AGENCE
DU NUMÉRIQUE
EN SANTÉ

Hub Santé

Dossier des Spécifications Techniques (DST)

Statut : Validé | *Classification : Restreinte* | *Version : v1.3*

Agence du Numérique en Santé
2 - 10 Rue d'Oradour-sur-Glane – 75015 Paris
esante.gouv.fr

Historique du document

Version	Rédigé par		Vérifié par		Validé par	
0.1	Romain Fouilland	Le 20/01/23				
	Motif et nature de la modification : Création du document					
0.1.1	Romain Fouilland	Le 06/02/2023				
	Motif et nature de la modification : Prise en compte des retours lors des 1-to-1 de présentations des spécifications					
0.1.2	Romain Fouilland	Le 24/02/2023				
	Motif et nature de la modification : Corrections (dates, génération de certificats) et ajout (heartbeat)					
1.0.0	Romain Fouilland Benjamin Bonche	Le 05/05/2023				
	Motif et nature de la modification : Stabilisation des spécifications suite au développement et déploiement du Hub					
1.1	Benjamin Bonche Romain Fouilland	Le 20/07/2023				
	Motif et nature de la modification : Corrections, mise à jour des règles de nommage du Hub, ajout de schémas (gestion des erreurs), ajout des règles d'autorisation					
1.2	Benjamin Bonche	Le 08/09/2023				
	Motif et nature de la modification : Ajout de la nomenclature des codes erreur, modification des permissions					
1.3	Benjamin Bonche	Le 16/03/2026				
	Motif et nature de la modification : Restructuration du document, ajout de la notion de périmètres fonctionnels, ajout d'un mécanisme de gestion multi-version des messages, enrichissement des cinématiques d'échange, ajout de précisions sur les conditions d'accès, de sécurité et de configuration côté clients.					

SOMMAIRE

1. INTRODUCTION

1.1. Objet du document

Ce document décrit en détail les principes et les spécifications techniques permettant de connecter et de gérer les échanges entre :

- Le « Hub Santé » de l'ANS
- Les logiciels d'éditeurs (appelés « Clients ») se conformant au Dossier des Spécifications Techniques (DST) : Logiciels de Régulation Médicale (LRM), logiciels de tablettes, logiciels de gestion des hélicoptères, concentrateurs régionaux, …

1.2. Contexte

Dans un cadre d'interopérabilité des services d'urgences aussi bien Santé qu'inter-forces, l'ANS pilote une solution d'échanges de messages appelée « Hub Santé » et la connecte avec les logiciels des Clients.

Afin de connecter ces Clients avec le Hub Santé dans sa version actuelle, l'ANS partage les présentes spécifications.

1.3. Référentiel

Ce document décrit l'ensemble des exigences et recommandations qui doivent être respectées par les Clients qui souhaitent se raccorder au Hub Santé.

EXI HUB XXX	L'ensemble des exigences sont identifiables par un encadré gris.

RECO HUB XXX	L'ensemble des recommandations sont identifiables par un encadré blanc.

1.4. *Repositories* GitHub partagés

1.4.1. Code Hub Santé des composants applicatifs et des clients

Afin de partager les travaux et les développements réalisés par l'équipe Hub Santé, un *repository* de code public a été créé sur le GitHub de l'ANS[1] à l'adresse suivante : https://github.com/ansforge/SAMU-Hub-Sante.

Ce *repository* contient :

- Un guide[2] pour lancer facilement les tutoriels[3] Java de RabbitMQ
- Des modèles de code d'un consommateur et d'un producteur implémentant les présentes spécifications et permettant de mutualiser les travaux
    - En Java[4] à titre d'illustration

[1] https://github.com/ansforge
[2] https://github.com/ansforge/SAMU-Hub-Sante/tree/main/clients/src/main/java/com/tutorials#readme
[3] https://www.rabbitmq.com/getstarted.html
[4] https://github.com/ansforge/SAMU-Hub-Sante/tree/main/clients/src/main/java/com/hubsante

  - En Spring Boot[5] utilisé par le *dispatcher* du Hub Santé

- Le code des pages web[6] du Hub Santé : landing page[7], LRM de test[8] et spécifications[9]

1.4.2. SDK du modèle de données

Afin de faciliter l'intégration des messages échangés avec le Hub Santé dans les systèmes des Clients, un *repository* de code public a été créé sur le GitHub de l'ANS[10] à l'adresse suivante : https://github.com/ansforge/SAMU-Hub-Modeles.

Ce *repository* contient :

- Un *README.md*[11] détaillant son utilisation et la correspondance des versions du *repository* avec celles des dossiers de spécifications fonctionnelles (DSF) et en fonction du périmètre métier (15-15, 15-nexsis, etc.)
- Les fichiers[12] techniques détaillant chaque modèle de données : schémas (*JSON Schemas* et *XSD*), document *Word* détaillant le modèle, diagramme *UML* illustratif et fichiers d'exemples
- Les *releases notes*[13] permettant de suivre les différentes versions et changements associés
- Le *package*[14] du SDK[15] dans le langage Java qui permet de l'intégrer facilement comme dépendance dans le code des Clients. Il contient les objets facilement manipulables et des méthodes de validation des messages.

1.5. Lexique

Abréviations	Significations
AC	*Autorité de Certification*[16]
AMQP	*Advanced Message Queuing Protocol*[17]
CISU	*Cadre d'Interopérabilité des Services d'Urgence*
CSR	*Certificate Signing Request*[18]
DSF	*Dossier des Spécifications Fonctionnelles : document spécifiant les cas d'usage couverts ainsi que les modèles de données et cinématiques associés*
DST	*Dossier des Spécifications Techniques : document spécifiant le fonctionnement du Hub Santé et les caractéristiques techniques du raccordement au Hub Santé*
EDXL	*Emergency Data Exchange Language*[19]
HubEx	*Plate-forme d'échanges – Hub d'échanges*

[5] Config https://github.com/ansforge/SAMU-Hub-Sante/blob/main/hub/dispatcher/src/main/resources/application.properties et code https://github.com/ansforge/SAMU-Hub-Sante/blob/main/hub/dispatcher/src/main/java/com/hubsante/hub/config/AmqpConfiguration.java
[6] https://github.com/ansforge/SAMU-Hub-Sante/tree/main/web
[7] https://hub.esante.gouv.fr/
[8] https://bac-a-sable.hub.esante.gouv.fr/lrm
[9] https://hub.esante.gouv.fr/specs/
[10] https://github.com/ansforge
[11] https://github.com/ansforge/SAMU-Hub-Modeles?tab=readme-ov-file#usage
[12] https://github.com/ansforge/SAMU-Hub-Modeles?tab=readme-ov-file#mod%C3%A8les
[13] https://github.com/ansforge/SAMU-Hub-Modeles/releases
[14] https://github.com/orgs/ansforge/packages?repo_name=SAMU-Hub-Modeles
[15] https://en.wikipedia.org/wiki/Software_development_kit
[16] https://fr.wikipedia.org/wiki/Autorit%C3%A9_de_certification
[17] https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol
[18] https://fr.wikipedia.org/wiki/Demande_de_signature_de_certificat
[19] https://en.wikipedia.org/wiki/EDXL

JSON	*JavaScript Object Notation*[20]
MOM	*Middleware Orienté Messages*[21]
PoC	*Proof of concept*
PTP	*Point To Point*
XML	*eXtensible Markup Language*[22]
XSD	*XML Schema Definition*[23]

[20] https://en.wikipedia.org/wiki/JSON
[21] https://fr.wikipedia.org/wiki/Message-oriented_middleware
[22] https://fr.wikipedia.org/wiki/Extensible_Markup_Language
[23] https://fr.wikipedia.org/wiki/XML_Schema

2. ARCHITECTURE

2.1. Fonctionnement général

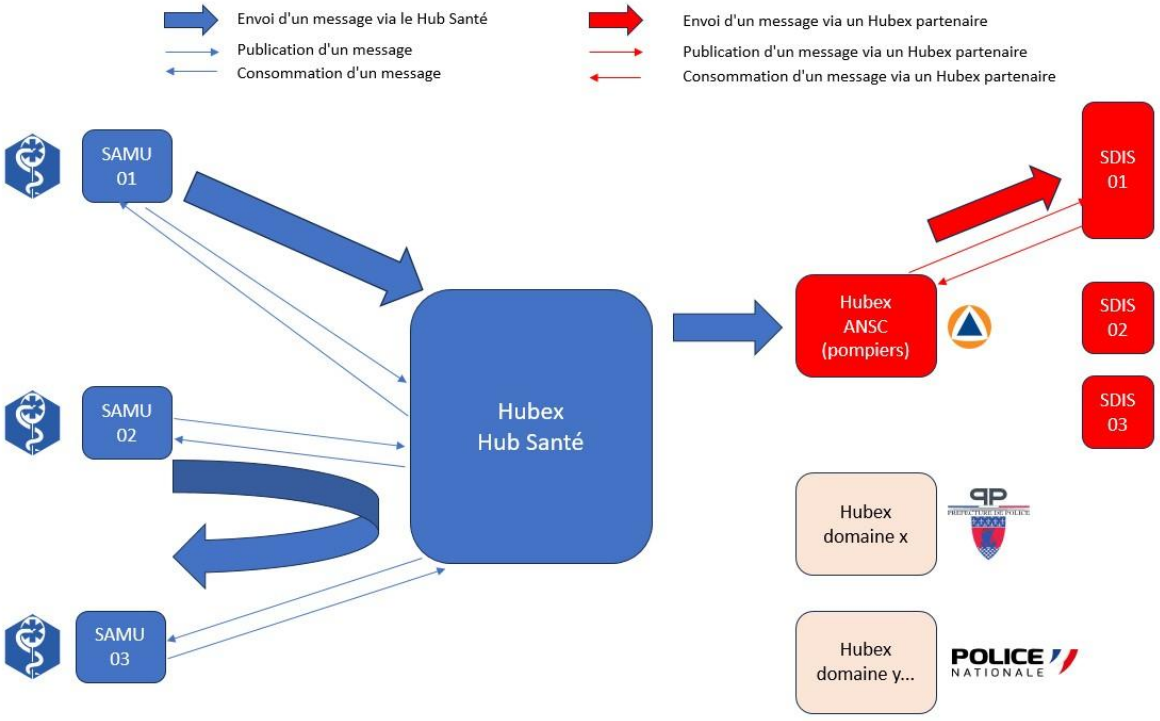
2.1.1. Présentation du maillage de HubEx

Les échanges entre les acteurs de l'urgence sont portés par des Hubs d'échanges (aussi appelés « HubEx ») métiers (un par force) interconnectés en un maillage. Les acteurs sont connectés à leur HubEx métier (et donc également appelés « Clients ») et peuvent échanger des messages avec n'importe quel autre acteur par l'intermédiaire de leur HubEx métier. Les communautés métiers sont ainsi sous la responsabilité technique de leur HubEx métier.

