



**Figure 26 – Transfer between recipient endpoint and recipient application**

### 5.9 Storing internal messages in components

A component (endpoint, broker) shall safely and temporary store the internal messages:

- Stored data include necessary information to recover the processing of the messages as if the component never stopped in case of failure, namely: internal messages (metadata and content) that the component creates or receives, tracking events (see 5.6.2) and, when applicable, additional transfer identifiers and transfer states (see 5.8).
- Safe storage means that the component effectively recovers consistent data after it stops and restarts, either on demand or on energy failure. Safe storage needs both reliable physical storage and atomic logic when writing.
- An endpoint stores the non-encrypted content of the messages.

For how long? A broker can forget an internal message after it settled its outbound transfer or after the message expired. For the purpose of the message delivery, an endpoint could delete a stored message as soon as it reaches the final delivery state (as defined in Table 2), however:

- A sender application could request the message-status sometime after the delivery of the message succeeded or failed: this requires longer storage of necessary information by the sender endpoint, e.g. as metadata, events and acknowledgements.
- A recipient endpoint needs to memorise the IDs of the received messages to check for duplicates for some more time after it transferred them successfully to a recipient application (see 5.8.3).
- Browsing the recent history of exchanged messages helps operations.

Therefore, an endpoint administrator shall be able configure what (sent / received messages, metadata / content) and how long the endpoint stores a message after it reached the final delivery state.

An endpoint could additionally archive the message proof sets as defined in 4.5.4.

**Table 2 – Final state of a message in an endpoint**

| Component | Final state of a message |
|---|---|
| Sender endpoint | The message-status is among RECEIVED or FAILED. |
| Recipient endpoint | The endpoint sent for the message either a receive acknowledgement or a failure acknowledgement or a tracing acknowledgement, or the message expired. |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 5.10 Message priority

The specification does not define a priority attribute for a message. The components (endpoints, brokers) could locally prioritise messages, e.g. according to the message-types.

### 5.11 Message delivery order

Assume an $A_1$-application sent an $M_1$-message followed by an $M_2$-message, the two with the same message-type, to an $A_2$-application. The specification does not require that, if $A_2$-application receives both messages, it should receive the $M_1$-message first; for this would restrict the use of potential multithreading (see 6.4) and could constrict performance.

### 5.12 Testing a route between two endpoints: tracing-messages

A tracing-message is an internal message used to check the end-to-end connectivity between two endpoints. A sender endpoint shall expose the *ConnectivityTest* service to send a tracing-message and a recipient endpoint acknowledge the tracing-messages.

- A sender application requests a connectivity test and indicates a recipient endpoint and a message-type.
- The sender endpoint creates and transfers a tracing-message to deliver to the recipient endpoint. Message metadata informs that it is a tracing-message (TRACING_MESSAGE – see Table 12). The content of a tracing-message is irrelevant but shall not be empty.
- The sender endpoint signs the message and encrypts the content. Therefore, the successful delivery includes the verification of the certificates' set-up and processing.
- The delivery route depends on the message-type as for any message.
- A broker does not distinguish the tracing-messages and the tracing-acknowledgements from others internal messages.
- When receiving the message, the recipient endpoint sends a tracing acknowledgement (TRACING_ACKNOWLEDGEMENT – see Table 12) and never delivers the message to an application.
- The sender endpoint sets the message-status directly to DELIVERED when receiving the corresponding tracing-acknowledgement.
- To know if a tracing-message reached the recipient endpoint, the sender application requests for the message-status, as for any message.

# 6 Transferring messages using the AMQP protocol

## 6.1 Main principles of the AMQP specification

### 6.1.1 Introduction

NOTE Subclause 6.1 outlines the principles of AMQP 1.0. It does not contain requirements but introduces terms and objects further used in the document. [AMQP xx] references the section xx of the AMQP specification (see Clause 2).

AMQP defines a *wire-level* protocol for *business messaging*. Wire-level means that AMQP describes the byte-flow structures, named after *frames*, exchanged on the telecommunication support between two TCP[4] peers. *Business messaging* means reliable transfer of messages between both peers (applications) in either direction.

- AMQP describes how the peers first establish a *connection*, then prepare and share a whole exchange context using *sessions* and *links* including flow control rules and agreed delivery policies, e.g. at-most-once, at-least-once or exactly-once. Then the peers interact transferring messages in compliance with the agreement.
- AMQP also defines *transaction messaging*, i.e. how to group interactions within *atomic transactions* to allow for the coordinated outcome of otherwise independent transfers. This extends to an arbitrary number of transfers spread across any number of distinct links in either direction.

Therefore, the AMQP definition allows for all common behaviours of business messaging. Note that:

- AMQP is a symmetric protocol and is agnostic to any coding language.
- AMQP does not describe how the peers manage and store the messages, either before or after the transfer. The links between them refer to *sources* and *targets*, which are names that each manages its own way. High-level implementations, such as java messaging service (JMS) come with additional safe and transactional storage: queues (see 6.2).

AMQP defines nine frames to open / close connections, to begin / end sessions and to attach / detach links between the peers, so they can reliably transfer messages and control the transfer flow as show in Figure 27.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

4 "Transmission Control Protocol" from TCP/IP.

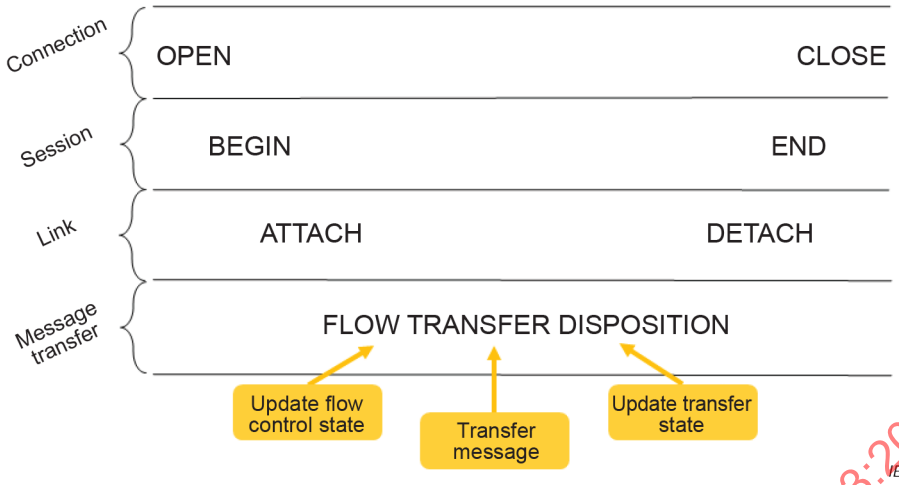



**Figure 27 – The nine AMQP frames**

### 6.1.2 Connection Open

Opening the connection between TCP peers entails the following steps:

- Exchange of the AMQP protocol version headers [AMQP 2.2].
- Exchange of protocol headers, specifically:
  - Negotiation and initiation of TLS [AMQP 5.2].
  - Negotiation of authentication method and performing authentication via SASL (Simple Authentication and Security Layer) [AMQP 5.3].
- Exchange of OPEN frames between the peers [AMQP 2.4.1, 2.7.1].

The negotiation of TLS can occur inline as listed above or the socket can be overlaid with TLS before the AMQP protocol version negotiation [AMQP 5.2.1]. The latter method is performed over TCP port 5671 (i.e. the IANA AMQPS service), the inline upgrade over port 5672.

The SASL handshake implements RFC 4422. Username and password models commonly use the SASL PLAIN. The authentication with a certificate uses SASL EXTERNAL, also defined in RFC 4422.

The OPEN frames [AMQP 2.7.1] negotiate the number of "channels" that both peers want to create, and what the maximum frame size, and therefore the maximum payload message size, is. They also negotiate some further parameters that only matter to AMQP stacks.

Once the peer party responds with OPEN, the connection is established. [AMQP 2.4.7] describes the full OPEN/CLOSE cycle state diagram. The only permitted action on top of a connection is session establishment.

### 6.1.3 Session begin

[AMQP 2.5, 2.7.2] Sessions are communication paths between the two peers that allow throttling the flow, based on receive and send windows. The session concept exists so that receivers can manage the number of outstanding frames which processing is pending. This is commonly an implementation detail inside the AMQP stacks.

To establish a session, the peers exchange BEGIN frames over an existing connection. Each session consumes one "channel" from each side of the connection, forming a bi-directional path. The only permitted action on top of a session is link establishment.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 6.1.4 Link attachment

Links are unidirectional transfer paths for messages, established over a session [AMQP 2.6]. The communication between both peers is *bi-directional*, meaning that they mutually exchange transfer states, but the messages in a link only flow in one direction, from the sender to the receiver.

Either party can establish (attach) links using the ATTACH frame [AMQP 2.7.3] in either direction between a source and a target: an outbound link for sending messages, an inbound link for receiving messages. The source and the target refer to resources that each peer manages.

Links can live longer than the connection on which they were established. The sender and the receiver can use their retained state to reattach the link on a new connection (and session) with precise control over delivery guarantees (e.g. ensuring the exactly-once delivery).

### 6.1.5 Message transfer

Message transfers are performed using three frames:

- FLOW is used for flow control, allowing messages be transferred or not.
- TRANSFER moves a message from the sender to the receiver [AMQP 2.7.5]. A message can be carried by a single transfer up to the maximum negotiated frame size for the connection. Larger messages are split across several TRANSFER frames gathered in a single transaction.
- DISPOSITION is sent by the receiver back to the sender indicating the processing state of the message.

The flow model is generally based on "link credit" [AMQP 2.6.7, 2.7.4].

- When receiving a message, the receiver can grant the sender one unit of link credit. If a message is available, the sender satisfies the link credit by transferring one message with the TRANSFER frame. The receiver could also grant 10, 100 or virtually unlimited link credits and the sender would keep sending available messages until the credit is used up.
- One unit of link credit is consumed when the message is "settled" using the DISPOSITION frame [AMQP 2.7.6]. Settling means that the message transfer is complete.

### 6.1.6 Link recovery and resends

When a connection drops, the AMQP protocol is capable of recovering the state of previously attached links over a new connection and session, and resume any interrupted and therefore unsettled transfers [AMQP 2.6.3/2.6.4].

Link resumption is *optional* and not consistently available also because it is difficult to surface into user APIs (application programming interfaces). The practice of throwing an exception upon link/session/connection loss and re-establishing a fresh connection/session/link is nearly as efficient. JMS implementation can probably not resume links.

### 6.1.7 Error management

Errors can occur at any level: connection [AMQP 2.8.16], session [AMQP 2.8.17], link [AMQP 2.8.18] or transfer, e.g. message is rejected [AMQP 3.4.3].

### 6.1.8 Message structure

[AMQP 3.2] "The term *message* is used with various connotations in the messaging world. The sender might like to think of the message as an immutable payload handed off to the messaging infrastructure for delivery. The receiver often thinks of the message as not only that immutable payload from the sender, but also various annotations supplied by the

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

messaging infrastructure along the way. To avoid confusion we formally define the term *bare message* to mean the message as supplied by the sender and the term *annotated message* to mean the message as seen at the receiver".

- An *annotated message* consists of the bare message plus sections for annotation at the head and tail. There are two classes of annotations: annotations that travel with the message indefinitely and annotations consumed by the next element in the delivery chain.
- The *bare message* consists of three sections: (standard) properties, application-properties, and application-data (the body).

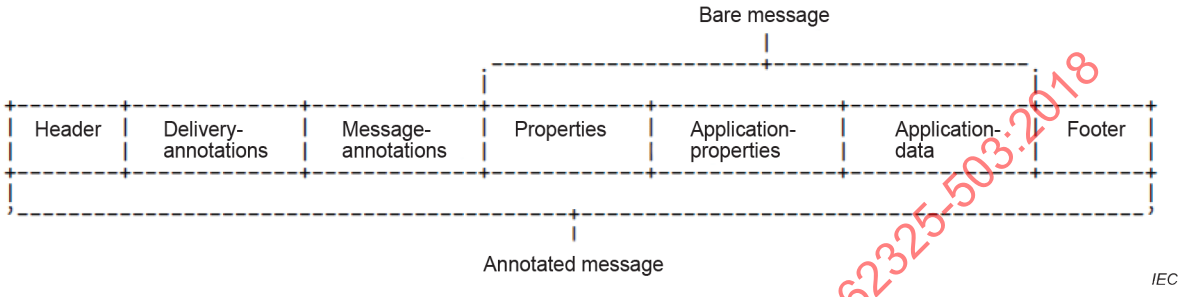



**Figure 28 – Structure of an AMQP message**

## 6.2 AMQP high-level implementation: the client/broker model

Warning: In the entire document apart from this subclause, the word "broker" refers the component defined in 4.4.1, i.e. a "MADES-broker". In this subclause, a broker is an implementation of one the AMQP peers that comes with additional features and named after an "AMQP-broker". Subclause 6.3 will explain how these two brokers relate.

High-level implementations seems to break the AMQP protocol symmetry. Indeed, they are client / server implementations in which the two peers, referred as client and broker, play different roles and implement different functions and services:

- The client opens connections and begins sessions. The client attaches outbound links to *produce* messages, i.e. to send messages that the broker stores in target queues. The client attaches inbound links to *consume* messages, i.e. to receive messages from broker source queues.
- The broker is a server that responds to the client requests. The broker hosts queues, i.e. safe and temporary storage for messages. The broker implements attachments and transactional accesses to queues by concurrent clients. The broker automatically transfers a message in a queue to one of the clients attached as consumer to that queue and with remaining link credit.

  NOTE 1 This includes the possibility to have multiple producers or consumers attached on a queue by a single client to take advantage and performance of multithreading features.

  NOTE 2 The broker "pushes" data towards the client in a way similar to *websocket*, which is a protocol over a single TCP connection and providing a full-duplex communication channel.

Actually, the dissymmetry appears because one of the peers, the AMQP-broker, implements sources and targets as first-in/first-out (FIFO) queues stored on safe disk storage, whereas the other peer, the AMQP-client, leaves such implementation open and provides APIs (e.g. produce, consume callback) to build applications.

A client / broker implementation hides AMQP details and provides high-level services dealing with queues, consumers and producers. In addition, the client can declare and commit transactions to settle the transfers, e.g. to ensure that the broker does not delete a consumed message from a source queue until it (i.e. the client) processes that received message successfully.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

Table 3 translates in the AMQP vocabulary the high-level services of the client / broker model. Subclause 6.3 describes using these high-level services how the MADES components exchange messages.

**Table 3 – Services of the client / broker model**

| Operation initiated by the client in client / broker model | Correspondence with AMQP specification |
|---|---|
| The client connects to the broker | The client opens a connection and begins a session with the broker. |
| The client attaches a consumer to queue X. | The client (receiver) attaches an inbound link with a broker (sender) source X. |
| The client attaches a producer to queue X | The client (sender) attaches an outbound link with a broker (receiver) target X. |
| The client produces a message towards queue X | The client (sender) transfers a message to the broker (receiver) using the attached outbound link which target is X. The broker stores the received message in queue X |
| The client consumes a message from queue X | The broker (sender) transfers a message to the client (receiver) out of queue X using the attached link which source is X. |

## 6.3 AMQP implementation in MADES components

AMQP communication between MADES components shall implement the client / broker model, as follows and as shown in Figure 29.

- A MADES-broker implements an AMQP-broker, hence its name.
- An endpoint implements an AMQP-client to transfer internal messages to and from MADES-broker queues, and to produce internal messages to other endpoints. Therefore, an endpoint also implements an AMQP-broker to receive directly internal messages.

NOTE Note the symmetric role of the endpoints when directly exchanging a message: the client/broker roles swap between the message and the acknowledgement (both are internal messages).

- Each queue in an AMQP-broker bears the name (i.e. the code) of an endpoint. An endpoint only consumes internal messages in a queue bearing its own name, i.e. one in every MADES-broker that it uses, and one internally for incoming direct transfer.
- An endpoint always produces an internal message to a queue bearing the name of the recipient endpoint of the internal message.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

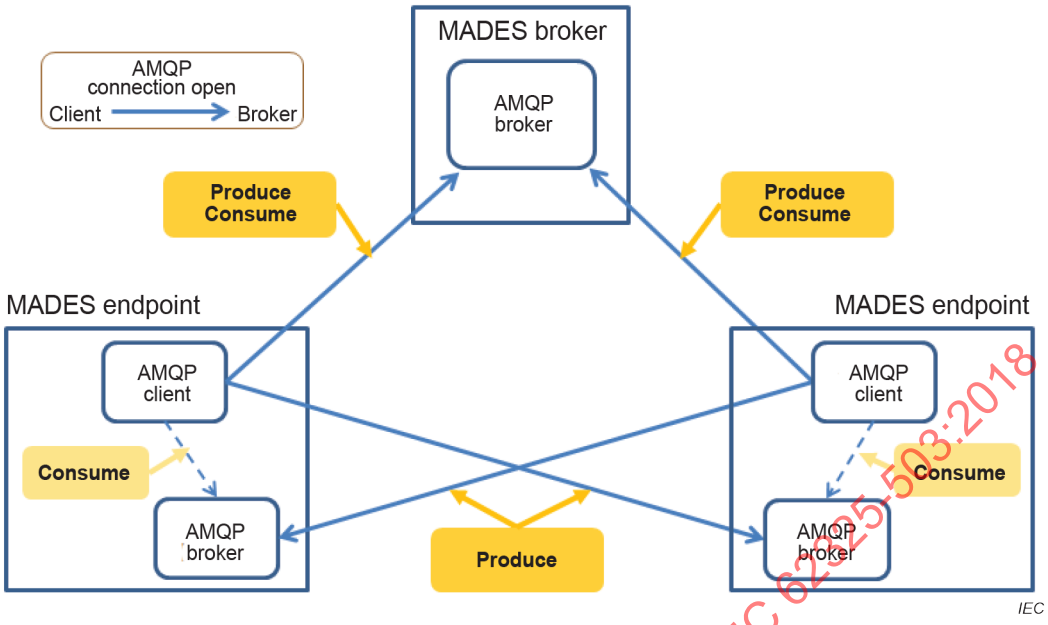



**Figure 29 – AMQP in MADES components**

The AMQP structure for transferring a MADES internal message is defined in 6.5.2 and includes the following fields:

- *senderCode* (application-properties): identifier of the sender endpoint.
- *receiverCode* (application-properties) identifier of the recipient endpoint.
- *subject* (properties): message-type.

Table 4 defines all the conditions, which a MADES component shall check when acting as a server (i.e. AMQP-broker). The client is always an endpoint. If a condition is not fulfilled, the server rejects the connection, the attachment or the message and, when applicable, returns an AMQP error [AMQP 2.8.15] to the client.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 4 – Rules for setting up connection/attachment and for message transfer**

| Operation | Conditions verified by server component for the operation to succeed |
|---|---|
| *(any operation)* | An operation shall never success if the client component code does not exist in configuration data at the time of the operation (e.g. after it has been revoked – see 8.6). |
| Connect | *(endpoint, broker)*<br>• A client connects using AMQPS (i.e. on the 5671 port with TLS occurring before AMQP protocol version negotiation).<br>• The connecting client authenticates using a valid authentication certificate delivered to an endpoint.<br>(*broker*) the broker allows the endpoint to connect (see 5.4).<br>(*endpoint*) The server endpoint accepts direct incoming connection. |
| Attach a consumer | (*broker*) The queue requested for the attachment bears the name (i.e. code) of the client endpoint.<br>*(endpoint)* This operation shall never success. |
| Attach a producer to queue X | (*broker*) X, the queue requested for the attachment, bears the name of a valid endpoint that the broker allows for connecting (see 5.4).<br>(*endpoint*) X, the queue requested for the attachment, is the code of the server endpoint. |
| Produce an internal message towards queue X | *(endpoint, broker)*<br>• The *receiverCode* field of the transferred internal message equals X and exists in configuration data at the time of the operation.<br>• The *senderCode* fields equals the connected endpoint code.<br>(*broker*) The *subject* field of the internal message is a message-type that the broker allows (see 5.4). |

The implementation shall provide means to configure the AMQP parameters in compliance with the MADES system governance, e.g. maximum size of the messages, exactly-once delivery policy.

The management of the queues by the broker could be:

- *static*: i.e. manually configured by the broker administrator;
- *dynamic*: (recommended) e.g. the broker creates a queue when validly referred for attaching a link (consumer or producer); the broker deletes the queue when empty and no more referred for some (configurable) time by the currently attached links.

## 6.4 Management of AMQP connections and attachments by an endpoint

An endpoint should open at most one connection at a time to a broker or to a peer endpoint.

NOTE One per node when a cluster.

An endpoint shall listen for internal messages, or equivalently have a consumer attached, to any broker from which it might receive a message, which are the following:

- The brokers referred in one of the *currently usable message-paths* of the endpoint.
- The brokers to which the endpoint recently sent a message that is non-expired and has the ACCEPTED message-status, for an acknowledgement should come back.

On small systems with predictable message traffic, the connections and the attachments of an endpoint could be configured statically, e.g. the administrator describes which attachments to which components are permanently open, either at start-up or when recovering from a connection loss. However, the endpoint should manage connections and attachments dynamically according to the message-paths and to the outgoing message traffic.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

To take advantage of the multithreading feature of endpoints and brokers, a consumer (resp. producer) could actually be a group of consumers (resp. producers). E.g., a group could then be managed as a whole, all members being attached and detached (nearly) simultaneously. The endpoint administrator could be able to define and configure such groups for performance reason. E.g. *3 consumers to broker B7; 3 producers to endpoint E3 through broker B5; 2 producers to endpoint E3* (direct outgoing transfer); *2 consumers to local queue* (direct incoming transfer).

## 6.5 Internal message format

### 6.5.1 Definitions, design and security checks

An internal message consists of two parts: content and metadata.

- The *content* of a (standard) message is the immutable document that a sender application requests the system to transport in an encrypted way; Subclause 5.6.2 defines the content of the acknowledgements.
- *Metadata* of an internal message is a set of attributes either provided by the sender application or created by the endpoint sending the internal message. Table 11 describes the entire list of the metadata attributes, which (not exhaustively here) includes message ID, message-type, expiration time, sender and receiver codes and, when applicable, signature and information on how the message is signed and encrypted (*processingMetadata*).

The AMQP bare message contains the internal message content and metadata. The AMQP body includes both. In addition, the AMQP properties sections partly include metadata (details in 6.5.2). The reason for redundancy is twofold: the body embeds metadata for backwards compatibility (with the draft technical specification) in XML format; a broker only uses AMQP fields to accept and route the message (see 6.3, Table 4).

### 6.5.2 AMQP format for transferring internal messages

The section describes the format of the AMQP message (see 6.1.8) to which an endpoint shall comply, when transferring an internal message (i.e. message or acknowledgement), to either a broker (indirect transfer) or an endpoint (direct transfer).

Table 5, Table 6 and Table 7 respectively describe the fields used in the AMQP sections: *header*, *properties* and *application-properties*. Lower layers of the AMQP stack can use the other fields (e.g. *message-id*) of the section *properties* [AMQP 3.2.4]. Each field in the section *application-properties* bears the name of the corresponding metadata element in Table 11. The body (Table 8 – *application-data* section) is a two-element *amqp-sequence* [AMQP 3.2.7].

**Table 5 – Internal message – AMQP format: header section**

| Field Name | AMQP type | Value | Required |
|---|---|---|---|
| ttl | milliseconds | Time-To-Live (i.e. the duration until the internal message expires) | True |

**Table 6 – Internal message – AMQP format: properties section**

| Field Name | AMQP type | Value (internal message metadata | Required |
|---|---|---|---|
| absolute-expiry-time | string | Expiration date and time of message (*expirationTime*). | True |
| subject | string | Message-type of the internal message (*messageType*). | True |
| correlation-id | string | ID of the original message when the internal message is an acknowledgement (*relatedMessageID*). | False |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 7 – Internal message – AMQP format: application-properties section**

| Field Name | AMQP type | Value | Required |
|---|---|---|---|
| messageID | string | ID of the internal message assigned by the endpoint. | True |
| receiverCode | string | Code of the endpoint that will receive the internal message | True |
| senderCode | string | Code of the endpoint that sends of the internal message | True |
| senderApplication | string | Display name of the sender application (when the internal message is a message). | False |
| baMessageID | string | ID of the message assigned by the sender (business) application (when the internal message is a message). | False |
| generated | dateTime | Date and time when the sending endpoint created the internal message. | True |
| internalType | string | Type of the internal message – see Table 12 (do not confuse with the message-type) | True |
| messageMversion | integer | MADES version of the specification with which the internal message complies – see 9 | True |

**Table 8 – Internal message – AMQP format: application-data section**

| Field Name | AMQP type | Value | Required |
|---|---|---|---|
| Element #1 | string | XML string of message metadata complying with the element *messageMetadata* of the XML schema given in 6.5.5. | True |
| Element #2 | binary | *Message* → Outcome of the encryption of the content provided by the sender application (see Table 29)<br>*Acknowledgement, tracing message* → Plain text message | True |

### 6.5.3 Encryption

A sender endpoint shall encrypt the content of every message using the encryption certificate of the recipient endpoint (see 8.1 and 4.5.3 for the encryption principles). The acknowledgements shall not be encrypted.

Encryption means that:

- the content of the message (*application-data*, element #2) is encrypted;
- the *processingMetadata* element (see Table 11) includes a *messageProcessor* item, which *processorID* equals "encryption", and which *processorData* describes how the content of the message was algorithmically encrypted.

Technically *processorData* is a collection of attributes, each a {name, type, value} triplet. The attribute named "Cipher" is mandatory, which indicates the used encryption technique.

The current specification only describes one encryption technique named "AES-256" as follows:

- A state of the art cryptographic generator produces a 256-bit random key. The message-content is encrypted with the "advanced encryption standard" (AES) algorithm and the generated key.
- The key is encrypted with the RSA (Rivest, Shamir, Adleman) algorithm and stored in the "Session key" attribute.
- The ID of the encryption certificate of the recipient endpoint , which public key is used by the RSA algorithm is stored in the "Certificate ID" attribute.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- Table 9 shows the resulting *processorData* attributes' collection.

**Table 9 – Encryption – Processing metadata attributes for the "AES-256" cipher**

| Metadata Attribute Name | Metadata Type | Description |
|---|---|---|
| Cipher | STRING | "AES-256" |
| Certificate ID | STRING | The ID of the encryption certificate of the recipient endpoint. |
| Session key | BYTE_ARRAY | The RSA-encrypted 256bit key used to encrypt the message-content. |

NOTE Other "Cipher" values can be defined using other key lengths and/or cryptographic algorithms (e.g. a new algorithm more resistant to attacks). The system governance defines which 'cipher' to used.

When receiving an internal message the recipient endpoint shall check if it is encrypted, i.e. if the message *processingMetadata* contains an "encryption" *messageProcessor*. In case it is and the Cipher value is "AES-256" then the endpoint shall:

1) Verify that the certificate ID used to encrypt the internal message is its own encryption certificate.
2) Verify that the encryption certificate was valid when the internal message was generated: the certificate expiration date-time is a certificate attribute; the generated date-time of an internal message is part of message metadata.
3) Decrypt the session key using RSA algorithm and the private key of the certificate.
4) Decrypt the message-content using the decrypted session key and the AES algorithm.

### 6.5.4 Signing

A sender endpoint shall sign every message (see 8.1 and 4.5.3 for the signing principles) using its signing certificate. The recipient endpoint shall sign the delivery acknowledgement (see 5.6.2). The manifest used to compute the fingerprint is:

content + *baMessageID* + *extension* + *generated* + *internalType* + *messageID* + *relatedMessageID* + *receiverCode* + *senderCode* + *senderApplication* + *messageType*.

Where:

- "content" is the *unencrypted* content provided by the sender application (see Table 29).
- the italic names refer to elements of internal message metadata as described in Table 11;
- "+" means the binary concatenation of the UTF-8 encoded attributes.

Signing means that the *processingMetadata* element (see Table 11) includes a *messageProcessor* item, which *processorID* equals "signature", and which *processorData* describes how the signature was algorithmically processed. Technically *processorData* is a collection of attributes, each a {name, type, value} triplet. The attribute named "Algorithm" is mandatory, which indicates the used signature technique.

The current specification only describes one signature technique named "SHA-512", as follows:

- The message hash (i.e. fingerprint) is computed from the manifest with the "secure hash algorithm" SHA-512 algorithm.
- The message hash is then encrypted with the RSA algorithm. The ID of the signing certificate of the signing endpoint, which private key is used by the RSA algorithm is stored in the "Certificate ID" attribute.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- An XML signature is generated, which complies with the W3C recommendation “XML Signature Syntax and Processing standard” *(See example in 6.5.6),* and stored in the "Signature" attribute. The XML signature includes the encrypted message hash.
- Table 10 shows the resulting *processorData* attributes' collection.

**Table 10 – Signing – Processing metadata attributes for the "SHA-512" Algorithm**

| Metadata Attribute Name | Metadata type | value |
|---|---|---|
| Algorithm | STRING | "SHA-512" |
| Certificate ID | STRING | The ID of the signing certificate of the endpoint that sent and signed the internal message. |
| Signature | STRING | The XML signature compliant with the “*XML Signature Syntax and Processing standard*”, embedded here as a string. |

NOTE Other "Algorithm" values can be defined.

When receiving an internal message an endpoint shall check if it is signed, i.e. if the message *processingMetadata* contains a "signature" *messageProcessor*. In case it is and the Algorithm value is "SHA-512" then the endpoint shall verify the signature as follows:

1) Verify that the signing certificate belongs to the endpoint that sends the message.
2) Verify that the signing certificate was valid when the internal message was generated (Certificate expiration date-time is a certificate attribute. The generated date-time of a message is part of message metadata).
3) Compute the fingerprint of the received internal message using the “Algorithm”.
4) Decrypt the encrypted message hash included in the XML signature using the public key of the signing certificate.
5) Verify that the fingerprints, outcomes of the operations 3 and 4, are equal.

### 6.5.5 Internal message metadata

Table 11 lists the metadata elements of an internal message.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 11 – MessageMetadata (type)**

| Element name | Element type | Description | Required | O [a] |
|---|---|---|---|---|
| messageID | string | The ID of the message assigned by the endpoint that creates the internal message – see 5.1. | True | E |
| receiverCode | string | *Message, Tracing-message* → The component ID of the recipient endpoint.<br>*Acknowledgement* → The *senderCode* of the original message. | True | A<br>E |
| messageType | string | *Message, Tracing-message* → The message-type as provided by the sender application – see 5.2.<br>*Acknowledgement* → The message-type of the original message. | True | A<br>E |
| extension | string | *Message* → The file extension of the document – used when the sender application or the recipient uses the FSSF interface – see 11.1.7.<br>*Acknowledgement, Tracing-message* → Not used. | False | A<br>∅ |
| generated | dateTime | The date and time when the endpoint created the internal message. | True | E |
| expirationTime | dateTime | *Message, Tracing-message* → The expiration date and time of the message – set by the sender endpoint when accepting the message – see 5.7.<br>*Acknowledgement* → The *expirationTime* of the original message. | True | E<br>E |
| senderCode | string | The component ID of the endpoint that created the internal message – see 5.1. | True | E |
| internalType | InternalMessageType (see Table 12) | The technical type of the message. | True | E |
| relatedMessageID | string | *Message, Tracing-message* → Not used.<br>*Acknowledgement* → The message ID of the original message. | False | E |
| senderApplication | string | *Message, Tracing-message* → The identifier of the application sending the document.<br>*Acknowledgement* → Not used. | False | A<br>∅ |
| baMessageID | string | *Message, Tracing-message* → An identifier of the document provided by the sender (business) application.<br>*Acknowledgement* → Not used. | False | A<br>∅ |
| processingMetadata | ProcessingMetadata (see Table 13) | Metadata added to describe how the internal message is signed and/or encrypted | False | E |
| messageMversion | integer | The MADES version with which the message complies – see Clause 9. | True | E |

[a] The "O" column stand for "Origin" and tells from where the element value comes, A: the sender application (see 11.1.2), E: the sending endpoint. ∅: a null or equivalently empty element.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 12 – InternalMessageType (type: string enumeration)**

| String Value | Description |
|---|---|
| STANDARD_MESSAGE | A message but not a tracing-message. |
| DELIVERY_ACKNOWLEDGEMENT | An acknowledgement notifying that the recipient endpoint accepted the original STANDARD_MESSAGE. – see 5.6.2. |
| RECEIVE_ACKNOWLEDGEMENT | An acknowledgement notifying that the recipient endpoint transferred the original STANDARD_MESSAGE to a recipient application. – see 5.6.2. |
| FAILURE_ACKNOWLEDGEMENT | A failure-acknowledgement. – see 5.6.2. |
| TRACING_MESSAGE | A tracing-message – see 5.12. |
| TRACING_ACKNOWLEDGEMENT | An acknowledgement notifying that the recipient endpoint accepted the original TRACING_MESSAGE – see 5.12. |

**Table 13 – ProcessingMetadata (type)**

| Element name | Element type (see Table 14) | Description | Required |
|---|---|---|---|
| messageProcessors | MessageProcessor[] | A collection of metadata, each from a used message processor | False |

**Table 14 – MessageProcessor (type)**

| Element name | Element type | Description | Required |
|---|---|---|---|
| processorID | string | The unique ID of the message processor: “signature” or “encryption” | True |
| processorData | Map (see Table 15) | A collection of triplets: {key, type, value} | True |

**Table 15 – Map (type)**

| Element name | Element type (see Table 16) | Description | Required |
|---|---|---|---|
| entries | MapEntry[] | A collection of data, each provided with a name and a value, i.e. a set composed of a key (name), a type (format) and a value (according to the type). | False |

**Table 16 – MapEntry (type)**

| Element name | Element type | Description | Required |
|---|---|---|---|
| key | string | The name of metadata | True |
| type | ValueType (see Table 17) | The type/format of metadata. | True |
| value | string | The value of metadata. | True |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 17 – ValueType (type: string enumeration)**

| String Value | Description |
|---|---|
| STRING | String |
| LONG | A 64-bit number expressed as string. Example number 42 is represented as string "42" (without quotes) |
| BYTE_ARRAY | ⇔ base64Binary type |
| BOOLEAN | A string equal to “true” or “false”. |

## XML schema for internal message metadata

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mades.entsoe.eu/internalMessaging"
   xmlns:mades="http://mades.entsoe.eu/internalMessaging">

 <xsd:simpleType name="InternalMessageType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="STANDARD_MESSAGE"/>
   <xsd:enumeration value="DELIVERY_ACKNOWLEDGEMENT"/>
   <xsd:enumeration value="RECEIVE_ACKNOWLEDGEMENT"/>
   <xsd:enumeration value="FAILURE_ACKNOWLEDGEMENT"/>
   <xsd:enumeration value="TRACING_MESSAGE"/>
   <xsd:enumeration value="TRACING_ACKNOWLEDGEMENT"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="ValueType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="STRING"/>
   <xsd:enumeration value="LONG"/>
   <xsd:enumeration value="BYTE_ARRAY"/>
   <xsd:enumeration value="BOOLEAN"/>
   </xsd:enumeration>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:complexType name="Entry">
  <xsd:sequence>
   <xsd:element name="key" type="xsd:string"/>
   <xsd:element name="type" type="mades:ValueType"/>
   <xsd:element name="value" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="Entries">
  <xsd:sequence>
   <xsd:element maxOccurs="unbounded" minOccurs="0" name="entry" type="mades:Entry"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="Map">
  <xsd:sequence>
   <xsd:element name="entries" type="mades:Entries"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MessageProcessor">
  <xsd:sequence>
   <xsd:element name="processorID" type="xsd:string"/>
   <xsd:element name="processorData" type="mades:Map"/>
  </xsd:sequence>
 </xsd:complexType>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
 <xsd:complexType name="MessageProcessors">
  <xsd:sequence>
   <xsd:element maxOccurs="unbounded" minOccurs="0" name="messageProcessor"
type="mades:MessageProcessor"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MessageProcessingMetadata">
  <xsd:sequence>
   <xsd:element name="messageProcessors" type="mades:MessageProcessors">
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MessageMetadata">
  <xsd:sequence>
   <xsd:element name="messageID" type="xsd:string"/>
   <xsd:element name="receiverCode" type="xsd:string"/>
   <xsd:element name="messageType" type="xsd:string"/>
   <xsd:element minOccurs="0" name="extension" nillable="true" type="xsd:string"/>
   <xsd:element name="generated" type="xsd:dateTime"/>
   <xsd:element minOccurs="0" name="expirationTime" nillable="true"
type="xsd:dateTime"/>
   <xsd:element name="senderCode" type="xsd:string"/>
   <xsd:element name="internalType" type="mades:InternalMessageType"/>
   <xsd:element minOccurs="0" name="relatedMessageID" nillable="true"
type="xsd:string"/>
   <xsd:element minOccurs="0" name="senderApplication" nillable="true"
type="xsd:string"/>
   <xsd:element minOccurs="0" name="baMessageID" nillable="true" type="xsd:string"/>
   <xsd:element name="processingMetadata" type="mades:MessageProcessingMetadata"/>
   <xsd:element name="messageMversion" type="xsd:int"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:element name="messageMetadata" type="mades:MessageMetadata"/>

</xsd:schema>
```

### 6.5.6 XML signature example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha512"/>
    <Reference URI="">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha512"/>
      <DigestValue>eVpInNsCIWzEjdrxxvongO2rnQ4=</DigestValue>
    </Reference>
  </SignedInfo>
    <SignatureValue>aD9HNiTmVxW+HnDOpSjzwDB+MypGTC7yb3/HUpAZKmEhRwQC0eBwYcZSRTqF8VdzmneH6abq2P+m
vHNXPC53i3mF58XDR5JFHHWLHq8B9HZm6/IYxcNy2cGW9yAVyQKe3uJXeV/95u9qMEwJhbOjvPIx
ZdbXqcCSorWqih7hdB86Nv2SIBfXMvWdIinwZfU/44RUptNyxQpP/Pw91Dd8YnMTNVwm2ax5oL1W
akIGKToS/yYid/Cgyb1xhGdfXEp30bqLusaLMYkbctpZ2WDn2w5I4mmOO78jndPUnaMT5gyFaonz
+K84xD+1/tZbTQ0adc9LE7XgAkpiiNjf2LW9tw==</SignatureValue>
  <KeyInfo>
    <KeyName>yyyyyyyyyyyyyyy</KeyName>
  </KeyInfo>
</Signature>
```

Where:

- DigestValue is the non-encrypted hash of the message.
- SignatureValue is the encrypted hash of the message.
- KeyName is the ID of the signer component.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# 7 Managing configuration data of the system

## 7.1 Rationale

Previous clauses described how a sender endpoint calculates the message route, how it builds a signed message with an encrypted content, how a broker authorises the endpoint connections and the messages, and how a recipient endpoint verifies the message signature, decrypts the message content and sends acknowledgements. At some moment, each component along the delivery needs information issued and managed by other components such as recipient endpoint message-paths, broker-restrictions, and sender / recipient certificates.

