Coalition pour la provenance et l'authenticité du contenu

Certificats de contenu

Spécifications techniques C2PA

2.2, 01/05/2025 :

Table des matières

Chapitre 1. Introduction

1.1. Aperçu

Avec l'accélération de la diffusion des contenus numériques et la multiplication des techniques de création et d'édition performantes, il est essentiel de déterminer la provenance des médias afin de garantir la transparence, la compréhension et, en fin de compte, la confiance.

Nous assistons actuellement à des défis extraordinaires en matière de confiance dans les médias. Alors que les plateformes sociales amplifient la portée et l'influence de certains contenus via des algorithmes toujours plus complexes et opaques, les contenus mal attribués et mal contextualisés se propagent rapidement. Qu'il s'agisse de désinformation involontaire ou de tromperie délibérée via la désinformation, les contenus non authentiques sont en augmentation.

Actuellement, ceux qui souhaitent inclure des métadonnées sur leur travail ne peuvent pas le faire de manière sécurisée, inviolable et standardisée sur toutes les plateformes. Sans ces informations provenant d'une source reconnue, les éditeurs et les consommateurs ne disposent pas du contexte essentiel pour déterminer l'authenticité des médias.

La provenance permet aux créateurs et aux éditeurs de contenu, indépendamment de leur situation géographique ou de leur niveau d'accès à la technologie, de divulguer des informations sur la manière dont un élément a été créé, comment il a été modifié et ce qui a été modifié. Chaque fois qu'un élément est modifié, sa provenance existante est conservée et chaque nouvelle modification est ajoutée à la provenance. De cette manière, le contenu avec provenance fournit des indicateurs d'authenticité afin que les consommateurs puissent être informés des modifications apportées au contenu. Cette provenance peut inclure les modifications apportées et la source de ces modifications. Cette capacité à fournir une provenance aux créateurs, aux éditeurs et aux consommateurs est essentielle pour faciliter la confiance en ligne.

Afin de répondre à ce problème à grande échelle pour les éditeurs, les créateurs et les consommateurs, la Coalition for Content Provenance and Authenticity (C2PA) a élaboré cette spécification technique visant à garantir la provenance et l'authenticité des contenus. Elle est conçue pour permettre l'adoption mondiale et volontaire de techniques de provenance numérique grâce à la création d'un riche écosystème d'applications compatibles avec la provenance numérique pour un large éventail de personnes et d'organisations, tout en répondant aux exigences de sécurité appropriées.

Cette spécification s'appuie, et continuera de s'appuyer, sur des scénarios, des flux de travail et des exigences recueillis auprès d'experts du secteur et d'organisations partenaires, notamment la Project Origin Alliance et la Content Authenticity Initiative (CAI). Il est également possible que les organismes de réglementation et les agences gouvernementales utilisent cette spécification pour établir des normes en matière de provenance numérique.

1.2. Champ d'application

Cette spécification décrit les aspects techniques de l'architecture C2PA, un modèle permettant de stocker et d'accéder à des informations vérifiables cryptographiquement dont la fiabilité peut être évaluée sur la base d'un modèle de confiance défini. Ce document contient des informations sur la manière de créer et de traiter un manifeste C2PA et ses composants, y compris l'utilisation de la technologie de signature numérique pour permettre la détection des altérations et établir la confiance.

Avant d'élaborer cette spécification, la C2PA a créé nos principes directeurs qui nous ont permis de rester concentrés sur la garantie que la spécification puisse être utilisée de manière à respecter la vie privée et le contrôle personnel des données, tout en gardant un œil critique sur les abus et les utilisations détournées potentiels. Par exemple, les responsables de la mise en œuvre de cette spécification sont fortement encouragés à donner aux créateurs et aux éditeurs de ressources multimédias la possibilité de contrôler si certaines données de provenance sont incluses.

IMPORTANT

Extrait de la section consacrée aux objectifs généraux des principes directeurs :

> Les spécifications C2PA NE DOIVENT PAS fournir de jugements de valeur quant à la « qualité » d'un ensemble donné de données de provenance, mais simplement indiquer si les assertions qu'il contient peuvent être validées comme étant associées à l'actif sous-jacent, correctement formées et exemptes de toute altération.

Il est important que la spécification n'ait pas d'impact négatif sur l'accessibilité du contenu pour les consommateurs.

D'autres documents de la C2PA traiteront de considérations spécifiques à la mise en œuvre, telles que les expériences utilisateur attendues et les détails de notre modélisation des menaces et des dommages.

1.3. Aperçu technique

Les informations C2PA comprennent une série de déclarations qui couvrent des domaines tels que la création d'actifs, les actions d'édition, les détails des périphériques de capture, les liens vers le contenu et bien d'autres sujets. Ces déclarations, appelées assertions, constituent la provenance d'un actif donné et représentent une série de signaux de confiance qui peuvent être utilisés par un être humain pour améliorer son opinion sur la fiabilité de cet actif. Les assertions sont regroupées avec des informations supplémentaires dans une entité signée numériquement appelée « revendication ». Cette revendication est signée numériquement par le générateur de revendications au nom du signataire, à l'aide des informations d'identification de signature du signataire, ce qui produit la signature de la revendication.