Les HubEx métiers considérés dans les travaux du Cadre d'Interopérabilité des Services d'Urgence (CISU) sont portés par :

- L'Agence du Numérique de Santé (ANS)
- L'Agence du Numérique de la Sécurité Civile (ANSC)
- La Gendarmerie Nationale (GN)
- La Police Nationale (PN)
- La Préfecture de Police de Paris (PP)

Le schéma suivant synthétise ce fonctionnement en maillage de HubEx et la centralisation des acteurs d'une force autour de leur HubEx métier :



2.1.2. Articulation avec les DSF

Les présentes spécifications détaillent le fonctionnement du Hub Santé ainsi que le raccordement au Hub Santé pour des Clients Santé. Ce raccordement est le prérequis technique à l'implémentation d'échanges fonctionnels tels que décrits dans les Dossiers de Spécifications Fonctionnelles (DSF) et portant notamment sur les échanges inter-Santé

(15-15, 15-SMUR, … entre Clients Santé) et les échanges 15-NexSIS (*i.e.* entre un LRM Santé et le logiciel national Pompiers NexSIS).

2.1.3. Orientations techniques macros

Messages asynchrones, AMQP et RabbitMQ

Les échanges entre HubEx ou entre un acteur et son HubEx métier (dans le cadre de ces spécifications entre un Client et le Hub Santé) se font via des opérations d'envoi et/ou de réception de messages en mode asynchrone selon une architecture Middleware Orienté Messages (MOM).

La solution technique proposée pour le réseau de HubEx à la suite des travaux techniques du CISU est basée sur le protocole Advanced Message Queuing Protocol (AMQP) 0-9-1[24] et son implémentation open-source[25].

Autorisations : acteurs et messages

La liste des Clients autorisés à échanger des messages sur le Hub Santé est maintenue par l'ANS. Les Clients ont le droit d'échanger uniquement avec les acteurs également enregistrés par les différents HubEx[26].

Les échanges autorisés sur le réseau de HubEx sont définis par des DSF précisant notamment les modèles de données attendus et qui sont contrôlés par les HubEx lors des échanges de messages. Les cinématiques d'échanges de plusieurs messages représentant des cas métiers spécifiques y sont aussi décrites.

Communications : point à point, files fixées et nommage

Le mode de communication est basé sur un mode d'échange point à point (PTP) via des files de messages.

Un Client émetteur publie donc un message pour un unique destinataire (c'est à dire un autre Client d'un HubEx métier) sur un échangeur d'envoi dédié nommé $hubsante$. L'émetteur utilise **son propre identifiant Client** comme clé de routage.

Un Client destinataire reçoit ses messages sur des files d'écoute spécifiques. L'ensemble des files sont portées par le Hub Santé. Chaque Client dispose de 3 files d'écoute selon la typologie des messages reçus :

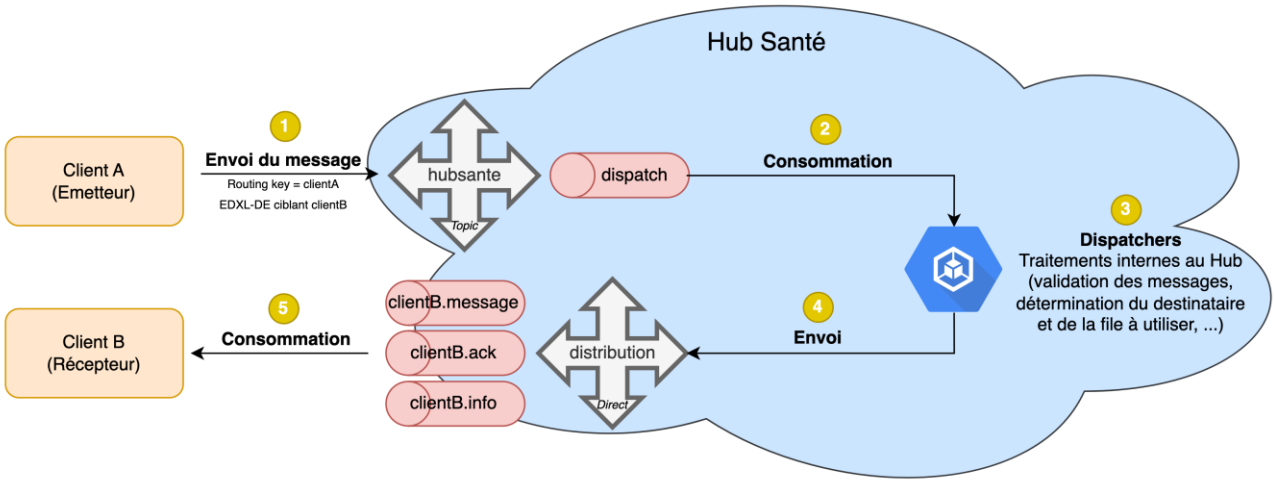
- « message » pour les échanges fonctionnels
- « ack » pour les acquittements de réception finale
- « info » pour les messages généraux d'informations, alertes et erreurs

La structuration du nom des files est $\{identifiantClient\}.\{typologie\}$ donnant, par exemple, *fr.health.samu010.message* ou *fr.health.samu020.ack*[27].

Grace à un mécanisme de routage interne au Hub Santé, chaque message publié sur l'échangeur d'envoi est transféré sur une file de messages accessible au destinataire. Le message est ensuite récupéré par le Client destinataire qui dispose alors des informations suffisantes pour traiter ce message au sein de son système d'information.

Le schéma ci-dessous illustre les différentes étapes du transit d'un message par le Hub Santé.

[24] https://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf
[25] https://www.rabbitmq.com/
[26] Le raccordement entre HubEx étant progressif et lié au déploiement des HubEx des différentes forces, l'ensemble des acteurs ne sera pas immédiatement accessible aux acteurs Santé raccordés au Hub Santé. Dans un premier temps, seuls les acteurs Santé et Pompiers sont accessibles.
[27] Voir la section Identifiants et logique de routage pour plus de détails sur les identifiants clients.



2.1.4. Environnements, URL et Autorité de Certification (AC) du Hub Santé

En plus des environnements internes utilisés par l'équipe Hub Santé pour ses développements et tests, trois environnements du Hub Santé sont disponibles.

Sur chacun de ces environnements, l'authentification auprès du serveur RabbitMQ se fait en mutual TLS (mTLS).

Chaque client doit donc être configuré de façon à d'une part reconnaître le serveur RabbitMQ comme digne de confiance, et d'autre part présenter un certificat client accepté par le serveur RabbitMQ.

L'IGC Santé est l'autorité de confiance retenue par le Hub Santé, sous ses deux versants, PROD et TEST, en fonction de l'environnement ciblé (cf détail environnement par environnement ci-dessous).

Le certificat serveur RabbitMQ est donc émis par cette autorité de confiance, il suffit donc au client d'ajouter l'AC IGC Santé dans sa liste d'autorités de confiance reconnues.

Le serveur RabbitMQ en fait de même : le certificat client présenté doit donc être également émis par la même autorité de confiance.

Bac-à-sable : pour les travaux, tests et recettes éditeurs

Cet environnement est à disposition des éditeurs souhaitant se raccorder au Hub Santé et réaliser les développements associés à l'implémentation de périmètres fonctionnels d'échanges du Hub Santé (tels que définis dans les DSF). Il est également utilisé pour réaliser les tests puis recettes des travaux des éditeurs en amont des déploiements sur les environnements de production.

L'enregistrement d'un Client sur le bac-à-sable se fait par prise de contact auprès de l'ANS. Le Client devra alors produire et transmettre un certificat client sur une URL identifiant son environnement de développement. Si fonctionnellement il s'agit bien d'un certificat client, il est nécessaire de demander à l'IGC Santé un certificat de type SSL_SERV, pour permettre le contrôle de la validité de l'URL fournie. Ce certificat, signé par l'IGC Santé TEST, sera intégré dans le Hub Santé.

Si le Client ne dispose pas des droits nécessaires sur l'IGC Santé TEST, le portail des industriels de l'ANS[28] est le point d'entrée pour l'accompagnement technique et la commande de certificats issus de l'IGC Santé grâce aux

[28] https://industriels.esante.gouv.fr/

formulaires « Industriels »[29]. Il doit ainsi remplir le formulaire F414 (en démarche simplifiée[30] ou en PDF[31]) pour obtenir des cartes et droits d'administrateurs techniques lui permettant ensuite de générer des certificats sur la plateforme IGC[32].