Such information is referred as *system configuration data*, which is stored in a *directory* in which each component has an entry. The principles of the specified solution for entering and publishing configuration data are presented in 4.4.3: *system configuration data is distributed over all components*, each component shall store the all of it and the component-directories be hubs in the distribution process as follows:

- Each component of the system implements a local directory that stores configuration data of the entire system within safe storage.

NOTE See safe storage definition in 5.9.

- A component-directory administrator grants access to the system to endpoints and brokers, through a registration process: they enter this way in his subsystem. Registration includes issuing the component's certificates, the component-directory being the certification authority (CA) for a subsystem.
- A component-directory administrator can revoke any of its registered components.
- Each endpoint and each broker pushes its own configuration data to its subsystem component-directory, also referred as its home component-directory.
- Any two component-directories share and cross synchronise configuration data of their subsystems, so that each component-directory eventually stores configuration data of the entire system.
- A component-directory share and synchronise configuration data of the entire system to any requesting endpoint or broker. Therefore, any endpoint can potentially send messages to any endpoint either directly or through any broker.

Clause 7 describes the services exposed by a component-directory so that the components can register, push configuration data, and share subsystems configuration data.

## 7.2 Directory content and information ownership

A subsystem directory contains one entry for each registered endpoint and broker, and one entry for the component-directory itself. Table 18 lists the attributes of an entry, and for each the type (or format) and the owner responsible to provide the attribute value among "C" (the component) or "D" (the home component-directory).

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 18 – Component-directory – content of an entry**

| Attribute | Type | Description | Owner | Mandatory |
|---|---|---|---|---|
| code | string | The ID of the component that the entry describes (entry key) – see 5.1. | D | True |
| type | enumeration | The type of the component that the entry describes among: BROKER, COMPONENT_DIRECTORY, ENDPOINT. | D | True |
| organization | string | The name of the organisation that owns the component. | C | True |
| person | string | The name(s) of the administrator(s) of the component, i.e. the person(s) to contact in case of operational issue. | C | True |
| email | string | The mail address(es) of the contact person(s). | C | True |
| phone | string | The phone number(s) of the contact person(s). | C | True |
| urls | string[][a] | An ordered list of URLs, generally two: primary or secondary (RFC 4266).<br>• *Broker* → AMQPS://...<br>• *Component-directory* → HTTPS://...<br>• *Endpoint* → AMQPS://... (or empty if the endpoint does not accept incoming direct transfer) | C | False |
| certificates | Certificate[] (see Table 19) | The collection of valid certificates owned by the component, whatever type and possibly more than one for some types.<br>• *Broker* → AUTHENTICATION<br>• *Component-directory* → AUTHENTICATION, INTEGRATED_CA<br>• *Endpoint* → AUTHENTICATION, SIGNING, ENCRYPTION | D | True |
| creation Timestamp | dateTime | The creation time of the entry. | D | True |
| modification Timestamp | dateTime | The last modification time of the entry. | D | True |
| component Directory | string | The code of the home component-directory of the component. | D | True |
| mades Implementation | MadesImplementation (see Table 20) | The name of the MADES compliant software product used by the component and the MADES version with which it complies. | C | True |
| paths | MessagePath[] (see Table 21) | The message-paths to send messages to the endpoint – see 5.3. | C | Only for endpoint |
| restriction | BrokerRestriction (see Table 22) | The broker-restriction – see 5.4. | C | Only for broker |

[a] "xxx[]" means a collection of attributes of type xxx. Here it is a collection of strings.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 19 – Certificate (type)**

| Attribute | Type | Description | Required |
|---|---|---|---|
| certificateID | string | The ID of the certificate – see 8.2. | True |
| type | string | The type of the certificate along: ENCRYPTION, SIGNING, AUTHENTICATION, INTEGRATED_CA. | True |
| certificate | base64Binary | The binary data of the certificate in DER (Distinguished Encoding Rules) format (see ITU-T Recommendation X.690). | True |

**Table 20 – MadesImplementation (type)**

| Attribute | Type | Description | Required |
|---|---|---|---|
| name | string | The name of the software or application implementing the component (information). | False |
| version | string | The version of the software or application implementing the component (information). | False |
| compatibleVersions | string | A comma-separated list of the MADES versions with which the component is currently compatible (information). | False |
| madesVersion | string | The version of MADES specification with which the component is currently compatible. – see Clause 9. | True |

**Table 21 – MessagePath (type)**

| 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><br><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 |

**Table 22 – BrokerRestriction (type)**

| Attribute | Type | Description | Required |
|---|---|---|---|
| messageTypes | string[] | The collection of message-types allowed passing through the broker. A message-type in the collection can end with the wildcard character (*) to match a set of message-types. An empty collection means that the broker allows for any message-type. | False |
| components | string[] | The collection of the codes of the endpoints allowed connecting to the broker (as sender/producer or recipient/consumer). An empty collection means the broker allows for any endpoint. | False |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

## 7.3 On the consistency of configuration data

### 7.3.1 Component consistency

A component (endpoint or broker) shall check the format and the *internal* consistency of its own configuration data, warn its administrator when inconsistent, and not push such inconsistencies to its home component-directory. E.g.:

- (*endpoint, broker*) Format error, e.g. non well-formed URLs, incorrect protocols (HTTPS or AMQPS);
- (*endpoint*) Consistent usage period (from-to) has "To" > "From";
- (*endpoint*) Two message-paths with the same message-type shall not have overlapping usage periods.

NOTE Note that "AB" and "A*" are different.

### 7.3.2 System consistency

A component should warn its administrator when some of its own configuration data is inconsistent or become inconsistent against configuration data of the entire system.

In order to prevent that independently designed components do not accept same data, a component shall not reject configuration data with the following inconsistencies, when received from another component through the synchronisation process:

- Some component codes used in messages-paths (*endpoint*) or restriction (*broker*) do not exist (*referential integrity*).
- Some message-paths are useless and the endpoint will reject messages having the corresponding message-type, e.g. the broker will reject the endpoint or the message-type (see 5.5).

A "A" directory-component shall reject configuration data synchronised from a "B" component-directory if a component of the "B" subsystem bears the code of a component already existing in the "A" subsystem or in any other previously correctly synchronised subsystems. "A" directory-component can reject either the whole "B" subsystem or only the duplicate components.

### 7.3.3 Distributed update implementation

A change in configuration data by the component administrator shall be updated in the component itself and in the home component-directory. Such a distributed update can produce an inconsistent state, which can be either:

- *Inconsistent state #1*: the component first stores the change locally and then pushes it to the component-directory. The push fails (e.g. due to a broken connection) and the change is only stored in the component.
- *Inconsistent state #2*: the component first pushes the change to the component-directory and then stores it locally. The component fails to store locally (e.g. the component stops unexpectedly) and the change is only stored in the component-directory.

A component shall implement the distributed update to avoid the inconsistent state #2.

NOTE The inconsistent state #1 is considered less risky since the component can automatically schedule a remedial action (new push), the component can notify the administrator about the push error, and the change has not been distributed over the system yet.

### 7.3.4 Eventual consistency

Since the distribution process of configuration data is asynchronous, a change can take some time to propagate all over the system.

This can cause temporary (but accepted) situations in which a sender endpoint accepts a message while the intermediate broker or the recipient endpoint rejects it.

The administrators should be aware that some messages could not be delivered. They should consider how to sequence the changes across components in case such changes could impact business critical messages. They also should consider using the message-paths usage periods (see 5.3).

## 7.4 Connection to a component-directory

For exchanging data with a component-directory, a client component (endpoint, broker, or component-directory) shall establish a secured communication channel using the HTTPS (HTTP over TLS) protocol, verify that the peer is a valid and trusted system component as stated in 8.4. Then the client issues HTTP requests through the channel and the server validates the request and responds.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 7.5 REST API implementation and available resources

After it successful establishes a connection to a server component-directory (see 7.4), a client component uses URLs to identify (or equivalently to locate) the resources among *registrations*, *endpoints*, *brokers*, *components*. It uses the following HTTP methods to search and operate on resources as listed in Table 23:

**Table 23 – HTTP operations**

| HTTP operation | Use case |
|---|---|
| POST | Create a resource |
| GET | Search for resources or get the details of a single resource |
| PUT | Update a resource |
| DELETE | Delete a resource |

An HTTP method can contain a body in either the request or the response, which is an attached XML formatted string. When it successfully creates a resource on a POST request, the server returns the location (i.e. the URL) of the new resource in the HTTP header. The component-directory API returns the standard return codes, listed in Table 24:

**Table 24 – HTTP return codes**

| HTTP return code | Description |
|---|---|
| 200 OK | Successful response to GET, PUT or DELETE. |
| 201 Created | Successful response to a POST, which is always combined with a URL in the HTTP header locating the just created resource. |
| 204 No Content | Successful response without returned content. |
| 400 Bad Request | The request is malformed (e.g. cannot parse the request body). |
| 401 Unauthorized | No or incorrect authentication. |
| 403 Forbidden | Authenticated user doesn't have the access right to the requested resource |
| 404 Not Found | Requested resource does not exist. |
| 409 Conflict | The request could not be completed due to a conflict with the current state of the target resource. Used for POST requests to create an already existing resource (non-unique key). |
| 415 Unsupported Media Type | Incorrect request content. |
| 422 Unprocessable Entity | Validation error. |
| 500 Internal Server Error | Unexpected error. |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

In case of an error (400 and over), the response also contains details about the error in the response body, which complies with the following schema:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mades.entsoe.eu/componentDirectory"
   xmlns="http://mades.entsoe.eu/componentDirectory">

 <xsd:element name="error">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="code" type="xsd:string"/>
    <xsd:element name="id" type="xsd:string"/>
    <xsd:element name="message" type="xsd:string"/>
    <xsd:element name="details" type="xsd:string"/>
    </xsd:element>
   </xsd:sequence>
```

```
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Table 25 describes the necessary operations for implementing the requirements stated in 7.1. The expected XML element bodies of the request and the response are indicated (*in italics*) and are described in the referred XML schema. Subclause 7.2 describes the values and formats of the attributes since they all are directory attributes. Additional information on the registration (resp. synchronisation) process and on the expected sequences of requests is available in 7.6 (respectively 7.7).

**Table 25 – Component-directory API**

| URL | Operation | Description |
|---|---|---|
| /api/v1/registrations | POST | Create a new registration request. The URL of the created registration request is returned in the HTTP header.<br>• Access: broker and endpoint<br>• Request: *registrationRequest* (see 7.8.2 – request)<br>• Response: *registrationRequest* (see 7.8.2 – response) |
| /api/v1/registrations/{ID} | GET | Get details about a PENDING registration request.<br>• Access: the component associated with registration {ID}.<br>• Request: empty<br>• Response: *registrationRequest* (see 7.8.2 – response) |
| /api/v1/endpoints/{code} | PUT | Push configuration data of the endpoint which code is {code}:<br>• Access: the endpoint which code is {code}<br>• Request: *endpoint* (see 7.8.3)<br>• Response: *endpoint* (see 7.8.3) |
| /api/v1/brokers/{code} | PUT | Push configuration data of the broker which code is {code}:<br>• Access: the broker which code is {code}<br>• Request: *broker* (see 7.8.3)<br>• Response: *broker* (see 7.8.3) |
| /api/v1/components | GET | Get configuration data of subsystems (synchronisation)<br>• Access: any component<br>• Request: *componentsQuery* (see 7.8.3)<br>• Response: *components* (see 7.8.3) |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

## 7.6 Registration process

When setting up a component, an endpoint or a broker administrator shall be able to:

- enter a primary and possibly a secondary https URL to access the component-directory services with which he wants to register;
- install an authentication certificate (private and public keys) given by the administrator of the component-directory;

  NOTE Only the registration process will use this certificate.

- install an encryption and/or a signing certificate (private and public keys) issued by EXTERNAL CAs (see 8.3.3).

The registration process shall work as follows:

- After the component administrator enters all of the registration information, the component connects to the component-directory using the "installation" authentication certificate and POSTs a registration request. The request includes generated public keys for the component-directory to issue the corresponding certificates. The request can include the available valid signing / encryption certificates issued by EXTERNAL CAs, if any, instead of generated public keys.

- The response provides a registration ID and a registration status. When PENDING the component then cyclically requests (GET) for the registration details until APPROVED or REJECTED.
- When APPROVED, the response includes certificates to install in the component and complying with format described in 8.2. The component-directory has created an entry for the component in the directory, to publish towards all components.
- After the component administrator configures message-paths (endpoint) or restriction (broker), the component is ready to send and receive messages.

To ensure business continuity, a component should request automatically for a registration extension by the home component-directory some time before a certificate expires. The extension process works as the registration with a noteworthy difference: the component authenticates using a valid authentication certificate.

The generated keypairs (i.e. private and public keys – see key size in 8.2) shall be either the outcome of an integrated keypair generator at the state of the (cryptographic) art, or imported in the endpoint.

NOTE The rules for a component-directory administrator to approve a new component in the system, the validity duration of the issued certificates, the recommended anticipation period for a registration extension are system governance rules. The approval of the registration extension of an existing component could be performed automatically, especially if the component-directory issues all the certificates of component.

## 7.7 Synchronisation process

### 7.7.1 Validity period of replicated data: time-to-live

The TTL (time-to-live) is the maximum duration of the validity of configuration data of a subsystem when replicated outside of the subsystem component-directory.

NOTE All the components with the same home component-directory belong to the same subsystem.

The motivation for a validity limit is that replicated data cannot be valid too long since original data might be updated. The TTL is defined at the system governance level and shall be used as follows:

- The component-directory administrator configures a single TTL value that applies to all of replicated configuration data of the subsystem.
- When responding to a synchronisation request, the component-directory grants transmitted configuration data with a validity limit: current time + TTL.
- When a component-directory propagates configuration data from another subsystem, i.e. replicating a replicate, it cannot grant a validity longer than the one he was granted.
- A component uses received configuration data only when it is valid. E.g. when processing a message, if configuration data misses, the message delivery fails. Therefore, a component has to refresh configuration data (i.e. resynchronise) before it expires.

### 7.7.2 Limitation of the synchronisation flow

To ease implementation, entire configuration data of a subsystem is distributed again when changed. In order to propagate quickly a change, the synchronisation cycle has to be short. Because changes are presumably not frequent, the specified solution is to postpone the validity of a replicate without sending data again in case there is no change.

A component-directory shall uniquely identify configuration data of it subsystem as a string. Unique identification means that distinct contents have distinct identifiers.

NOTE There are several possible implementations. E.g. sequence number, last modification time, UUID, fingerprints.

The identifier is named after *contentID*.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

The synchronisation process shall work the following way:

- A synchronisation request from an endpoint or a broker identifies configuration data that the component already has, i.e. a list of couples *{component-directory code, contentID}*, each couple being associated to a subsystem. The list is empty on the first synchronisation of the component.
- The response contains a list of triplets *{component-directory code, contentID, time-to-live}*, in addition to updated configuration data (i.e. directory entries: endpoint, broker, and component-directory).
- The client updates its local replicate in a "delete and replace" mode for each subsystem. Some subsystems can appear or disappear.

The synchronisation process between two components-directories uses the same request with the following differences:

- The request only contains a single couple that identifies the server component-directory and the content that the client already has.
- The request indicates that the client only looks for data configuration of the server subsystem (*onlyHomeComponents* is set to true).

NOTE A message between two endpoints registered with two different component-directories can only be delivered after both synchronised at least once.

### 7.7.3 Configuration of the synchronisation process

The administrator of a component (endpoint, broker or component-directory) shall be able to configure:

- A synchronisation cycle: It should be short enough (e.g. minutes) in order to be quickly informed in case of a component is compromised and revoked.
- An ordered list of component-directories (codes) to synchronise with the first available.

NOTE The default is that a component synchronises with it home component-directory. URLs to access a component-directory are included in configuration data.

The administrator of a component-directory shall be able to configure:

- The TTL value of the subsystem. It should be long enough (e.g. hours) to make the directory-components non-critical components with respect to the messaging service availability.
- A list of component-directories (codes and access URLs) with which to synchronise.

## 7.8 XML schemas of the APIs requests and responses

### 7.8.1 Shared types

The shared types are referred in schemas as: *<xsd:include schemaLocation="types.xsd"/>*

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mades.entsoe.eu/componentDirectory"
   xmlns="http://mades.entsoe.eu/componentDirectory">

 <xsd:complexType name="ComponentContactInformation" abstract="true">
  <xsd:sequence>
   <xsd:element name="organization" type="xsd:string"/>
   <xsd:element name="person" type="xsd:string"/>
   <xsd:element name="email" type="xsd:string"/>
   <xsd:element name="phone" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:simpleType name="ComponentType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="BROKER"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
   <xsd:enumeration value="COMPONENT_DIRECTORY"/>
   <xsd:enumeration value="ENDPOINT"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="CertificateType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="AUTHENTICATION"/>
   <xsd:enumeration value="ENCRYPTION"/>
   <xsd:enumeration value="SIGNING"/>
   <xsd:enumeration value="INTEGRATED_CA"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:complexType name="Certificate">
  <xsd:sequence>
   <xsd:element name="certificateID" type="xsd:string"/>
   <xsd:element name="type" type="CertificateType"/>
   <xsd:element name="certificate" type="xsd:base64Binary"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MadesImplementation">
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="version" type="xsd:string"/>
  <xsd:attribute name="compatibleVersions" type="xsd:string"/>
  <xsd:attribute name="madesVersion" type="xsd:string" use="required"/>
 </xsd:complexType>

 <xsd:complexType name="ComponentInformationBase" abstract="true">
  <xsd:complexContent>
   <xsd:extension base="ComponentContactInformation">
    <xsd:sequence>
     <xsd:element name="code" type="xsd:string"/>
     <xsd:element name="type" type="ComponentType"/>
     <xsd:element minOccurs="0" name="urls">
      <xsd:complexType>
       <xsd:sequence>
        <xsd:element name="url" maxOccurs="unbounded" type="xsd:string"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
     <xsd:element name="certificates">
      <xsd:complexType>
       <xsd:sequence>
        <xsd:element name="certificate" type="Certificate" maxOccurs="unbounded"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
     <xsd:element minOccurs="0" name="creationTimestamp" type="xsd:dateTime"/>
     <xsd:element minOccurs="0" name="modificationTimestamp" type="xsd:dateTime"/>
     <xsd:element name="componentDirectory" minOccurs="0" type="xsd:string"/>
     <xsd:element name="madesImplementation" type="MadesImplementation"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="ComponentCodesList">
  <xsd:sequence>
   <xsd:element minOccurs="0" maxOccurs="unbounded" name="component"
type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MessagePath">
  <xsd:sequence>
   <xsd:element minOccurs="0" name="senderComponent" type="ComponentCodesList"/>
   <xsd:element name="messageType" type="xsd:string"/>
   <xsd:element name="path" type="xsd:string"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
   <xsd:element name="validFrom" type="xsd:dateTime"/>
   <xsd:element minOccurs="0" name="validUntil" type="xsd:dateTime"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MessagePathList">
  <xsd:sequence>
   <xsd:element name="path" minOccurs="0" maxOccurs="unbounded" type="MessagePath"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="EndpointInformation">
  <xsd:complexContent>
   <xsd:extension base="ComponentInformationBase">
    <xsd:sequence>
     <xsd:element minOccurs="0" name="paths" type="MessagePathList"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="BrokerRestriction">
  <xsd:sequence>
   <xsd:element minOccurs="0" name="components" type="ComponentCodesList"/>
   <xsd:element minOccurs="0" name="messageTypes">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="unbounded" name="messageType"
type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="BrokerInformation">
  <xsd:complexContent>
   <xsd:extension base="ComponentInformationBase">
    <xsd:sequence>
     <xsd:element name="restriction" type="BrokerRestriction"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="ComponentInformation">
  <xsd:choice>
   <xsd:element name="EndpointInformation" type="EndpointInformation"/>
   <xsd:element name="BrokerInformation" type="BrokerInformation"/>
  </xsd:choice>
 </xsd:complexType>

 <xsd:complexType name="ComponentDirectoryInformation">
  <xsd:complexContent>
   <xsd:extension base="ComponentInformationBase"/>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:schema>
```

### 7.8.2 registrations resource

XML schema for the HTTP request:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mades.entsoe.eu/componentDirectory"
   xmlns="http://mades.entsoe.eu/componentDirectory">

 <xsd:include schemaLocation="types.xsd"/> (see 7.8.1)
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
<xsd:complexType name="PublicKey">
 <xsd:sequence>
  <xsd:element name="type" type="CertificateType"/>
  <xsd:element name="publicKey" type="xsd:base64Binary"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:element name="registrationRequest">
 <xsd:complexType>
  <xsd:complexContent>
   <xsd:extension base="ComponentContactInformation">
    <xsd:sequence>
     <xsd:element name="code" type="xsd:string"/>

     <xsd:element name="publicKeys">
      <xsd:complexType>
       <xsd:sequence>
        <xsd:element name="publicKey" type="PublicKey" maxOccurs="unbounded"/>
       </xsd:sequence>
      </xsd:complexType>

      <xsd:element minOccurs="0" name="certificates">
       <xsd:complexType>
        <xsd:sequence>
         <xsd:element name="certificate" type="Certificate" maxOccurs="unbounded"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>

     </xsd:element>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:element>
</xsd:schema>
```

XML schema for the HTTP response:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mades.entsoe.eu/componentDirectory"
   xmlns="http://mades.entsoe.eu/componentDirectory">

 <xsd:include schemaLocation="types.xsd"/> (see 7.8.1)