Ces assertions, revendications et signatures de revendication sont toutes regroupées dans une unité vérifiable appelée « manifeste C2PA » (voir figure 1, « Un manifeste C2PA et ses composants ») par un composant matériel ou logiciel appelé « générateur de revendications ». L'ensemble des manifestes C2PA, tels qu'ils sont stockés dans les informations d'identification du contenu de l'actif, représentent ses données de provenance.

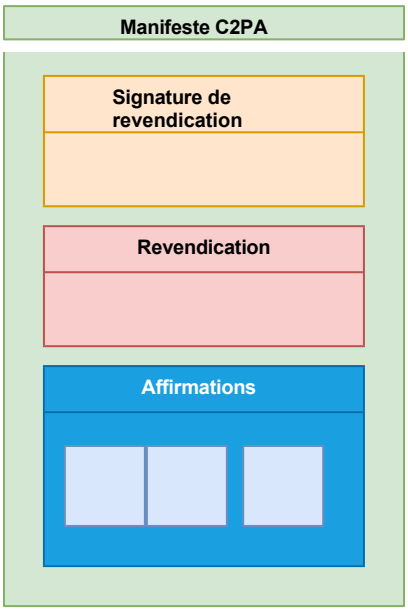

Figure 1. Un manifeste C2PA et ses éléments constitutifs

1.3.1. Établir la confiance

La base des décisions de confiance dans C2PA, notre modèle de confiance, est l'identité du signataire associé à la clé de signature cryptographique utilisée pour signer une revendication dans un manifeste C2PA. Les signatures des revendications des manifestes C2PA, lorsqu'elles sont associées à des horodatages fiables, peuvent être soumises à un processus de validation indéfini afin de déterminer si les revendications ont été signées alors que les informations d'identification de signature étaient valides et n'avaient pas été révoquées.

1.3.2. Un exemple

Un scénario très courant serait celui d'un utilisateur prenant une photo avec son appareil photo (ou son téléphone) compatible C2PA. Dans ce cas, l'appareil photo créerait un manifeste contenant certaines assertions, notamment des informations sur l'appareil photo lui-même, une vignette de l'image et certains hachages cryptographiques qui lient la photo au manifeste. Ces assertions seraient ensuite répertoriées dans la revendication, qui serait signée numériquement, puis l'ensemble du manifeste C2PA (voir figure 2, « Exemple de manifeste C2PA d'une photo ») serait intégré dans le fichier JPEG de sortie. Ce manifeste C2PA resterait valide indéfiniment.

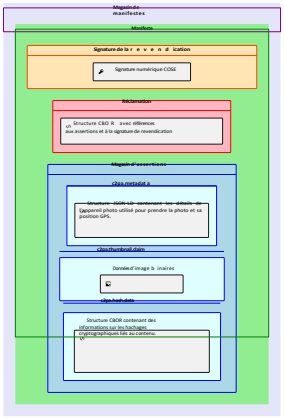

Figure 2. Exemple de manifeste C2PA d'une photographie

Un consommateur de manifeste, tel qu'un validateur C2PA, aide les utilisateurs à établir la fiabilité de l'actif en validant d'abord la signature numérique et les informations d'identification associées. Il vérifie également la validité de chacune des assertions et présente les informations qu'elles contiennent, ainsi que la signature, à l'utilisateur de manière à ce qu'il puisse prendre une décision éclairée quant à la fiabilité du contenu numérique.

1.3.3. Objectifs de conception

Lors de la création de l'architecture C2PA, il était important de définir des objectifs clairs pour le travail afin de garantir que la technologie soit utilisable sur un large éventail d'implémentations matérielles et logicielles à travers le monde et accessible à tous. Ces objectifs sont présentés dans le tableau 1, « Objectifs de conception C2PA ».