Les URL de l'environnement de bac-à-sable sont les suivantes :

- Hub Santé : **amqps://messaging.bac-a-sable.hub.esante.gouv.fr:5671**
- Site de test et recette : https://bac-a-sable.hub.esante.gouv.fr/lrm/

Pré-production : pour les tests et formations métiers

Cet environnement, iso à la production, permet de réaliser les tests et formations sur une instance déployée chez le métier (au SAMU par exemple) en amont du déploiement d'un lien en production.

L'enregistrement d'un Client sur la pré-production se fait suite à une recette validée des travaux sur le périmètre demandé sur l'environnement de bac-à-sable. Le Client devra alors transmettre à l'ANS un certificat client sur une URL identifiant son environnement de pré-production. Si fonctionnellement il s'agit bien d'un certificat client, il est nécessaire de demander à l'IGC Santé un certificat de type SSL_SERV, pour permettre le contrôle de la validité de l'URL fournie. Ce certificat sera signé par l'**IGC Santé PROD** (généralement fourni par la structure métier : le SAMU par exemple).

Si le Client (ou la structure métier équipée) ne dispose pas des droits nécessaires sur l'IGC Santé PROD, il doit remplir le formulaire F413 (en démarche simplifiée[33] ou en PDF[34]) qui lui permet de déclarer des administrateurs techniques qui pourront ensuite de générer des certificats sur la plateforme IGC. Au besoin, si des droits d'administrateur technique existent déjà, le formulaire F503 (en démarche simplifiée[35] ou en PDF[36]) permet de mettre à jour les domaines couverts par ces droits.

L'URL du Hub Santé sur l'environnement de pré-production est : **amqps://messaging.pre-prod.hub.esante.gouv.fr:5671**

Production

Cet environnement porte l'échange des messages en production.

L'enregistrement d'un Client sur la production se fait suite à une recette validée des travaux sur le périmètre demandé sur l'environnement de pré-production. Le Client devra alors transmettre à l'ANS un certificat client sur une URL identifiant son environnement de production. Si fonctionnellement il s'agit bien d'un certificat client, il est nécessaire de demander à l'IGC Santé un certificat de type SSL_SERV, pour permettre le contrôle de la validité de l'URL fournie. Ce certificat sera signé par l'**IGC Santé PROD** (généralement fourni par la structure métier : le SAMU par exemple).

Si le Client ne dispose pas des droits nécessaires sur l'IGC Santé PROD, la procédure à suivre est la même que celle détaillée pour l'environnement de pré-production ci-dessus.

L'URL du Hub Santé sur l'environnement de production est : **amqps://messaging.hub.esante.gouv.fr:5671**

2.2. Sécurisation des échanges

[29] https://esante.gouv.fr/index-des-formulaires
[30] https://www.demarches-simplifiees.fr/commencer/f414
[31] https://esante.gouv.fr/sites/default/files/media_entity/documents/F414.pdf
[32] https://pfc.eservices.esante.gouv.fr/pfcng-ihm/authentication.xhtml
[33] https://www.demarches-simplifiees.fr/commencer/f413
[34] https://esante.gouv.fr/sites/default/files/media_entity/documents/F413.pdf
[35] https://www.demarches-simplifiees.fr/commencer/f503
[36] https://esante.gouv.fr/sites/default/files/media_entity/documents/F503.pdf

2.2.1. Protocoles

Le Hub Santé est accessible directement depuis Internet, via les adresses spécifiées pour chacun des environnements mentionnés ci-dessus.

La sécurisation des échanges repose sur le protocole **AMQPS** (AMQP sur TLS), garantissant la confidentialité et l'authenticité des communications dans les deux sens (Client ↔ Hub Santé). Le canal AMQPS est établi au moyen d'une **authentification mutuelle par certificats** (mTLS). RabbitMQ assure directement la gestion de la sécurisation TLS et mTLS : il autorise et authentifie un client à partir du certificat que celui-ci présente.

La connexion doit obligatoirement être établie en **TLS v1.2 ou TLS v1.3.** Les suites de chiffrement TLS acceptées par le Hub Santé sont les suivantes :

- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-CHACHA20-POLY1305
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_GCM_SHA256
- TLS_AES_128_CCM_SHA256

Lors de l'établissement de la session TLS, le Hub Santé et le Client présentent chacun un **certificat X.509 serveur SSL_SERVEUR (SERV_SSL)** délivré par l'IGC Santé (en TEST pour le bac à sable, et en PROD pour les environnements pré-production et production — voir la section *Environnements, URL et Autorité de Certification (AC) du Hub Santé*).

- Le **Client** doit vérifier que le certificat serveur présenté correspond bien à celui du Hub Santé.
- Le **Hub Santé** s'assure que le certificat du Client figure parmi les certificats autorisés. Il vérifie également la cohérence entre :
  - L'identifiant du SAMU utilisé dans les messages,
  - La clé de routage en écriture,
  - Les files de messages en lecture,
  - Et le certificat présenté, en s'appuyant sur les mécanismes d'authentification et d'autorisation de RabbitMQ.

EXI HUB 001	Pour se connecter au Hub Santé, le Client DOIT accepter l'IGC Santé comme autorité de confiance.

EXI HUB 002	Pour se connecter au Hub Santé, le Client DOIT établir une connexion AMQP Sécurisée avec authentification mutuelle et un certificat X.509 serveur SSL_SERVEUR (SERV_SSL) délivré par l'IGC Santé.

EXI HUB 003	Pour se connecter au Hub Santé, le Client DOIT établir une connexion AMQP Sécurisée via TLS 1.2 ou TLS 1.3

2.2.2. Persistance des messages

Entrée et sortie des messagesEnveloppe EDXL-DEEnveloppe EDXL-DEDans l'implémentation RabbitMQ de la norme AMPQ 0-9-1, il est possible d'ajouter un header *delivery_mode* sur les messages envoyés par un client. Celui-ci permet la personnalisation par le client de la manière dont RabbitMQ traite les messages et impacte sa récupération en cas de panne du *broker*.

La valeur par défaut est *Transient* (header *delivery_mode* valorisé à 1). Dans ce mode, le *broker* enregistre les messages en mémoire pendant leur traitement. En cas de redémarrage, ces messages sont **perdus**. Les messages envoyés avec le header *delivery_mode* valorisé à 2 sont traités en mode *Persistent*. Ils sont écrits sur le disque et récupérés en cas de redémarrage. La documentation RabbitMQ[37] est disponible pour de plus d'information à ce sujet.

Pour assurer une meilleure tolérance aux pannes, le Hub Santé force les clients à utiliser le header *delivery_mode* valorisé à 2 pour que les messages soient traités en mode *Persistent*. Une vérification est effectuée en interne lors du traitement de chaque message. Le message est rejeté si le header ne possède pas la bonne valeur et un message d'erreur est envoyé à l'expéditeur par le Hub Santé.

EXI HUB 004	Le client DOIT utiliser le header *delivery_mode* avec la valeur « 2 » (*mode Persistent*) sur les messages envoyés.

Pour le moment, aucun stockage des messages (approche d'Event Hub) en dehors des besoins de fonctionnement de RabbitMQ n'est envisagé en production. Néanmoins, afin de faciliter les travaux d'implémentation, de correction et d'amélioration continue du Hub Santé, tout ou partie des messages de tests pourra être stocké.

2.2.3. Ouvertures d'accès

Il peut être nécessaire de solliciter les équipes informatiques de l'établissement de santé afin d'autoriser l'ouverture des domaines utilisés. À minima, l'accès à l'URL de RabbitMQ sur le port 5671 doit être garanti.

Il peut également être pertinent d'ajouter les URL des sites de présentation et de démonstration sur les postes des utilisateurs, afin de leur faciliter l'accès à l'information et à la formation.

2.3. Périmètres fonctionnels

Les messages supportés par le Hub Santé se regroupent par périmètres fonctionnels : échanges entre SAMUs (15-15), entre SAMU et SI pompiers (15-18), entre SAMUs et SMURs, ou tout autre périmètre qui pourra être ajouté à l'avenir.

Pour isoler les échanges par périmètre, une logique de *virtual hosts* (vhosts) est appliquée sur le serveur RabbitMQ.

Chaque version d'un périmètre est déployée sur son propre « Virtual Host ».

Le client se connecte à chaque « Virtual Host » des versions qu'il supporte.

Ainsi, un SAMU qui souhaiterait utiliser le vhost 15-15 (échanges inter SAMUs) et le vhost 15-NexSIS (échanges SAMU-SDIS) doit se connecter sur les deux vhosts. Chacun de ces vhosts héberge alors ses trois files clients, qui fonctionnent de façon indépendante.

Pour plus de clarté, le nom du vhost est formé par la concaténation du nom du périmètre et du numéro de version du DSF. Par exemple : "15-15_v1.5", "15-nexsis_v1.9".

2.4. Echange de messages multi-version

[37] Documentation RabbitMQ sur le header *delivery_mode* : https://www.rabbitmq.com/docs/queues#storage

Le format des messages supportés peut être amené à évoluer au fil du temps, au gré des échanges et des retours utilisateurs. De même, les périmètres fonctionnels peuvent s'enrichir de nouveaux messages au fur et à mesure de l'avancée de travaux.

Pour permettre aux clients du Hub Santé d'intégrer ces évolutions à leur rythme, tout en maintenant leur capacité à échanger des informations avec leurs partenaires, un système de compatibilité entre versions a été mis en place.

Les versions des DSF supportées par chaque client sont configurées au niveau du Hub Santé et un composant de conversion est déployé sur le Hub Santé. Les appels à ce composant sont pris en charge par le Hub Santé, de sorte que son utilisation est transparente pour les partenaires raccordés : chacun publie et consomme les messages dans la version qu'il a déclaré supporter.

Les seuls messages convertis sont ceux qui existent dans les deux versions, mais dont le contenu, où les règles de cardinalité, ont changé. Les données portées par un message introduit dans une version récente ne sont pas converties dans une version antérieure.