 <xsd:element name="registrationRequest">
  <xsd:complexType>
   <xsd:complexContent>
    <xsd:extension base="ComponentContactInformation">
     <xsd:sequence>
      <xsd:element name="id" type="xsd:string"/>
      <xsd:element name="code" type="xsd:string"/>
      <xsd:element name="status">
       <xsd:simpleType>
        <xsd:restriction base="xsd:string">
         <xsd:enumeration value="PENDING"/>
         <xsd:enumeration value="APPROVED"/>
         <xsd:enumeration value="REJECTED"/>
        </xsd:restriction>
       </xsd:simpleType>
      </xsd:element>
      <xsd:element name="reason" minOccurs="0" type="xsd:string"/>
      <xsd:element minOccurs="0" name="certificates">
       <xsd:complexType>
        <xsd:sequence>
         <xsd:element name="certificate" type="Certificate" maxOccurs="unbounded"/>
        </xsd:sequence>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
      </xsd:complexType>
     </xsd:element>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:element>
</xsd:schema>
```

### 7.8.3 endpoints, brokers and components resources

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mades.entsoe.eu/componentDirectory"
   xmlns="http://mades.entsoe.eu/componentDirectory">

 <xsd:include schemaLocation="types.xsd"/> (see 7.8.1)

 <xsd:element name="endpoint" type="EndpointInformation"/>

 <xsd:element name="broker" type="BrokerInformation"/>

 <xsd:element name="components">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element minOccurs="0" name="components">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element name="endpoint" type="EndpointInformation"/>
       <xsd:element name="broker" type="BrokerInformation"/>
       <xsd:element name="componentDirectory" type="ComponentDirectoryInformation"/>
      </xsd:choice>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="metadata">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="componentDirectoryMetadata" maxOccurs="unbounded">
        <xsd:complexType>
         <xsd:sequence>
          <xsd:element name="componentDirectory" type="xsd:string"/>
          <xsd:element name="ttl" type="xsd:long"/>
          <xsd:element name="contentID" type="xsd:string"/>
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <xsd:element name="componentsQuery">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="componentDirectory" minOccurs="0" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="componentDirectory" type="xsd:string"/>
       <xsd:element name="contentID" type="xsd:string"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
    <xsd:element minOccurs="0" name="onlyHomeComponents" type="xsd:boolean"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# 8 Managing the certificates

## 8.1 Definitions and principles

NOTE Subclause 8.1 recalls basic cryptographic principles and does not contain requirements.

The security of a MADES system is based on a public key infrastructure (PKI). Such infrastructure binds certificates to both the components and the parties that use the system. Indeed, the components crosscheck their identities before exchanging information, the sender parties want that the only intended recipients can read the documents, and each party want to authenticate the senders of the documents he received.

Certificates use asymmetric cryptography based on private and public keys. On the contrary of symmetric cryptography, encryption is done using one key and decryption using the other key (which is different, and hence the asymmetry). Whereas the public key is easily deduced from the private key, the reverse operation is a hard mathematical challenge.

Encryption:

- A document is encrypted with a randomly generated symmetric key attached to the document in a secret way, being encrypted itself with the recipient's public key.
- To decrypt the document, the recipient first decrypts the "encrypted symmetric key" using its private key, and then decrypts the document with the symmetric key.

Signing:

- A signature is an encrypted fingerprint of a list of resources, named after the *signature manifest*. Technically a fingerprint is a "hash" outcome of a strong one-way transformation (e.g. SHA-512). Subclause 6.5.3 defines the manifest of a MADES message.
- The algorithm that computes the fingerprint does not require any key, so anyone owning the manifest can compute the fingerprint. Building another meaningful manifest with the same fingerprint is a hard mathematical challenge. The signature is the fingerprint encrypted with the sender's private key.
- Thus, anyone owning the manifest, the signature and the sender's public key can verify that the sender also owned the manifest when generating the signature. Therefore, the sender cannot repudiate a manifest he signed.
- Signing refers to the full process, i.e. computing the fingerprint and encrypting it.
- Verifying a signature includes computing the fingerprint from the manifest, decrypting the signature, and checking that the two equals.

A certification authority (CA) is an entity that issues certificates. A certificate contains a public key and a name, is signed with the private key of the certificate of the issuer, and has an expiration date, which is sooner than the expiration date of the certificate of the issuer. When signing a certificate, the issuer CA certifies the ownership of the keys (private and public) by the party whose name is included in the certificate. Other parties can verify the certificate signature using the certificate of the issuer. There, if parties trust a CA, they can then rely upon the signatures generated using the certificates that the CA has issued.

The certificate of a CA can itself be issued and signed by another CA, the latter delegating to the former the right to issue certificates. The certification chain of a certificate is the delegation sequence of CAs, i.e. the list of the certificates of all CAs' from the issuer of the certificate to a self-signed certificate, referred as the root certificate.

A valid certificate is a non-expired certificate. An expired certificate is never used for authentication, encryption or for signing a document. However, it can still be used to decrypt old documents or verify their signatures, and thus to prove whatever necessary.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 8.2 Certificates: format and unique ID

All components (endpoints, brokers and component-directories) use certificates to be authenticated by the communication peers (transport-layer security), to sign and to encrypt the messages (message-level security) when required.

The format of the certificates shall comply with the ISO/IEC 9594-8:2017 (Rec. ITU-T X.509 (2016)) standard, and the certificates' keys have a length of 2048 bits.

The exact concatenation of the standardised attributes "issuer" and "serial number" of a certificate forms a unique ID, referred as the "certificate ID".

### 8.3 Used certificates and issuers certificates authorities

#### 8.3.1 Overview

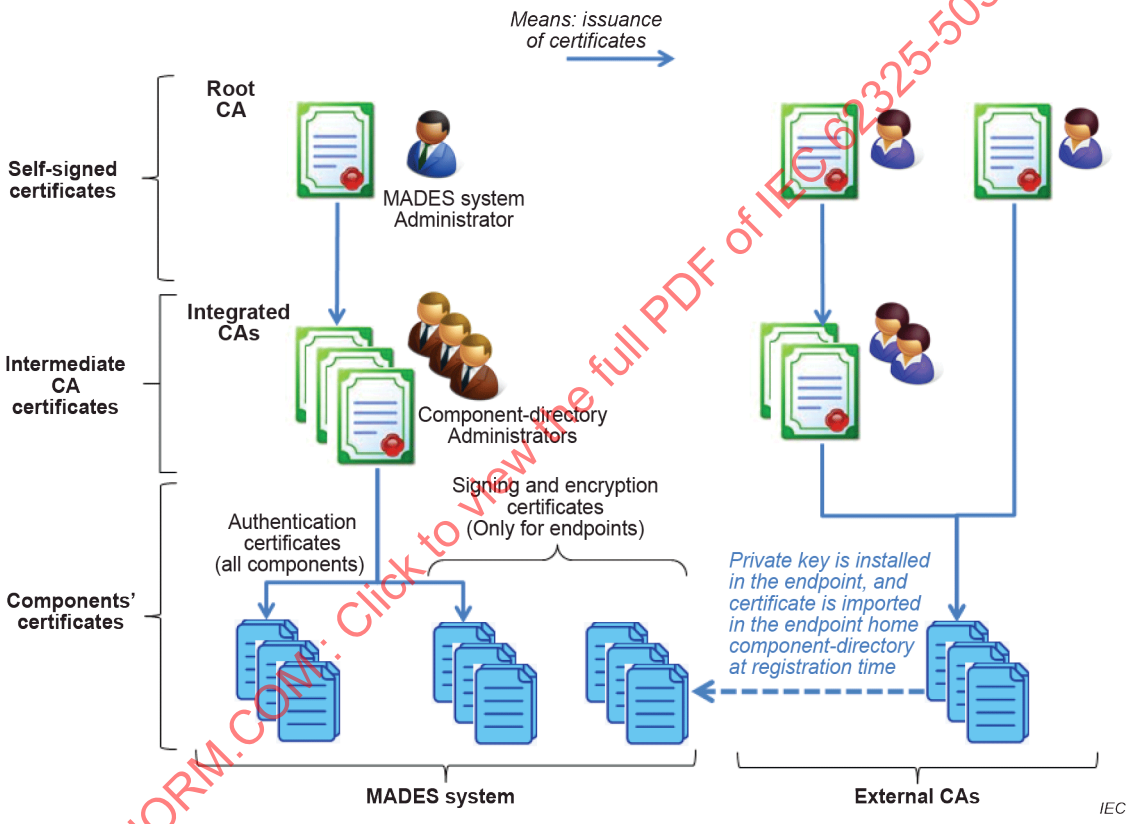



**Figure 30 – Certificates and certification authorities (CAs) of a MADES system**

Figure 30 describes the certification authorities and the certificates used in a MADES system.

#### 8.3.2 Transport-layer security (authorise data exchanges)

Each component (endpoint, broker) shall own an authentication certificate issued and published by its home component-directory.

The system organisation issues the authentication certificates as follows:

- The organisation owns a ROOT CA certificate.
- Each administrator of a component-directory owns an INTEGRATED CA certificate issued by the ROOT CA.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- The organisation delegates to each component-directory administrator the right to issue the authentication certificates of the subsystem, i.e. the authentication certificate of the component-directory itself and of the registered components (endpoints, brokers).

Each component shall authenticate using the authentication certificate whether it acts as a client or a server. Whatever the operation using the authentication certificate, it shall fail when the certificate has expired.

#### 8.3.3 Message-level security (protect message confidentiality and authenticate message issuer)

Each endpoint shall own an encryption certificate published in component-directories for the others endpoints to encrypt the messages they sent. The endpoint uses the corresponding private key to decrypt the messages it receives.

Each endpoint shall own a signing certificate published in component-directories for the others endpoints to verify the signature of the messages they receive. The endpoint uses the corresponding private key to sign the messages it sends.

The signing and encryption certificates of an endpoint can be issued by the home component-directory administrator using the INTEGRATED CA, or by an EXTERNAL CA trusted by the network parties and not necessary issued by one the main public trusting organisations (see 7.6).

The endpoint shall never encrypt or sign a message using an expired certificate. The endpoint shall not decrypt or verify a message signature using a certificate that was expired at the time the message was created (creation time is included in message metadata).

### 8.4 Trusting the certificates of others components

#### 8.4.1 Authentication

A component shall only communicate with a peer component if the authentication certificate presented by the peer component:

- belongs to the ROOT CA certification chain;
- is successfully verified.

During the TLS authentication phase, each peer shall convey to the other peer the following ordered certificate chain:

1) Its own authentication certificate.
2) The INTEGRATED CA certificate that is certified by the ROOT CA and that certifies the authentication certificate.

A component shall trust the ROOT CA and any authentication certificate issued by an INTEGRATED CA.

#### 8.4.2 Signing and encryption

An endpoint shall trust the signing and encryption certificates provided by a component-directory.

### 8.5 Renewing the (nearly) expired certificates

The certificates of a component should be renewed before they expire. All the certificates of a component shall be renewed at the same time.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

Therefore, the validity periods of the old and the new certificates overlap. The home component-directory shall still publish a valid certificate after it expired for a configurable time (e.g. more than 24 hours). The selection rules when multiple valid certificates shall be the following:

- A sender endpoint selects, among its valid signing certificates, the one which expiration time is the nearest.
- A sender endpoint selects, among the valid encryption certificates of a recipient endpoint, the one which expiration time is the nearest.
- A connecting client selects anyone of its valid authentication certificates.

Two component shall be able to communicate if their authentication certificates belong to two distinct certificate chains. Therefore, when the ROOT CA is renewed, components can communicate whether their certificate belongs to the old or to the new certification chain.

### 8.6 Revoking a component

A component-directory administrator should revoke a component of his subsystem only for security reasons, and when there is a reasonable doubt that the component or one of its certificates is compromised or could be misused.

The component-directory administrator shall be able to revoke any component of his subsystem: the component is then removed from configuration data and the change is propagated. The consequences are:

- As soon as the change is propagated to a sender endpoint, any message to a revoked endpoint is not accepted; also is any message that should pass through a revoked broker, for no message-path exists (see 5.5).
- As soon as the change is propagated to a broker, the broker rejects any connection, or attachment from a revoked endpoint, and rejects any message from or to a revoked endpoint (see Table 4).
- As soon as the change is propagated to a recipient endpoint, the signature of any message received from a revoked endpoint cannot be verified and the message delivery fails (see 7.7.1).

## 9 Managing the version of the MADES specification

### 9.1 MADES version of this document

The current document specifies the MADES version "2", namely:

- The message transfer uses AMQP 1.0 protocol: 1 (major), 0 (minor), 0 (revision).
- The AMQP format for a message is described in 6.5.2.
- The default signing algorithm is: hash generated with SHA-512, hash encrypted with RSA (see 6.5.4).
- The default encryption algorithm is: data encrypted with AES-256, key encrypted with RSA (see 6.5.3).
- The directory services API is described in 7.5: /api/v1/<resources> (*registrations, endpoints, brokers, components*).
- The webservices interface for the applications is described in 11.1: *SendMessage, ReceiveMessage, CheckMessageStatus, ConnectivityTest, ConfirmReceiveMessage* (targetNamespace="http://mades.entsoe.eu/2/")

### 9.2 Issue, version meaning, upgrading recommendations

The version of the MADES specification technically appears in two places:

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- In the *directory*, as mandatory information of configuration data of each component: the version of the specification with which the component complies (see Table 20).

  A component complies with a version if it behaves as specified for that version: namely the way it exposes and uses services, and the format it understands for the internal messages.

- In each *internal message*, both in the AMQP *application-properties* section (see Table 7) and in a metadata element of the *application-data* section (see Table 8): the version of the specification with which the internal message format complies.

Let us consider a system with all components complying with the version (N-1), named after (N-1)-components. When the N-specification comes out, the components should upgrade. A big-bang rollout, i.e. a simultaneous upgrade of all the components, is complex to coordinate when many parties and risky regarding business continuity.

Therefore, it is probably not acceptable on a running system in most cases. A smooth rollout requires that an upgraded endpoint can successfully exchange messages with a non-upgraded one. Technically, this means that a N-endpoint and a (N-1)-endpoint can exchange (N-1)-messages and a N-broker accepts messages with either version. This implies that:

- A sender endpoint shall know the complying version of a recipient endpoint, so the former can compose a message the latter can understand → This is why the version of a component is in the directory, as the highest version with which the component complies and its preferred version to receive messages.
- Either a recipient endpoint or a broker shall know with which version an internal message complies → This is why the version is included in the message.

Upgrading the system implies ensuring backward compatibility if the required AMQP protocol evolves: the protocol includes a version negotiation between the peers when opening a connection [AMQP 2.2].

An evolution of the directory services should be backward compatible. Otherwise, a new version of the API shall be specified (e.g. /api/v2/<resources>). A smooth upgrade scenario would then be to deploy first component-directories compatible with either version. Then, the migrated endpoints and brokers use the new API, while the non-migrated use the old API.

An evolution of the endpoint / application interface (webservice) should be backward compatible. Otherwise, new services shall be specified. A smooth upgrade scenario would then be that a party deploys an endpoint compatible with either version and then the applications migrate to the new services.

### 9.3 Changing the signature or the encryption algorithms

The internal message format (see 6.5) includes a flexible metadata structure for hosting information that the sender and the recipient should exchange to compute correctly the cryptographic algorithms.

To rollout new signature or encryption algorithms, all endpoints of the system should first upgrade independently to be able to send and receive messages signed and encrypted by the new algorithm, but not sending such messages. The key point is that any endpoint can then receive messages generated by either algorithm (old or new).

Sending messages generated by the new algorithm can then be switched on for one endpoint, then progressively for all endpoints.

## 10 Administrating and operating the components

This clause contains some requirements to ease the integration of the components in information systems.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- A component shall log the encountered errors, the configuration changes and the transferred internal messages (e.g. date, from, to, message-type) within a safe storage and for a configurable time window.
- A client component shall be configurable to open connections to a server component through a network proxy server. Note that AMQPS connections work through SOCKS proxies and not through HTTP proxies.

NOTE SOCKS (Socket Secure) is a protocol for exchanging packets between a client and a server through a proxy.

- A party could implement a component exposing HTTPS or AMQPS services in a high available mode. Availability techniques such as redundancy and switchover might not be seamless to a client component, and the server administrator will then provide several access URLs (e.g. primary, secondary). In such situations, a client component shall select dynamically the one URL that gains effective access to the service.
- A component shall be hosted in a server that is synchronised with a reliable time source. A standard protocol such as the Network Time Protocol (NTP) should be used.
- URLs should be configured as a fully qualified domain name (FQDN) rather than an IP address.
- A component should support automatic reload of the locally configured parameters, i.e. without being manually restarted.
- A component should warn the administrator when a certificate soon expires (configurable anticipation).
- A component should support HTTPS and authentication for the administrator connections to a graphical user interface if any.
- An endpoint should support HTTPS and authentication for the application web service interface.

# 11 Interfaces for the applications

## 11.1 Endpoint webservice interface for applications

### 11.1.1 Overview

The endpoint webservice interface provides the applications an access to the system. There are five available services:

- *SendMessage* – to upload a message into the endpoint for sending it to another endpoint.
- *ReceiveMessage* – to download an available message from the endpoint.
- *CheckMessageStatus* – to check the current message-status of a previously sent message.
- *ConnectivityTest* – to check if a path (endpoint + message-type) is reachable.
- *ConfirmReceiveMessage* – to inform the endpoint of the correct reception of a previously downloaded message.

In case a service encounters an unrecoverable error, it returns information on the error. When not explicitly described for a service, the set of the returned elements is listed in Table 26 and the *errorCode* values are those listed in Table 27.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 26 – Endpoint interface – Generic error**

| Element name | Description | Element type |
|---|---|---|
| errorCode | A code representing the type of error. | string |
| errorID | Unique identification of the error. | string |
| errorMessage | An English readable text describing the error. | string |
| errorDetails | (optional) Additional English readable details about the error context. | string |

**Table 27 – Endpoint interface – Value for errorCode**

| String value for errorCode | Description |
|---|---|
| INVALID_PARAMETERS | The provided parameters (i.e. the request elements) are incomplete, are not in the expected format or do not have the expected syntax. |
| AUTHENTICATION_ERROR | The peer component cannot be authenticated |
| VALIDATION_ERROR | The message is not valid (content size exceeded, unknown sender/recipient, signature is not valid etc.). |
| INTERNAL_ERROR | Internal application error. The content of request did not cause the error, but the application encounters a non-recoverable error (*Null Pointer Exception* in code, full database etc.) |
| CONCURRENT_ERROR | The server component is already processing a concurrent request from the same client. |

### 11.1.2 SendMessage service

An application uses the *SendMessage* service to upload a message into the endpoint in order to send the message to another endpoint.

Table 28 describes the request elements. Table 30 describes the response elements. Table 31 describes the additional error elements for the service (elements in addition to those described in 11.1.1).

**Table 28 – SendMessage – Request elements**

| Element name | Description | Element type | Required |
|---|---|---|---|
| message | Sending context, content and requested destination of the message. | SentMessage (see Table 29) | True |
| conversationID | Unique identifier associated with the request. | string | False |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

Concerning *conversationID*: There are situations where the sender application does not receive back or fails to store durably the returned message ID, for example in case of failure of the endpoint, of the network or of the application itself. Therefore, the application does not know whether the message was or was not correctly transferred to the endpoint. There are two subsequent issues:

- If the message was correctly transferred and stored in the endpoint, the application does not know the message ID needed for further processing, such as checking the message-status.
- Considering that losing a message is a non-acceptable risk, the application will send the message again when the connection with the endpoint recovers. The drawback is that the message can then be sent twice with two different IDs.

Therefore, resending the message using the same *conversationID* value solves both issues. Indeed, upon request to send a message with a *conversationID* value already used for an existing stored and sent message, the sender endpoint shall not send the message again but

just return to the caller application the ID of the already existing message. It is recommended that *conversationID:= senderApplication + baMessageID.*

**Table 29 – SentMessage (type)**

| Element name | Element type | Description | Required |
|---|---|---|---|
| receiverCode | string | The component ID of the recipient endpoint (see 5.1) – Pattern[a]: [A-Za-z0-9-@]+ | True |
| messageType | string | The message-type of the message (see 5.2) – Pattern: [A-Za-z0-9]+ | True |
| baMessageID | string | An identifier of the document provided by the sender (business) application. Information is transported “as is” to the recipient application – Pattern: [A-Za-z0-9]* | False |
| senderApplication | string | The identifier of the sender application. Information is transported “as is” to the recipient application – Pattern: [A-Za-z0-9]* | False |
| content | base64Binary | The content of the message, i.e. the document.<br>There is no constraint about the structure of the document processed as a stream of bytes. E.g., it can be a human-readable XML document, or multiples files compressed in ZIP format. | True |
| [a] Pattern is the « regular expression » that the element value matches. | | | |

**Table 30 – SendMessage – Response elements**

| Element name | Description | Element type |
|---|---|---|
| messageID | The UUID (Universal Unique ID) of the message composed and stored by the endpoint – see 5.1. | string |

**Table 31 – SendMessage – Additional error elements**

| Element name | Description | Element type |
|---|---|---|
| receiverCode | The component ID of the requested recipient endpoint for the message. | string |

### 11.1.3 ReceiveMessage service

An application uses the *ReceiveMessage* service to download a message from the endpoint. Table 32 describes the request elements. Table 33 describes the response elements. Table 35 describes the additional error elements for the service.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 32 – ReceiveMessage – Request elements**

| Element name | Description | Element type | Required |
|---|---|---|---|
| messageType | The message-type of the requested message – see 5.2.<br>Pattern: [A-Za-z0-9]+ | string | True |
| downloadMessage | The service returns, if any, the first received and pending message having the requested message-type. "First" means according to the priority defined in 5.10.<br>The content (or document) of the message is or is not returned according to the value of this element:<br>true:= returned;<br>false:= not returned. | boolean | True |

**Table 33 – ReceiveMessage – Response elements**

| Element name | Description | Element type |
|---|---|---|
| receivedMessage | Sending context and possibly content of a message. | ReceivedMessage (see Table 34) |
| remainingMessagesCount | The number of remaining messages received by the endpoint, matching the requested message-type and waiting for delivery.<br>In case the service returns the content of a message, the message is not included in the count of the remaining messages. | integer |

**Table 34 – ReceivedMessage (type)**

| Element name | Element type | Description |
|---|---|---|
| messageID | string | The UUID (Universal Unique ID) of the message – see 5.1. |
| receiverCode | string | The component ID of the recipient endpoint of the message – see 5.1. |
| senderCode | string | The component ID of the sender endpoint of the message – see 5.1. |
| messageType | string | The message-type of the message currently transferred to the application. |
| content | base64Binary | The content of the message as provided by the sender application in the *SendMessage* service. |
| senderApplication | string | The identifier of the sender application, if any and as provided by the sender application in the *SendMessage* service. |
| baMessageID | string | The identifier of the message assigned by the sending (business) application, if any, and provided in the *SendMessage* service. |

**Table 35 – ReceiveMessage – Additional error elements**

| Element name | Description | Element type |
|---|---|---|
| messageType | The requested message-type. | string |

Until the recipient application confirms to the recipient endpoint that the message is correctly transferred using the *ConfirmReceiveMessage* service (see 11.1.4), the endpoint shall consider that the message has not been transferred, is still pending and shall transfer it again next time an application requests for the message-type. This ensures that no message is lost. Consequently, the applications should be aware that, in some failure or recovery situations, they can possibly receive an already delivered message (i.e. having a known message ID).

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 11.1.4 ConfirmReceiveMessage service

An application uses the *ConfirmReceiveMessage* service to confirm the download transfer of a message from a recipient endpoint.

An application should not reject a message for a functional reason; the expected confirmation is only technical. The business functional acceptance (i.e. the compliance of the message with business rules) is another issue (see note in 5.6.1). If a transferred message is not confirmed back, for example in case of failure, the endpoint will provide it again at the next *ReceiveMessage* call.

In case an application encounters an unrecoverable error when processing a transferred message, and when the error comes from the message itself (e.g. inconsistent elements) and not from the application (e.g. file system full), the application should confirm the transfer, log the error and possibly alert, otherwise the message will indefinitely be retransferred by the endpoint until it is confirmed.

Table 36 describes the request elements. Table 37 describes the response elements. Table 38 describes the additional error elements for the service.

**Table 36 – ConfirmReceiveMessage – Request elements**

| Element name | Description | Element type | Required |
|---|---|---|---|
| messageID | The UUID (Universal Unique ID) of the message which transfer to the application is confirmed – see 5.1. | string | True |

**Table 37 – ConfirmReceiveMessage – Response elements**

| Element name | Description | Element type |
|---|---|---|
| messageID | The UUID (Universal Unique ID) of the message which transfer has just been confirmed. | string |

**Table 38 – ConfirmReceiveMessage – Additional error elements**

| Element name | Description | Element type |
|---|---|---|
| messageID | The requested message ID. | string |

### 11.1.5 CheckMessageStatus service

An application uses the *CheckMessageStatus* service to check the current message-status of a message. Table 39 describes the request elements. Table 40 describes the response elements. Table 44 describes the additional error elements for the service.

**Table 39 – CheckMessageStatus – Request elements**

| Element name | Description | Element type | Required |
|---|---|---|---|
| messageID | The UUID (Universal Unique ID) of the message, which status is requested – see 5.1. | string | True |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 40 – CheckMessageStatus – Response elements**

| Element name | Description | Element type |
|---|---|---|
| messageStatus | All of the information about the message delivery. | MessageStatus (see Table 41) |

**Table 41 – MessageStatus (type)**

| Element name | Element type | Description |
|---|---|---|
| messageID | string | The UUID (Universal Unique ID) of the message which message-status is reported in this data structure – see 5.1. |
| state | MessageState (see Table 43) | The message-status of the requested message. |
| receiverCode | string | The component ID of the recipient endpoint of the message. |
| senderCode | string | The component ID of the sender endpoint of the message. |
| messageType | string | The message-type of the message – see 5.2. |
| senderApplication | string | The identifier of sender application, if any and as provided in the *SendMessage* service. |
| baMessageID | string | The identifier of the message assigned by the sender (business) application, if any and as provided in the *SendMessage* service. |
| sendTimestamp | dateTime | The time when the message was created by the sender endpoint (The *generated* element of the *MessageMetadata* type – see Table 11). |
| receiveTimestamp | dateTime | The "reception time" of the message in the recipient endpoint. It is the time when the received acknowledgement was generated. |
| trace | MessageTraceItem [] (see Table 42) | The collection of the traces reporting the events about the message delivery. |

**Table 42 – MessageTraceItem (type)**

| Element name | Element type | Description | Required |
|---|---|---|---|
| timestamp | dateTime | The date and time of the reported event. | True |
| state | MessageTraceState (see Table 43) | The reported event. | True |
| component | string | The ID of the component (see 5.1) where the event happened. | True |
| Component description | string | The display name of the component where the event happened. | True |
| Details | string | The English readable details about the event. | False |

**Table 43 – MessageState or MessageTraceState (Type: string enumeration)**

| String value | Description |
|---|---|
| ACCEPTED | The sender endpoint accepted the message. |
| DELIVERED | The recipient endpoint accepted the message. |
| RECEIVED | A recipient application accepted the message. |
| FAILED | The processing of the message failed and the delivery stopped. |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 44 – CheckMessageStatus – Additional error elements**

| Element name | Description | Element type |
|---|---|---|
| messageID | The requested message ID. | string |

### 11.1.6 ConnectivityTest service

An application uses the *ConnectivityTest* service to check if another endpoint is reachable through the system. The service sends a tracing message which message-status can be requested later using the *CheckMessageStatus* service. The connectivity is successful, i.e. the tracing-message has reached the recipient endpoint, when the message-status is DELIVERED.

Table 45 describes the request elements. Table 46 describes the response elements. Table 47 describes the additional error elements for the service.

**Table 45 – ConnectivityTest – Request elements**

| Element name | Description | Element type | Required |
|---|---|---|---|
| receiverCode | The component ID of the recipient endpoint which connectivity is checked.<br>Pattern: [A-Za-z0-9-@]+ | string | True |
| messageType | The message-type of the message – see 5.2 | string | True |

**Table 46 – ConnectivityTest – Response elements**

| Element name | Description | Element type |
|---|---|---|
| messageID | The message ID of the tracing-message. | string |

**Table 47 – ConnectivityTest – Additional error elements**

| Element name | Description | Element type |
|---|---|---|
| receiverCode | The component ID of the recipient endpoint which connectivity check was requested. | String |

### 11.1.7 WSDL for the endpoint webservice interface

The services are described using the Web Services Description Language (WSDL) 1.1[5]. See http://www.w3.org/TR/wsdl and Figure 31.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

5 Figure 31 from http://en.wikipedia.org/wiki/Web_Services_Description_Language.

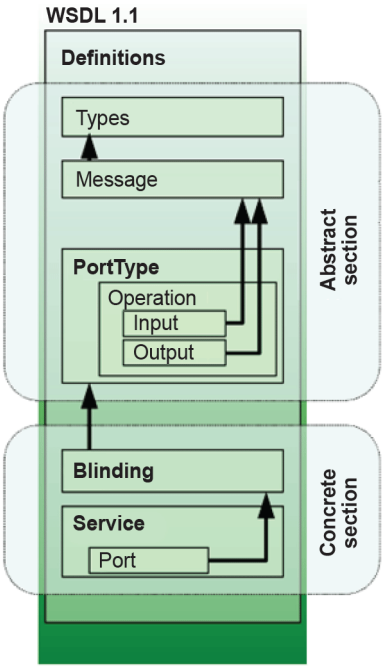



**Figure 31 – WSDL 1.1 definitions**

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="MadesEndpointV2" targetNamespace="http://mades.entsoe.eu/2/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:mades="http://mades.entsoe.eu/2/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">

  <wsdl:types>
    <xsd:schema targetNamespace="http://mades.entsoe.eu/2/">

      <xsd:element name="SendMessageRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="message" type="mades:SentMessage"/>
            <xsd:element minOccurs="0" name="conversationID" nillable="true"
type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="SendMessageResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="messageID" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="SendMessageError">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="errorCode" type="xsd:string"/>
            <xsd:element name="errorID" type="xsd:string"/>
            <xsd:element name="errorMessage" type="xsd:string"/>
            <xsd:element minOccurs="0" name="receiverCode" nillable="true"
type="xsd:string"/>
            <xsd:element minOccurs="0" name="errorDetails" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
      </xsd:element>

      <xsd:element name="ReceiveMessageRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="messageType" type="xsd:string"/>
            <xsd:element name="downloadMessage" type="xsd:boolean"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="ReceiveMessageResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element minOccurs="0" name="receivedMessage" nillable="true"
type="mades:ReceivedMessage"/>
            <xsd:element name="remainingMessagesCount" type="xsd:long"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="ReceiveMessageError">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="errorCode" type="xsd:string"/>
            <xsd:element name="errorID" type="xsd:string"/>
            <xsd:element name="errorMessage" type="xsd:string"/>
            <xsd:element minOccurs="0" name="messageType" nillable="true"
type="xsd:string"/>
            <xsd:element minOccurs="0" name="errorDetails" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="ConfirmReceiveMessageRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="messageID" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="ConfirmReceiveMessageResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="messageID" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="ConfirmReceiveMessageError">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="errorCode" type="xsd:string"/>
            <xsd:element name="errorID" type="xsd:string"/>
            <xsd:element name="errorMessage" type="xsd:string"/>
            <xsd:element minOccurs="0" name="messageID" nillable="true"
type="xsd:string"/>
            <xsd:element minOccurs="0" name="errorDetails" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:complexType name="SentMessage">
        <xsd:sequence>
          <xsd:element name="receiverCode" type="xsd:string"/>
          <xsd:element name="messageType" type="xsd:string"/>
          <xsd:element name="content" type="xsd:base64Binary"/>
          <xsd:element minOccurs="0" name="senderApplication" nillable="true"
type="xsd:string"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
          <xsd:element minOccurs="0" name="baMessageID" nillable="true"
type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="ReceivedMessage">
        <xsd:sequence>
          <xsd:element name="messageID" type="xsd:string"/>
          <xsd:element name="receiverCode" type="xsd:string"/>
          <xsd:element name="senderCode" type="xsd:string"/>
          <xsd:element name="messageType" type="xsd:string"/>
          <xsd:element name="content" type="xsd:base64Binary"/>
          <xsd:element minOccurs="0" name="senderApplication" nillable="true"
type="xsd:string"/>
          <xsd:element minOccurs="0" name="baMessageID" nillable="true"
type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="MessageStatus">
        <xsd:sequence>
          <xsd:element name="messageID" type="xsd:string"/>
          <xsd:element name="state" type="mades:MessageState"/>
          <xsd:element name="receiverCode" type="xsd:string"/>
          <xsd:element name="senderCode" type="xsd:string"/>
          <xsd:element name="messageType" type="xsd:string"/>
          <xsd:element minOccurs="0" name="senderApplication" nillable="true"
type="xsd:string"/>
          <xsd:element minOccurs="0" name="baMessageID" nillable="true"
type="xsd:string"/>
          <xsd:element name="sendTimestamp" type="xsd:dateTime"/>
          <xsd:element minOccurs="0" name="receiveTimestamp" nillable="true"
type="xsd:dateTime"/>
          <xsd:element name="trace" nillable="true" type="mades:MessageTrace"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="MessageTrace">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" name="trace"
type="mades:MessageTraceItem"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="MessageTraceItem">
        <xsd:sequence>
          <xsd:element name="timestamp" type="xsd:dateTime"/>
          <xsd:element name="state" type="mades:MessageTraceState"/>
          <xsd:element name="component" type="xsd:string"/>
          <xsd:element name="componentDescription" type="xsd:string"/>
          <xsd:element name="details" nillable="true" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:element name="ConnectivityTestRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="receiverCode" type="xsd:string"/>
            <xsd:element name="messageType" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="ConnectivityTestResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="messageID" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
      <xsd:element name="ConnectivityTestError">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="errorCode" type="xsd:string"/>
            <xsd:element name="errorID" type="xsd:string"/>
            <xsd:element name="errorMessage" type="xsd:string"/>
            <xsd:element minOccurs="0" name="receiverCode" nillable="true"
type="xsd:string"/>
            <xsd:element minOccurs="0" name="errorDetails" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="CheckMessageStatusRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="messageID" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="CheckMessageStatusResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="messageStatus" type="mades:MessageStatus"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="CheckMessageStatusError">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="errorCode" type="xsd:string"/>
            <xsd:element name="errorID" type="xsd:string"/>
            <xsd:element name="errorMessage" type="xsd:string"/>
            <xsd:element minOccurs="0" name="messageID" nillable="true"
type="xsd:string"/>
            <xsd:element minOccurs="0" name="errorDetails" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:simpleType name="MessageState">
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="ACCEPTED"/>
          <xsd:enumeration value="DELIVERED"/>
          <xsd:enumeration value="RECEIVED"/>
          <xsd:enumeration value="FAILED"/>
        </xsd:restriction>
      </xsd:simpleType>

      <xsd:simpleType name="MessageTraceState">
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="ACCEPTED"/>
          <xsd:enumeration value="DELIVERED"/>
          <xsd:enumeration value="RECEIVED"/>
          <xsd:enumeration value="FAILED"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:schema>
  </wsdl:types>

  <wsdl:message name="SendMessageRequest">
    <wsdl:part name="parameters" element="mades:SendMessageRequest"/>
  </wsdl:message>

  <wsdl:message name="SendMessageResponse">
    <wsdl:part name="parameters" element="mades:SendMessageResponse"/>
  </wsdl:message>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
  <wsdl:message name="ConnectivityTestFault">
    <wsdl:part name="fault" element="mades:ConnectivityTestError"/>
  </wsdl:message>

  <wsdl:message name="ReceiveMessageRequest">
    <wsdl:part name="parameters" element="mades:ReceiveMessageRequest"/>
  </wsdl:message>

  <wsdl:message name="ConfirmReceiveMessageRequest">
    <wsdl:part name="parameters" element="mades:ConfirmReceiveMessageRequest"/>
  </wsdl:message>

  <wsdl:message name="ConnectivityTestRequest">
    <wsdl:part name="parameters" element="mades:ConnectivityTestRequest"/>
  </wsdl:message>

  <wsdl:message name="CheckMessageStatusResponse">
    <wsdl:part name="parameters" element="mades:CheckMessageStatusResponse"/>
  </wsdl:message>

  <wsdl:message name="ConfirmReceiveMessageResponse">
    <wsdl:part name="parameters" element="mades:ConfirmReceiveMessageResponse"/>
  </wsdl:message>

  <wsdl:message name="ReceiveMessageFault">
    <wsdl:part name="fault" element="mades:ReceiveMessageError"/>
  </wsdl:message>

  <wsdl:message name="CheckMessageStatusFault">
    <wsdl:part name="fault" element="mades:CheckMessageStatusError"/>
  </wsdl:message>

  <wsdl:message name="CheckMessageStatusRequest">
    <wsdl:part name="parameters" element="mades:CheckMessageStatusRequest"/>
  </wsdl:message>

  <wsdl:message name="ConfirmReceiveMessageFault">
    <wsdl:part name="fault" element="mades:ConfirmReceiveMessageError"/>
  </wsdl:message>

  <wsdl:message name="SendMessageFault">
    <wsdl:part name="fault" element="mades:SendMessageError"/>
  </wsdl:message>

  <wsdl:message name="ReceiveMessageResponse">