Les modifications apportées à un message entre deux versions (ajout de champ, suppression de champ, changement de nomenclature, …) sont gérées par le mécanisme de conversion. Dans ce cas, des règles de conversion ont été établies et sont référencées dans un document dédié, accessible sur le site du Hub Santé[38].

Pour ces raisons, et pour des questions de maintenabilité, il n'est pas prévu d'étendre ce dispositif au-delà d'un nombre raisonnable de montées de version. La sortie d'une nouvelle version occasionne à terme le décommissionnement de la version supportée la plus ancienne. Ce mécanisme de montée de version doit donc être vu comme un outil d'accompagnement des partenaires, offrant le temps nécessaire pour planifier et mettre en œuvre les évolutions.

2.5. Contrôle des messages publiés par périmètre

Afin de mieux maîtriser les usages, faciliter la compréhension et la détection des erreurs, une limitation et un contrôle des types de messages publiés sur les *virtual hosts* (vhosts) RabbitMQ est en place afin d'empêcher la transmission de messages non disponibles sur un périmètre donné.

Si un message non autorisé est envoyé, le client reçoit un message d'erreur sur sa file *.info* indiquant que le message n'est pas supporté.

Le DSF précise les messages disponibles sur chaque périmètre.

3. INTERFAÇAGE

3.1. Connexion

[38] https://hub.esante.gouv.fr/pages/accompagnement.html#technique

La connexion utilise le protocole AMQP 0-9-1. Chaque client se connecte au Hub Santé via un accès sécurisé par certificat remplaçant ainsi l'usage d'un couple identifiant / mot de passe.

Le chiffrement du flux, l'autorisation et l'authentification des clients sont ainsi réalisés via des certificats X509 signés par l'AC IGC Santé, voir la partie Sécurisation des échanges pour plus de détails.

L'URL d'accès au RabbitMQ du Hub Santé est messaging.hub.esante.gouv.fr.

Afin de gérer différentes versions (notamment les itérations successives des spécifications fonctionnelles), le Hub Santé utilise le concept technique de « Virtual Hosts ».

Ainsi, et conformément aux spécifications d'AMQP 0-9-1[39], l'**URI complète** (sans identifiant et mot de passe car l'authentification est portée par les certificats) est donc de la forme :

$$"amqps://messaging.hub.esante.gouv.fr:5671" ["/" vhost]$$

Afin d'**activer les échanges en TLS**, le Client doit configurer son connecteur.

La documentation de RabbitMQ a une page très complète sur le TLS[40] et donne des pistes pour Java[41] et .NET[42]. D'autre part, le GitHub propose deux exemples d'implémentation : l'un en Java dans les Consumer/Producer, l'autre en Spring Boot pour le Dispatcher du Hub Santé. Le protocole TLS doit être utilisé en version 1.2 ou 1.3.

Les Clients doivent établir leur connexion avec :

- La « Peer Verification » (via un truststore permettant de valider le certificat du serveur),
- La « Server Hostname Verification » (qui vérifie la correspondance entre le nom de domaine du serveur et le certificat présenté).

EXI HUB 005	Le client DOIT activer la « peer verification » dans la connexion établie avec le Hub Santé.

EXI HUB 006	Le client DOIT activer la « server hostname verification » dans la connexion établie avec le Hub Santé.

Afin d'**utiliser les certificats pour l'authentification** et non un couple identifiant et mot de passe, le Client doit configurer son connecteur afin de définir le mécanisme d'authentification sur la valeur : EXTERNAL. La documentation de RabbitMQ propose quelques pistes pour Java, .NET et Erlang[43]. D'autre part, le dépôt GitHub SAMU-Hub-Santé offre deux exemples d'implémentation[44] en Java et en Spring Boot[45].

EXI HUB 007	Le client DOIT utiliser le mécanisme d'authentification « EXTERNAL » pour utiliser l'authentification par certificat.

Après s'être authentifié à partir du Common Name (CN) de son certificat, le Client est autorisé :

[39] https://www.rabbitmq.com/uri-spec.html
[40] https://www.rabbitmq.com/ssl.html
[41] https://www.rabbitmq.com/ssl.html#java-client
[42] https://www.rabbitmq.com/ssl.html#dotnet-client
[43] https://www.rabbitmq.com/access-control.html#client-mechanism-configuration
[44] https://github.com/ansforge/SAMU-Hub-Sante/blob/0.6/client/src/main/java/com/hubsante/Consumer.java#L60
[45] https://github.com/ansforge/SAMU-Hub-Sante/blob/main/hub/dispatcher/src/main/java/com/hubsante/hub/config/AmqpConfiguration.java#L39

- À lire les messages publiés uniquement sur ses trois files d'écoute ({identifiantClient}.message, {identifiantClient}.ack et {identifiantClient}.info)
- À publier un message sur l'échangeur d'envoi *hubsante* uniquement en fournissant son identifiant clientId en tant que clé de routage

Pour faciliter l'identification de la connexion, le Client doit fournir un identifiant personnalisé, connu sous le nom **client-provided connection name**, correspondant à son clientID. La démarche à suivre pour positionner cet identifiant varie selon le type du client, et la documentation officielle de RabbitMQ donne des indications pour les clients en Java et en .NET[46].

EXI HUB 008	Le client DOIT utiliser son ClientID pour nommer la connexion ouverte.

Le Client dispose uniquement de ces permissions (lecture sur ses files d'écoute, écriture sur l'échangeur d'envoi) et n'a donc aucune permission de configuration. Cette dernière permission est inutile car la configuration des ressources est gérée par le Hub Santé.

Il incombe donc au Client de se connecter aux files en mode ***passif*** [47].

EXI HUB 009	Le client DOIT se connecter aux files en mode *passif*.

3.2. Entrée et sortie des messages

Afin de garantir la reprise d'activité en cas de problème sur le Hub Santé, les files de messages sont déclarées comme persistantes. Ainsi, leurs métadonnées sont stockées directement sur le disque et non seulement en mémoire vive ce qui permet de les récupérer lorsque le système est relancé.

Cependant, pour pouvoir récupérer les messages présents sur les files, tous les messages en entrée des HubEx doivent être également insérés avec un mode persistant faute de quoi ils ne seront pas récupérés lorsque le système est relancé[48].

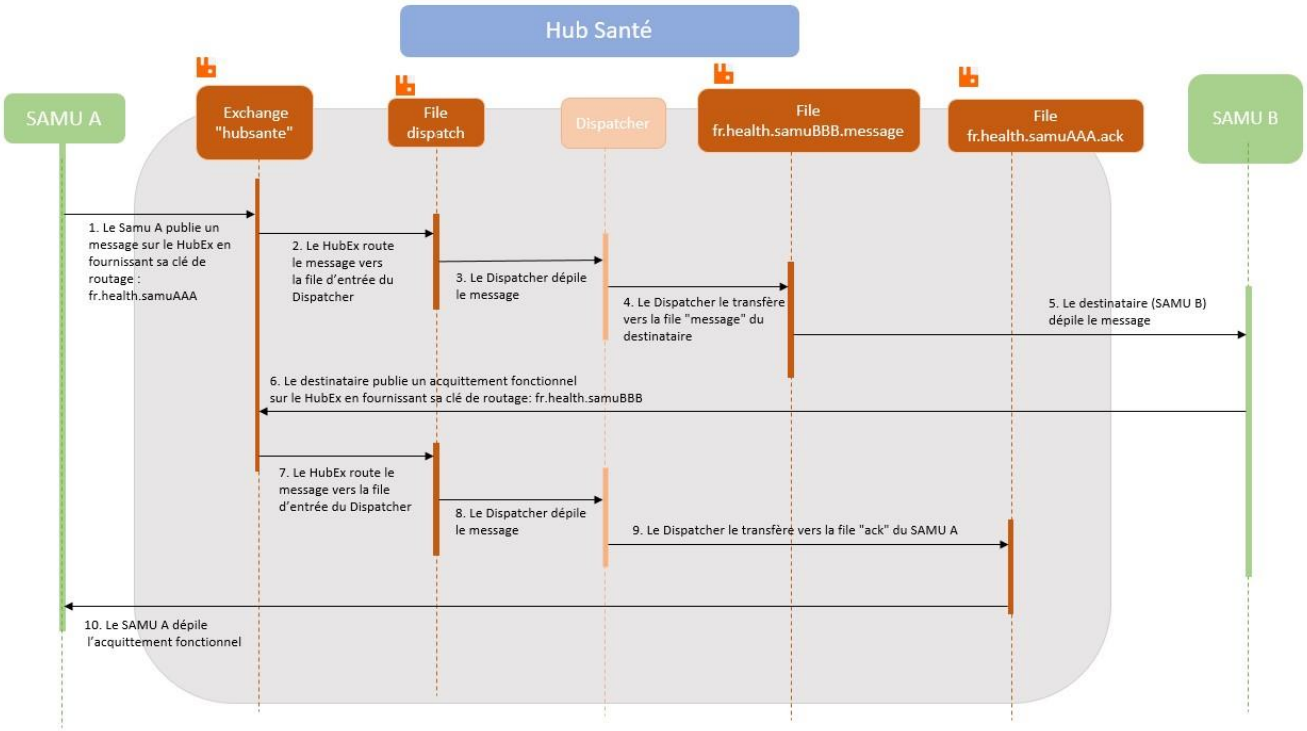
Le dépôt GitHub SAMU-Hub-Santé offre un exemple d'implémentation en Java et en Sprint Boot.

3.3. Flot des messages et acquittements

Le schéma suivant illustre le cheminement d'un message au sein du Hub Santé. Dans cet exemple, le Client LRM "Samu A" génère un message contenant les données d'un appel à destination du "Samu B". Le client publie ensuite son message sur l'échangeur *hubsante* du Hub Santé en fournissant comme clé de routage son Identifiant.