    <wsdl:part name="parameters" element="mades:ReceiveMessageResponse"/>
  </wsdl:message>

  <wsdl:message name="ConnectivityTestResponse">
    <wsdl:part name="parameters" element="mades:ConnectivityTestResponse"/>
  </wsdl:message>

  <wsdl:portType name="MadesEndpointV2">
    <wsdl:operation name="SendMessage">
      <wsdl:input message="mades:SendMessageRequest"/>
      <wsdl:output message="mades:SendMessageResponse"/>
      <wsdl:fault name="SendMessageError" message="mades:SendMessageFault"/>
    </wsdl:operation>
    <wsdl:operation name="ReceiveMessage">
      <wsdl:input message="mades:ReceiveMessageRequest"/>
      <wsdl:output message="mades:ReceiveMessageResponse"/>
      <wsdl:fault name="ReceiveMessageError" message="mades:ReceiveMessageFault"/>
    </wsdl:operation>
    <wsdl:operation name="ConfirmReceiveMessage">
      <wsdl:input message="mades:ConfirmReceiveMessageRequest"/>
      <wsdl:output message="mades:ConfirmReceiveMessageResponse"/>
      <wsdl:fault name="ConfirmReceiveMessageError"
message="mades:ConfirmReceiveMessageFault"/>
    </wsdl:operation>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
    <wsdl:operation name="ConnectivityTest">
      <wsdl:input message="mades:ConnectivityTestRequest"/>
      <wsdl:output message="mades:ConnectivityTestResponse"/>
      <wsdl:fault name="ConnectivityTestError"
message="mades:ConnectivityTestFault"/>
    </wsdl:operation>
    <wsdl:operation name="CheckMessageStatus">
      <wsdl:input message="mades:CheckMessageStatusRequest"/>
      <wsdl:output message="mades:CheckMessageStatusResponse"/>
      <wsdl:fault name="CheckMessageStatusError"
message="mades:CheckMessageStatusFault"/>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="MadesEndpointSOAP12" type="mades:MadesEndpointV2">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="SendMessage">
      <soap12:operation soapAction="http://mades.entsoe.eu/2/SendMessage"/>
      <wsdl:input> <soap12:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap12:body use="literal"/> </wsdl:output>
      <wsdl:fault name="SendMessageError"> <soap12:fault name="SendMessageError"
use="literal"/> </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="ReceiveMessage">
      <soap12:operation soapAction="http://mades.entsoe.eu/2/ReceiveMessage"/>
      <wsdl:input> <soap12:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap12:body use="literal"/> </wsdl:output>
      <wsdl:fault name="ReceiveMessageError"> <soap12:fault
name="ReceiveMessageError" use="literal"/> </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="ConfirmReceiveMessage">
      <soap12:operation soapAction="http://mades.entsoe.eu/2/ConfirmReceiveMessage"/>
      <wsdl:input> <soap12:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap12:body use="literal"/> </wsdl:output>
      <wsdl:fault name="ConfirmReceiveMessageError"> <soap12:fault
name="ConfirmReceiveMessageError" use="literal"/> </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="ConnectivityTest">
      <soap12:operation soapAction="http://mades.entsoe.eu/2/ConnectivityTest"/>
      <wsdl:input> <soap12:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap12:body use="literal"/> </wsdl:output>
      <wsdl:fault name="ConnectivityTestError"> <soap12:fault
name="ConnectivityTestError" use="literal"/> </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="CheckMessageStatus">
      <soap12:operation soapAction="http://mades.entsoe.eu/2/CheckMessageStatus"/>
      <wsdl:input> <soap12:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap12:body use="literal"/> </wsdl:output>
      <wsdl:fault name="CheckMessageStatusError"> <soap12:fault
name="CheckMessageStatusError" use="literal"/> </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:binding name="MadesEndpointSOAP11" type="mades:MadesEndpointV2">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="SendMessage">
      <soap:operation soapAction="http://mades.entsoe.eu/2/SendMessage"/>
      <wsdl:input> <soap:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap:body use="literal"/> </wsdl:output>
      <wsdl:fault name="SendMessageError"> <soap:fault name="SendMessageError"
use="literal"/> </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="ReceiveMessage">
      <soap:operation soapAction="http://mades.entsoe.eu/2/ReceiveMessage"/>
      <wsdl:input> <soap:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap:body use="literal"/> </wsdl:output>
      <wsdl:fault name="ReceiveMessageError"> <soap:fault name="ReceiveMessageError"
use="literal"/> </wsdl:fault>
    </wsdl:operation>
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
    <wsdl:operation name="ConfirmReceiveMessage">
      <soap:operation soapAction="http://mades.entsoe.eu/2/ConfirmReceiveMessage"/>
      <wsdl:input> <soap:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap:body use="literal"/> </wsdl:output>
      <wsdl:fault name="ConfirmReceiveMessageError"> <soap:fault
name="ConfirmReceiveMessageError" use="literal"/> </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="ConnectivityTest">
      <soap:operation soapAction="http://mades.entsoe.eu/2/ConnectivityTest"/>
      <wsdl:input> <soap:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap:body use="literal"/> </wsdl:output>
      <wsdl:fault name="ConnectivityTestError"> <soap:fault
name="ConnectivityTestError" use="literal"/> </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="CheckMessageStatus">
      <soap:operation soapAction="http://mades.entsoe.eu/2/CheckMessageStatus"/>
      <wsdl:input> <soap:body use="literal"/> </wsdl:input>
      <wsdl:output> <soap:body use="literal"/> </wsdl:output>
      <wsdl:fault name="CheckMessageStatusError"> <soap:fault
name="CheckMessageStatusError" use="literal"/> </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="MadesEndpointV2Service">
    <wsdl:port name="MadesEndpointSOAP12" binding="mades:MadesEndpointSOAP12">
      <soap12:address location="http://mades.entsoe.eu/2/"/>
    </wsdl:port>
    <wsdl:port name="MadesEndpointSOAP11" binding="mades:MadesEndpointSOAP11">
      <soap:address location="http://mades.entsoe.eu/2/"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

## 11.2 File System Shared Folders (FSSF)

### 11.2.1 Overview

FSSF (File System Shared Folders) is an additional and recommended interface for an application to exchange documents as files with the endpoint. The endpoint accesses the documents in a local file system and the exchanges with applications relies on the FTP protocol server, possibly secured. The principles are the following:

- A sender application writes in a shared and unique OUT-folder a document that the endpoint has to send.
- A recipient application reads in an IN-folder a document that the endpoint received.
- Additional information necessary for the message delivery is part of the filename. Such information is the request / response necessary elements of the *SendMessage* and *ReceiveMessage* services.
- The organisation of the folders is local to each endpoint and configurable.

When implemented, a file interface with the endpoint shall comply with the FSSF interface as described in Subclause 11.2.

NOTE There are differences between interfacing the endpoint using FSSF and using the webservice interface. The *CheckMessageStatus* and *ConnectivityTest* services are not supported. The *ConfirmReceiveMessage* operation is implicit; i.e. a message moves to the RECEIVED status when the recipient endpoint has successfully written its content in a file of the IN-folder.

### 11.2.2 Folders and file naming convention

Filenames are standardised. A filename is a set of tokens joined by underscore characters ("_"). Each token shall only contains alphanumeric or hyphen characters, i.e. contain no accented characters, no white spaces and no special characters. Joining underscores shall be present even when an optional token is missing or empty.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

There are four types of folders used by the FSSF interface listed in Table 48 with the corresponding filename syntax described with tokens presented in Table 49.

**Table 48 – FSSF – Folders and filename format**

| Type | Folder / Writer | Description and Filename format |
|---|---|---|
| Files to be sent | OUT / applications | The folder contains the files written by applications and that the endpoint has to send.<br>The sender endpoint removes from the folder a file correctly processed (i.e. accepted file is deleted) or not (i.e. rejected files is moved to the OUT_ERROR folder).<br>Filenames:<br><SenderApp>_<Recipient>_<MessType>_<BAmessageID>.<Ext> |
| Failed files | OUT_ERROR / Sender endpoint | The folder contains the files that the sender endpoint did not process correctly.<br>It is up to the endpoint administrator to analyse and clean up the folder.<br>The filenames can match or not the “files to send” filename format. Note that not matching the filename format is a reason for the file be rejected. |
| Received files | IN / Recipient endpoint | The folder contains files which content is a message received by the endpoint. The filename is built from metadata of the received message.<br>The recipient application shall remove a file from the folder after it processed it.<br>Filenames:<br><SenderApp>_<Sender>_<MessType>_<BAmessageID>_<MessageID>.<Ext> |
| Log files | OUT_LOG / Sender endpoint | The folder contains one log file for each message accepted by the endpoint.<br>The file contains English readable text. Each line reports an event about the message delivery, and is the concatenation of the *MessageTraceItem* structure, as provided in the *CheckMessageStatus* service response.<br>A new line is appended to the file each time an event regarding the message is notified to the sender endpoint.<br>It is up to the endpoint administrator to clean up the OUT_LOG folder.<br>The filename is the exact name of the sent file with an added “.log” extension:<br><SenderApp>_<Recipient>_<MessType>_<BAmessageID>.<Ext>.log |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Table 49 – FSSF – Tokens used to generate the filenames**

| Token name | Mandatory or optional | Token description |
|---|---|---|
| <BAmessageID> | Optional | An identifier of the document provided by the sender (business) application. Information is transported “as is” to the recipient application. – Pattern: [A-Za-z0-9-]* |
| <MessType> | Mandatory | The message-type (see 5.2). – Pattern: [A-Za-z0-9-]* |
| <Ext> | Optional | The file extension – Pattern: [A-Za-z0-9-]* |
| <MessageID> | Mandatory | The UUID (Universal Unique ID) of the message composed by the sender endpoint (see 5.1) – Pattern: [A-Za-z0-9-]+ |
| <Sender> | Mandatory | The component code of the sender endpoint. – Pattern: [A-Za-z0-9-@]+ |
| <SenderApp> | Optional | The identifier of the sender application. – Pattern: [A-Za-z0-9-]* |
| <Receiver> | Mandatory | The component code of the recipient endpoint. – Pattern: [A-Za-z0-9-@]+ |

Additional rules:

- The to-be-sent filenames without extension shall not end with the dot character (“.”).
- The sender endpoint ignore files in the OUT-folder with extensions “TMP” or “tmp”.
- The sender endpoint fails to send the files matching one of the following conditions:
  - The filename does not match the “files to-be-sent” filename format.
  - The filename is longer than 200 characters.
  - The file is empty.

### 11.2.3 Concurrent access to files

As the applications and the endpoint concurrently access the files, special attention is required to avoid access conflicts and data losses.

- The file-reader ignores the files which extensions are “TMP” or “tmp”.
- The file-writer write data first in a temporary file with extension “TMP” or “tmp”, and then rename it changing the extension, as “rename” is an atomic operation on every file system.

Data can be lost if a file is overridden by another file having the same filename. To avoid this, each file should have a unique filename:

- OUT – OUT_ERROR – OUT_LOG: It is highly recommended that the applications use <SenderApp> and <BAmessageID> to ensure they cannot use the same filename.
- IN: the use of <MessageID> in the filename ensures that the contents of two different received messages always generate two different files.

### 11.2.4 Configuring FSSF

The administrator can configure the endpoint with following information:

- An OUT folder name.
- An OUT_ERROR folder name.
- An OUT_LOG folder name.
- A list of message-types, each associated with an IN folder and a default extension (possibly empty).
- A default IN folder and a default extension (possibly empty).

The recipient endpoint writes the content of a received message in a file in the IN folder associated to the message-type, having the file extension provided in the *extension* attribute of message metadata (see Table 11), and use the default values if any and when applicable.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# Bibliography

IETF RFC 4266, *The gopher URI Scheme,*
https://tools.ietf.org/html/rfc4266

NOTE The first definition for many URI schemes appeared in RFC 1738. Because that document has been made obsolete, RFC 4266 copies the gopher URI scheme from it to allow that material to remain on standards track.

IETF RFC 4634, *US Secure Hash Algorithms (SHA and HMAC-SHA),*
https://tools.ietf.org/html/rfc4634

W3C Recommendation 10 June 2008, *XML Signature Syntax and Processing (Second Edition)*
http://www.w3.org/TR/xmldsig-core/

IETF RFC 4122, *A universally unique identifier (UUID) URN namespace,*
https://tools.ietf.org/html/rfc4122

IETF RFC 4422, *Simple Authentication and Security Layer (SASL),*
https://tools.ietf.org/html/rfc4422

IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2,*
https://tools.ietf.org/html/rfc5246

ITU-T Recommendation X.690, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER),*
http://www.itu.int/rec/T-REC-X.690/en

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# SOMMAIRE



IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

# CADRE POUR LES COMMUNICATIONS POUR LE MARCHÉ DE L'ÉNERGIE –

## Partie 503: Lignes directrices concernant les échanges de données du marché pour le profil défini dans l'IEC 62325-351

## AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

La Norme internationale IEC 62325-503 a été établie par le comité d'études 57 de l'IEC: Gestion des systèmes de puissance et échanges d'informations associés.

Cette édition annule et remplace l'IEC TS 62325-503 parue en 2014.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

a) Utilisation de l'ISO/IEC 19464:2014, Advanced Message Queuing Protocol (AMQP) v1.0 specification (disponible en anglais seulement);
b) Répartition du nœud décrit dans l'IEC TS 62325-503:2014 dans un courtier qui met en œuvre la fonction de transmission de messages et un annuaire;

c) Renforcement de l'opérabilité et de la résilience du système de communication avec capacité pour un point d'extrémité d'envoyer et de recevoir des messages par l'intermédiaire de plusieurs courtiers;

d) Avantage issu de la normalisation, des performances et de l'évolutivité du protocole AMQP de transfert de messages.

Le texte de cette norme est issu des documents suivants:

| CDV | Rapport de vote |
|---|---|
| 57/1936/CDV | 57/1983/RVC |

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette Norme internationale.

Ce document a été rédigé selon les Directives ISO/IEC, Partie 2.

Dans ce document, les caractères suivants sont utilisés:

*Facilitation de la visibilité de l'information dans les tableaux et diagrammes: en italique*

Une liste de toutes les parties de la série IEC 62325, publiées sous le titre général *Cadre pour les communications pour le marché de l'énergie* peut être consultée sur le site web de l'IEC.

Le comité a décidé que le contenu de ce document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives au document recherché. À cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

**IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.**

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# INTRODUCTION

Le présent document fait partie de la série IEC 62325 concernant les communications relatives au marché déréglementé de l'énergie.

Le principal objectif de la série de normes IEC 62325 est de produire des documents destinés à faciliter l'intégration de logiciels d'application pour le marché, développés de façon indépendante par différents fournisseurs, dans un système de gestion de marché et entre des systèmes de gestion de marché et des systèmes participant au marché. Pour ce faire, des échanges de messages sont définis afin de permettre à ces applications ou systèmes d'accéder aux données publiques et d'échanger des informations, indépendamment de la façon dont ces informations sont représentées en interne.

Le modèle d'information commun (CIM, *common information model*), spécifie la base d'une sémantique d'échange des messages. Les spécifications d'un profil marché de style européen qui couvrent les besoins des marchés de l'électricité conçus selon le style européen sont définies dans l'IEC 62325-351. Ces marchés de l'électricité respectent la réglementation européenne et appliquent les concepts d'accès tiers et de découpage des marchés en zones. Les normes internationales IEC 62325-451-n spécifient le contenu des messages échangés.

L'objet du présent document est de fournir des lignes directrices dédiées à l'échange des messages susmentionnés. Un participant aux marchés européens (opérateur, régies de distribution, etc.) peut tirer profit d'une plate-forme commune harmonisée et sûre d'échange des messages avec les gestionnaires de réseaux de transport (GRT), réduisant ainsi le coût de mise en place de différentes plates-formes TI assurant l'interface avec toutes les parties concernées.

Le présent document représente une étape importante permettant aux parties de participer à des marchés de l'électricité autres que leurs marchés nationaux. Elles peuvent utiliser le même système d'échange d'informations ou un système analogue pour participer à deux marchés ou plus dans toute l'Europe.

À l'origine, le présent document s'appuyait sur les travaux du groupe de travail EDI du Réseau européen des gestionnaires de réseaux de transport d'électricité (ENTSO-E – *European Network of Transmission System Operators*).

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# CADRE POUR LES COMMUNICATIONS POUR LE MARCHÉ DE L'ÉNERGIE –

## Partie 503: Lignes directrices concernant les échanges de données du marché pour le profil défini dans l'IEC 62325-351

## 1 Domaine d'application

La présente partie de l'IEC 62325 est destinée aux marchés européens de l'électricité.

Le présent document spécifie une norme relative à une plate-forme de communication que chaque gestionnaire de réseau de transport (GRT) en Europe peut utiliser pour échanger en toute fiabilité et sécurité des documents destinés au marché de l'énergie. Par conséquent, un participant aux marchés européens (GRT, centre régional de surveillance, régie de distribution, bourse d'électricité, etc.) peut tirer profit d'une plate-forme commune unique harmonisée et sûre d'échange des messages avec les autres participants, réduisant ainsi le coût de mise en place de différentes plates-formes de technologies de l'information (TI) assurant l'interface avec toutes les parties concernées.

MADES (Norme d'échange de données marché ou *MArket Data Exchange Standard* en anglais) est l'acronyme qui désigne la présente norme.

MADES constitue une spécification d'une plate-forme de communication commune décentralisée basée sur les normes TI internationales:

- Du point de vue d'une application, MADES spécifie les interfaces logicielles d'échange de documents électroniques avec des applications homologues. Ces interfaces permettent principalement d'envoyer et de recevoir des documents avec un système de communication appelé MADES (ou «système MADES», voire simplement «système»). L'expéditeur peut s'enquérir de l'état de livraison d'un document et le destinataire émet un message en retour, à savoir un accusé de réception, à réception du document. Cette situation permet d'utiliser un système MADES pour échanger des documents dans des processus métiers exigeant une livraison fiable.
- MADES spécifie également les services non perceptibles par les applications tels que la localisation du destinataire, l'état de connexion du destinataire, l'acheminement des messages et la sécurité. Les services comprennent l'annuaire, l'authentification, la signature, le chiffrement, le suivi, la trace et l'archivage temporaire des messages.

MADES a pour objet de produire une norme d'échange sécurisé de messages basée sur des protocoles de communication normalisés et d'appliquer les meilleures pratiques TI pour échanger des données sur tout réseau de communication TCP/IP afin de faciliter les échanges d'informations de métier à métier (B2B) tels que décrits dans les séries IEC 62325351 et IEC 62325-451.

Un système MADES fonctionne comme une organisation postale: l'objet transporté est un «message» dans lequel le document de l'expéditeur est incorporé en toute sécurité dans une enveloppe contenant des métadonnées qui constituent des informations nécessaires pour le transport, le suivi et la livraison.

## 2 Références normatives

Les documents suivants cités dans le texte constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC TS 61970-2, *Energy management system application program interface (EMS-API) – Part 2: Glossary* (disponible en anglais seulement)

ISO/IEC 19464:2014 *Information technology – Advanced Message Queuing Protocol (AMQP) v1.0 specification,* https://www.amqp.org/ (développé par l'OASIS open standards consortium) (disponible en anglais seulement)

ISO/IEC 9594-8:2017, *Information technology – Open systems interconnection – The Directory – Part 8: Public-key and attribute certificate frameworks* (disponible en anglais seulement)

## 3 Termes et définitions

Pour les besoins du présent document, les termes et les définitions de l'IEC TS 61970-2 ainsi que les suivants, s'appliquent.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

NOTE Pour des définitions générales, se référer à l'IEC 60050, *Vocabulaire Electrotechnique International.*

**3.1**
**protocole avancé de mise en file d'attente de messages**
**AMQP**
protocole Internet ouvert pour la transmission de messages d'activité, comme défini dans l'IEC 19464:2014

Note 1 à l'article: L'abréviation «AMQP» est dérivée du terme anglais développé correspondant «advanced message queuing protocol».

**3.2**
**protocole sécurisé et avancé de mise en file d'attente de messages**
**AMQPS**
combinaison du protocole de transmission de messages d'activité défini dans l'IEC 19464 et de la sécurité de la couche transport (TLS – *transport layer security*)

Note 1 à l'article: L'abréviation «AMQPS» est dérivée du terme anglais développé correspondant «advanced message queuing protocol secured with transport layer security»

**3.3**
**norme d'échange de données marché**
**MADES**
spécification décrite dans le présent document pour le profil marché de style européen

Note 1 à l'article: L'abréviation «MADES» est dérivée du terme anglais développé correspondant «market data exchange standard».

**3.4**
**transfert d'état représentatif**
**REST**
mode effectif d'interopérabilité entre des systèmes d'ordinateurs par une demande d'accès et de manipulation des représentations textuelles de ressources par application d'un ensemble prédéfini d'opérations sans état

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

Note 1 à l'article: L'abréviation «REST» est dérivée du terme anglais développé correspondant «representational state transfer».

**3.5**
**couche authentification et sécurité simple**
**SASL**
cadre d'authentification et de sécurité des données dans les protocoles Internet

Note 1 à l'article: L'abréviation «SASL» est dérivée du terme anglais développé correspondant «simple authentication and security layer».

**3.6**
**protocole d'accès à des objets simples**
**SOAP**
spécification de protocole pour l'échange d'informations structurées dans la mise en œuvre de services Web

Note 1 à l'article: L'abréviation «SOAP» est dérivée du terme anglais développé correspondant «simple object access protocol».

**3.7**
**gestionnaire de réseau de transport**
**GRT**
entité impliquée dans le transport de l'énergie électrique ou du gaz naturel

**3.8**
**sécurité de la couche transport**
**TLS**
protocole cryptographique assurant le caractère privé et l'intégrité des données entre deux applications informatiques communicantes

Note 1 à l'article: L'abréviation «TLS» est dérivée du terme anglais développé correspondant «transport layer security».

## 4 Concepts évolués

### 4.1 À quoi sert MADES ?

NOTE L'Article 4 décrit le contexte métier d'utilisation de la spécification et en introduit les principaux concepts. En revanche, il ne comporte aucune exigence.

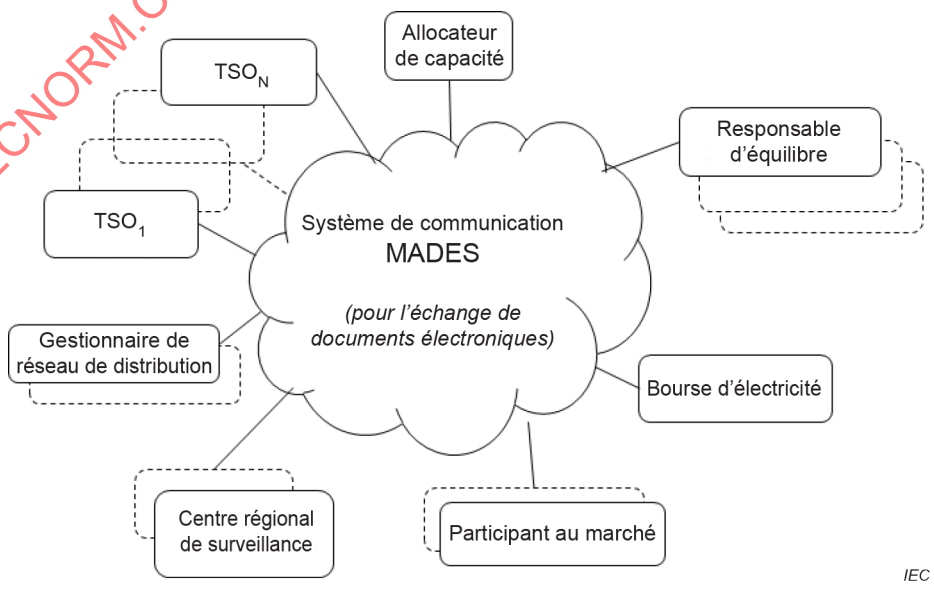



**Figure 1 – Aperçu général de MADES**

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

Le premier objectif de MADES est de fournir aux gestionnaires de réseaux de transport (GRT) un système de communication normalisé permettant d'échanger en toute sécurité des documents électroniques entre eux et avec les autres parties impliquées dans le marché européen de l'électricité comme représenté à la Figure 1.

Ces documents sont principalement ceux utilisés dans le marché de l'énergie et décrits dans les séries IEC 62325-351 et IEC 62325-451. Les parties impliquées comprennent les GRT, centres régionaux de surveillance, bourses d'électricité, gestionnaires de réseau de distribution (GRD), responsables d'équilibre, allocateurs de la capacité de transport, opérateurs de marchés, opérateurs de capacité, producteurs, etc.

MADES permet à chaque partie d'héberger un point d'accès au système de communication: les applications de son système d'information peuvent alors utiliser le point d'accès pour envoyer et recevoir en toute sécurité des documents avec d'autres parties.

D'un point de vue technique, MADES est une norme d'échange de données constituée de protocoles de communication normalisés et qui applique des normes et les meilleures pratiques des technologies de l'information pour créer un mécanisme d'échange de données sur tout réseau et toute infrastructure de communication TCP/IP afin de faciliter les échanges d'informations métier à métier.

Les nouvelles règles de marché engendrent de nouveaux processus et activités métiers, et exigent généralement de nouveaux échanges d'informations entre les parties impliquées. Par expérience, les échanges de données répondent aux objectifs métiers lorsque la solution technique retenue résulte d'un accord entre toutes les parties impliquées qui rassemble diverses contraintes y compris le calendrier de mise en œuvre, les offres des fournisseurs, l'infrastructure et les liaisons de communication déjà existantes, les capacités d'intégration des systèmes d'information existants, la confidentialité des informations échangées, les risques juridiques, les performances, etc.

Lorsque les processus métiers exigent un échange d'informations entre plusieurs systèmes ou parties, les solutions bilatérales peuvent devenir complexes, la mise en œuvre de chaque interface nécessitant du temps, ainsi que des moyens financiers et autres ressources pour son développement et sa maintenance. Conséquence notable: certaines parties impliquées dans les processus métiers qui exigent des échanges avec de nombreuses autres parties doivent installer et utiliser différents outils et solutions de communication afin d'assurer une interface avec plusieurs systèmes d'information. Le processus conduisant à un marché européen de l'électricité plus intégré engendre de nombreuses situations de ce type avec coordination de nouveaux processus entre les acteurs dont les rôles sont réglementés. Les participants au marché qui interviennent dans plusieurs pays recherchent également une interface harmonisée.

MADES peut prendre en charge tout processus métier quel que soit le format de document transmis (par exemple, XML, binaire), ainsi que la séquence d'échanges.

MADES est indépendant de l'infrastructure de communication physique sous-jacente, qui peut être tout réseau de protocole Internet (IP), tel que l'Internet, une infrastructure privée physique ou un réseau privé virtuel (RPV) à points d'accès multiples.

MADES repose uniquement sur des normes TI non exclusives relatives aux protocoles de communications, l'intégrité des données, l'authentification des entités homologues (signature), la confidentialité (chiffrement), l'annuaire des parties accessibles (par exemple, HTTP, AMQP, TLS, X.509), comme représenté à la Figure 2.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

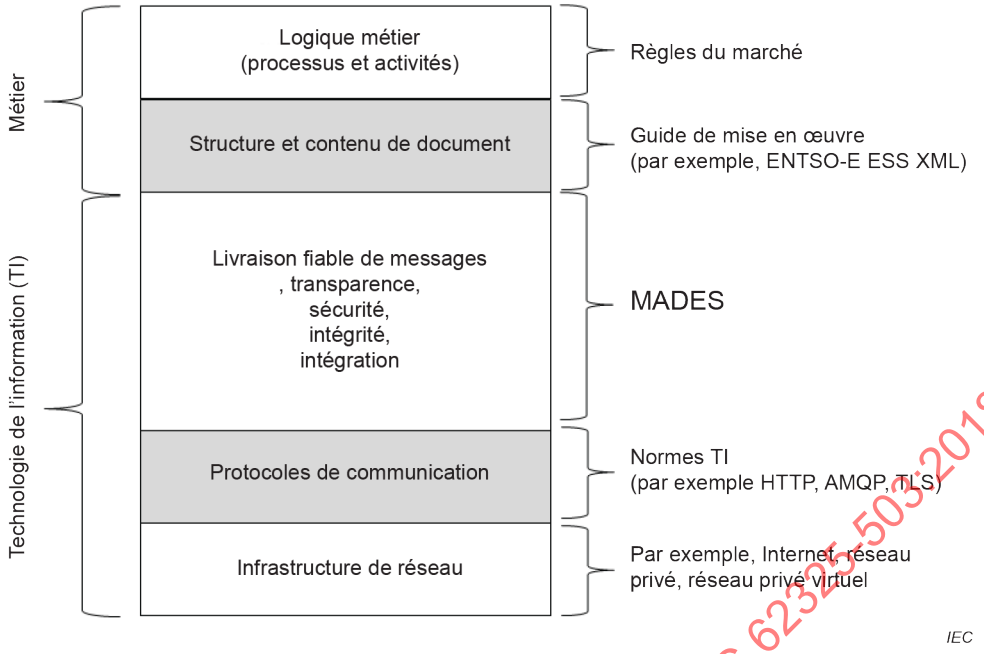



**Figure 2 – Domaine d'application de MADES dans une architecture en couches**

## 4.2 Présentation

La norme MADES spécifie un système de communication de livraison de messages entre des parties autorisées. Ce type de système présente les principales caractéristiques suivantes:

1) Livraison fiable de messages– Une partie (émettrice) connectée peut envoyer un message à une autre partie (destinataire) connectée ou qui peut se connecter au système. L'expéditeur est informé de l'absence d'accusé de réception d'un message par le destinataire à l'échéance, quelle qu'en soit la raison.
2) Transparence – Le système n'est pas une boîte noire: il assure le suivi de chaque message afin de recueillir des informations fiables sur la livraison.
3) Sécurité – Seul le destinataire du message peut lire le contenu (confidentialité) et identifier l'expéditeur sans aucune ambiguïté (signature).
4) Intégrité – Le système garantit l'absence de modification du contenu d'un message au cours du processus de livraison.
5) Intégration – De nombreuses technologies peuvent être intégrées aux services utilisés pour l'envoi et la réception de messages. Cette caractéristique facilite le développement et l'intégration dans tout système d'information.

Les quatre premières principales caractéristiques (livraison fiable de messages, transparence, sécurité et intégrité) constituent des capacités du système, tandis que la dernière caractéristique (intégration) est une caractéristique de conception des composants MADES.

## 4.3 Livraison fiable et transparente de messages

La principale caractéristique d'un système MADES est la fonction de livraison de messages, telle que représentée à la Figure 3.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018


**Figure 3 – Livraison de messages MADES**

Un expéditeur souhaite ou a besoin de transférer un document à un destinataire en utilisant le système. L'expéditeur et le destinataire sont tous deux des applications qui utilisent une interface de programmation d'application (API – *application-programming interface*) pour permettre la connexion à leurs points d'accès respectifs, appelés *points d'extrémité*.

Les applications homologues ne voient le système qu'au travers de l'interface définie. Le document transmis d'une application à l'autre peut être du texte ou des données binaires. En effet, le système achemine un message qui comprend le document contenant des métadonnées supplémentaires, qui inclut les informations d'acheminement et de transmission du message en toute sécurité, telles qu'un identifiant unique, qui permet d'identifier les points d'extrémité homologues et les informations cryptographiques associées à la signature du message et au chiffrement du document.

Le système assure le suivi de tout message accepté par le point d'extrémité d'un expéditeur. Le point d'extrémité peut rejeter une demande d'envoi d'un document par une application métier, par exemple lorsque le destinataire est inconnu.

L'application de l'expéditeur peut vérifier à tout moment l'état d'un message envoyé, à savoir ACCEPTED (ACCEPTÉ), DELIVERED (LIVRE), RECEIVED (REÇU) ou FAILED (ÉCHEC) (voir 5.6.1).

Le point d'extrémité expéditeur sait que le système a livré un message au point d'extrémité destinataire, lorsqu'il reçoit un accusé de réception émis par ce dernier. Le point d'extrémité expéditeur définit l'état du message à FAILED lorsqu'il expire, c'est-à-dire s'il n'a pas reçu l'accusé de réception à l'échéance, quelle qu'en soit la raison.

Le système achemine à la fois:

- Un message normal – message constitué d'un point d'extrémité expéditeur afin d'envoyer un document électronique sur demande d'une application expéditrice.
- Un accusé de réception – message constitué d'un point d'extrémité destinataire afin d'assurer le suivi de la livraison d'un message normal[1] entre deux points d'extrémité. Les applications n'ont pas connaissance des accusés de réception qui restent internes au système.

[1] Il s'agit d'un accusé de réception technique, et non d'un accusé de réception tel que défini dans l'IEC 62325-451-1.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

NOTE Dans le reste du document, le terme «message» fait référence à «message normal», et l'expression «message interne» fait généralement référence soit à un message (normal), soit à un accusé de réception.

## 4.4 Composants d'un système MADES

### 4.4.1 Point d'extrémité, courtier et annuaire des composants (component-directory)

Un système MADES est un ensemble de composants de trois types, c'est-à-dire un point d'extrémité, un courtier et un annuaire des composants (component-directory). Deux composants interagissent selon un mode client / serveur tel que représenté à la Figure 4. Il s'agit de la vue d'ensemble que le présent article explique de manière détaillée.

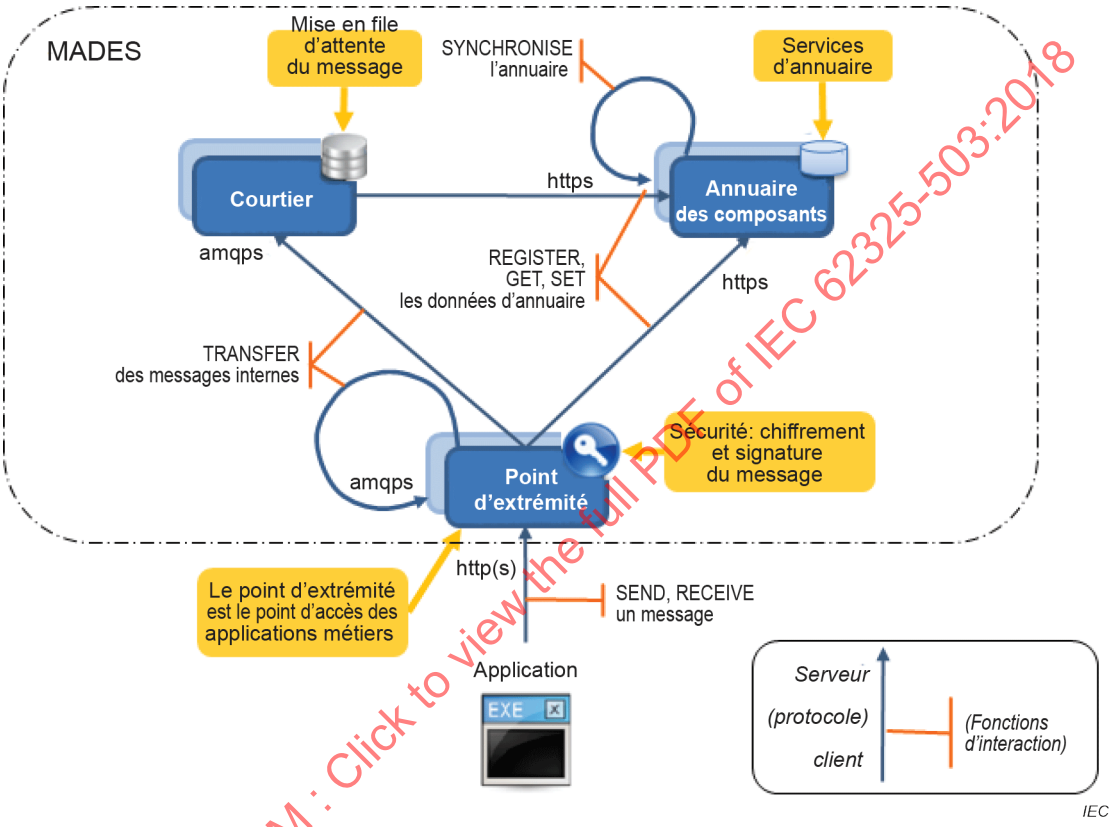



**Figure 4 – Composants, interactions et protocoles d'un système MADES**

Une partie qui utilise le système héberge un point d'extrémité. Une application (*client*[2]) utilise l'interface exposée par le point d'extrémité (*serveur*) pour envoyer et recevoir des messages. L'interface utilise le protocole HTTP(S).

Le présent document fait référence au point d'extrémité d'un expéditeur (d'un destinataire respectivement) lorsqu'il s'agit de cibler les opérations effectuées par le point d'extrémité pour l'envoi (la réception respectivement) d'un message. Les points d'extrémité assurent la sécurité de bout en bout, c'est-à-dire que le point d'extrémité de l'expéditeur signe et chiffre le message, tandis que le point d'extrémité destinataire déchiffre le message et vérifie la signature de l'expéditeur.

Pour le transfert d'un message, le point d'extrémité expéditeur (*client*) le dépose dans un courtier (*serveur*) et le point d'extrémité destinataire *(client)* le télécharge à partir de ce courtier. Le courtier archive provisoirement le message dans une file d'attente; ce processus est qualifié ci-après de transfert *indirect*. Le point d'extrémité expéditeur (*client*) peut également déposer directement le message dans le point d'extrémité destinataire (*serveur*);

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

2 Le composant client ouvre une connexion. Le composant serveur accepte la connexion.

ce processus est qualifié ci-après de transfert *direct*. Quel que soit le parcours, le protocole de transfert du message est AMQPS, qui signifie Protocole sécurisé et avancé de mise en file d'attente de messages (TLS).

Le transfert d'un accusé de réception fonctionne selon le même principe, les rôles des points d'extrémité homologues étant toutefois inversés, le destinataire étant l'expéditeur et réciproquement. Un courtier fonctionne uniquement avec des messages internes et ne différencie pas les messages des accusés de réception.

Les points d'extrémité et les courtiers (*tous les clients*) obtiennent les données de configuration du système (par exemple, identifications des composants existants, certificats, parcours de livraison autorisés) à partir d'un annuaire des composants (*un serveur*), qui met en œuvre un carnet d'adresses. Lorsque le système est composé de plusieurs annuaires des composants, ils se synchronisent entre eux. Les services de l'annuaire utilisent le protocole HTTPS.

### 4.4.2 Parcours de livraison et accusés de réception

La Figure 5 présente les parcours possibles pour le transfert d'un message:

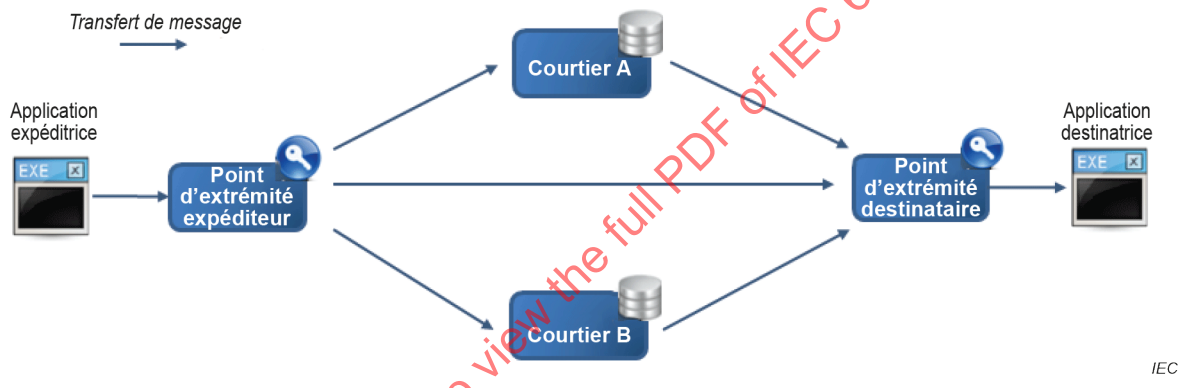



**Figure 5 – Parcours possibles pour la livraison d'un message**

Un point d'extrémité expéditeur peut transférer un message à un point d'extrémité destinataire soit au moyen d'un courtier intermédiaire, soit directement. L'autorisation ou l'interdiction d'un transfert direct est un choix de sécurité (voir 4.5.2).

La Figure 5 indique que le système peut inclure plusieurs courtiers et que les messages entre deux points d'extrémité donnés peuvent suivre des parcours différents. Le parcours dépend du type de document à transmettre, appelé ci-après *type de message* (message-type), qui lie généralement le message à un certain processus métier (voir 5.2). Le point d'extrémité expéditeur choisit le parcours selon le chemin du message (message-path) (voir 5.3) associé au type de message, selon la configuration du point d'extrémité *destinataire*, par ailleurs publié et accessible dans les annuaires des composants.

La Figure 6 présente les protocoles de communication utilisés pour les parcours possibles d'un message.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

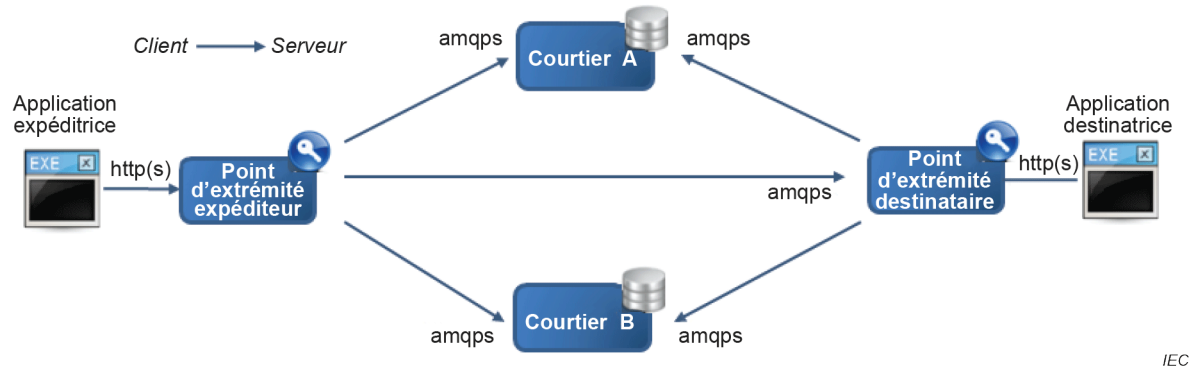



**Figure 6 – Protocoles de communication pour la livraison d'un message**

Un accusé de réception est un message interne transmis par le point d'extrémité destinataire au point d'extrémité expéditeur afin de l'informer de la réception d'un message. Le parcours de l'accusé de réception est l'*exact parcours inverse* du message correspondant.

NOTE:

- L'application expéditrice définit le type de message, mais ne choisit pas le parcours. La modification du chemin du message modifie le parcours, mais n'a pas d'impact sur les applications.
- Un équilibrage de charge relatif au trafic de messages par l'intermédiaire de plusieurs courtiers n'est pas possible. Entre deux points d'extrémité donnés, tous les messages avec le même type de message suivent le même parcours. Certains courtiers peuvent être configurés en haute disponibilité.
- Le parcours entre un point d'extrémité expéditeur et un point d'extrémité destinataire comporte au plus un courtier.

### 4.4.3 Partage des données de configuration du système

La mise en œuvre en toute transparence de l'acheminement et de la sécurité d'un message dans des applications signifie que les points d'extrémité et les courtiers ont accès aux données de configuration qui décrivent le système. Cette configuration inclut les informations suivantes pour chaque composant:

- identifiant unique,
- certificats de sécurité pour l'authentification, la signature et le chiffrement,
- emplacement du réseau de télécommunications pour l'accès aux services que le composant utilise,
- règles applicables à la livraison d'un message: dans le cas d'un point d'extrémité, les règles précisent les parcours pour lui envoyer des messages; dans le cas d'un courtier, les règles précisent les messages qu'il accepte, c'est-à-dire les types de messages et les points d'extrémité qui peuvent les émettre,
- Informations pratiques concernant les entités à contacter.

Chaque point d'extrémité et chaque courtier s'enregistrent auprès d'un annuaire des composants unique, appelé ci-après leur *annuaire de référence*. Un rôle important de l'administrateur d'un annuaire des composants réside dans les fonctions suivantes: il doit accepter de nouvelles parties, établir une relation de confiance avec elles et enregistrer leurs composants. Ce rôle comprend les opérations inverses: par exemple, désenregistrer des composants, notamment ceux potentiellement compromis. Un annuaire des composants et l'ensemble de ses composants enregistrés forment un *sous-système*.

La Figure 7 présente le flux de données entre un annuaire des composants "A" et ses composants enregistrés:

(A) Lors de l'acceptation d'une demande d'enregistrement, l'annuaire des composants crée une nouvelle entrée pour le composant. Il émet et archive également les certificats du composant.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

(B) L'administrateur d'un composant enregistré saisit toutes les données de configuration concernant le composant, qui sont ensuite transmises pour mise à jour à l'annuaire de référence du composant.

(C) Un composant enregistré stocke le contenu exhaustif de l'annuaire dans une mémoire cache locale et demande une synchronisation cyclique. Lors d'une modification, le composant reçoit une copie complète de l'annuaire, et non des incréments.

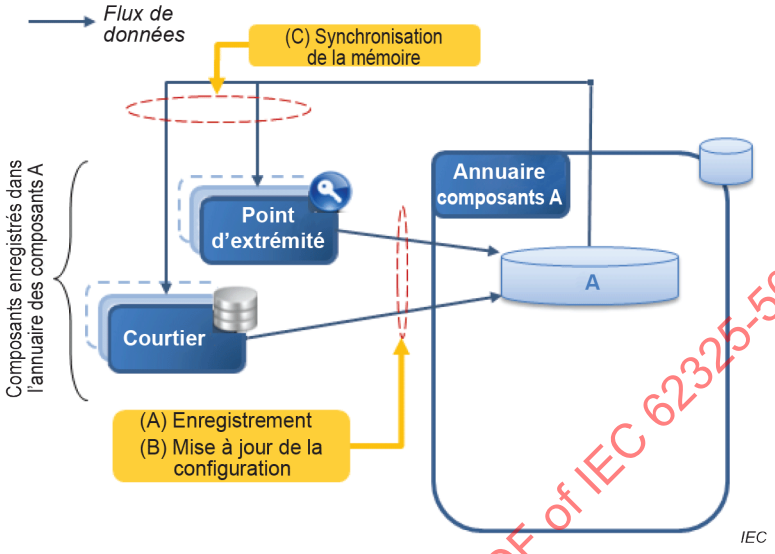



**Figure 7 – Flux de données entre un annuaire des composants et ses composants enregistrés**

La spécification de l'annuaire des composants a un double objectif:

1) Réduire le plus possible les tâches administratives: la saisie de données est limitée. En outre, alors qu'il convient que le premier enregistrement exige probablement toujours une acceptation manuelle, le renouvellement des certificats peut être[3] automatique (voir 7.6) dans le cas où la demande provient d'un composant enregistré et valide.
2) Concevoir un composant non critique, c'est-à-dire qui ne bloque pas le transfert d'un message lorsqu'il est arrêté. En effet, chaque point d'extrémité et chaque courtier stockent les données de l'annuaire dans une mémoire cache locale et les rafraîchissent, elles restent valides pendant un laps de temps configurable au cours duquel l'annuaire des composants est arrêté.

La Figure 8 présente le même annuaire de composant "A", à présent intégré à un système plus large comportant deux autres sous-systèmes, c'est-à-dire les annuaires des composants: "B" et "C".

- La synchronisation de chaque couple d'annuaires des composants s'effectue de manière bidirectionnelle, c'est-à-dire que chaque annuaire demande à l'autre annuaire le contenu de ses sous-systèmes;
- Un point d'extrémité enregistré "A" (un courtier respectivement) reçoit à présent le contenu exhaustif des annuaires du système ("A" + "B" + "C") lors de la synchronisation avec son annuaire de référence;
- un composant enregistré "B" ou "C" se synchronise avec l'annuaire des composants "A" lorsque son annuaire de référence est arrêté. Inversement, un composant enregistré "A" peut se synchroniser avec un autre annuaire des composants (ce flux de données n'est pas représenté à la Figure 8).

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

3 Il s'agit d'une décision relevant de la gouvernance du système.

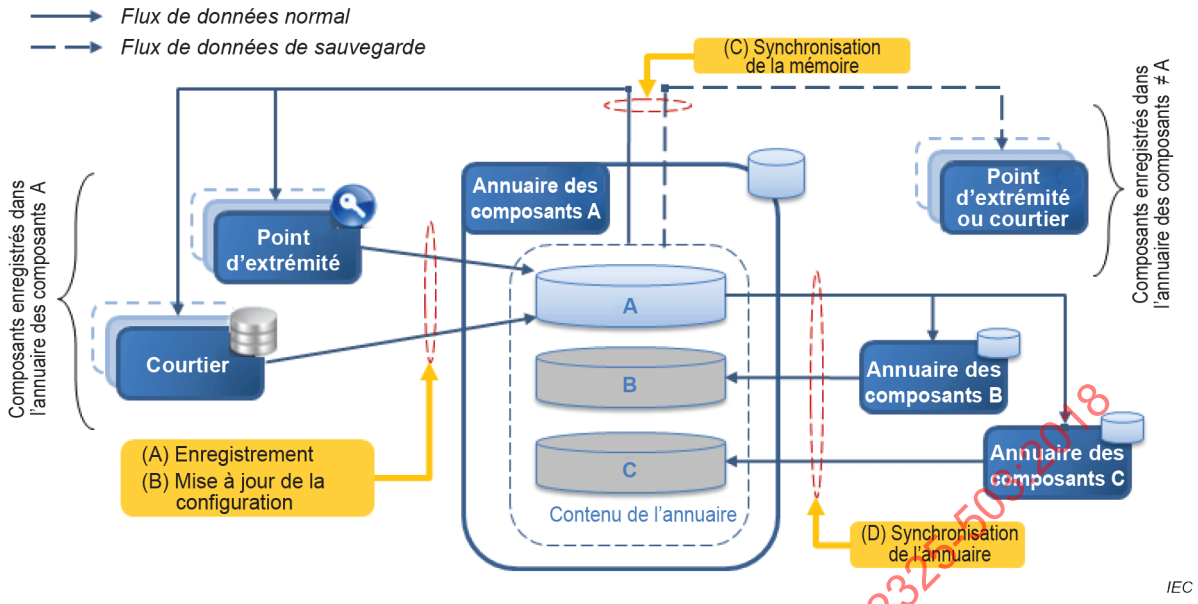



**Figure 8 – Flux de données avec plusieurs annuaires des composants**

NOTE La mise en mémoire cache locale du contenu exhaustif de l'annuaire constitue le point clé pour la non-criticité d'un annuaire des composants dans le système. L'hypothèse de conception sous-jacente est que les données de configuration restent peu nombreuses et ne varient pas fréquemment même lorsqu'une centaine de composants environ constituent le système.

Tous les services exposés par un annuaire des composants sont accessibles par le protocole HTTPS comme représenté à la Figure 9.

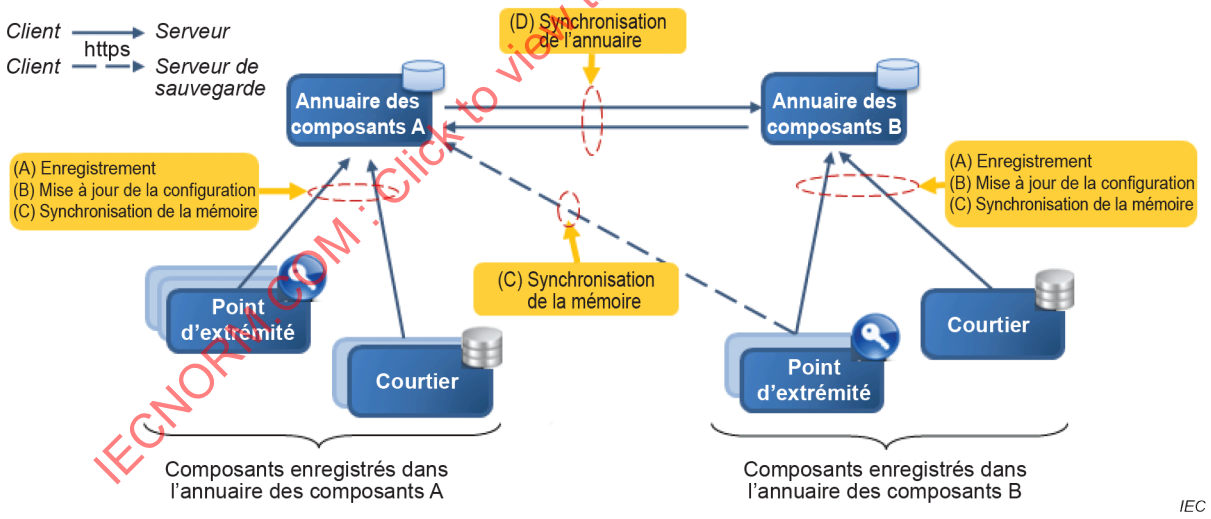



**Figure 9 – Services et protocoles d'un annuaire des composants**

### 4.4.4 Interfaces exposées par les composants

La Figure 10 présente les interfaces définies par la spécification dans l'Article 5, et mises en œuvre, c'est-à-dire utilisées et exposées par un composant conforme.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

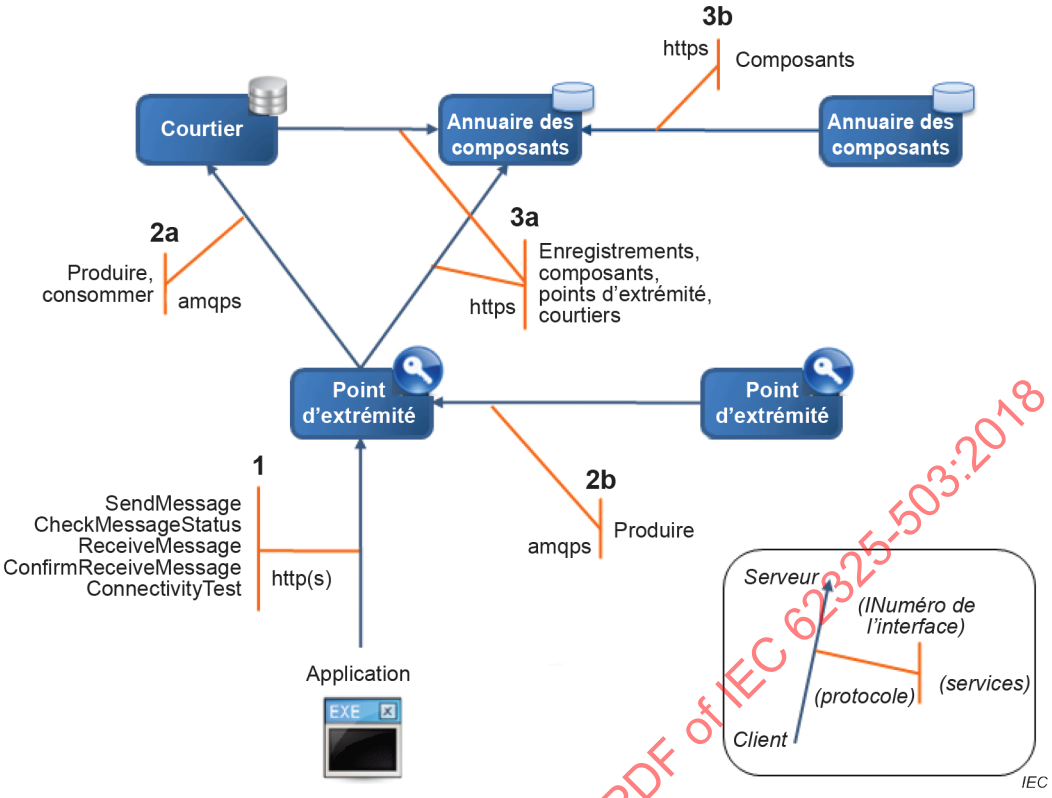



**Figure 10 – Interfaces, services et protocoles MADES**

(1) Interface des points d'extrémité pour les applications utilisant les services Web HTTP(S) SOAP – voir 11.1.

- Une application envoie un message (*SendMessage*) par l'intermédiaire du système ou demande l'état de livraison d'un message envoyé précédemment (*CheckMessageStatus*).
- Une application demande le téléchargement d'un message reçu (*ReceiveMessage*), puis confirme en retour au système qu'il a archivé [4] ou traité en toute sécurité le message (*ConfirmReceiveMessage*).
- Une application demande au système d'envoyer un message d'essai à une partie homologue (*ConnectivityTest*) afin de vérifier le bon fonctionnement de la communication avec cette partie pour un type de message donné.

(2a) Interface avec courtier pour les points d'extrémité utilisant le protocole AMQP – voir 6.3.

- AMQP est un protocole dédié à l'interopérabilité entre divers intergiciels de messagerie: il ne décrit aucune interface, mais la structure des trames d'octets échangées entre deux entités homologues TCP (*pour le transfert*).
- Un courtier MADES est un serveur qui gère les files d'attente de messages (internes) jusqu'à leur réception par le destinataire.
- Lorsque le courtier l'a identifié avec succès, un point d'extrémité agit comme un producteur pour envoyer des messages (*Produire*) et comme un consommateur pour recevoir des messages (*Consommer*).

(2b) Interface avec point d'extrémité pour les autres points d'extrémité utilisant le protocole AMQP – voir 6.3.

4 Une mémoire fiable est définie en 5.9.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- L'interface de transfert direct entre les points d'extrémité est identique à l'interface (2a) sans le service *Consommer*. En effet, un point d'extrémité ne contient pas de files d'attente dans lesquelles les autres points d'extrémité peuvent consommer des messages.

(3a) Interface d'annuaire des composants utilisée par les points d'extrémité et les courtiers – voir 7.5.

- Les points d'extrémité et les courtiers utilisent l'interface pour s'enregistrer auprès de leur annuaire de référence auquel ils transmettent les données de configuration. Ils utilisent également cette interface pour tenir à jour une copie locale de l'annuaire du système.
- L'interface applique les principes REST. Les demandes et les réponses utilisent le format XML et sont spécifiées à l'aide de schémas également XML.
- Les ressources accessibles à un composant authentifié sont les suivantes:
  - *enregistrements:* demander l'accès au système, obtenir et renouveler les certificats.
  - *Composants:* obtenir les informations d'annuaire concernant tous les composants enregistrés dans le système afin de conserver une copie locale de l'annuaire. La réponse renvoie le contenu exhaustif de l'annuaire ou aucune donnée lorsque l'annuaire local est déjà mis à jour.
  - *points* d'extrémité (courtiers respectivement): obtenir les données de configuration d'un point d'extrémité (un courtier respectivement) donné ou modifier ce type d'informations. Seul un composant peut modifier ses propres données de configuration.

(3b) Interface d'annuaire des composants utilisée par d'autres annuaires des composants – voir 7.5.

- Un annuaire des composants utilise l'interface pour se synchroniser avec un autre annuaire des composants du système.
- Pour effectuer la synchronisation, un annuaire des composants "A" demande les ressources '*composants*' d'un annuaire des composants "B" exactement comme le font les points d'extrémité et les courtiers lorsqu'ils utilisent l'interface (3b). Toutefois, il ne demande que le contenu de l'annuaire homologue limité aux composants du sous-système "B".

### 4.4.5 Exemples d'architecture des systèmes MADES

Un système MADES peut prendre la forme d'une architecture minimale constituée de deux points d'extrémité et d'un annuaire des composants sans (Figure 11) ou avec un courtier (Figure 12), ou d'un réseau de points d'extrémité avec plusieurs courtiers et annuaires des composants (Figure 13, Figure 14).

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

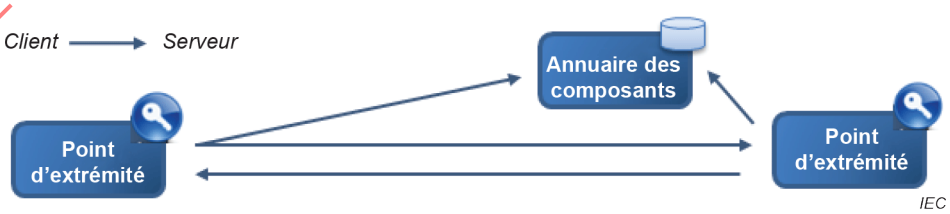



**Figure 11 – Système MADES minimal (sans courtier)**

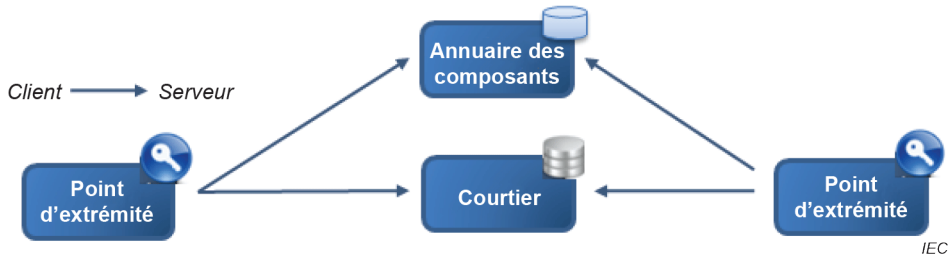
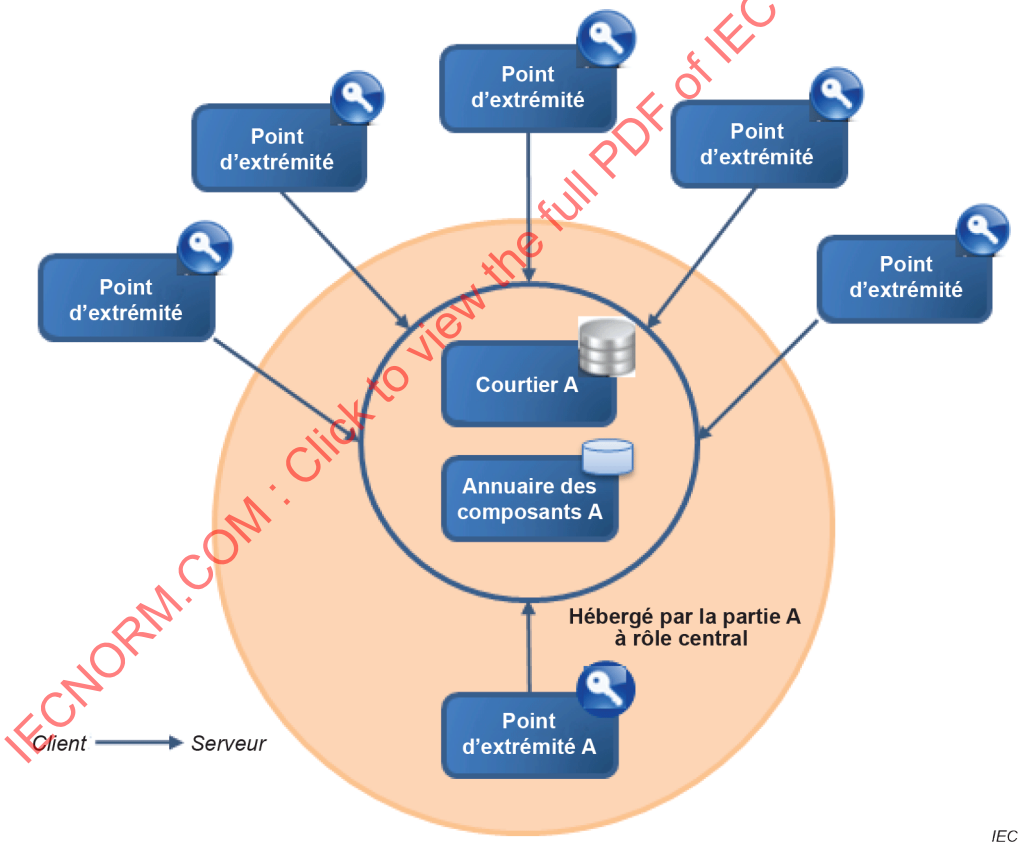



**Figure 12 – Système MADES minimal (avec courtier)**

L'une des deux parties communicantes utilisant l'architecture décrite à la Figure 12 peut héberger soit le courtier, soit l'annuaire des composants, voire les deux.

Une architecture type comprend une des parties prenantes à un processus métier, par exemple "A", dont le rôle est central (Figure 13). Par conséquent, cette partie prenante peut héberger un courtier et un annuaire des composants. Concentrateur dans le cadre du processus métier, elle possède et héberge également un point d'extrémité permettant les échanges avec les autres parties prenantes.




IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Figure 13 – Système MADES avec une partie dont le rôle est central**

Un autre processus métier peut impliquer les mêmes ou d'autres parties, voire un sous-groupe, dans lequel une partie, par exemple "B", exerce à présent un rôle central. Les parties peuvent partager le courtier existant "A". Elles peuvent toutefois utiliser également un autre courtier dédié (courtier "B") pour des raisons telles que la localisation d'hébergement, la criticité des activités, la délimitation des responsabilités, le contrat sur les niveaux de service (SLA – *service-level agreement*), la planification de projets, etc., ce qui entraîne une architecture à plusieurs courtiers comme représenté à la Figure 14.

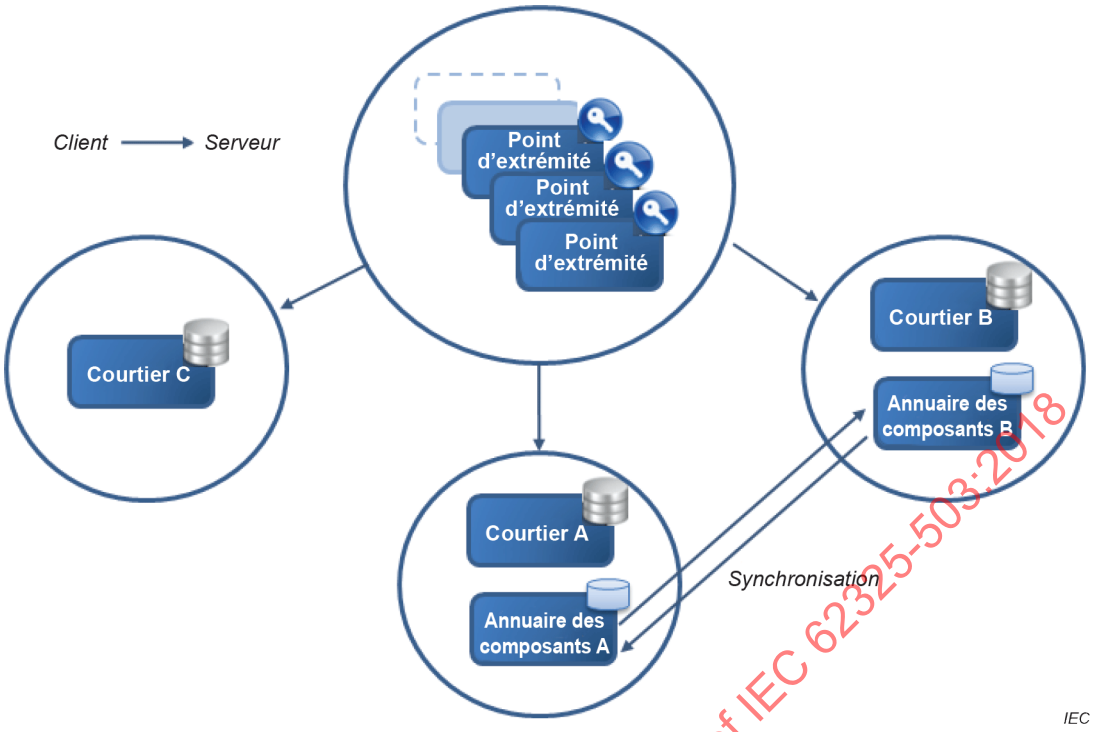



**Figure 14 – Système MADES avec plusieurs courtiers**

La Figure 14 présente deux annuaires des composants hébergés respectivement par les parties qui possèdent le courtier A et le courtier B. Il existe deux annuaires des composants, mais il peut y en avoir plus. Les règles sont les suivantes:

- Les annuaires des composants se synchronisent entre eux.
- Le système comporte deux annuaires des composants. Cette configuration est recommandée pour la continuité d'activité bien que l'annuaire des composants ne soit pas un composant critique, et puisse être arrêté pendant une certaine durée sans altérer la livraison de messages.

Dans l'exemple de la Figure 14, les deux parties A et B hébergent un annuaire des composants et constituent des bureaux d'enregistrement pour chaque sous-système, c'est-à-dire un sous-ensemble de points d'extrémité et de courtiers du système. Toutefois, la partie C peut également héberger un annuaire des composants.

La Figure 15 présente la même architecture vue d'une partie impliquée dans chaque processus métier. La partie possède un seul point d'extrémité enregistré dans un annuaire des composants unique. Dans ce cas de figure, la partie accepte le transfert direct et les connexions entrantes avec le point d'extrémité. L'architecture permet des échanges bilatéraux directs entre les parties sans incidence sur les courtiers.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

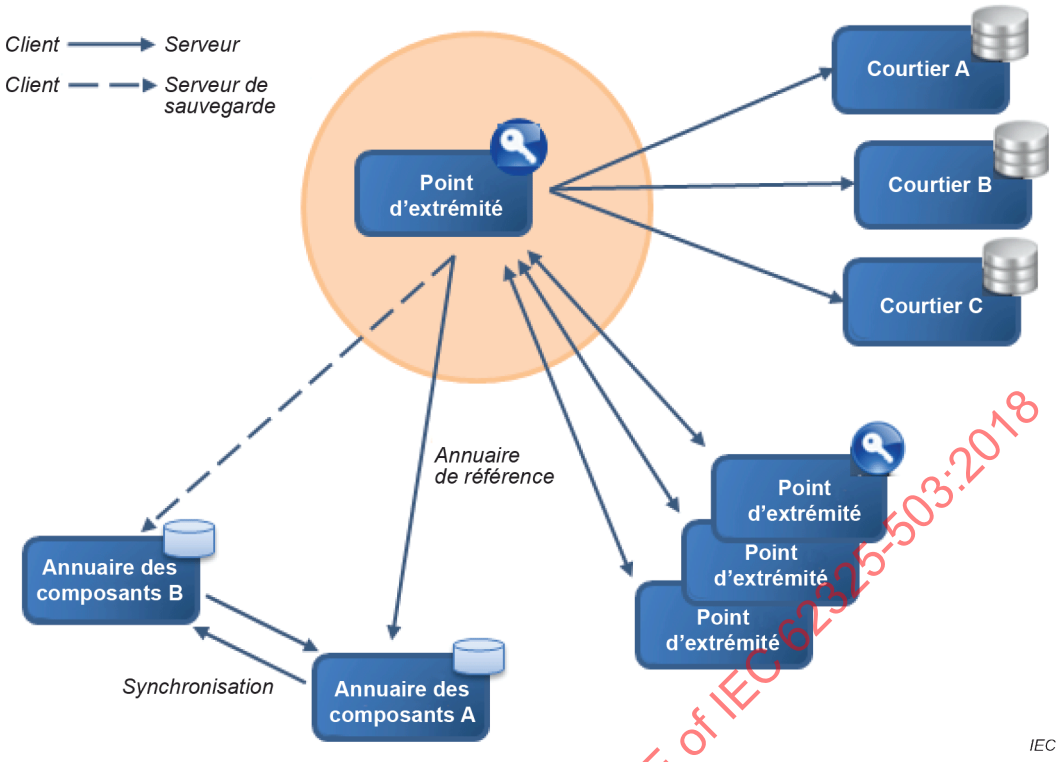



**Figure 15 – Utilisation d'un seul point d'extrémité pour plusieurs processus métiers**

La partie peut également choisir d'héberger plusieurs points d'extrémité, chacun de ces points étant dédié à un sous-ensemble de processus métiers ou de messages pour des raisons telles que la localisation d'hébergement, la criticité des activités, l'organisation interne et la délimitation des responsabilités, le contrat sur les niveaux de service (SLA), la planification de projets, etc. Chaque point d'extrémité doit alors s'enregistrer dans un annuaire des composants, et les points d'extrémité apparaîtront dans le système avec des identifiants différents comme s'ils étaient détenus par des parties distinctes. Ces points d'extrémité ne peuvent pas se comporter en mode maître / esclave, de même qu'ils ne peuvent pas effectuer un équilibrage de charge du trafic de messages entrants à destination de la partie[5].

## 4.5 Sécurité et intégrité des messages

### 4.5.1 Objectifs et solution de sécurité

Les objectifs de la sécurité MADES sont les suivants:

- La solution de sécurité est transparente pour les applications – aucune mise en œuvre spécifique n'est exigée pour la communication d'une application en toute sécurité.
- Tous les canaux de communication sont chiffrés.
- Seul le destinataire prévu peut lire un message.
- Le destinataire d'un message peut identifier l'expéditeur sans aucune ambiguïté.
- Non-répudiation – Un message associé à son accusé de réception forme une preuve non ambiguë de l'envoi du message par l'expéditeur et de sa réception par le destinataire.

La solution de sécurité spécifiée est conforme à l'infrastructure à clé publique X.509 (PKI – *public key infrastructure*) qui utilise des certificats, et couvre deux niveaux de sécurité: une sécurité de la couche transport et une sécurité au niveau du message (message-level).

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

5 La haute disponibilité d'un point d'extrémité constitue une question essentielle de sa conception.

- Un canal chiffré permet la communication de deux composants sur la couche transport. De plus, les deux composants s'authentifient toujours réciproquement sans aucune ambiguïté et en premier lieu, puis vérifient qu'ils sont des composants enregistrés valides du système avant d'échanger des données.
- Dans le cas de la sécurité au niveau du message, chaque message est signé puis chiffré.

Chaque composant possède un certificat d'authentification utilisé pour la sécurité de la couche transport. Chaque point d'extrémité possède deux certificats supplémentaires pour la sécurité au niveau du message: un certificat pour la signature des messages envoyés et l'autre certificat pour le déchiffrement des messages reçus.

### 4.5.2 Sécurité de la couche transport

La sécurité des données échangées entre deux composants repose tout d'abord sur la couche de protocole de transport. Deux composants communiquant utilisent le protocole «sécurité de la couche transport» (TLS) pour chiffrer le canal de communication. HTTPS (AMQPS respectivement) est la version sécurisée TLS de HTTP (AMQP respectivement).

Quels que soient les composants homologues communiquant, ils s'authentifient mutuellement à l'aide des certificats X.509 représentés à la Figure 16.

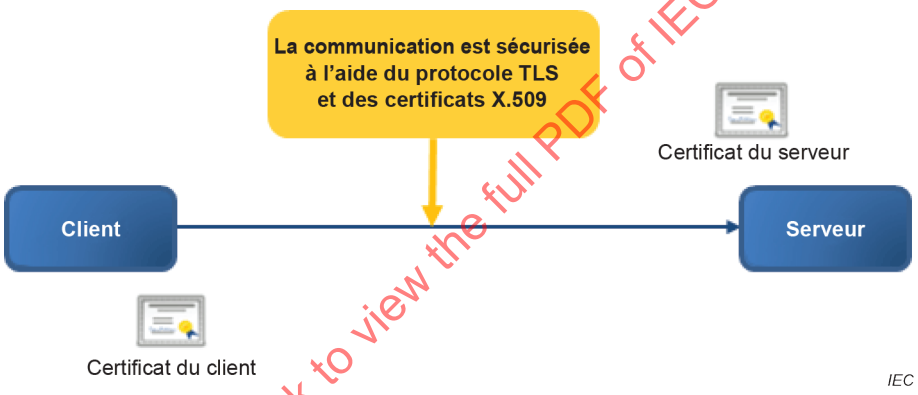



**Figure 16 – Sécurité de transport MADES**

Le client est toujours à l'origine de la communication, c'est-à-dire de la connexion IP. La Figure 10 présente tous les composants homologues possibles.

Cette configuration produit une architecture hautement sécurisée lorsque le point d'extrémité agit uniquement comme client, et un pare-feu supplémentaire peut renforcer la protection par un blocage de toutes les connexions entrantes comme représenté à la Figure 17.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

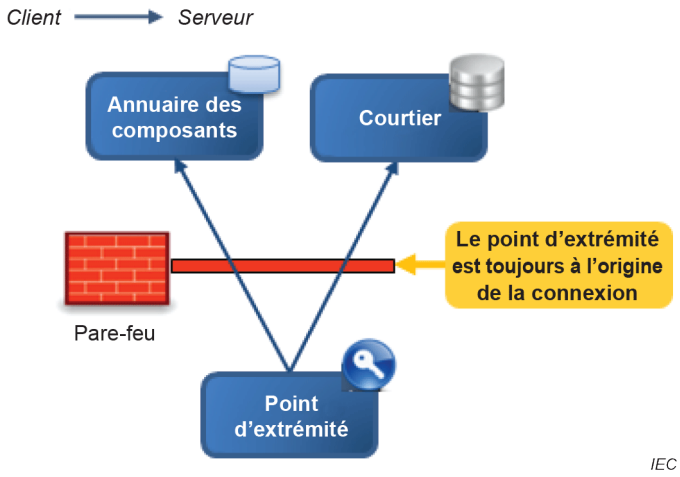



**Figure 17 – Sécurité: point d'extrémité protégé**

Lorsque les parties conviennent d'un transfert direct des messages entre les points d'extrémité, le pare-feu doit autoriser les connexions entrantes, éventuellement limitées aux points d'entrée homologues prévus comme représenté à la Figure 18.

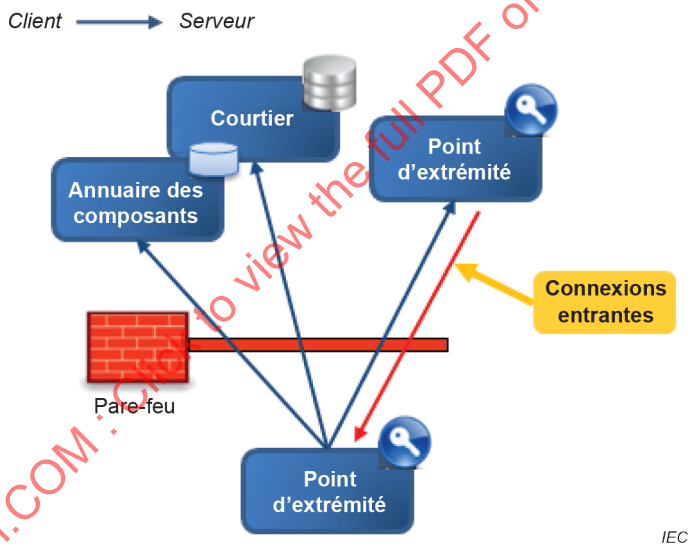



**Figure 18 – Sécurité: point d'extrémité exposé**

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

NOTE Les rôles sont symétriques et les deux points d'extrémité homologues doivent autoriser les connexions entrantes même lorsque le flux de messages s'effectue dans un seul sens, car les accusés de réception transitent alors dans l'autre sens.

### 4.5.3 Sécurité au niveau du message: signature et chiffrement

L'utilisation de signatures numériques permet d'identifier sans aucune ambiguïté l'expéditeur de chaque message, ainsi que l'accusé de réception reçu, comme représenté à la Figure 19.

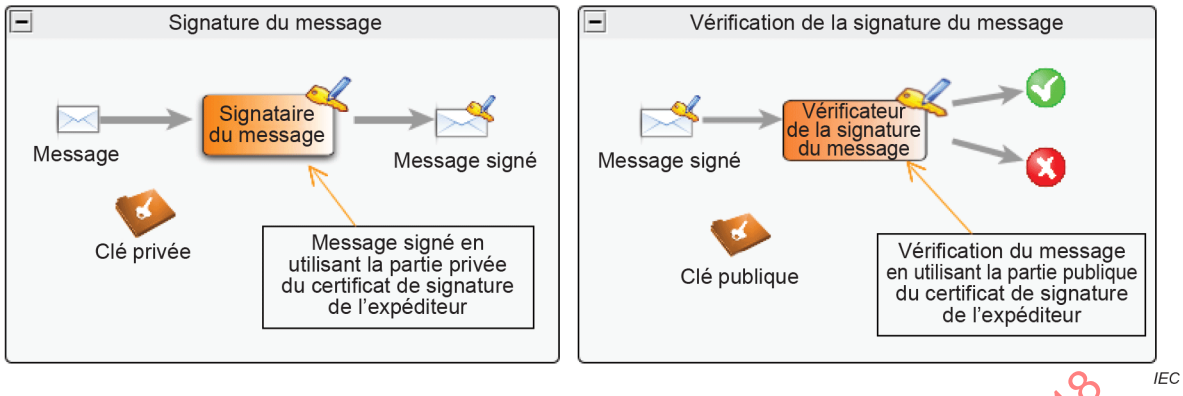



**Figure 19 – Signature de message et vérification de la signature**

Lors de l'envoi d'un message ou d'un accusé de réception, le point d'extrémité le signe à l'aide de la clé privée de son propre certificat de signature. Le point d'extrémité qui reçoit le message authentifie le point d'extrémité de l'expéditeur par vérification de la signature, c'est-à-dire à l'aide de la clé publique du certificat de signature de ce même point d'extrémité.

Le point d'extrémité expéditeur chiffre le contenu du message à l'aide de la clé publique du certificat de chiffrement du point d'extrémité destinataire (voir 6.5.3 pour l'algorithme). Par conséquent, seul le point d'extrémité destinataire peut lire ce contenu (Figure 20).

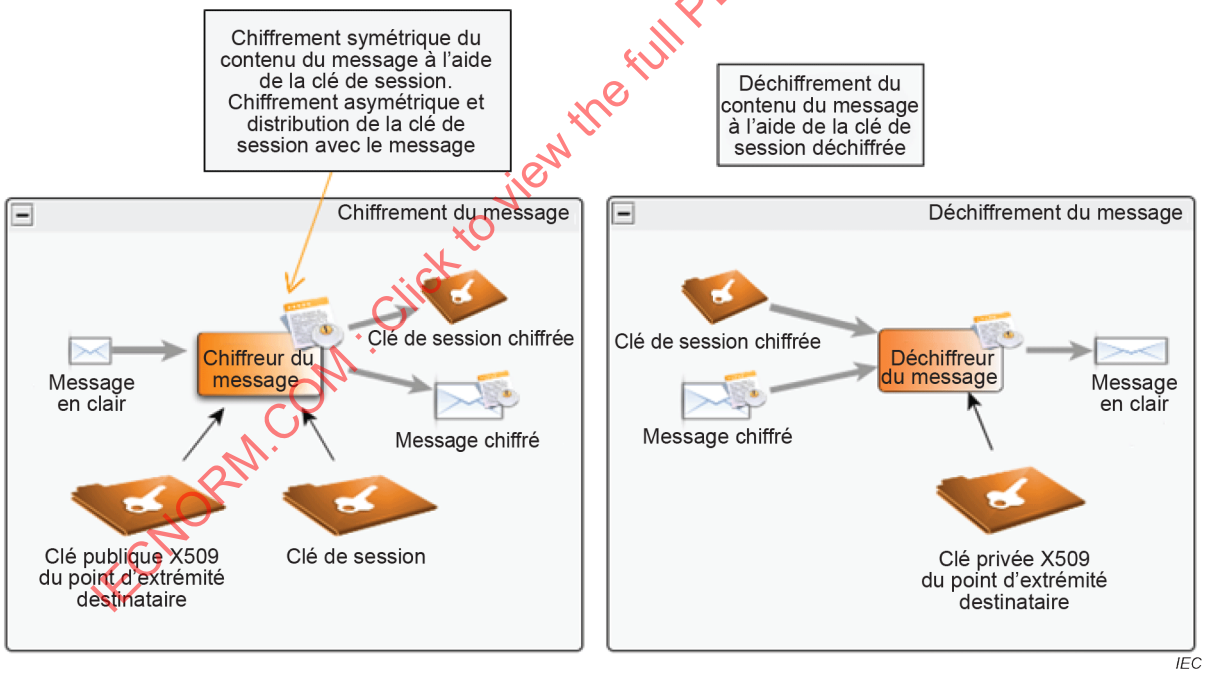



**Figure 20 – Chiffrement et déchiffrement de message**

Le point d'extrémité de l'expéditeur chiffre le document de l'application, c'est-à-dire le contenu du message, à l'aide d'une clé de session générée aléatoirement. Cette clé est alors chiffrée à l'aide de la clé publique du certificat de chiffrement du point d'extrémité destinataire. La clé chiffrée est incluse dans le message et acheminée vers le point d'extrémité destinataire.

Ce dernier déchiffre la clé contenue dans le message à l'aide de la clé privée de son certificat de chiffrement, puis l'utilise pour déchiffrer le contenu du message.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 4.5.4 Non-répudiation

La Figure 21 représente le processus de signature d'un message et de son accusé de réception correspondant afin de permettre à chaque entité homologue de prouver que l'autre entité a envoyé soit le message, soit l'accusé de réception.

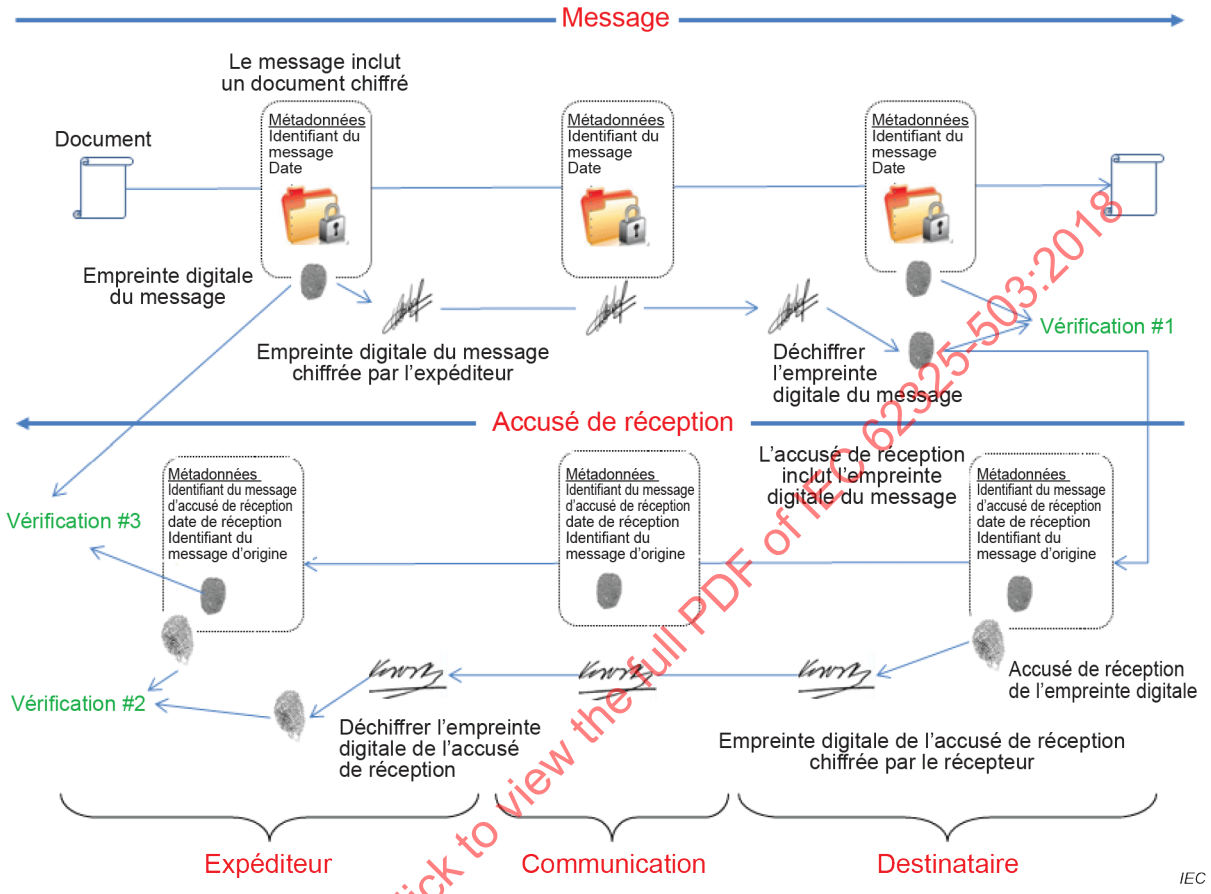



**Figure 21 – Non-répudiation**

- Livraison du message:

  Un message intègre le document de l'expéditeur. L'empreinte digitale du message identifie de façon unique le document avec ses métadonnées, telles que l'identifiant unique du message (*Msg ID* ou *ID du message*), ainsi que la date et l'heure d'envoi (*Date*).

  L'empreinte digitale et le document de l'expéditeur sont tous deux chiffrés et livrés au destinataire. L'empreinte digitale est chiffrée de sorte que toute personne peut identifier de façon unique l'expéditeur (*signature*). Le document est chiffré de sorte que seul le destinataire peut le lire (*chiffrement*).

  Le destinataire déchiffre l'empreinte digitale et le document. Il vérifie (*check #1*) que l'empreinte digitale qu'il peut calculer à partir du message reçu correspond à l'empreinte digitale signée acheminée (*vérification de signature*). L'ensemble constitué du message chiffré, de la signature du message (c'est-à-dire l'empreinte digitale chiffrée) et du certificat de signature de l'expéditeur atteste que l'expéditeur a émis le message.

- Accusé de réception:

  Le destinataire renvoie l'accusé de réception à l'expéditeur, en utilisant un processus analogue. L'accusé de réception n'intègre pas de document, mais l'identifiant unique du

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

message d'origine (*Original Msg ID ou identifiant du message d'origine*), ainsi que son empreinte digitale[6]. L'accusé de réception est signé, mais non chiffré.

Lorsque l'expéditeur reçoit l'accusé de réception, il identifie (*check #2*) l'entité qui l'a envoyée (*vérification de signature*). Il vérifie également que l'empreinte digitale du document ayant fait l'objet d'un accusé de réception équivaut à l'empreinte digitale du message d'origine (*check #3*).

L'ensemble constitué du message d'origine, de l'accusé de réception signé et du certificat de signature du destinataire atteste que le destinataire a reçu le document.

NOTE Pour disposer d'un ensemble de preuves indépendant ne nécessitant pas le certificat de chiffrement du destinataire, la signature doit précéder le chiffrement.

# 5 Livraison des messages

## 5.1 Identification unique des composants et des messages

Chaque composant, qu'il s'agisse d'un point d'extrémité, d'un courtier ou d'un annuaire des composants, doit avoir un identifiant (ID) unique dans le système. Le programme d'identification est une question relevant de la gouvernance du système.

- L' «identifiant du composant» (appelé également «code du composant») sert à identifier le composant lors d'un échange avec les autres composants.
- Une application utilise l'identifiant d'un point d'extrémité pour identifier le destinataire lors de l'envoi d'un document. Inversement, une application reçoit toujours un document associé à l'identifiant du point d'extrémité expéditeur.
- Les identifiants des points d'extrémité expéditeur et destinataire sont inclus dans les métadonnées de chaque message.

Lorsqu'un point d'extrémité génère un message interne (message ou accusé de réception), il doit l'identifier à l'aide d'un UUID (identifiant unique universel ou *Universal Unique Identifier* en anglais), tel que défini dans l'IETF RFC 4122.

NOTE Lors de la livraison d'un document, un point d'extrémité destinataire fournit à l'application destinatrice une identité garantie (c'est-à-dire authentifiée) de l'expéditeur, à savoir l'identifiant du composant point d'extrémité expéditeur. Toutefois, l'identité de l'expéditeur est souvent incluse également dans le document lui-même, et la vérification de la correspondance des deux identités incombe à l'application qui analyse le document.

## 5.2 Attribut «type de message» (message-type) d'un message

Un système MADES peut prendre en charge plusieurs processus métiers simultanés.

Une partie "P" dont un point d'extrémité est connecté au système peut utiliser plusieurs applications, en mettant en œuvre des fonctions venant à l'appui des activités internes et des échanges avec les autres parties conformément aux rôles qu'elle exerce dans les processus métiers.

Les applications demandent au point d'extrémité d'envoyer des documents aux autres parties. Le point d'extrémité prend en charge des demandes simultanées des applications.

Les autres parties, tout en remplissant leurs propres rôles dans ces processus métiers, peuvent également envoyer certains documents à la partie "P". Pour une répartition correcte des documents reçus entre les applications, chaque message contient un type de message.

Le type de message est une information textuelle obligatoire fournie par une application expéditrice, intégrée aux métadonnées du message, acheminée jusqu'au point d'extrémité

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

6 L'accusé de réception désigné ici est le DELIVERY_ACKNOWLEDGEMENT (voir 5.6.2).

destinataire, et utilisée par une application destinatrice pour récupérer uniquement les documents qui l'intéresse.

Chaque partie peut librement organiser les activités, fonctions et applications dans son système d'information d'une manière transparente pour les autres parties. Toutefois, toutes les parties doivent convenir des types de messages comme partie intégrante de la conception globale des échanges d'informations.

NOTE Les types de messages sont comparables aux numéros de ports: des applications concurrentes dans un ordinateur utilisent différents numéros de ports pour un acheminement correct des données reçues. L'utilisation des numéros de ports est largement partagée (par exemple, 21:FTP, 22:SSH, 25:SMTP), mais ne fait pas partie intégrante du protocole IP qui accepte tout entier, quel qu'il soit.

### 5.3 Parcours d'un message vers le point d'extrémité d'un destinataire: chemins de message

Chaque administrateur de point d'extrémité doit être capable de configurer une liste de chemins de messages qui décrivent de quelle manière les points d'extrémité homologues envoient des messages à son propre point d'extrémité. La configuration signifie la création et la maintenance d'un ensemble de données transmis à l'annuaire de référence du point d'extrémité, puis diffusé à tous les composants du système via le processus de synchronisation entre annuaires.

Les attributs d'un chemin de message sont les suivants:

- Chemin (Path) – Parcours de livraison des messages au point d'extrémité:

  "DIRECT" ou "INDIRECT:<broker code>"
- Type de message (Message-type) – Texte, comportant éventuellement un caractère générique (*) à la fin (par exemple, ABC*), qui indique les types de messages qui peuvent utiliser le chemin.
- Expéditeurs (Senders) – Ensemble de points d'extrémité homologues, c'est-à-dire une liste de codes de points d'extrémité qui peuvent utiliser le chemin. Un caractère générique (*) désigne n'importe quel point d'extrémité.
- De (From) – heure (inclusive) à partir de laquelle le chemin peut être utilisé.
- À (To) – Heure (exclusive) jusqu'à laquelle le chemin peut être utilisé; pas d'heure limite lorsqu'elle n'est pas définie.

Il peut exister plusieurs chemins de messages avec le même type de messages et des périodes d'utilisation non chevauchantes[7]. La période d'utilisation peut servir à informer à l'avance l'ensemble des points d'extrémité du système de la planification d'une modification d'un parcours de livraison.

Pour envoyer un message avec le type de message $B$ à un point d'extrémité destinataire $R$ à l'heure $T$, un point d'extrémité expéditeur $S$ doit rechercher un chemin de messages en utilisant l'algorithme suivant:

Soit $P$ un ensemble de chemins de message. $P$ est l'ensemble de départ de tous les chemins de messages du point d'extrémité destinataire $R$. Chaque étape de l'algorithme exclut certains éléments de l'ensemble $P$.

Étape 1: conserver dans l'ensemble $P$ uniquement les éléments $p$ qui vérifient:

- $p$.From ≤ T < $p$.To: c'est-à-dire utilisable à l'heure T= Maintenant
- $B$ correspond à p.Message-type: c'est-à-dire qu'il correspond au type de message à envoyer;

7 Les dispositions de 7.3 traitent de la cohérence des données de configuration.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- Si $P$ est vide, passer alors à l'étape 4.

Étape 2:

- Soit $Q$ le sous-ensemble de $P$ contenant les éléments $p$ vérifiant $B$=$p$.Message-type (c'est-à-dire correspondance parfaite).
- Si Q n'est pas vide, alors $P := Q$, et passer à l'étape 4.

Étape 3:

- Soit $n$ la valeur maximale de length($p$.Message-type) pour tous les éléments $p$ dans l'ensemble $P$.
- Soit $Q$ le sous-ensemble de $P$ contenant les éléments $p$ vérifiant $n$=length($p$.Message-type).
- $P := Q$ (Noter que $Q$ ne peut pas être vide, puisque $P$ ne l'est pas).

Étape 4: Aucun chemin de message n'est trouvé si l'une des conditions suivantes s'applique:

- $P$ est vide ou contient plus d'un élément.
- Le point d'extrémité destinataire n'est pas dans la liste des entités homologues autorisées, à savoir: $R$ n'est pas dans $p$.Senders où $p$ est l'élément unique dans l'ensemble $P$.
- $p$, l'élément unique dans $P$, décrit un transfert direct vers $R$, mais le point d'extrémité (actuel) expéditeur S n'autorise pas les connexions entrantes, rendant impossible tout retour d'accusé de réception.

Exemple:

- Dans le contexte d'un processus métier (par exemple BP1), un point d'extrémité échange des messages avec les types de messages BP1-A, BP1-B, BP1-C. Toutefois, seuls les points d'extrémité E1 et E2 peuvent envoyer des messages BP1-C et doivent le faire avec un courtier différent.
- Une configuration correcte inclut deux chemins de messages:

  #1: message-type BP1-*: tout point d'extrémité utilise le Broker1 à compter du 01/01/2000 et de façon permanente

  #2: message-type BP1-C: les points d'extrémité E1 et E2 utilisent le Broker2 à compter du 01/01/2000 et de façon permanente
- L'algorithme ne détermine aucun chemin si le point d'extrémité E3 souhaite envoyer un message BP1-C à l'autre point d'extrémité. Effectivement, aucun des types de messages ayant apparemment une plus grande correspondance ne peut s'appliquer, tels que BP1-*, BP1*, BP*, B*voire *, car il existe un chemin de messages BP1-C défini.

### 5.4 Restriction des parcours par un courtier

Chaque administrateur d'un courtier doit être capable de configurer la liste des points d'extrémité autorisés à utiliser le courtier et la liste des types de messages autorisés à transiter par le courtier.

- La configuration signifie la création et la maintenance de données transmises à l'annuaire de référence, puis diffusées à tous les composants du système via le processus de synchronisation entre annuaires.
- Une liste vide de points d'extrémité (de types de messages respectivement) signifie que le courtier admet tout point d'extrémité (type de message respectivement).
- Un type de message peut s'achever par le caractère générique (*) afin de correspondre à un ensemble de types de messages.
- Le courtier rejette toute connexion provenant d'un point d'extrémité non autorisé et rejette tout message interne (message ou accusé de réception) dont le type de messages n'est pas autorisé.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

NOTES:

- Pour qu'un message transite par un courtier, les points d'extrémité expéditeur et destinataire doivent tous deux s'y connecter, le premier afin d'envoyer le message et le second afin d'envoyer l'accusé de réception.
- Une modification des restrictions par l'administrateur d'un courtier ne peut jamais entraîner l'acheminement de certains messages vers d'autres courtiers. En effet, cette modification ne peut que bloquer ou débloquer certains parcours, car l'algorithme d'acheminement (voir 5.3) n'utilise pas les restrictions des courtiers.

## 5.5 Acceptation d'un message par un point d'extrémité expéditeur

Un point d'extrémité expéditeur doit accepter un message provenant d'une application expéditrice uniquement lorsque les conditions suivantes sont satisfaites:

- Le point d'extrémité destinataire existe.
- Il détient un certificat de chiffrement valide.
- Il existe un chemin pour le type de messages vers le point d'extrémité destinataire (résultat de l'algorithme d'acheminement – voir 5.5).
- Dans le cas où le chemin de messages représente un transfert indirect par un courtier intermédiaire, le courtier existe et autorise les points d'extrémité expéditeur et destinataire, ainsi que le type de message (restrictions du courtier – voir 5.4).

## 5.6 Suivi de la livraison d'un message

### 5.6.1 Attribut «état du message» (message-status) d'un message

Le système assure le suivi du processus de livraison de chaque message. L'état de livraison d'un message, appelé ci-après état du message (message-status), indique que le point d'extrémité expéditeur est informé de la livraison. Une application expéditrice peut demander l'état d'un message envoyé[8] récemment, en indiquant l'identifiant retourné par le système lorsque le message y est entré.

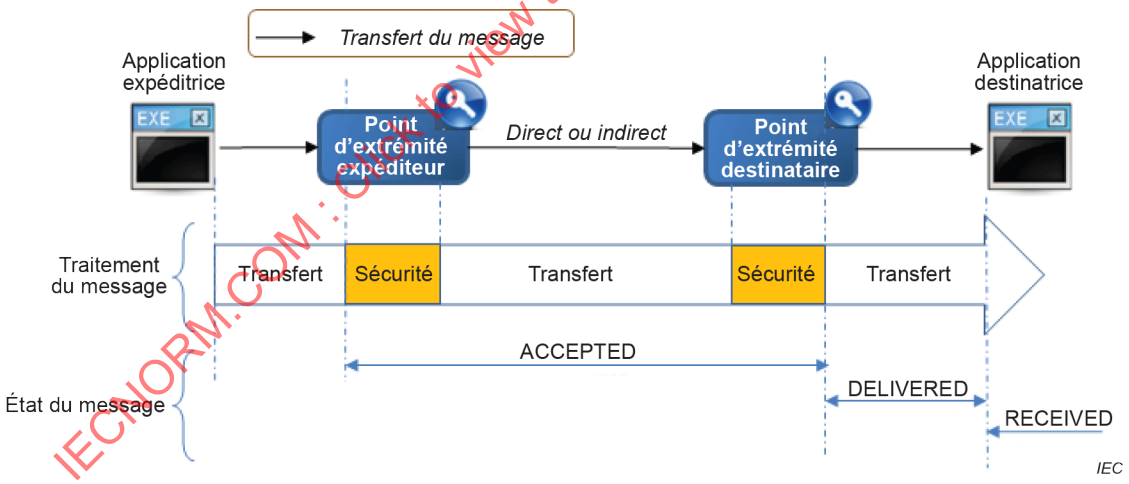



**Figure 22 – État du message tout au long du processus de livraison**

La Figure 22 présente les valeurs de l'état du message tout au long du processus de livraison. Le point d'extrémité expéditeur met en forme le message et effectue les opérations de sécurité, tandis que le point d'extrémité destinataire réalise les opérations inverses.

- ACCEPTED – Le système (en fait le point d'extrémité expéditeur) a accepté la demande d'envoi provenant de l'application expéditrice, a créé une structure de données interne pour le message et l'a archivée en toute sécurité.

8 Voir 5.9 concernant la période accessible passée.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

- DELIVERED – Le point d'extrémité destinataire a reçu le message, qui a subi avec succès les vérifications de sécurité (c'est-à-dire que le contenu est déchiffré et que la signature est vérifiée), et l'a archivé en toute sécurité.
- RECEIVED – Le point d'extrémité destinataire a transféré le message vers une application destinatrice qui a confirmé la réception.
- FAILED – Le système ne peut pas garantir que le point d'extrémité destinataire ou qu'une application destinatrice a reçu le message ou le recevra. Dans certaines situations, l'erreur consignée dans un rapport signifie que le système ne peut pas remettre le message ou ne le remettra pas. Par exemple, le message a expiré.

Un point d'extrémité expéditeur peut (facultatif) différencier avec la valeur DELIVERING (LIVRAISON EFFECTIVE), la partie de l'état du message ACCEPTED entre le transfert technique avec succès du message et la réception de l'accusé de réception.

NOTE La valeur RECEIVED signifie uniquement qu'une application destinatrice a accepté le message d'un point de vue technique. Une analyse ultérieure peut produire un «accusé de réception fonctionnel» indiquant si la partie homologue accepte ou rejette le contenu du message selon les règles commerciales. Cet «accusé de réception fonctionnel» peut alors constituer un nouveau message que le système remet au point d'extrémité expéditeur d'origine.

### 5.6.2 Événements de livraison et accusés de réception

Les points d'extrémité doivent signaler les événements de suivi avec les caractéristiques décrites dans le Tableau 1, et le point d'extrémité expéditeur doit mettre à jour l'état du message en conséquence, tel que représenté à la Figure 23.

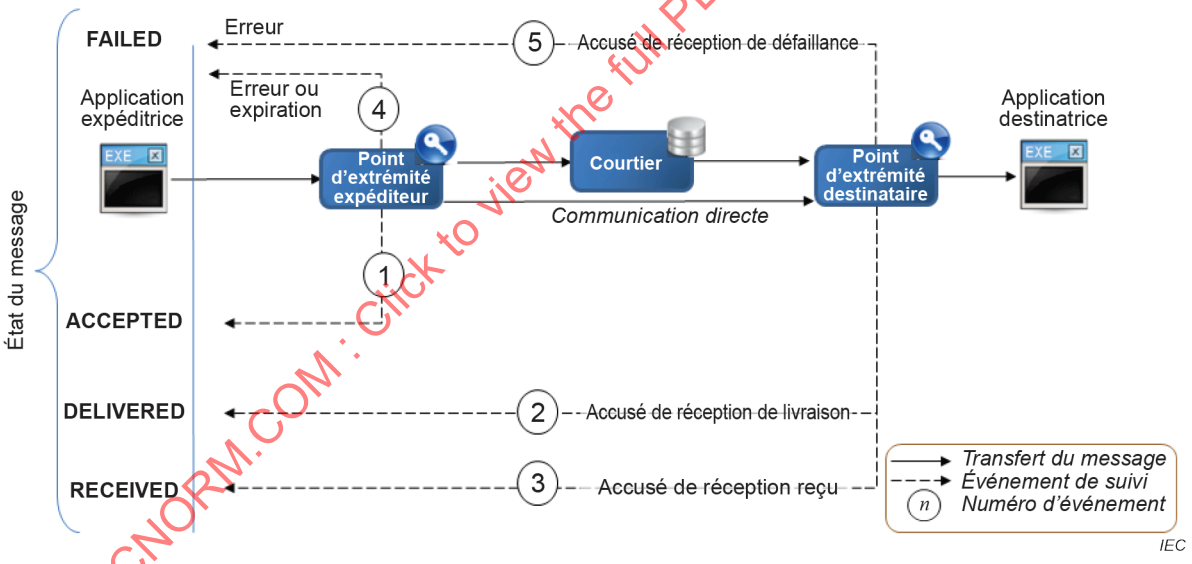



**Figure 23 – Événements de suivi lors de la livraison d'un message**

Seuls les points d'extrémité signalent les événements, un point d'extrémité expéditeur signale un événement par un traitement interne et un point d'extrémité destinataire signale un événement par l'envoi d'un accusé de réception.

Les erreurs qui se produisent lors du transfert d'un message entre un point d'extrémité et un courtier doivent être consignées dans un journal par les deux composants, c'est-à-dire alors que le point d'extrémité expéditeur effectue un téléchargement montant et le point d'extrémité destinataire effectue un téléchargement descendant. Ces erreurs peuvent être signalées soit par le protocole AMQP (voir 6.1.7), soit par le courtier du fait d'une restriction (voir 6.3). Étant donné que les impossibilités d'acheminement sont vérifiées au cours du processus d'acceptation d'un message (voir 5.5), un logiciel stable peut, dans la plupart des situations, réessayer de transférer les messages en attente. Les dispositions de 5.7 décrivent comment signaler l'expiration du message.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

Un accusé de réception:

- signale un événement lié à un message appelé message d'origine (original-message) dont l'ID est inclus dans les métadonnées de l'accusé de réception;
- a le même type de message que le message d'origine;
- n'est jamais chiffré;
- chemine via le parcours inverse du message d'origine, par l'intermédiaire du courtier X s'il a été reçu via ce dernier, directement vers le point d'extrémité expéditeur s'il a été reçu directement.

**Tableau 1 – Caractéristiques des événements de suivi**

| Numéro d'événement | Caractéristiques de l'événement |
|---|---|
| 1 | État du message: ACCEPTED – voir les conditions d'acceptation en 5.5<br>Notification de l'événement: interne au point d'extrémité expéditeur |
| 2 | État du message: DELIVERED – Le point d'extrémité expéditeur doit définir l'état du message sur FAILED et non DELIVERED en cas d'échec de la vérification #3 décrite en 4.5.4.<br>Notification de l'événement: Accusé de réception par le point d'extrémité expéditeur<br>✓ Contenu: Empreinte digitale non chiffrée du message d'origine.<br>✓ Type interne: DELIVERY_ACKNOWLEDGEMENT<br>✓ Signé: **Oui** / Chiffré: Non |
| 3 | État du message: RECEIVED<br>Notification de l'événement: Accusé de réception par le point d'extrémité expéditeur<br>✓ Contenu: non pertinent, mais au moins un caractère.<br>✓ Type interne: RECEIVE_ACKNOWLEDGEMENT<br>✓ Signé: Non / Chiffré: Non |
| 4 | État du message: FAILED<br>Notification de l'événement: interne au point d'extrémité expéditeur |
| 5 | État du message: FAILED<br>Notification de l'événement: Accusé de réception par le point d'extrémité expéditeur<br>✓ Contenu: une description lisible en français de l'erreur constatée ou du motif de la non-livraison du message.<br>✓ Type interne: FAILURE_ACKNOWLEDGEMENT<br>✓ Signé: Non / Chiffré: Non |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

## 5.7 Expiration d'un message

L'expiration d'un message est un mécanisme qui permet de notifier à une application expéditrice la non-réception d'un message par le point d'extrémité destinataire dans les délais impartis.

Une heure d'expiration doit être associée à chaque message interne et gérée comme suit:

- L'administrateur d'un point d'extrémité peut configurer localement les durées maximales de livraison des messages qu'il envoie. La configuration signifie la possibilité d'associer des durées maximales aux types de messages, y compris une valeur par défaut appliquée lorsqu'aucune association n'est définie. Le comptage du temps commence lorsque le point d'extrémité expéditeur accepte le message (voir 5.5).
- L'heure d'expiration d'un accusé de réception est égale à l'heure d'expiration du message d'origine.

- L'heure d'expiration est incluse dans les métadonnées de la structure du message interne (Tableau 11: expirationTime) et dans les sections du message AMQP (Tableau 5: ttl; Tableau 6: absolute-expiry-time).
- Un composant ne transfère pas un message interne expiré à un autre composant ou à une application, et il accepte, puis ignore un message interne s'il a été reçu hors délai.
- Un point d'extrémité expéditeur positionne à FAILED l'état d'un message expiré dont l'état est ACCEPTED, c'est-à-dire qu'il déclare l'échec de la livraison d'un message lorsqu'aucun accusé de réception n'a signalé sa réception par le point d'extrémité destinataire.

NOTE Le point d'extrémité destinataire n'émet pas d'accusé de réception pour signaler l'expiration d'un message. Une valeur par défaut de la durée maximale de livraison de chaque message garantit qu'aucun message ne peut rester indéfiniment en cours de livraison (c'est-à-dire devenir un message «zombie»), quelle qu'en soit la raison.

## 5.8 Transfert fiable d'un message

### 5.8.1 Justifications

Étant donné qu'un message chemine par une chaîne de composants de livraison, chaque composant transfère au composant suivant la responsabilité de l'archivage sécurisé (voir 5.9) et de la livraison du message. La livraison garantie de messages entre des applications homologues repose sur des transferts fiables entre les différents composants. Elle ne repose pas sur une vérification d'exhaustivité de bout en bout, le destinataire demandant à l'expéditeur de renvoyer les messages manquants.

Soit un composant et un message qui y circule. À un moment donné, le composant peut être le seul de la chaîne de livraison possédant le message. Par conséquent, il stocke chaque message en transit dans une mémoire fiable; dans le cas contraire, le message peut être perdu en cas de défaillance.

Considérons à présent une étape de la livraison d'un message, entre un expéditeur et un destinataire. Les deux entités partagent un identifiant du message en transit attribué par l'expéditeur, de sorte que deux messages en transit simultanément entre ces deux entités n'ont jamais le même identifiant. L'identifiant peut être soit un attribut existant du message tel qu'un identifiant du message, soit un identifiant ad hoc dont la portée se limite au transfert entre les deux entités.

Pour ne perdre aucun message, l'expéditeur solde le transfert après confirmation de l'archivage sécurisé du message par le récepteur. Le solde signifie, du point de vue de l'expéditeur, que le transfert du message est achevé, que le message ne fera pas l'objet d'un nouveau transfert, et qu'il l'oublie.

Lors du rétablissement du canal de communication par suite d'une défaillance, l'expéditeur transfère une nouvelle fois les messages non soldés et non expirés parce que le récepteur peut ne pas les avoir reçus ou ne pas les avoir archivés en toute sécurité. Pour éviter toute duplication de message, le récepteur conserve (dans une mémoire fiable) les identifiants des messages reçus récemment. Par conséquent, il peut ignorer un message déjà reçu. Lorsqu'il est signifié que l'expéditeur a soldé un message, le récepteur peut l'oublier: cette opération solde le transfert du point de vue du récepteur.

Un composant ne doit pas attendre que le composant précédent solde un message pour le transférer au composant suivant. Pour un message donné, il est même possible qu'un composant solde son transfert sortant avant son transfert entrant; toutefois, dans ce cas, le composant mémorise son identifiant et surveille les messages entrants pour éliminer les duplications encore possibles. En effet, il ne peut pas solder le transfert entrant tant que l'expéditeur n'a pas d'abord confirmé le solde.

Pour un transfert fiable, l'expéditeur et le récepteur échangent des informations comme représenté à la Figure 24. Chaque entité peut solder un message expiré indépendamment des confirmations de l'autre.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

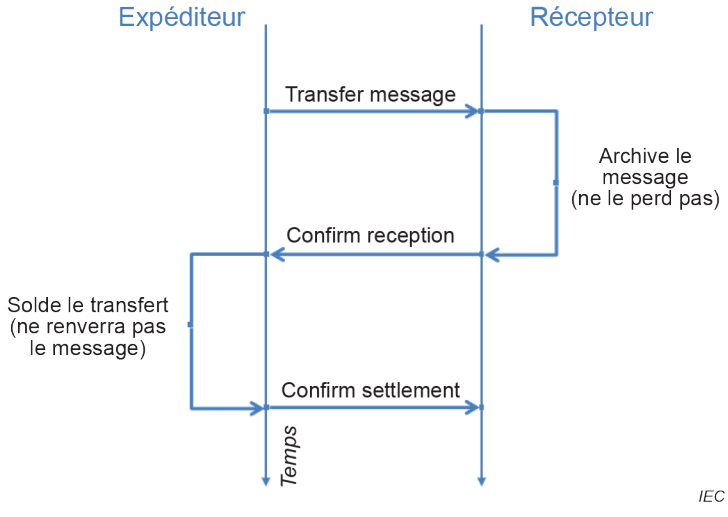



**Figure 24 – Transfert fiable**

Différentes mises en œuvre sont possibles lorsque les entités homologues échangent les états des transferts non soldés, soit de manière individuelle, soit en groupe, lors du flux de messages et lors du rétablissement du canal de communication.

Les Paragraphes 5.8.2 à 5.8.4 indiquent comment MADES définit des transferts fiables au cours de la livraison entre des applications et des composants, comme représenté à la Figure 22.

### 5.8.2 Transfert entre une application expéditrice et un point d'extrémité expéditeur

La communication utilise le protocole HTTPS, l'application expéditrice étant le client HTTP et le récepteur (c'est-à-dire le point d'extrémité expéditeur) étant le serveur HTTP, comme représenté à la Figure 25.

La demande *SendMessage* (voir 11.1.2) correspond au transfert du message de (*transfer message*) et la réponse correspond à la réception de confirmation (*confirm reception*). La demande inclut un paramètre *conversationID*, qui est l'identifiant du message en transit.

Il n'existe aucune *confirmation de solde* (confirm settlement). Toutefois, le récepteur (c'est-à-dire le point d'extrémité expéditeur) doit conserver le message, y compris l'attribut *conversationID*, l'identifiant du message et l'état du message, quelque temps après le succès ou l'échec de la livraison, uniquement ne serait-ce que pour répondre à une demande *CheckMesssageStatus* (voir 5.9):

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

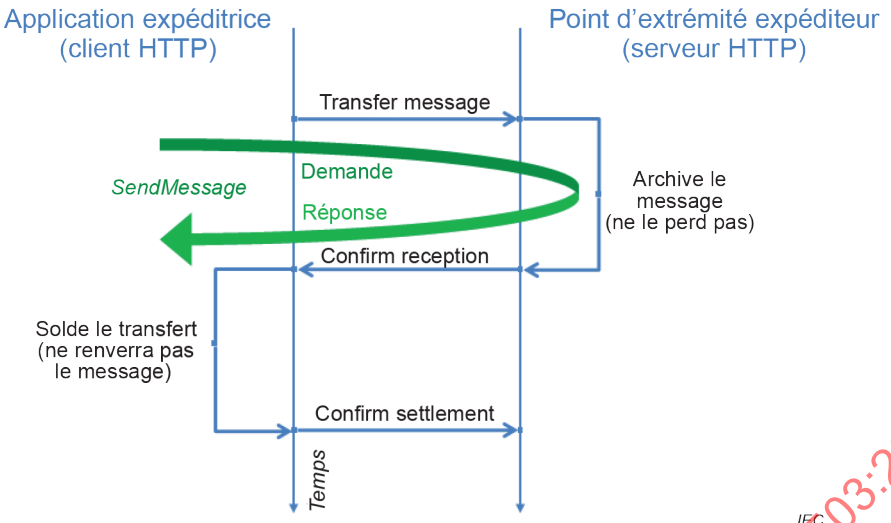



**Figure 25 – Transfert entre l'application expéditrice et le point d'extrémité expéditeur**

### 5.8.3 Transfert entre les composants utilisant le protocole AMQP

Qu'il soit direct (en une étape) ou indirect (en deux étapes), le transfert de messages entre deux composants (point d'extrémité, courtier) est mis en œuvre via le protocole AMQP (voir l'Article 6).

Les trames AMQP échangées entre deux entités homologues AMQP, un expéditeur et un récepteur, sont des trames de TRANSFER destinées à transférer les messages et des trames de DISPOSITION destinées à mettre à jour l'état de transfert (c'est-à-dire des confirmations) – Informations détaillées, voir [AMQP 2.6.12].

L'expéditeur et le récepteur configurent la politique de solde des messages selon une garantie de livraison définie (*at-least-once* (*au moins une fois*) ou exactly-once (*exactement une fois*)), et peuvent solder un transfert en cas d'expiration du message, indépendamment de l'état de transfert de l'autre partie.

Les exigences de MADES concernant la mise en œuvre du protocole AMQP sont spécifiées à l'Article 6. De plus, un point d'extrémité destinataire doit vérifier et arrêter les messages internes dupliqués entrants éventuels sur la base de l'identifiant du message. En effet, le rétablissement total d'un élément de la chaîne de livraison peut ne pas s'effectuer correctement par suite d'un état défectueux. Cet élément peut par conséquent préférer générer un message en double (par exemple, voir 6.1 – rétablissement de liaisons facultatif) qui ne sera pas propagé par le point d'extrémité destinataire.

### 5.8.4 Transfert entre le point d'extrémité destinataire et l'application destinatrice

La communication utilise le protocole HTTPS, le récepteur (c'est-à-dire l'application destinatrice) étant le client HTTP et l'expéditeur (c'est-à-dire le point d'extrémité destinataire) étant le serveur HTTP, comme représenté à la Figure 26.

La réponse *ReceiveMessage* (voir 11.1.3) correspond au *transfert du* message. L'identifiant du message (voir 5.1) identifie le message en transit.

La demande *ConfirmReceiveMessage* (voir 11.1.4) correspond à la *réception de confirmation* et la réponse correspond à la *confirmation de solde*.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

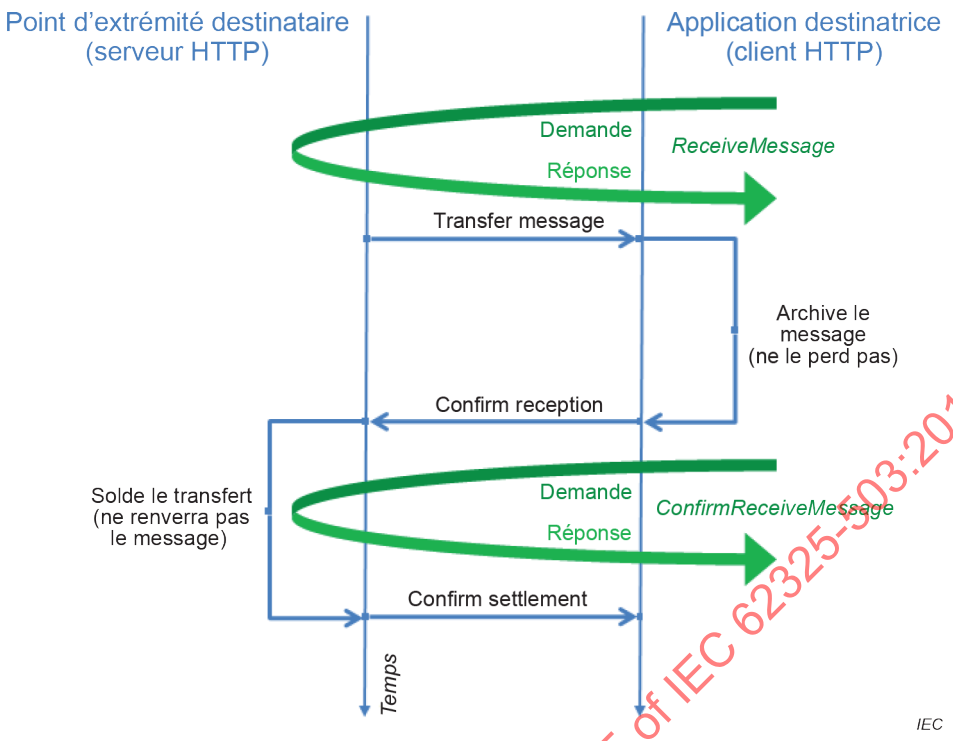



**Figure 26 – Transfert entre le point d'extrémité destinataire et l'application destinatrice**

### 5.9 Archivage des messages internes dans les composants

Un composant (point d'extrémité, courtier) doit mémoriser les messages internes en toute sécurité et de manière provisoire:

- Les données stockées comprennent les informations nécessaires au rétablissement du traitement des messages comme si le composant n'avait jamais été interrompu par une défaillance, à savoir: messages internes (métadonnées et contenu) que le composant génère ou reçoit, événements de suivi (voir 5.6.2) et, le cas échéant, identifiants et états de transfert supplémentaires (voir 5.8).
- Un archivage sécurisé signifie que le composant récupère effectivement des données cohérentes après l'interruption et le redémarrage du traitement, soit sur demande, soit du fait d'une panne d'alimentation. Un stockage physique fiable et une logique d'écriture atomique sont nécessaires à un archivage sécurisé.
- Un point d'extrémité archive le contenu non chiffré des messages.

Quelle durée d'archivage ? Un courtier peut oublier un message interne après le solde de son transfert sortant ou après son expiration. Un point d'extrémité peut, à des fins de livraison du message, supprimer un message archivé dès qu'il atteint l'état final de livraison (tel que défini dans le Tableau 2).

- Toutefois: Une application expéditrice peut demander l'état du message quelque temps après le succès ou l'échec de sa livraison: cette opération exige une durée d'archivage plus longue des informations nécessaires au point d'extrémité expéditeur, notamment des métadonnées, des événements et des accusés de réception.
- Il est nécessaire qu'un point d'extrémité destinataire mémorise les identifiants des messages reçus pour bloquer les messages dupliqués pendant une durée supplémentaire après qu'il les a transférés avec succès à une application destinatrice (voir 5.8.3).
- Le parcours de l'historique récent des messages échangés facilite l'exploitation.

Par conséquent, l'administrateur d'un point d'extrémité doit être capable de configurer les éléments (quoi) (messages envoyés / reçus, métadonnées / contenu) et la durée (combien de

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

temps) d'archivage d'un message par le point d'extrémité après qu'il a atteint l'état final de livraison.