[46] https://www.rabbitmq.com/docs/connections#client-provided-names
[47] Java : https://www.rabbitmq.com/api-guide.html#passive-declaration, .NET : https://www.rabbitmq.com/dotnet-api-guide.html#passive-declaration, Ruby : http://rubybunny.info/articles/queues.html#checking_if_a_queue_exists, Python : https://pika.readthedocs.io/en/stable/modules/channel.html#pika.channel.Channel.queue_declare
[48] https://www.rabbitmq.com/queues.html#durability

Le Hub Santé récupère le message et le transfère sur la file du client destinataire.



Afin de garantir la fiabilité du système, des acquittements sont nécessaires pour :

- Informer chaque acteur lorsque le message a bien été traité par l'intermédiaire suivant
- Informer l'émetteur que le message a bien été reçu par le destinataire

Pour informer l'émetteur du traitement ou des actions prises par le destinataire, les messages fonctionnels doivent être utilisés.

3.3.1. Informer du transfert de responsabilité – acquittement technique

Le protocole AMQ 0-9-1 prévoit une logique d'acquittement technique sous la forme de *Consumer Acknowledgement*[49].

Tant qu'un consumer n'a pas formellement acquitté un message, celui-ci est réputé non consommé et le serveur RabbitMQ ne le purge pas de la file. Il s'agit donc d'un mécanisme purement technique, entre le client et le serveur RabbitMQ, pas d'un accusé de réception à destination du système émetteur du message (sur ce point, voir la section suivante "acquittement de réception finale").

Ces acquittements doivent être intégrés à la fin du code de traitement d'un message. En cas d'oubli, les messages resteraient dans la file d'envoi et RabbitMQ tenterait de les renvoyer au client destinataire ce qui conduirait à différents problèmes (duplication de messages, blocage des files, problèmes de mémoire, …).

EXI HUB 010	Le client DOIT acquitter techniquement de la réception des messages avec un « Consumer Acknowledgement » du protocole AMQP.

[49] https://www.rabbitmq.com/confirms.html#consumer-acknowledgements

3.3.2. Informer de la réception – acquittement de réception finale (message RC-REF)

La validation auprès de l'émetteur, appelée *Publisher Confirms* par RabbitMQ, n'est pas nativement intégrée dans le protocole AMQP 0-9-1. Afin d'offrir cette fonctionnalité, un deuxième type de files $\{IdentifiantClient\}.ack$ permet de remonter les accusés de réception finale.

Une fois le message intégré dans le système du Client destinataire, ce dernier doit en informer le Client émetteur en lui envoyant un acquittement de réception finale sur ces files de messages. Le cheminement est similaire au message envoyé mais pris en sens inverse.

Seuls les messages reçus sur la file ".message" nécessitent l'envoi d'un message d'acquittement, contrairement à ceux reçus sur les files ".ack" et ".info".

D'un point de vue fonctionnel, tant que l'acquittement de réception finale n'est pas reçu côté émetteur, le Client émetteur doit considérer que le message n'a pas été délivré au Client destinataire et doit en informer ses utilisateurs pour que ceux-ci puissent éventuellement contacter le destinataire.

Le format des acquittements de réception finale est défini dans la partie Message de référence et format des acquittements de réception finale: il s'agit du message RC-REF.

Cet acquittement de réception finale n'est en aucun cas un message à portée fonctionnelle (acceptation du dossier, réponse favorable à une demande d'engagement de ressource, etc.). Il doit bien être compris comme l'équivalent d'un accusé de réception lors d'un échange postal : le message a bien été reçu, sans préjudice des éventuelles actions engagées par la suite. Il doit donc être émis dès l'intégration du message dans le système destinataire.

EXI HUB 011	Le Client destinataire DOIT envoyer un message RC-REF pour informer le Client émetteur de la bonne réception et intégration du message dans son système.

3.3.3. Informer d'une erreur à l'intégration d'un message – message RC-REF "refused"

Dans le cas où une erreur se produit lors de l'intégration du message dans le système, un message de non-acquittement doit être envoyé à la place du message d'acquittement par le client destinataire pour indiquer au client émetteur la réception correcte mais un traitement échoué du message.

Il s'agit d'un message RC-REF avec le booléen "refused" valorisé à "true". L'envoi de ce message doit être accompagné par l'envoi d'un message d'info contenant les précisions sur la nature de ou des erreurs produites.

EXI HUB 012	En cas d'erreur à l'intégration d'un message, le Client destinataire DOIT envoyer un message RC-REF avec le booléen « refused » valorisé à « true » pour informer le Client émetteur.

EXI HUB 013	En cas d'erreur à l'intégration du message, le Client destinataire DOIT envoyer au Client émetteur un message RS-ERROR contenant les précisions sur la nature de ou des erreurs produites.

3.3.4. Informer du traitement – message fonctionnel

L'accusé de réception finale n'informe que de la réception du message par le Client destinataire et ne donne aucune information sur le traitement et les actions réalisés par le destinataire.

Pour garder des messages d'acquittements les plus simples possibles et devant la pluralité des réponses possibles, ces informations doivent faire l'objet de nouveaux messages fonctionnels et ne pas passer par les acquittements.

3.3.5. Gestion des erreurs

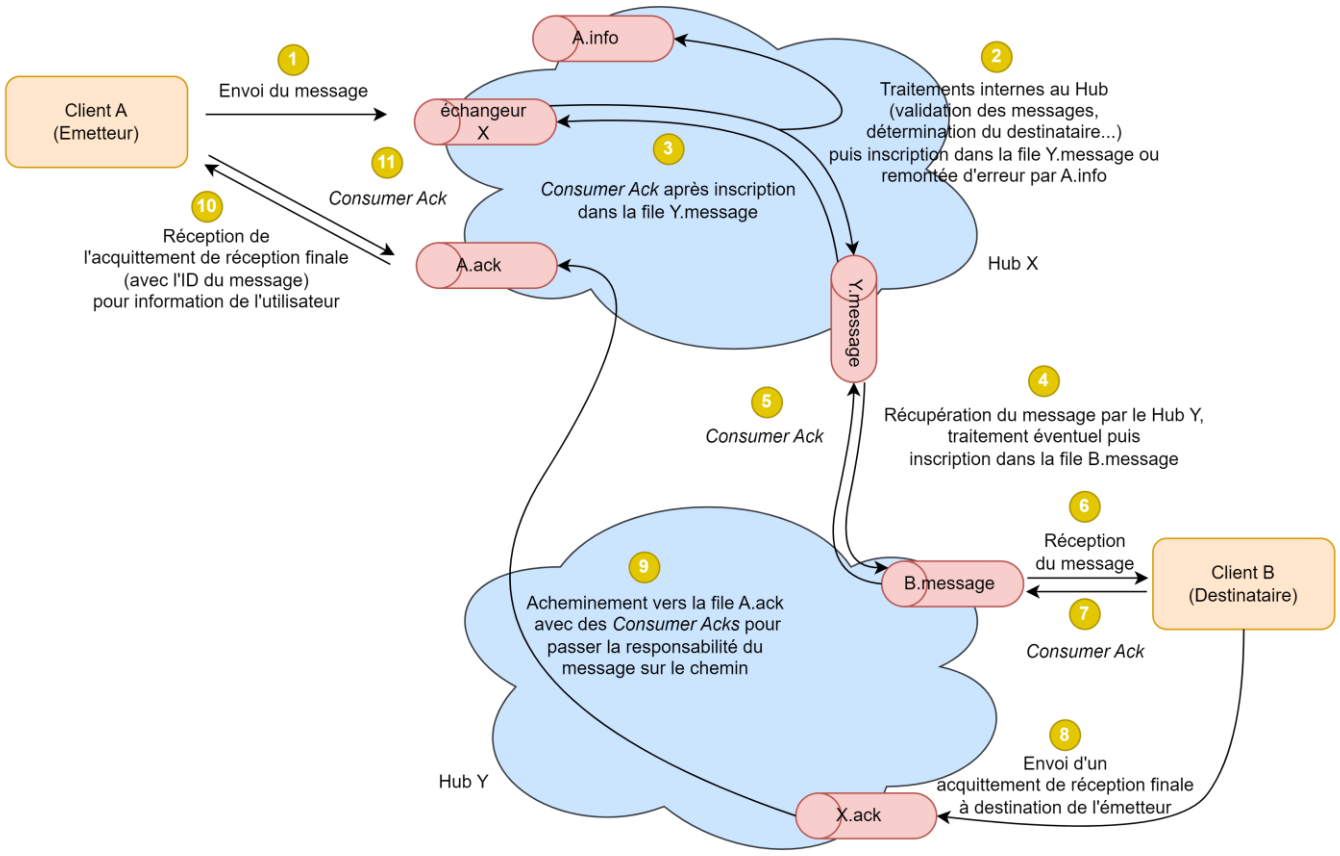
Un troisième type de file, $\{IdentifiantClient\}.info$, est mis en place pour remonter des informations et de potentielles erreurs aux émetteurs et destinataires des messages. Une erreur survenue au cours d'un envoi de message peut aboutir à l'envoi d'un message d'erreur dans les cas suivants :

- Définition non conforme du Content-Type dans les propriétés du message AMQP (voir la section Format des messages : JSON et XML)
- Incohérence entre la clé de routage utilisée à la publication et le champ $SenderID$ dans l'Enveloppe EDXL-DE (voir la section Enveloppe EDXL-DE)
- Non-conformité du message envoyé aux modèles de données spécifiées dans les différents DSF
- Expiration du message avant dépilement par le destinataire
- Rejet du message par le destinataire lors de son dépilement