Un point d'extrémité peut de plus archiver les ensembles de preuve de messages tels que définis en 4.5.4.

**Tableau 2 – État final d'un message dans un point d'extrémité**

| Composant | État final d'un message |
|---|---|
| Point d'extrémité expéditeur | L'état du message est RECEIVED ou FAILED. |
| Point d'extrémité destinataire | Le point d'extrémité a envoyé soit un accusé de réception positif, soit un accusé de réception négatif, soit un accusé de réception de traçage du message, ou le message a expiré. |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 5.10 Priorité des messages

La spécification ne définit aucun attribut de priorité d'un message. Les composants (points d'extrémité, courtiers) peuvent hiérarchiser localement par ordre de priorité les messages, par exemple, selon les types de messages.

### 5.11 Ordre de livraison des messages

Par hypothèse, une application $A_1$ a envoyé un message $M_1$ suivi d'un message $M_2$ (même type de message dans les deux cas) à une application $A_2$. La spécification n'exige pas de l'application $A_2$, dans le cas où elle reçoit les deux messages, qu'il convient qu'elle reçoive en premier lieu le message $M_1$, car ceci limiterait la possibilité de traitements concurrents (voir 6.4) et pourrait réduire les performances.

### 5.12 Vérification par essai d'un parcours entre deux points d'extrémité: messages de traçage (*tracing-messages*)

Un message de traçage est un message interne qui vérifie la connectivité de bout en bout entre deux points d'extrémité. Un point d'extrémité expéditeur doit exposer le service *ConnectivityTest* pour envoyer un message de traçage et un point d'extrémité destinataire accuse réception des messages de traçage.

- Une application expéditrice demande un essai de connectivité et indique un point d'extrémité destinataire, ainsi qu'un type de message.
- Le point d'extrémité expéditeur génère et transfère un message de traçage afin de le remettre au point d'extrémité destinataire. Les métadonnées d'un message indiquent qu'il s'agit d'un message de traçage (TRACING_MESSAGE – voir Tableau 12). Le contenu d'un message de traçage n'est pas pertinent, mais ne doit pas être vide.
- Le point d'extrémité expéditeur signe le message et chiffre son contenu. Par conséquent, la livraison avec succès du message comprend la vérification de la configuration des certificats et de leur utilisation.
- Le parcours de livraison dépend du type de message comme tel est le cas pour tout message.
- Un courtier ne différencie pas les messages et les accusés de réception de traçage des autres messages internes.
- À réception du message, le point d'extrémité destinataire envoie un accusé de réception de traçage (TRACING_ACKNOWLEDGEMENT – voir Tableau 12) et ne remet jamais le message à une application.
- Le point d'extrémité expéditeur positionne l'état du message directement sur DELIVERED à réception de l'accusé de réception de traçage correspondant.

- Afin de déterminer si un message de traçage a atteint le point d'extrémité destinataire, l'application expéditrice demande l'état du message, comme pour tout message.

# 6 Transfert de messages via le protocole AMQP

## 6.1 Principes essentiels de la spécification AMQP

### 6.1.1 Introduction

NOTE Les dispositions de 6.1 présentent les principes du protocole AMQP 1.0. Elles ne comportent aucune exigence, mais introduisent les termes et objets utilisés plus loin dans le document. [AMQP xx] référence la section xx de la spécification AMQP (voir l'Article 2).

Le protocole AMQP définit un protocole *filaire* (*wire-level)* pour la *transmission de messages d'activité* (business messaging). Le qualificatif «filaire» signifie que le protocole AMQP décrit les structures des trames d'octets, appelées ci-après *trames*, échangées sur le support de télécommunications entre deux entités homologues [9] TCP. La transmission de messages d'activité désigne le transfert fiable de messages entre les deux entités homologues (applications) dans chaque direction.

- Le protocole AMQP décrit comment les entités homologues établissent tout d'abord une connexion (connection), puis préparent et partagent un contexte d'échange complet utilisant les sessions et les *liaisons* (*links)*, y compris les règles de contrôle de flux et les politiques de livraison convenues, par exemple, «au plus une fois» (at-most-once), «au moins une fois» ou «exactement une fois». Les entités homologues s'échangent ensuite les messages conformément à l'accord.
- Le protocole AMQP permet également de définir des *transactions d'échanges*, c'est-à-dire de regrouper des interactions en une *transaction indivisible* dont le traitement est coordonné, alors qu'elles seraient traitées indépendamment sinon. Une transaction peut être élargie à un nombre arbitraire de transferts répartis sur un nombre indéfini de liaisons distinctes dans chaque direction.

Par conséquent, le protocole AMQP permet tous les comportements classiques de transmission de messages d'activité. Noter que:

- Le protocole AMQP est un protocole symétrique souple par rapport à tout langage de codage.
- Il ne décrit pas comment les entités homologues gèrent et archivent les messages, que ce soit avant ou après le transfert. Les liaisons entre les entités indiquent des *sources* et des *cibles* (targets), qui sont des noms que chaque entité gère comme elle l'entend. Les mises en œuvre évoluées, telles que le service de transmission de messages java (JMS – *java messaging service*), sont associées à un archivage transactionnel sécurisé supplémentaire, à savoir des files d'attente (voir 6.2).

Le protocole AMQP définit neuf trames d'ouverture / fermeture de connexions, de démarrage / fin de sessions et d'établissement / suppressions de liaisons entre entités homologues, de sorte qu'elles puissent s'échanger en toute fiabilité des messages et contrôler le flux d'échanges comme représenté à la Figure 27.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

9 «Protocole de commande de transmission» issu du protocole TCP/IP.

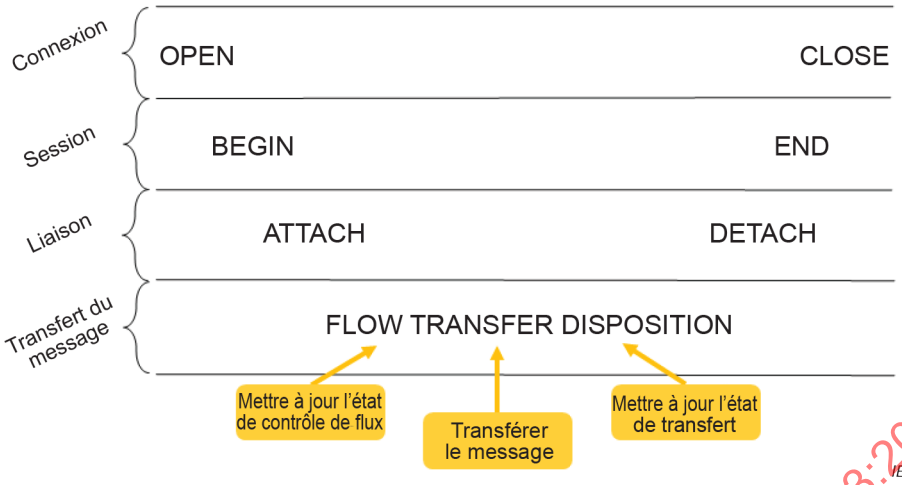



**Figure 27 – Les neuf trames AMQP**

### 6.1.2 Ouvrir une connexion

L'ouverture d'une connexion entre des entités homologues TCP implique les étapes suivantes:

- Échange des en-têtes de version de protocole AMQP [AMQP 2.2].
- Échange des en-têtes de protocole, plus particulièrement:
  - Négociation et déclenchement d'une session TLS [AMQP 5.2].
  - Négociation de la méthode d'authentification et exécution de l'authentification via la SASL (Couche authentification et sécurité simple) [AMQP 5.3].
- Échange de trames OPEN entre les entités homologues [AMQP 2.4.1, 2.7.1].

La négociation de la session TLS peut être réalisée en ligne comme décrit ci-dessus ou bien directement sur la «socket» avant la négociation de la version de protocole AMQP [AMQP 5.2.1]. Cette dernière méthode s'applique sur le port TCP 5671 (c'est-à-dire le service IANA AMQPS), la réalisation en ligne s'applique sur le port 5672.

L'établissement d'une liaison SASL met en œuvre le protocole RFC 4422. L'authentification par nom d'utilisateur et mot de passe utilise le protocole SASL PLAIN. L'authentification par certificat utilise le protocole SASL EXTERNAL, également défini dans le protocole RFC 4422.

Les trames OPEN [AMQP 2.7.1] permettent de négocier le nombre de «canaux» que les deux entités souhaitent créer, ainsi que la taille maximale d'une trame, et par conséquent la taille maximale de la charge utile du message. Elles permettent également de négocier certains paramètres supplémentaires qui servent uniquement pour les piles logicielles AMQP.

La connexion est établie une fois que la partie homologue répond par OPEN. Le protocole [AMQP 2.4.7] décrit le diagramme d'état complet du cycle OPEN/CLOSE. L'établissement d'une session est la seule action autorisée au début d'une connexion.

### 6.1.3 Démarrer une session

[AMQP 2.5, 2.7.2] Les sessions sont des chemins de communication entre les deux entités homologues qui permettent de limiter le flux, sur la base de fenêtres de réception et d'envoi. Le concept de session permet aux récepteurs de gérer le nombre de trames en cours dont le traitement est en attente. Il s'agit communément d'un élément détaillé de mise en œuvre interne aux piles logicielles AMQP.

Pour établir une session, les entités homologues échangent les trames BEGIN sur une connexion existante. Chaque session utilise un «canal» de chaque côté de la connexion, pour

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

former un chemin bidirectionnel. L'établissement d'une liaison est la seule action autorisée au début d'une session.

### 6.1.4 Établissement d'une liaison

Les liaisons sont des chemins de transfert unidirectionnel des messages, établies sur une session [AMQP 2.6]. La communication entre les deux entités homologues est *bidirectionnelle*, signifiant qu'elles échangent mutuellement des états de transfert, mais le flux des messages dans le cadre d'une liaison s'effectue uniquement dans une direction, de l'expéditeur au récepteur.

Chaque partie peut établir (connecter) des liaisons via la trame ATTACH [AMQP 2.7.3] dans chaque direction entre une source et une cible: une liaison sortante pour l'envoi de messages et une liaison entrante pour la réception de messages. La source et la cible font référence aux ressources gérées par chaque entité homologue.

Les liaisons peuvent durer plus longtemps que la connexion sur laquelle elles ont été établies. L'expéditeur et le récepteur peuvent conserver leur état pour rétablir la liaison sur une nouvelle connexion (et session) avec un contrôle précis des garanties de livraison (par exemple, en garantissant la livraison «exactement une fois»).

### 6.1.5 Transfert de messages

Les transferts de messages utilisent trois trames:

- La trame FLOW permet de contrôler le flux, en autorisant ou non le transfert de messages.
- La trame TRANSFER achemine un message de l'expéditeur au récepteur [AMQP 2.7.5]. Un message peut être acheminé par un seul transfert jusqu'à la taille maximale de trame négociée pour la connexion. Les messages plus importants sont répartis entre plusieurs trames TRANSFER regroupées dans une seule transaction.
- La trame DISPOSITION est renvoyée à l'expéditeur par le récepteur en indiquant l'état de traitement du message.

Le modèle de flux repose généralement sur le «crédit de liaison» [AMQP 2.6.7, 2.7.4].

- À réception d'un message, le récepteur peut accorder à l'expéditeur une unité de crédit de liaison. Si un message est disponible, l'expéditeur utilise le crédit de liaison en transférant un message avec la trame TRANSFER. Le récepteur peut également accorder 10, 100 ou un nombre virtuellement non limité de crédits de liaison et l'expéditeur continue d'envoyer des messages disponibles jusqu'à l'épuisement du crédit.
- Une unité de crédit de liaison est utilisée lorsque le message est «soldé» via la trame DISPOSITION [AMQP 2.7.6]. Le solde signifie que le transfert de messages est effectué.

### 6.1.6 Reprise de liaisons et nouveaux envois

En cas de perte d'une connexion, le protocole AMQP est en mesure de recouvrer l'état des liaisons préalablement établies sur une nouvelle connexion et session, et reprend les transferts interrompus et par conséquent non soldés [AMQP 2.6.3/2.6.4].

La reprise de liaisons est *facultative* et n'est souvent pas disponible parce qu'il est difficile de l'exposer dans les API (Interfaces de programmation d'application). La levée d'une exception par suite de la perte d'une liaison/session/connexion et le rétablissement d'une nouvelle connexion/session/liaison sont pratiquement aussi efficaces. Les mises en œuvre JMS ne reprennent en général pas les liaisons.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 6.1.7 Gestion des erreurs

Des erreurs peuvent se produire à tout niveau: connexion [AMQP 2.8.16], session [AMQP 2.8.17], liaison [AMQP 2.8.18] ou transfert, par exemple, rejet du message [AMQP 3.4.3].

### 6.1.8 Structure du message

[AMQP 3.2] «Le terme *message* revêt diverses connotations dans le domaine de la transmission de messages. L'expéditeur peut souhaiter considérer le message comme une charge utile immuable transmise à l'infrastructure de transmission de messages pour livraison. Le récepteur considère souvent le message comme étant non seulement la charge utile immuable de l'expéditeur, mais avec diverses annotations fournies par l'infrastructure de transmission de messages tout au long du processus. Pour éviter toute confusion, le terme *message nu* (*bare message*) est défini comme le message fourni par l'expéditeur et le terme *message annoté* (*annotated message*) est défini comme le message perçu au niveau du récepteur».

- Un *message annoté* se compose du message nu et de sections pour annotation au début et à la fin. Il existe deux classes d'annotations: les annotations qui accompagnent le message indéfiniment et les annotations utilisées par l'élément suivant dans la chaîne de livraison.
- Le *message nu* se compose de trois sections: les propriétés (normalisées), les propriétés d'application (application-properties) et les données d'application (application-data) (le corps du message).

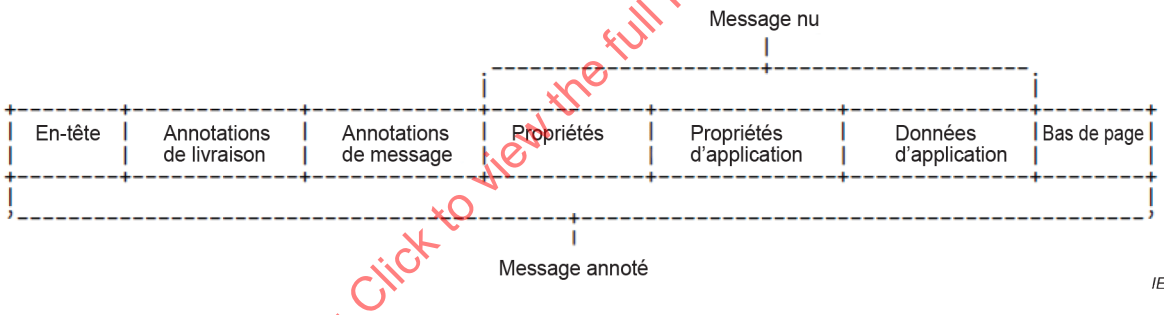



**Figure 28 – Structure d'un message AMQP**

## 6.2 Mise en œuvre évoluée du protocole AMQP: modèle client/courtier

Avertissement: dans l'ensemble du document sauf le présent paragraphe, le terme «courtier» fait référence au composant défini en 4.4.1, c'est-à-dire un «courtier MADES». Dans le présent paragraphe, un courtier représente la mise en œuvre de l'une des entités homologues AMQP avec des caractéristiques supplémentaires et est appelé ci-après «courtier AMQP». Le paragraphe 6.3 explique le mode de rapport entre ces deux courtiers.

Les mises en œuvre évoluées semblent rompre la symétrie du protocole AMQP. En effet, ce sont des mises en œuvre client / serveur dans lesquelles les deux entités homologues, désignées comme client et courtier, ont des rôles différents et mettent en œuvre des fonctions et des services différents:

- Le client ouvre des connexions et démarre des sessions. Le client établit des liaisons sortantes afin de *produire* (produce) des messages, c'est-à-dire envoyer des messages que le courtier archive dans des files d'attente cibles. Le client établit des liaisons entrantes afin de *consommer* (consume) des messages, c'est-à-dire recevoir des messages provenant des files d'attente sources du courtier.
- Le courtier est un serveur qui répond aux demandes du client. Le courtier héberge les files d'attente, c'est-à-dire qu'il réalise l'archivage sécurisé temporaire des messages. Le courtier met en œuvre les connexions et les accès transactionnels aux files d'attente par

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

des clients concurrents[10]. Le courtier transfère automatiquement un message dans une file d'attente à un des clients connecté en tant que consommateur à cette file d'attente[11] et avec le crédit de liaison restant.

En effet, la dissymétrie se produit parce que l'une des entités homologues, le courtier AMQP, met en œuvre des sources et des cibles utilisées comme files d'attente premier entré/premier sorti (FIFO) archivées dans une mémoire disque sécurisée, tandis que l'autre entité homologue, le client AMQP, laisse la mise en œuvre libre et offre des API (par exemple, produire, utiliser un rappel) pour construire des applications.

Une mise en œuvre client / courtier dissimule les informations détaillées AMQP et fournit des services évolués traitant des files d'attente, des consommateurs et des producteurs. De plus, le client peut déclarer et engager des transactions pour solder les transferts, par exemple, pour assurer que le courtier ne supprime pas un message utilisé d'une file d'attente source jusqu'à ce qu'il (c'est-à-dire le client) a traité avec succès ce message reçu.

Le Tableau 3 traduit dans le vocabulaire AMQP les services évolués du modèle client / courtier. Le paragraphe 6.3 décrit comment les composants MADES échangent des messages à l'aide de ces services.

**Tableau 3 – Services du modèle client / courtier**

| Opération déclenchée par le client dans le modèle client / courtier | Correspondance avec la spécification AMQP |
|---|---|
| Le client se connecte au courtier | Le client ouvre une connexion et démarre une session avec le courtier. |
| Le client connecte un consommateur à la file d'attente X. | Le client (récepteur) établit une liaison entrante à une source courtier (expéditeur) X. |
| Le client connecte un producteur à la file d'attente X. | Le client (expéditeur) établit une liaison sortante à une cible courtier (récepteur) X. |
| Le client produit un message à destination de la file d'attente X | Le client (expéditeur) transfère un message au courtier (récepteur) via la liaison sortante établie dont la cible est X. Le courtier archive le message reçu dans la file d'attente X |
| Le client utilise consomme un message de la file d'attente X | Le courtier (expéditeur) transfère un message au client (récepteur) provenant de la file d'attente X, via la liaison établie dont la source est X. |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

## 6.3 Mise en œuvre du protocole AMQP dans les composants MADES

Une communication AMQP entre les composants MADES doit mettre en œuvre le modèle client / courtier comme suit et tel que représenté à la Figure 29.

- Un courtier MADES met en œuvre un courtier AMQP, d'où son nom.
- Un point d'extrémité met en œuvre un client AMQP afin de transférer des messages internes à destination et en provenance des files d'attente du courtier MADES, et de produire des messages internes destinés à d'autres points d'extrémité. Par conséquent, un point d'extrémité met également en œuvre un courtier AMQP afin de recevoir directement les messages internes[12].
- Chaque file d'attente dans un courtier AMQP porte le nom (c'est-à-dire le code) d'un point d'extrémité. Un point d'extrémité consomme uniquement les messages internes d'une file

---

[10] Ceci comprend la possibilité pour un seul client de connecter plusieurs producteurs ou consommateurs à une file d'attente pour tirer profit des possibilités d'exécution concurrentes.

[11] Le courtier «dirige» les données vers le client de façon analogue au protocole websocket, qui utilise une connexion TCP unique et un canal de communication bidirectionnelle simultanée.

[12] Noter le rôle symétrique des points d'extrémité lors de l'échange direct d'un message: les rôles client/courtier se permutent entre le message et l'accusé de réception (tous deux étant des messages internes).

d'attente qui porte son propre nom, c'est-à-dire une file d'attente dans chaque courtier MADES qu'il utilise, et une file d'attente interne pour le transfert direct entrant.

- Un point d'extrémité produit toujours un message interne dans la file d'attente qui porte le nom du point d'extrémité destinataire du message interne.

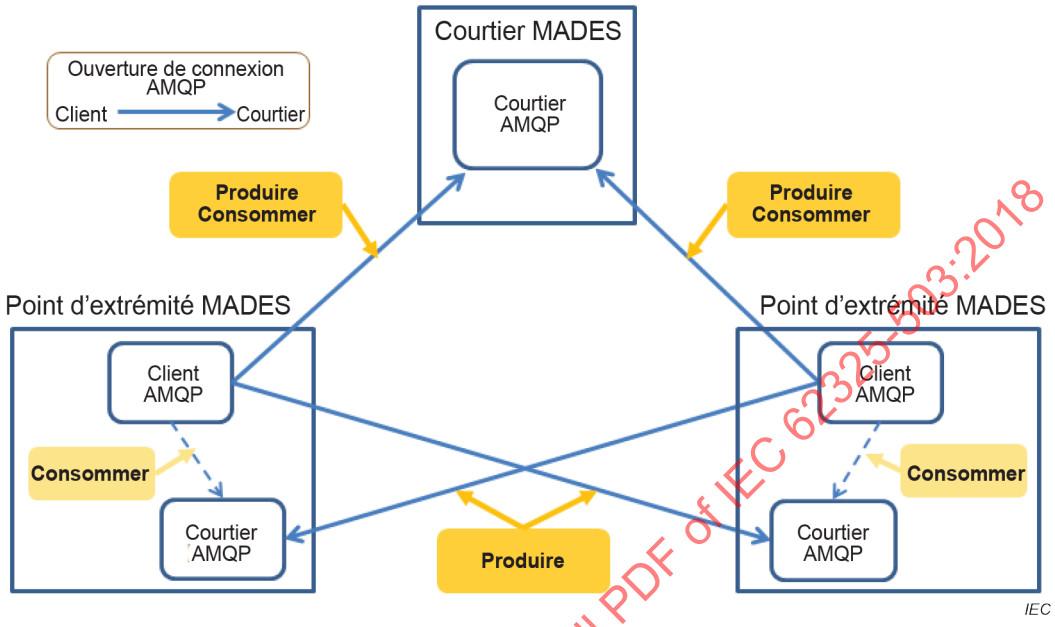



**Figure 29 – Protocole AMQP dans les composants MADES**

La structure AMQP de transfert d'un message interne MADES est définie en 6.5.2 et inclut les champs suivants:

- *senderCode* (propriétés d'application): identifiant du point d'extrémité expéditeur.
- *receiverCode* (propriétés d'application): identifiant du point d'extrémité destinataire.
- *subject* (propriétés): type de message.

Le Tableau 4 définit toutes les conditions qu'un composant MADES doit vérifier lorsqu'il agit en tant que serveur (c'est-à-dire courtier AMQP). Le client est toujours un point d'extrémité. Lorsqu'une condition n'est pas satisfaite, le serveur rejette la connexion, l'établissement de liaison ou le message et, le cas échéant, retourne une erreur AMQP [AMQP 2.8.15] au client.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Tableau 4 – Règles de mise en place d'une connexion/d'un établissement de liaison et règles de transfert de messages**

| Opération | Conditions vérifiées par le composant du serveur pour assurer le succès de l'opération |
|---|---|
| *(toute opération)* | Une opération ne doit jamais aboutir avec succès si le code de composant du client n'existe pas dans les données de configuration au moment de son exécution (par exemple, si le composant a été révoqué – voir 8.6). |
| Connexion | *(point d'extrémité, courtier)*<br>• Un client se connecte via le protocole AMQPS (c'est-à-dire sur le port 5671 avec établissement d'une session TLS préalablement à la négociation de la version du protocole AMQP).<br>• Le client connecté s'authentifie via un certificat valide délivré au point d'extrémité.<br>(*courtier*) le courtier autorise la connexion du point d'extrémité (voir 5.4).<br>(*point d'extrémité*) Le point d'extrémité serveur autorise une connexion entrante directe. |
| Connecter un consommateur | (*courtier*) La file d'attente demandée pour la connexion porte le nom (c'est-à-dire le code) du point d'extrémité client.<br>(*point d'extrémité*) Cette opération ne doit jamais aboutir. |
| Connecter un producteur à la file d'attente X. | (*courtier*) X, la file d'attente demandée pour la connexion porte le nom d'un point d'extrémité valide que le courtier autorise pour la connexion (voir 5.4).<br>(*point d'extrémité*) X, la file d'attente demandée pour la connexion est le code du point d'extrémité serveur. |
| Produire un message interne à destination de la file d'attente X | *(point d'extrémité, courtier)*<br>• Le champ *receiverCode* du message interne transféré est égal à X et est présent dans les données de configuration au moment de l'opération.<br>• Le champ *senderCode* est égal au code de point d'extrémité connecté.<br>(*courtier*) Le champ *subject* du message interne est un type de message que le courtier autorise (voir 5.4). |

La mise en œuvre doit prévoir les moyens de configurer les paramètres AMQP conformément à la gouvernance du système MADES, par exemple, la taille maximale des messages, la politique de livraison "exactement une seule livraison".

La gestion des files d'attente par le courtier peut être:

- *statique:* c'est-à-dire configurée manuellement par l'administrateur du courtier;
- *dynamique*: (recommandée) par exemple, le courtier crée une file d'attente lorsqu'elle est valablement référencée lors de l'établissement d'une liaison (consommateur ou producteur); le courtier supprime la file d'attente lorsqu'elle est vide et qu'elle n'est plus utilisée depuis un certain temps (durée configurable) dans le cadre des liaisons encore établies.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 6.4 Gestion des connexions et des établissements de liaisons AMQP par un point d'extrémité

Il convient qu'un point d'extrémité ouvre au plus une connexion à la fois[13] avec un courtier ou un point d'extrémité homologue.

Un point d'extrémité doit consommer les messages internes, et de la même façon un de ses consommateurs doit être connecté à tout courtier duquel il peut recevoir un message. Les courtiers sont les suivants:

[13] Une par nœud dans le cas d'un groupe.

- Les courtiers auxquels il est fait référence dans l'un des *chemins de message actuellement utilisables (currently usable message-paths)* du point d'extrémité.
- Les courtiers auxquels le point d'extrémité a envoyé récemment un message non expiré et dont l'état est ACCEPTED, car un accusé de réception peut venir de lui.

Pour les petits systèmes dont le trafic de messages est prévisible, les connexions et les établissements de liaisons d'un point d'extrémité peuvent être configurés de manière statique. Par exemple, l'administrateur décrit les composants vers lesquels sont établies des liaisons permanentes, soit au démarrage, soit lors d'une reprise par suite d'une perte de connexion. Toutefois, il convient que le point d'extrémité gère les connexions et les établissements de liaisons de manière dynamique en accord avec les chemins de messages et le trafic des messages sortants.

Afin de tirer profit des possibilités de traitements concurrents par les points d'extrémité et les courtiers, un consommateur (un producteur respectivement) peut effectivement être un groupe de consommateurs (de producteurs respectivement). Par exemple, un groupe peut alors être géré dans son ensemble, tous les membres étant connectés et déconnectés (presque) simultanément. L'administrateur du point d'extrémité peut être capable de définir et de configurer de tels groupes pour des raisons de performances. Par exemple, *3 consommateurs connectés au courtier B7; 3 producteurs connectés au point d'extrémité E3 par l'intermédiaire du courtier B5; 2 producteurs connectés au point d'extrémité E3* (transfert sortant direct); *2 consommateurs connectés à la file d'attente locale* (transfert entrant direct).

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 6.5 Format de message interne

#### 6.5.1 Définitions, conception et vérifications de sécurité

Un message interne est composé de deux parties: contenu et métadonnées.

- Le *contenu* d'un message (normal) est le document immuable qu'une application expéditrice demande au système d'acheminer de façon chiffrée. Les dispositions de 5.6.2 définissent le contenu des accusés de réception.
- Les *métadonnées* d'un message interne constituent un ensemble d'attributs soit fourni par l'application expéditrice, soit créé par le point d'extrémité qui envoie le message interne. Le Tableau 11 décrit la liste complète des attributs des métadonnées, qui inclut (pas de manière exhaustive ici) l'identifiant du message, le type de message, le délai d'expiration, les codes expéditeur et récepteur et, le cas échéant, la signature et les informations concernant le mode de signature et de chiffrement du message (*processingMetadata*).

Le message nu AMQP comprend le contenu et les métadonnées du message interne. Le corps du message AMQP inclut les deux. De plus, les sections propres aux propriétés AMQP incluent en partie des métadonnées (informations détaillées en 6.5.2). Cette redondance a une double origine: le corps du message intègre des métadonnées de rétrocompatibilité[14] au format XML et un courtier utilise uniquement des champs AMQP pour accepter et acheminer le message (voir 6.3, Tableau 4).

#### 6.5.2 Format AMQP de transfert des messages internes

La section décrit le format du message AMQP (voir 6.1.8) auquel un point d'extrémité doit se conformer, lors du transfert d'un message interne (c'est-à-dire message ou accusé de réception) vers un courtier (transfert indirect) ou un point d'extrémité (transfert direct).

Le Tableau 5, le Tableau 6 et le Tableau 7 décrivent respectivement les champs utilisés dans les sections AMQP: *Header* (*en-tête*) *properties* (*propriétés*) et application-properties (*propriétés d'application*). Les couches inférieures de la pile logicielle AMQP peuvent utiliser les autres champs (par exemple, *message-id*) de la section *properties* [AMQP 3.2.4]. Chaque champ de la section *application-properties* porte le nom de l'élément de métadonnées

[14] Avec le projet de spécification technique.

correspondant dans le Tableau 11. Le corps du message (Tableau 8 – section *application-data*) est une *amqp-sequence* à deux éléments [AMQP 3.2.7].

**Tableau 5 – Message interne – Format AMQP: section Header**

| Nom de champ | Type AMQP | Valeur | Exigé |
|---|---|---|---|
| ttl | Milliseconds (millisecondes) | Time-To-Live (durée de vie) (c'est-à-dire la durée de validité du message interne jusqu'à son expiration) | Vrai |

**Tableau 6 – Message interne – Format AMQP: section Properties**

| Nom de champ | Type AMQP | Valeur (métadonnées de message interne) | Exigé |
|---|---|---|---|
| absolute-expiry-time | string (chaîne) | Date et heure d'expiration du message (*expirationTime*). | Vrai |
| subject | string | Type de message interne (*messageType*). | Vrai |
| correlation-id | string | Identifiant du message d'origine lorsque le message interne est un accusé de réception (*relatedMessageID*). | Faux |

**Tableau 7 – Message interne – Format AMQP: section Application-properties**

| Nom de champ | Type AMQP | Valeur | Exigé |
|---|---|---|---|
| messageID | string | Identifiant du message interne attribué par le point d'extrémité. | Vrai |
| receiverCode | string | Code du point d'extrémité qui reçoit le message interne | Vrai |
| senderCode | string | Code du point d'extrémité qui envoie le message interne | Vrai |
| senderApplication | string | Nom affiché de l'application expéditrice (lorsque le message interne est un message). | Faux |
| baMessageID | string | Identifiant du message attribué par l'application (métier) expéditrice (lorsque le message interne est un message). | Faux |
| generated | dateTime | Date et heure de création du message interne par le point d'extrémité expéditeur. | Vrai |
| internalType | string | Type du message interne – voir Tableau 12 (ne pas confondre avec le type de message) | Vrai |
| messageMversion | Integer (entier) | Version MADES de la spécification à laquelle le message interne satisfait – voir 9 | Vrai |

**Tableau 8 – Message interne – Format AMQP: section Application-data**

| Nom de champ | Type AMQP | Valeur | Exigé |
|---|---|---|---|
| Element #1 | string | Chaîne XML des métadonnées du message conforme à l'élément *messageMetadata* du schéma XML donné en 6.5.5. | Vrai |
| Element #2 | binary (binaire) | *Message* → Résultat du chiffrement du contenu fourni par l'application expéditrice (voir Tableau 29)<br>Accusé de réception, message de traçage → Message à texte clair. | Vrai |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 6.5.3 Chiffrement

Un point d'extrémité expéditeur doit chiffrer le contenu de chaque message au moyen du certificat de chiffrement du point d'extrémité destinataire (voir 8.1 et 4.5.3 pour les principes de chiffrement). Les accusés de réception ne doivent pas être chiffrés.

Le chiffrement signifie que:

- le contenu du message (*application-data*, élément #2) est chiffré;
- l'élément *processingMetadata* (voir Tableau 11) inclut un élément *messageProcessor*, dont le *processorID* vaut «encryption» (en anglais) et dont les *processorData* décrivent l'algorithmique de chiffrement du contenu du message.

D'un point de vue technique, les *processorData* constituent un ensemble d'attributs, chacun de ces attributs {name, type, value} étant un triplé. L'attribut appelé "Cipher» («Chiffre») est obligatoire, qui indique la technique de chiffrement utilisée.

La spécification actuelle décrit uniquement une technique de chiffrement appelée 'AES-256' comme suit:

- Un générateur cryptographique à l'état de la technique produit une clé aléatoire de 256 bits. Le contenu du message est chiffré à l'aide de l'algorithme de «norme de chiffrement avancé» (AES – *advanced encryption standard*) et de la clé générée.
- La clé est chiffrée à l'aide de l'algorithme RSA (Rivest, Shamir, Adleman) et stockée dans l'attribut "Session key'' («Clé de session»).
- L'identifiant du certificat de chiffrement du point d'extrémité destinataire dont la clé publique est utilisée par l'algorithme RSA est stocké dans l'attribut "Certificate ID".
- Le Tableau 9 présente l'ensemble des attributs *processorData* obtenus.

**Tableau 9 – Chiffrement – Attributs de métadonnées de traitement pour le chiffre 'AES-256'**

| Nom d'attribut de métadonnées | Type de métadonnées | Description |
|---|---|---|
| Cipher | STRING | "AES-256" |
| Certificate ID | STRING | L'identifiant du certificat de chiffrement du point d'extrémité destinataire. |
| Session key | BYTE_ARRAY | Clé de 256 bits à chiffrement RSA utilisée pour chiffrer le contenu du message. |

NOTE D'autres valeurs ''Cipher'' peuvent être définies en utilisant d'autres longueurs de clé et/ou algorithmes cryptographiques (par exemple, un nouvel algorithme plus résistant aux attaques). La gouvernance du système définit quel «algorithme» utiliser.

À réception d'un message interne, le point d'extrémité destinataire doit vérifier s'il est chiffré (c'est-à-dire si les *processingMetadata* du message contiennent un *messageProcessor* de «chiffrement»). Si tel est le cas et si la valeur contenue dans l'attribut «Cipher» est 'AES-256', alors le point d'extrémité doit:

1) Vérifier que l'identifiant de certificat utilisé pour chiffrer le message interne est son propre certificat de chiffrement.
2) Vérifier que le certificat de chiffrement était valide au moment de la production du message interne: le délai d'expiration du certificat est un attribut du certificat et le délai généré d'un message interne fait partie intégrante des métadonnées du message.
3) Déchiffrer la clé de session à l'aide de l'algorithme RSA et de la clé privée du certificat.
4) Déchiffrer le contenu du message à l'aide de la clé de session déchiffrée et de l'algorithme AES.

### 6.5.4 Signature

Un point d'extrémité expéditeur doit signer chaque message (voir 8.1 et 4.5.3 pour les principes de signature) à l'aide de son certificat de signature. Le point d'extrémité destinataire doit signer l'accusé de réception de livraison (voir 5.6.2). Le programme utilisé pour calculer l'empreinte digitale est le suivant:

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

content + *baMessageID* + *extension* + *generated* + *internalType* + *messageID* + *relatedMessageID* + *receiverCode* + *senderCode* + *senderApplication* + *messageType*.

Où:

- "content" («contenu») est le contenu *non chiffré* fourni par l'application expéditrice (voir Tableau 29).
- Les noms en italique font référence aux éléments des métadonnées d'un message interne tels que décrits dans le Tableau 11;
- "+" désigne la concaténation binaire des attributs codés en UTF-8.

La signature signifie que l'élément *processingMetadata* (voir Tableau 11) inclut un élément *messageProcessor*, dont le *processorID* vaut «signature» et dont les *processorData* décrivent le mode de traitement algorithmique de la signature. D'un point de vue technique, les *processorData* constituent un ensemble d'attributs, chacun de ces attributs {name, type, value} étant un triplé. L'attribut appelé "Algorithm" («Algorithme») est obligatoire, qui indique la technique de signature utilisée.

La spécification actuelle décrit uniquement une technique de signature appelée 'SHA-512' comme suit:

- Le hachage du message (c'est-à-dire l'empreinte digitale) est calculé avec l'algorithme de hachage sécurisé ("secure hash algorithm") SHA-512.
- Le hachage du message est alors chiffré à l'aide de l'algorithme RSA. L'identifiant du certificat de signature du point d'extrémité signataire dont la clé privée est utilisée par l'algorithme RSA est stocké dans l'attribut «Certificat ID».
- Une signature XML est créée, conforme à la recommandation W3C («Norme de syntaxe et de traitement de signature XML») ('Signature Syntax and Processing standard' *(Voir exemple en 6.5.6),* et stockée dans l'attribut «Signature». La signature XML inclut le hachage du message chiffré.
- Le Tableau 10 présente l'ensemble des attributs *processorData* obtenus.

**Tableau 10 – Signature – Attributs de métadonnées de traitement pour l'algorithme 'SHA-512'**

| Nom d'attribut de métadonnées | Type de métadonnées | valeur |
|---|---|---|
| Algorithm | STRING | "SHA-512" |
| Certificate ID | STRING | Identifiant du certificat de signature du point d'extrémité qui a envoyé et signé le message interne. |
| Signature | STRING | Signature XML conforme à la «*norme de syntaxe et de traitement de signature XML*» intégrée ici en tant que chaîne. |

NOTE D'autres valeurs "Algorithm" peuvent être définies.

À réception d'un message interne, un point d'extrémité doit vérifier s'il est signé (c'est-à-dire si les *processingMetadata* du message contiennent un *messageProcessor* de «signature»). Si tel est le cas et si la valeur "Algorithm" est 'SHA-512', alors le point d'extrémité doit vérifier la signature comme suit:

1) Vérifier que le certificat de signature appartient au point d'extrémité qui envoie le message.
2) Vérifier que le certificat de signature était valide au moment de la création du message interne (le délai d'expiration du certificat est un attribut du certificat). Le délai produit d'un message fait partie intégrante des métadonnées du message).
3) Calculer l'empreinte digitale du message interne reçu à l'aide de l'algorithme.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

4) Déchiffrer le hachage du message chiffré inclus dans la signature XML à l'aide de la clé publique du certificat de signature.

5) Vérifier que les empreintes digitales, résultats des opérations 3 et 4, sont identiques.

### 6.5.5 Métadonnées de message interne

Le Tableau 11 énumère les éléments de métadonnées d'un message interne.

**Tableau 11 – Métadonnées de message (type)**

| Nom d'élément | Type d'élément | Description | Exigé | O[15] |
|---|---|---|---|---|
| messageID | string | Identifiant du message attribué par le point d'extrémité qui crée le message interne – voir 5.1. | Vrai | E |
| receiverCode | string | *Message, Message de traçage* → Identifiant de composant du point d'extrémité destinataire.<br>*Accusé de réception* → *SenderCode* du message d'origine. | Vrai | A<br>E |
| messageType | string | *Message, Message de traçage* → Type de message fourni par l'application expéditrice – voir 5.2.<br>*Accusé de réception* → – Type du message d'origine. | Vrai | A<br>E |
| extension | string | *Message* → L'extension de fichier du document – utilisée lorsque l'application expéditrice ou le destinataire utilise l'interface FSSF – voir 11.1.7.<br>*Accusé de réception, Message de traçage* → Non utilisé. | Faux | A<br>∅ |
| generated | dateTime | Date et heure de création du message interne par le point d'extrémité. | Vrai | E |
| expirationTime | dateTime | *Message, Message de traçage* → Date et heure d'expiration du message – définies par le point d'extrémité expéditeur lors de l'acceptation du message – voir 5.7.<br>*Accusé de réception* → *expirationTime* du message d'origine. | Vrai | E<br>E |
| senderCode | string | Identifiant de composant du point d'extrémité qui a créé le message interne – voir 5.1. | Vrai | E |
| internalType | InternalMessageType (voir Tableau 12) | Type technique du message. | Vrai | E |
| relatedMessageID | string | *Message, Message de traçage* → Non utilisé.<br>*Accusé de réception* → Identifiant du message d'origine. | Faux | E |
| senderApplication | string | *Message, Message de traçage* → Identifiant de l'application qui envoie le document.<br>*Accusé de réception* → Non utilisé. | Faux | A<br>∅ |
| baMessageID | string | *Message, Message de traçage* → Identifiant du document fourni par l'application (métier) expéditeur.<br>*Accusé de réception* → Non utilisé. | Faux | A<br>∅ |
| processingMetadata | ProcessingMetadata (voir Tableau 13) | Ajout des métadonnées pour décrire le mode de signature et/ou chiffrement du message interne | Faux | E |
| messageMversion | Integer | Version MADES avec laquelle le message est conforme – voir Article 9. | Vrai | E |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

15 La colonne O signifie «Origine» et indique la provenance de la valeur de l'élément, A: application expéditrice (voir 11.1.2), E: point d'extrémité expéditeur. ∅: valeur nulle ou élément vide de manière équivalente.

**Tableau 12 – InternalMessageType (type: énumération de chaînes)**

| Valeur de chaîne | Description |
|---|---|
| STANDARD_MESSAGE | Message, mais non message de traçage. |
| DELIVERY_ACKNOWLEDGEMENT | Accusé de réception signifiant l'acceptation par le point d'extrémité destinataire du STANDARD_MESSAGE d'origine. – voir 5.6.2. |
| RECEIVE_ACKNOWLEDGEMENT | Accusé de réception signifiant le transfert par le point d'extrémité destinataire du STANDARD_MESSAGE d'origine à une application destinatrice. – voir 5.6.2. |
| FAILURE_ACKNOWLEDGEMENT | Accusé de réception d'une défaillance. – voir 5.6.2. |
| TRACING_MESSAGE | Message de traçage – voir 5.12. |
| TRACING_ACKNOWLEDGEMENT | Accusé de réception signifiant l'acceptation par le point d'extrémité destinataire du TRACING_MESSAGE d'origine – voir 5.12. |

**Tableau 13 – ProcessingMetadata (type)**

| Nom d'élément | Type d'élément (voir Tableau 14) | Description | Exigé |
|---|---|---|---|
| messageProcessors | MessageProcessor[] | Ensemble de métadonnées, chaque métadonnée provenant d'un processeur de message utilisé | Faux |

**Tableau 14 – MessageProcessor (type)**

| Nom d'élément | Type d'élément | Description | Exigé |
|---|---|---|---|
| processorID | string | Identifiant unique du processeur de message «signature» ou «chiffrement» | Vrai |
| processorData | Map (correspondance) (voir Tableau 15) | Ensemble de triplés: {key, type, value} ({clé, type, valeur}) | Vrai |

**Tableau 15 – Map (type)**

| Nom d'élément | Type d'élément (voir Tableau 17) | Description | Exigé |
|---|---|---|---|
| entries (entrées) | MapEntry[] | Ensemble de données, chaque donnée étant fournie avec un nom et une valeur, c'est-à-dire un ensemble constitué d'une clé (nom), d'un type (format) et d'une valeur (selon le type). | Faux |

**Tableau 16 – MapEntry (type)**

| Nom d'élément | Type d'élément | Description | Exigé |
|---|---|---|---|
| clé (key) | string | Nom des métadonnées | Vrai |
| type | ValueType (voir Tableau 17) | Type/format des métadonnées. | Vrai |
| Value (valeur) | string | Valeur des métadonnées. | Vrai |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

**Tableau 17 – ValueType (type: énumération de chaînes)**

| Valeur de chaîne | Description |
|---|---|
| STRING | Chaîne |
| LONG | Numéro à 64 bits exprimé sous forme de chaîne. Exemple: le numéro 42 est représenté comme étant la chaîne 42 (sans guillemets) |
| BYTE_ARRAY | ⇔ type base64Binary |
| BOOLEAN | Chaîne égale à «vraie» ou «fausse». |

## Schéma XML des métadonnées de message interne

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mades.entsoe.eu/internalMessaging"
   xmlns:mades="http://mades.entsoe.eu/internalMessaging">

 <xsd:simpleType name="InternalMessageType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="STANDARD_MESSAGE"/>
   <xsd:enumeration value="DELIVERY_ACKNOWLEDGEMENT"/>
   <xsd:enumeration value="RECEIVE_ACKNOWLEDGEMENT"/>
   <xsd:enumeration value="FAILURE_ACKNOWLEDGEMENT"/>
   <xsd:enumeration value="TRACING_MESSAGE"/>
   <xsd:enumeration value="TRACING_ACKNOWLEDGEMENT"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="ValueType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="STRING"/>
   <xsd:enumeration value="LONG"/>
   <xsd:enumeration value="BYTE_ARRAY"/>
   <xsd:enumeration value="BOOLEAN"/>
   </xsd:enumeration>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:complexType name="Entry">
  <xsd:sequence>
   <xsd:element name="key" type="xsd:string"/>
   <xsd:element name="type" type="mades:ValueType"/>
   <xsd:element name="value" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="Entries">
  <xsd:sequence>
   <xsd:element maxOccurs="unbounded" minOccurs="0" name="entry" type="mades:Entry"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="Map">
  <xsd:sequence>
   <xsd:element name="entries" type="mades:Entries"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MessageProcessor">
  <xsd:sequence>
   <xsd:element name="processorID" type="xsd:string"/>
   <xsd:element name="processorData" type="mades:Map"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MessageProcessors">
```

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

```
  <xsd:sequence>
   <xsd:element maxOccurs="unbounded" minOccurs="0" name="messageProcessor"
type="mades:MessageProcessor"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MessageProcessingMetadata">
  <xsd:sequence>
   <xsd:element name="messageProcessors" type="mades:MessageProcessors">
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="MessageMetadata">
  <xsd:sequence>
   <xsd:element name="messageID" type="xsd:string"/>
   <xsd:element name="receiverCode" type="xsd:string"/>
   <xsd:element name="messageType" type="xsd:string"/>
   <xsd:element minOccurs="0" name="extension" nillable="true" type="xsd:string"/>
   <xsd:element name="generated" type="xsd:dateTime"/>
   <xsd:element minOccurs="0" name="expirationTime" nillable="true"
type="xsd:dateTime"/>
   <xsd:element name="senderCode" type="xsd:string"/>
   <xsd:element name="internalType" type="mades:InternalMessageType"/>
   <xsd:element minOccurs="0" name="relatedMessageID" nillable="true"
type="xsd:string"/>
   <xsd:element minOccurs="0" name="senderApplication" nillable="true"
type="xsd:string"/>
   <xsd:element minOccurs="0" name="baMessageID" nillable="true" type="xsd:string"/>
   <xsd:element name="processingMetadata" type="mades:MessageProcessingMetadata"/>
   <xsd:element name="messageMversion" type="xsd:int"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:element name="messageMetadata" type="mades:MessageMetadata"/>

</xsd:schema>
```