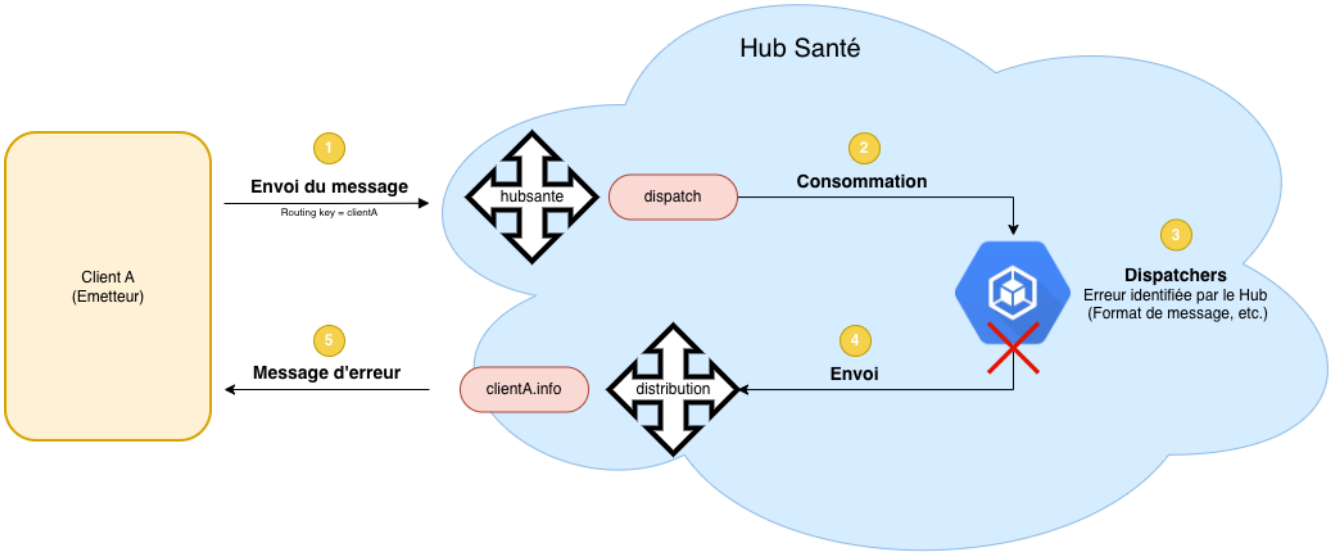
Les rapports d'erreur respectent la structure définie en partie Format des messages d'erreur: message RS-ERROR.

3.3.6. Schémas récapitulatifs

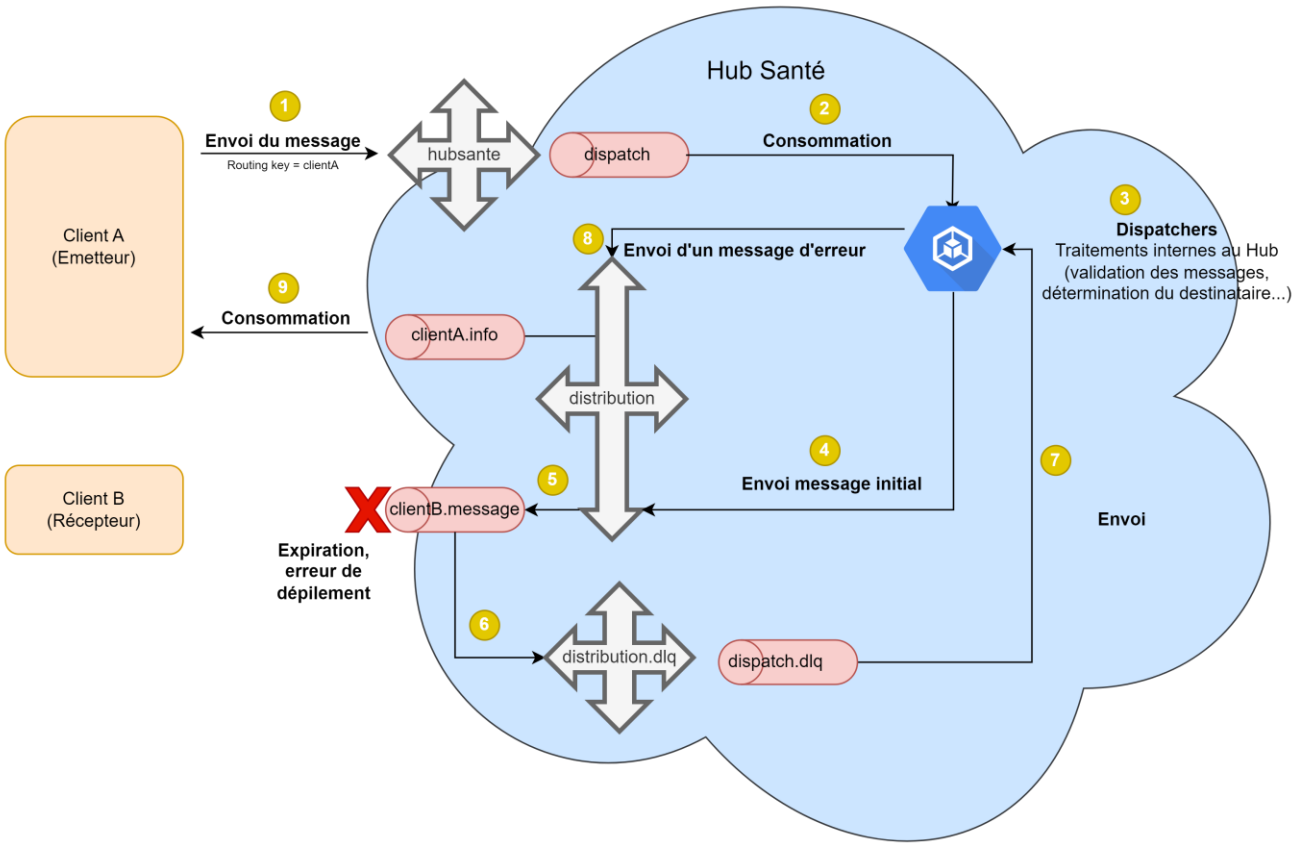
Le schéma suivant illustre le cheminement d'un message entre deux Clients liés à des HubEx différents ainsi que les acquittements techniques (*Consumer Acks*) et l'acquittement de réception finale (message *ack*).



Le schéma suivant illustre le cheminement d'un message rejeté par le Hub Santé.



Le schéma suivant illustre le cheminement d'un message accepté par le Hub mais non dépilé (expiration du délai, erreur interne côté destinataire...).



3.4. Messages

Un message transitant par le réseau de HubEx est composé :

- D'une enveloppe de type EDXL-DE contenant les informations utiles au routage du message. Il s'agit de la seule partie exploitée par les HubEx.
- D'un corps de message contenant les informations fonctionnelles à transmettre au Client destinataire.

Pour l'ensemble des cas d'usages portés par le Hub Santé, un contrat d'interface (prenant la forme d'un Dossier des Spécifications Fonctionnelles – DSF) est partagé et accompagné d'un modèle de données décrivant le format des messages autorisés (JSON Schema en JSON, XSD en XML).

Le Hub valide la conformité des messages transmis par rapport aux modèles de données des DSF.

3.4.1. Format des messages : JSON et XML

Par défaut, les messages échangés doivent être en JSON (JavaScript Object Notation).

Pour certains cas d'usage, notamment le lien 15-NexSIS basé sur les travaux CISU, les messages pourront également être échangés en XML (eXtensible Markup Language).

La conversion entre les deux formats et la correspondance entre les contrats d'interface et modèles de données seront assurés par le Hub Santé. Ainsi, les Clients peuvent ne s'occuper que du contrat d'interface et du modèle de données dans le format de leur convenance.

Pour permettre au Hub de prendre en charge ces deux formats, il est nécessaire de définir le $Content - Type$ au niveau des propriétés du message AMQP. Cela se fait généralement au niveau du client RabbitMQ. Deux valeurs sont acceptées par le Hub Santé : $application/json$ et $application/xml$. Toute autre valeur, ou si le $Content - Type$ n'est pas explicitement défini, entraînera un échec de transmission et l'envoi d'un message d'erreur correspondant sur la file info de l'émetteur.

EXI HUB 014	Le Client DOIT définir le *content-type* au niveau des propriétés du message AMQP avec l'une des deux valeurs : « *application/json* » ou « *application/xml* ».

3.4.2. Gestion de l'expiration des messages

Le protocole AMQP 0-9-1, utilisé dans le Hub Santé par le biais de son implémentation via RabbitMQ, indique, dans la spécification des headers disponibles dans les messages AMQP, la présence d'un champ $Expiration$.

Ce header applique des règles de Time-To-Live **directement au niveau des messages AMQP**. Au bout de la durée précisée dans le header, un message non consommé sur une queue sera traité comme expiré.

Au sein du Hub Santé, ce header **ne doit pas être valorisé par les clients**. Tout message reçu avec un header $Expiration$ valorisé verra sa valeur remise à zéro avec la valeur $null$ à l'ingestion par le Hub. La gestion de l'expiration des messages doit se faire côté client par le biais du champ $dateTimeExpires$ présent dans l'enveloppe EDXL (cf DSF et description ci-dessous).

Le header $Expiration$ est utilisé en interne pour surcharger la politique par défaut d'expiration des messages. Cette politique par défaut est définie au niveau des queues et provoque l'expiration de tout message en attente passé 24 heures. Si le champ $dateTimeExpires$ est valorisé et indique un délai plus court que cette valeur par défaut, le Hub définira le header sur ce délai pour ce message.

EXI HUB 015	Le Client ne DOIT pas utiliser le header « Expiration » au niveau des messages AMQP.

EXI HUB 016	Pour surcharger la date d'expiration du message, le Client DOIT utiliser le champ $dateTimeExpires$ présent dans l'enveloppe EDXL.

3.4.3. Encodage des messages

La charge utile transportée par les messages AMQP doit être encodée sous forme de chaîne de caractère en suivant le format UTF-8.