### 6.5.6 Exemple de signature XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha512"/>
    <Reference URI="">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha512"/>
      <DigestValue>eVpInNsCIWzEjdrxxvongO2rnQ4=</DigestValue>
    </Reference>
  </SignedInfo>
    <SignatureValue>aD9HNiTmVxW+HnDOpSjzwDB+MypGTC7yb3/HUpAZKmEhRwQC0eBwYcZSRTqF8VdzmneH6abq2P+m
vHNXPC53i3mF58XDR5JFHHWLHq8B9HZm6/IYxcNy2cGW9yAVyQKe3uJXeV/95u9qMEwJhbOjvPIx
ZdbXqcCSorWqih7hdB86Nv2SIBfXMvWdIinwZfU/44RUptNyxQpP/Pw91Dd8YnMTNVwm2ax5oL1W
akIGKToS/yYid/Cgyb1xhGdfXEp30bqLusaLMYkbctpZ2WDn2w5I4mmOO78jndPUnaMT5gyFaonz
+K84xD+1/tZbTQ0adc9LE7XgAkpiiNjf2LW9tw==</SignatureValue>
  <KeyInfo>
    <KeyName>yyyyyyyyyyyyyyy</KeyName>
  </KeyInfo>
</Signature>
```

Où:

- DigestValue est le hachage non chiffré du message.
- SignatureValue est le hachage chiffré du message.
- KeyName est l'identifiant du composant signataire.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

# 7 Gestion des données de configuration du système

## 7.1 Justifications

Les articles précédents ont décrit comment un point d'extrémité expéditeur calcule le parcours d'un message, comment il constitue un message signé dont le contenu est chiffré, comment un courtier autorise les connexions des points d'extrémité et les messages, et comment un point d'extrémité destinataire vérifie la signature du message, déchiffre son contenu et envoie des accusés de réception. À un moment donné, chaque composant du processus de livraison a besoin d'informations publiées et gérées par d'autres composants tels que les chemins de message du point d'extrémité destinataire, les restrictions du courtier et les certificats de l'expéditeur / destinataire.

Ces informations sont désignées comme étant des *données de configuration système* (*system configuration data)*, stockées dans un *annuaire* dans lequel chaque composant a une entrée. Les principes de la solution spécifiée pour la saisie et la publication des données de configuration sont présentés en 4.4.3: *les données de configuration système sont réparties sur tous les composants*, chaque composant doit stocker toutes ces données et les annuaires des composants sont des concentrateurs dans le processus de publication:

- Chaque composant du système met en œuvre un annuaire local qui stocke les données de configuration du système complet dans une mémoire fiable[16].
- L'administrateur d'un annuaire des composants accorde l'accès au système aux points d'extrémité et aux courtiers, par l'intermédiaire d'un processus d'enregistrement: les points d'extrémité et les courtiers s'insèrent ainsi dans son sous-système. L'enregistrement comprend l'émission des certificats des composants, l'annuaire des composants étant l'autorité de certification (CA – *certification authority*) d'un sous-système.
- L'administrateur d'un annuaire des composants peut révoquer l'un de ses composants enregistrés.
- Chaque point d'extrémité et chaque courtier transmettent leurs propres données de configuration dans l'annuaire des composants de leur sous-système, également dénommé leur annuaire de référence.
- Deux annuaires des composants partagent et réalisent une synchronisation croisée des données de configuration de leurs sous-systèmes, de sorte que chacun des annuaires contient finalement les données de configuration du système complet.
- Un annuaire des composants partage et synchronise les données de configuration du système complet avec tout point d'extrémité ou courtier qui en fait la demande. Par conséquent, tout point d'extrémité peut potentiellement envoyer des messages à tout point d'extrémité soit directement, soit par l'intermédiaire d'un courtier.

L'Article 7 décrit les services exposés par un annuaire des composants de sorte que les composants peuvent enregistrer, leur transmettre les données de configuration et accéder aux données de configuration de tous les sous-systèmes.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

## 7.2 Contenu d'un annuaire et détention des informations

L'annuaire d'un sous-système contient une entrée pour chaque point d'extrémité et chaque courtier enregistrés, et une entrée pour l'annuaire des composants lui-même. Le Tableau 18 énumère les attributs d'une entrée, ainsi que pour chaque entrée, le type (ou format), ainsi que le détenteur des informations chargé de fournir la valeur d'attribut: C (le composant) ou D (l'annuaire de référence).

---

16 Voir la définition de «mémoire fiable» en 5.9.

**Tableau 18 – Annuaire des composants – contenu d'une entrée**

| Attribut | Type | Description | Détenteur | Obligatoire |
|---|---|---|---|---|
| code | string | Identifiant du composant décrit par l'entité (clé d'entrée) – voir 5.1. | D | Vrai |
| type | enumeration (énumération) | Type de composant que l'entrée décrit parmi: BROKER, COMPONENT_DIRECTORY, ENDPOINT. | D | Vrai |
| organization (organisation) | string | Nom de l'organisation détentrice du composant. | C | Vrai |
| person (personne) | string | Nom(s) du ou des administrateurs du composant, c'est-à-dire la ou les personnes à contacter en cas de problème de fonctionnement. | C | Vrai |
| email (courrier électronique) | string | Adresse(s) électronique(s) de la ou des personnes à contacter | C | Vrai |
| phone (téléphone) | string | Numéro(s) de téléphone de la ou des personnes à contacter. | C | Vrai |
| Urls (URL) | string[][17] | Liste ordonnée d'adresses URL, généralement deux: principale ou secondaire (RFC 4266).<br>• *Courtier* → AMQPS://…<br>• *Annuaire des composants* → HTTPS://…<br>• *Point d'extrémité* → AMQPS://… (ou vide si le point d'extrémité n'accepte pas de transfert direct entrant) | C | Faux |
| certificates (certificats) | Certificate[] (voir Tableau 19) | Ensemble de certificats valides détenus par le composant, quel que soit le type et éventuellement plus d'un certificat pour certains types.<br>• *Courtier* → AUTHENTICATION<br>• *Annuaire des composants* → AUTHENTICATION, INTEGRATED_CA<br>• *Point d'extrémité* → AUTHENTICATION, SIGNING, ENCRYPTION | D | Vrai |
| creation□ Timestamp (horodatage de création)□ | dateTime | Heure de création de l'entrée. | D | Vrai |
| modification□ Timestamp (horodatage de modification)□ | dateTime | Heure de la dernière modification de l'entrée. | D | Vrai |
| component□ Directory (annuaire des composants)□ | string | Code de l'annuaire des composants de référence du composant. | D | Vrai |
| mades□ Implementation (mise en œuvre MADES)□ | MadesImplementation (voir Tableau 20) | Nom du produit logiciel conforme à MADES utilisé par le composant et version MADES à laquelle il est conforme. | C | Vrai |
| paths (chemins) | MessagePath[] (voir Tableau 21) | Chemins de message permettant d'envoyer des messages au point d'extrémité – voir 5.3. | C | Uniquement pour le point d'extrémité |
| restriction | BrokerRestriction (voir Tableau 22) | Restriction du courtier – voir 5.4. | C | Uniquement pour le courtier |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

17 "xxx[]" désigne un ensemble d'attributs de type xxx. Il s'agit ici d'un ensemble de chaînes.

**Tableau 19 – Certificat (type)**

| Attribut | Type | Description | Exigé |
|---|---|---|---|
| certificateID | string | Identifiant du certificat – voir 8.2. | Vrai |
| type | string | Type du certificat tout au long du processus: ENCRYPTION, SIGNING, AUTHENTICATION, INTEGRATED_CA. | Vrai |
| certificate | base64Binary | Données binaires du certificat au format DER (Règles de codage distinctives ou *Distinguished Encoding Rules* en anglais) (voir Recommandation X.690 de l'UIT-T). | Vrai |

**Tableau 20 – MadesImplementation (type)**

| Attribut | Type | Description | Exigé |
|---|---|---|---|
| name | string | Nom du logiciel ou de l'application qui met en œuvre le composant (information). | Faux |
| version | string | Version du logiciel ou de l'application qui met en œuvre le composant (information). | Faux |
| compatibleVersions | string | Liste à virgules des versions MADES avec lesquelles le composant est actuellement compatible (information). | Faux |
| madesVersion | string | Version de la spécification MADES avec laquelle le composant est actuellement compatible. – voir Article 9. | Vrai |

**Tableau 21 – MessagePath (type)**

| Attribut | Type | Description | Exigé |
|---|---|---|---|
| messageType | string | Texte, comportant éventuellement un caractère générique (*) à la fin (par exemple, ABC*), qui indique le type des messages qui peuvent utiliser le chemin. | Vrai |
| path | string | Format: {DIRECT \| INDIRECT }:<broker code>□<br><broker code> obligatoire dans le cas d'un chemin INDIRECT.□ | Vrai |
| senderComponent | string[] | Ensemble des codes des points d'extrémité expéditeurs qui peuvent utiliser le chemin pour envoyer un message. Un caractère générique (*) désigne n'importe quel point d'extrémité. | Vrai |
| validFrom | dateTime | heure (inclusive) à partir de laquelle le chemin est utilisable. | Vrai |
| validUntil | dateTime | Heure (exclusive) jusqu'à laquelle le chemin est utilisable; pas d'heure limite lorsqu'elle n'est pas définie. | Faux |

**Tableau 22 – BrokerRestriction (type)**

| Attribut | Type | Description | Exigé |
|---|---|---|---|
| messageTypes | string[] | Ensemble des types de messages autorisés qui transitent par le courtier. Un type de message inclus dans cet ensemble peut s'achever par le caractère générique (*) afin de correspondre à un ensemble de types de messages. Un ensemble vide signifie que le courtier autorise tout type de message. | Faux |
| components (composants) | string[] | Ensemble des codes des points d'extrémité autorisés qui se connectent au courtier (en tant qu'expéditeur/producteur ou destinataire/consommateur). Un ensemble vide signifie que le courtier autorise tout point d'extrémité. | Faux |

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

### 7.3 Informations concernant la cohérence des données de configuration

#### 7.3.1 Cohérence des composants

Un composant (point d'extrémité ou courtier) doit vérifier le format et la cohérence *interne* de ses propres données de configuration, avertir son administrateur en cas d'incohérence, et ne pas introduire ce type d'incohérences dans son annuaire de référence. Par exemple:

- (*point d'extrémité, courtier)* Erreur de format, par exemple, adresses URL erronées, protocoles incorrects (HTTPS ou AMQPS);
- (*point d'extrémité*) Une période d'utilisation constante (de-à) comporte la relation "To" > "From";
- (*point d'extrémité*) Deux chemins de message avec le même type de message [18] ne doivent pas comporter de périodes d'utilisation qui se chevauchent.

#### 7.3.2 Cohérence du système

Il convient qu'un composant avertisse son administrateur de l'incohérence avérée ou nouvelle de certaines de ses propres données de configuration par rapport aux données de configuration du système complet.

Pour éviter que des composants conçus de façon indépendante n'acceptent pas les mêmes données, un composant ne doit pas rejeter les données de configuration présentant les incohérences suivantes, lorsqu'elles proviennent d'un autre composant par l'intermédiaire du processus de synchronisation:

- Certains codes de composants utilisés dans les chemins de messages (*point d'extrémité)* ou l'élément de restriction (*courtier*) n'existent pas (intégrité référentielle) (referential integrity).
- Certains chemins de messages sont inutiles et le point d'extrémité rejette les messages comportant le type de message correspondant, par exemple, le courtier rejette le point d'extrémité ou le type de message (voir 5.5).

Un annuaire des composants A doit rejeter les données de configuration synchronisées à partir de l'annuaire des composants B si un composant du sous-système B porte le code d'un composant déjà existant dans le sous-système A ou dans les autres sous-systèmes précédemment synchronisés de façon correcte. Un annuaire des composants A peut rejeter soit l'ensemble du sous-système B, soit uniquement les composants dupliqués.

#### 7.3.3 Mise en œuvre de la mise à jour répartie

Une modification des données de configuration par l'administrateur d'un composant doit être mise à jour au niveau du composant proprement dit et au niveau de son annuaire de référence. Cette mise à jour répartie peut produire un état incohérent, qui peut être le suivant:

- *État incohérent #1:* le composant archive tout d'abord localement la modification, puis l'introduit dans l'annuaire des composants. L'introduction échoue (par exemple, en raison d'une connexion interrompue) et la modification n'est stockée que dans le composant.
- *État incohérent #2*: le composant introduit tout d'abord la modification dans l'annuaire des composants, puis la stocke localement. Le composant ne parvient pas à stocker la modification localement (par exemple, arrêt intempestif du composant) qui est alors stockée uniquement dans l'annuaire des composants.

Un composant doit mettre en œuvre la mise à jour répartie afin d'éviter l'état incohérent #2.

IECNORM.COM : Click to view the full PDF of IEC 62325-503:2018

---

[18] Noter que AB et A sont différents.