EXI HUB 017	Le Client DOIT encoder le contenu des messages AMQP sous forme de chaîne de caractère en suivant le format UTF-8.

3.4.4. Nommage des modèles

Les modèles sont nommés suivant la structure suivante :

[NORME]-[CAS D'USAGE]

Les normes possibles sont :

- **EDXL :** Emergency Data Exchange Language[50]
- **RC :** Référentiel CISU pour les modèles publiés par les travaux inter-forces
- **RS :** Référentiel Santé pour les modèles publiés par les travaux Santé

Des exemples de cas d'usage possibles sont :

- Techniques
  - **DE** : *Distribution Element*
  - **ERROR** : Erreur
  - **REF** : Référence
- Fonctionnels
  - **EDA :** Échange de Dossier / Affaire
  - **DR :** Demande de ressource

La liste exhaustive des cas d'usages par périmètre est disponible dans le DSF du périmètre correspondant.

Ces normes et cas d'usage peuvent être combinées pour construire les modèles suivants :

- **EDXL-DE** : Enveloppe
- **RC-DE** : En-tête de messages
- **RC-EDA** : Échange de dossier / affaire 15-NexSIS
- **RS-EDA** : Échange de dossier / affaire 15-15
- **RC-REF** : Reference à un message précédent, utilisé notamment pour l'acquittement
- **RS-ERROR :** Message d'erreur

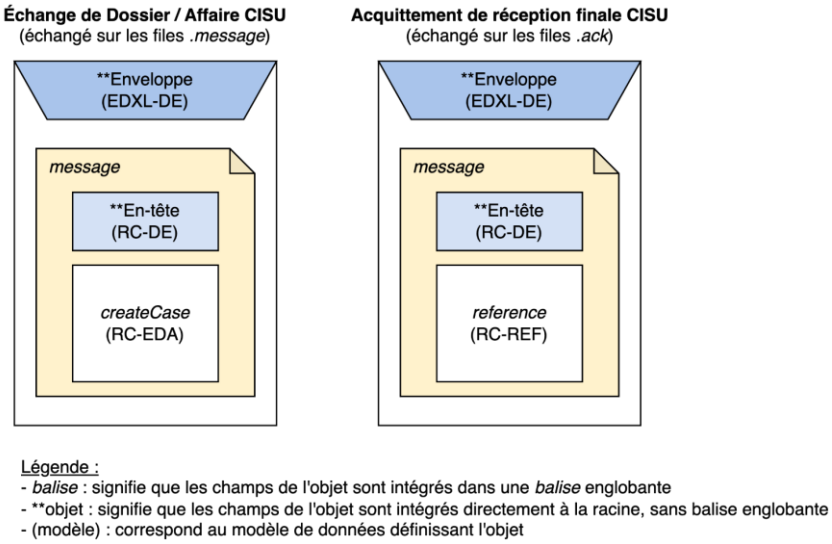
3.4.5. Enveloppe et en-tête de message

Les messages fonctionnels (décrits dans les Dossiers des Spécifications Fonctionnelles – DSF) sont contenus dans les attributs ***content*** de l'enveloppe EDXL-DE (voir la partie Enveloppe EDXL-DE) et comportent systématiquement deux éléments :

[50] https://docs.oasis-open.org/emergency/edxl-de/v2.0/edxl-de-v2.0.html

- Un en-tête de message suivant le format RC-DE (voir la partie En-tête RC-DE)
- Un message suivant le format suivant :
    - Un acquittement de réception finale : RC-REF (voir la partie Message de référence et format des acquittements de réception finale)
    - Un objet fonctionnel défini dans un DSF : RC-EDA, RS-EDA, …

Le schéma ci-dessous détaille synthétiquement la construction de deux trames échangées entre deux partenaires via le Hub Santé.



3.4.6. Identifiants et logique de routage

L'approche retenue par le Hub Santé actuellement est basée sur une logique de domaines imbriqués et permet d'échanger à plusieurs échelles. Néanmoins, elle est encore en travaux au niveau de CISU et pourra être amenée à évoluer.

L'identifiant se construit selon la règle : <pays>.<domaine>.<acteur>, où chaque partie précise respectivement le territoire, le domaine fonctionnel et l'organisation ciblée.

Ainsi, l'identifiant d'un SAMU aurait la forme $fr.health.samuXXX$ permettant d'indiquer qu'il s'agit :

- D'un acteur français
- Du domaine de la Santé
- Correspondant au SAMU du département XXX

XXX étant le code CRRA (ex : 680 pour Mulhouse, 2A0 pour Ajaccio ; 64A pour Bayonne).

Ces identifiants permettent aux différents HubEx de pouvoir router les messages en interne ou vers le Hub adapté.

3.4.7. Enveloppe EDXL-DE

Le tableau ci-dessous précise les balises qui doivent être envoyées et qui sont nécessaires au routage des messages. Ces balises sont spécifiées dans le document EDXL-DE (Emergency Data Exchange Language - Distribution Element), version 2.0 du 19 septembre 2013[51].

Bien que le format par défaut du Hub Santé soit le JSON, pour certains cas d'usage (notamment le lien 15-NexSIS), cette enveloppe peut également être envoyée sous une forme XML. Les mêmes règles de format et de cardinalité s'appliquent. Le nommage des champs est identique dans les deux cas, à l'exception de l'élément encapsulant le message fonctionnel en lui-même : pour plus de clarté, dans le cas d'une enveloppe en JSON, il a été décidé de renommer les champs **contentXML** en **jsonContent** et **embeddedXMLContent** en **embeddedJsonContent**.

[51] https://docs.oasis-open.org/emergency/edxl-de/v2.0/edxl-de-v2.0.html

edxlDistribution

Paramètre	Description	Cardinalité	Type
distributionID	Identifiant unique de ce message d'urgence attribué par l'expéditeur et comme étant unique pour cet expéditeur.	[1..1]	**Chaîne de caractères** devant respecter le format suivant : **<senderId>_<internalId>** senderId : voir paramètre senderID internalId : identifiant au format UUID défini par l'expéditeur mais étant garanti unique dans son système Ex : $fr.health.samuXXX\_cdf683c3-e01e-4055-bff1-3f918d467fa4$
senderID	Identifiant de l'émetteur parmi une liste donnée.	[1..1]	**Chaîne de caractères** parmi la liste des acteurs raccordés au réseau de HubEx et suivant les règles de nommage. Ex : $fr.health.samuXXX$
dateTimeSent	Date et heure auxquelles le message a été envoyé.	[1..1]	**Date au format ISO 8601 (horodatage complet avec décalage horaire par rapport au fuseau GMT)** Ex : 2022-06-21T23:59:00+02:00 (21 juin 2022 à 23h59, heure de Paris)
dateTimeExpires	Date et heure auxquelles le message doit expirer : c'est-à-dire que les données doivent être détruites et non délivrées au-delà de cette date.	[1..1]	**Date au format ISO 8601** (cf supra)
distributionStatus	Statut du message à choisir parmi plusieurs valeurs. *Actuellement sans effet dans le routage du Hub Santé mais utilisé à terme pour faire passer des messages de tests en recette ou investigation.*	[1..1]	Valeurs énumérées parmi la liste ci-dessous : **Actual, Exercise** Les valeurs ci-dessous spécifiées par la spécification EDXL ne sont actuellement pas utilisées : System, Test, Unknown, NoAppropriateDefault
distributionKind	Permet de préciser le type de message. *Le routage s'appuie sur ce champ : le Hub traite les valeurs Report, Update, Cancel pour transférer le message sur la file message du destinataire, la valeur Ack vers la file ack et la valeur Error vers la file info.*	[1..1]	Valeurs énumérées parmi la liste ci-dessous : **Report, Update, Cancel, Ack, Error** Les valeurs ci-dessous spécifiées par la spécification EDXL ne sont actuellement pas utilisées : Request, Response, Dispatch, SensorConfiguration, SensorControl, SensorStatus, SensorDetection, Unknown, NoAppropriateDefault
descriptor	Description des sous-éléments liés au message.	[1..1]	Voir tableau **descriptor** ci-après.
Content	Contenu du message.	[1..1]	Voir tableau **content** ci-après.

descriptor

Paramètre	Description	Cardinalité	Type
combinedConfidentiality	Non utilisé	Non utilisé	Non utilisé
language	Langage du message échangé.	[1..1]	**Étiquette** de langue complète de l'IETF[52]. Ex : fr-FR
senderRole	Non utilisé	Non utilisé	Non utilisé
recipientRole	Non utilisé	Non utilisé	Non utilisé
keyword	Non utilisé	Non utilisé	Non utilisé
explicitAddress	Identifiants de l'expéditeur et du destinataire du message.	[1..1]	Voir tableau **explicitAddress** ci-après.
targetAreas	Non utilisé	Non utilisé	Non utilisé
urgency	Non utilisé	Non utilisé	Non utilisé
Severity	Non utilisé	Non utilisé	Non utilisé
certainty	Non utilisé	Non utilisé	Non utilisé
incidentID	Non utilisé	Non utilisé	Non utilisé
incidentDescription	Non utilisé	Non utilisé	Non utilisé
Link	Non utilisé	Non utilisé	Non utilisé
extension	Non utilisé	Non utilisé	Non utilisé

content

Paramètre	Description	Cardinalité	Type
contentObject	Contenu du message.	[1..n] (un tableau contenant un seul élément est attendu)	Voir tableau **contentObject** ci-après.
Exlink	Non utilisé	Non utilisé	Non utilisé
other	Non utilisé	Non utilisé	Non utilisé

[52] https://fr.wikipedia.org/wiki/%C3%89tiquette_d%27identification_de_langues_IETF

contentObject			
Paramètre	**Description**	**Cardinalité**	**Type**
contentDescriptor	Non utilisé	Non utilisé	Non utilisé
contentXML	Contient les données dans une structure XML donnée, dans un unique élément enfant **embeddedXMLContent**.	[1..1] (l'objet obligatoire est lié au type – JSON ou XML – du message envoyé)	Contenu du message au format **XML**.
jsonContent ***(alternative en JSON)***	Contient les données dans un schéma JSON donné, dans un unique élément enfant **embeddedJsonContent**.		Contenu du message en **JSON**.
otherContent	Non utilisé	Non utilisé	Non utilisé

3.4.8. Adressage destinataire

La norme EDXL-DE permet de spécifier plusieurs balises <*explicitAddress*>. Cependant, il a été retenu de ne permettre l'émission que vers un unique destinataire et de n'avoir que des messages point-à-point. Le message complet doit donc être dupliqué s'il est nécessaire de l'envoyer à plusieurs destinataires.

Le tableau ci-dessous précise les balises qui doivent être envoyées pour l'utilisation de l'élément <*explicitAddress*> :

explicitAddress			
Paramètre	**Description**	**Cardinalité**	**Type**
explicitAddressScheme	Identifiant du SI pilotant le Hub	[1..1]	Pour tout message envoyé au Hub Santé, la **valeur est fixe : "hubex".**
explicitAddressValue	Identifiant du destinataire du message parmi une liste donnée	[1..1]	**Chaîne de caractères** parmi la liste des acteurs raccordés au réseau de HubEx et suivant les règles de nommage. Ex : $fr.health.samuXXX$ *Nota : il s'agit de la même liste que pour la balise senderID.*

3.4.9. En-tête RC-DE

Le référentiel CISU (RC), utilisé pour spécifier le format de certains messages d'urgence comme le partage d'affaire ou l'acquittement de réception finale d'un autre message d'urgence, contient un certain nombre de champs communs à tous les types de message et permettant de les rendre autoportant même sans leur enveloppe EDXL-DE.

Ces champs sont définis dans le référentiel CISU – Distribution Element, ou RC-DE.

Les messages de type CISU peuvent être compris comme des extensions du RC-DE : par exemple, le message de référence – qui suit la spécification RC-REF – correspond à un RC-DE étendu par l'élément **Reference**. Le message d'Échange de Dossier/Affaire, défini dans le DSF 15-NexSIS et qui suit la spécification RC-EDA, correspond à un RC-DE étendu par l'élément **CreateCase**.

Le tableau ci-dessous précise les balises qui doivent être envoyées pour l'utilisation de l'élément <RC-DE> :

RC-DE			
Paramètre	**Description**	**Cardinalité**	**Type**
messageId	Identifiant du message interne. Identique au paramètre ***distributionID*** de l'enveloppe EDXL-DE	[1..1]	Chaîne de caractères identiques au distributionID (cf 3.4.3).
sender		[1..1]	Voir tableau **sender** ci-après
sentAt	Date et heure auxquelles le message a été envoyé	[1..1]	**Date au format ISO 8601** (cf supra)
status	Statut du message à choisir parmi plusieurs valeurs.	[1..1]	Valeurs issues de la spécification EDXL-DE.distributionStatus (cf 3.4.3), énumérées parmi la liste ci-dessous : **Actual**, **Exercise, System**. Les valeurs ci-dessous ne sont actuellement pas utilisées : Test, Unknown, NoAppropriateDefault
kind	Permet de préciser le type de message.	[1..1]	Valeurs issues de la spécification EDXL-DE/distributionKind (cf 3.4.3), énumérées parmi la liste ci-dessous : **Report, Update, Cancel, Ack, Error**. Les valeurs ci-dessous ne sont actuellement pas utilisées : Request, Response, Dispatch, SensorConfiguration, SensorControl, SensorStatus, SensorDetection, Unknown, NoAppropriateDefault.
recipient	Liste les destinataires du message. Il n'existe qu'un destinataire par enveloppe EDXL-DE (cf 2.1.3), mais ce paramètre permet de lister l'ensemble des destinataires ayant reçu un message au contenu similaire. Ce paramètre peut se comprendre comme l'équivalent d'un champ « Copie » dans un courrier postal : plusieurs enveloppes au même contenu, avec référence de l'ensemble des destinataires dans chaque courrier.	[1..1]	Liste de [1..n] recipient (voir tableau ci-après)

L'élément unique **recipient** contient une liste de **recipient** sur le modèle ci-dessous :

recipient			
Paramètre	**Description**	**Cardinalité**	**Type**
recipient	Identifiant du destinataire	[1..n]	Voir tableau **recipient** ci-après

sender			
Paramètre	**Description**	**Cardinalité**	**Type**

name	Identifiant de l'acteur.	[1..1]	Se compose du senderID tronqué du préfixe "<code_pays>.<domaine>". Exemple : Le senderID "fr.health.samuXXX" devient "samuXXX".
URI	URI de l'acteur.	[1..1]	Se compose sur le modèle suivant : <scheme>:<senderID>, où le scheme est fixe : « **hubex** » (cf 3.4.4). Cf senderId au point 3.4.7 Exemple : hubex:fr.health.samuXXX

recipient			
Paramètre	**Description**	**Cardinalité**	**Type**
name	Identifiant de l'acteur.	[1..1]	Se compose de la valeur du champ explicitAdressValue de l'enveloppe EDXL tronquée du préfixe "<code_pays>.<domaine>". Exemple : Le destintaire "fr.health.samuXXX" devient "samuXXX".
URI	URI de l'acteur.	[1..1]	Se compose sur le modèle suivant : <scheme>:<explicitAdressValue>, où le scheme est fixe : « **hubex** » (cf 3.4.4). Cf explicitAdressValue au point 3.4.8 Exemple : hubex:fr.health.samuXXX

3.4.10. Message de référence et format des acquittements de réception finale

Le message de référence RC-REF permet de faire référence à un message précédemment partagé. Il reprend la *distributionID* du message concerné ainsi que l'ensemble des paramètres du RC-DE, selon le modèle suivant :

RC-REF			
Paramètre	**Description**	**Cardinalité**	**Type**
Ensemble des paramètres du RC-DE			

reference	Référence attachée au message.	[1..1]	Voir tableau **reference** ci-après.

reference			
Paramètre	**Description**	**Cardinalité**	**Type**
distributionID	Identifiant unique du message d'urgence référencé par le message RC-REF.	[1..1]	Chaîne de caractères, issue du champ **distributionID** de l'enveloppe **EDXL-DE** du message acquitté (cf 3.4.3).
refused	Indique si le message acquitté a été refusé ; si le message est refusé l'acquittement doit être accompagné par un message d'erreur	[0..1]	Booléen
errorDistributionID	Identifiant unique du message d'erreur accompagnant l'acquittement	[0..1]	Chaîne de caractères, issue du champ **distributionID** de l'enveloppe **EDXL-DE** du message d'erreur accompagnant l'acquittement (cf 3.4.3).
step	Nomenclature permettant d'identifier les différentes étapes d'intégration et de consultation du message dans le système émetteur	[0..1]	Valeurs énumérées parmi la liste suivante : **RECU**, **ERREUR**, **INTEGRE**, **CONSULTE**, **SUPPRIME**

Les messages RC-REF permettent notamment de transmettre les acquittements de réception finale. Pour cela, le message doit :

- Faire référence au message à acquitter (par son *distributionID*)
- Spécifier EDXL-DE.distributionKind à Ack
- Spécifier RC-DE.kind à Ack

3.4.11. Format des messages d'erreur

Les messages d'erreur envoyés par le Hub Santé sur les files INFO des acteurs sont insérés dans des enveloppes EDXL-DE, comme l'ensemble des messages transitant par le réseau de Hubs.

Leur contenu (dans *embeddedJsonContent* ou *embeddedXMLContent*) respecte le modèle suivant :

Message d'erreur			
Paramètre	**Description**	**Cardinalité**	**Type**
errorCode	Le code de l'erreur ayant conduit au rejet du message.	[1..1]	Voir tableau **ErrorCode** ci-après.
errorCause	La cause de l'erreur. Le distributionID de l'enveloppe EDXL y est précisé si le message a pu être désérialisé, ainsi que	[1..1]	**Chaîne de caractères**

	des éléments plus précis suivant l'erreur relevée.		
referencedDistributionID	DistributionID du message source	[1..1]	**Chaîne de caractères**
sourceMessage	Le message rejeté	[0..1]	Contenu du message rejeté sans contrainte sur ce qu'il contient

ErrorCode		
statusCode	**statusString**	**description**
100	DELIVERY_MODE_INCONSISTENCY	Le message a été rejeté car il n'a pas été publié en mode persistant.
101	NOT_ALLOWED_CONTENT_TYPE	Le message a été rejeté car l'un des deux Content-Type autorisés n'a pas été spécifié.
102	UNRECOGNIZED_MESSAGE_FORMAT	Le message n'a pas pu être désérialisé.
200	SENDER_INCONSISTENCY	Le champ "sender" de l'enveloppe EDXL ne correspond pas à la clé de routage utilisée pour la publication du message.
300	INVALID_MESSAGE	Le message n'est pas conforme aux spécifications techniques (JSON Schema ou XSD).
400	EXPIRED_MESSAGE_BEFORE_ROUTING	Le message n'a pas été reçu par son destinataire, il a expiré sur le Hub avant de lui être délivré.
500	DEAD_LETTERED_QUEUE	Le message n'a pas été reçu par son destinataire, il a expiré avant qu'il ne le dépile.
501	UNROUTABLE_MESSAGE	Le message n'a pas pu être routé vers une queue.

3.4.12. Schémas et exemples de messages

Les schémas (AsyncAPI et XSD) ainsi que des exemples de messages (JSON et XML) sont fournis dans le dossier *docs/DST* du GitHub partagé (voir la section *Repositories* GitHub partagé pour plus de détails).