Tableau 1. Objectifs de conception du C2PA

Objectif	**Description**
Confidentialité	Permettre aux utilisateurs de contrôler la confidentialité de leurs informations, y compris les données de consommation et les informations enregistrées dans la provenance
Responsabilité	Veiller à ce que les consommateurs puissent déterminer la provenance d'un actif
Évolutivité	Permettre la création/consommation/validation de la provenance des médias à la même échelle que la création/consommation de médias sur le web.
Extensibilité	Garantir que les futurs fournisseurs de métadonnées et d'informations d'identification puissent ajouter leurs informations sans avoir besoin de l'intervention ou de l'approbation de la C2PA
Interopérabilité	Garantir que différentes implémentations puissent fonctionner ensemble sans ambiguïté
Applicabilité à l'ensemble du flux de travail	Conserver la provenance de l'actif à travers plusieurs outils, depuis sa création jusqu'à toutes les modifications et publications/distributions ultérieures.
Minimalisme technologique	Ne créer que le minimum de technologies nouvelles requises dans la spécification en s'appuyant sur des techniques antérieures bien établies
Sécurité	Concevoir de manière à garantir que les consommateurs puissent avoir confiance dans l'intégrité et la source de provenance, et veiller à ce que la conception soit examinée par des experts.
Omniprésence du contenu	Permettre l'inclusion de la provenance pour tous les types de médias courants, y compris les documents.
Flexibilité locale	Permettre le stockage et la consommation/validation de la provenance en ligne et hors ligne (actifs uniquement).
Universalité mondiale	Conçu pour répondre aux besoins des utilisateurs intéressés à travers le monde
Accessibilité	Veiller à ce que la technologie puisse être utilisée conformément aux normes d'accessibilité reconnues, telles que les WCAG
Dommages et utilisation abusive	Concevoir pour prévenir et atténuer les dommages potentiels, y compris les menaces pour les droits humains et les risques disproportionnés pour les groupes vulnérables
Évolution	Révision continue des spécifications par rapport à ces objectifs afin de garantir qu'ils restent notre priorité.

Chapitre 2. Glossaire

2.1. Termes introductifs

2.1.1. Acteur

Une personne ou un élément non humain (matériel ou logiciel) qui participe à l'écosystème C2PA. Par exemple : un appareil photo (dispositif de capture), un logiciel de retouche d'images, un service cloud ou la personne qui utilise ces outils.

REMARQUE Une organisation ou un groupe d'acteurs peut également être considéré comme un acteur dans l'écosystème C2PA.

2.1.2. Générateur de revendication

Acteur non humain (matériel ou logiciel) qui génère la revendication relative à un actif ainsi que la signature de la revendication, conduisant ainsi au manifeste C2PA associé à l'actif.

2.1.3. Signataire

Le détenteur d'une clé privée utilisée pour signer la revendication. Le signataire est identifié par le sujet de l'attestation.

2.1.4. Consommateur de manifeste

Un acteur qui consomme un actif associé à un manifeste C2PA dans le but d'obtenir les données de provenance du manifeste C2PA.

2.1.5. Validateur

Un consommateur de manifeste dont le rôle est d'effectuer les actions décrites dans la validation.

2.1.6. Action

Une opération effectuée par un acteur sur un actif. Par exemple, « créer », « intégrer » ou « appliquer un filtre ».

2.2. Actifs et contenu

2.2.1. Contenu numérique

La partie d'un actif qui représente le contenu réel, comme les pixels d'une image, ainsi que toutes les métadonnées techniques supplémentaires nécessaires à la compréhension du contenu (par exemple, un profil de couleur ou des paramètres d'encodage).

2.2.2. Métadonnées de l'actif

Informations non techniques sur l'actif et son contenu numérique.

2.2.3. Ressource

Fichier ou flux de données contenant du contenu numérique, des métadonnées d'actif et, éventuellement, un manifeste C2PA.

REMARQUE	Aux fins de la présente définition, nous élargirons la définition classique du terme « fichier » afin d'y inclure les données natives du cloud et générées dynamiquement.

2.2.4. Ressource dérivée

Un actif dérivé est un actif créé à partir d'un actif existant et sur lequel des actions ont été effectuées afin de modifier son contenu numérique.

EXEMPLE : un flux audio qui a été raccourci ou un document auquel des pages ont été ajoutées.

2.2.5. Représentation d'un actif

Représentation d'un actif (soit en tant que partie d'un actif, soit en tant qu'actif entièrement nouveau) auquel une action de « transformation non éditoriale » (par exemple, recodage ou redimensionnement) a été appliquée.

EXEMPLE : fichier vidéo réencodé pour réduire la résolution d'écran ou la bande passante réseau.

2.2.6. Ressource composée

Une ressource composée est une ressource créée à partir d'un ensemble de plusieurs parties ou fragments de contenu numérique (appelés « ingrédients ») provenant d'une ou plusieurs autres ressources. Lorsqu'elle est créée à partir d'une ressource existante, il s'agit d'un cas particulier de ressource dérivée. Cependant, une ressource composée peut également être créée à partir d'une « page blanche ».

EXEMPLES :

- Une vidéo créée en important des clips vidéo et des segments audio existants dans une « ardoise vierge ».
- Une image où une autre image est importée et superposée à l'image de départ.

2.2.7. Transformation éditoriale

Type de transformation qui modifie soit l'intention, soit la signification, soit les deux, du contenu numérique.

2.3. Aspects fondamentaux de la C2PA

2.3.1. Assertion

Structure de données qui représente une déclaration faite (ou « créée ») par le signataire ou simplement recueillie au moment de la génération de la revendication, concernant l'actif. Ces données font partie du manifeste C2PA.

2.3.2. Réclamation

Une structure de données signée numériquement et inviolable qui fait référence à un ensemble d'assertions concernant un actif et aux informations nécessaires pour représenter le contenu contraignant. Si certaines assertions ont été supprimées, une déclaration à cet effet est incluse. Ces données font partie du manifeste C2PA.

2.3.3. Signature de la revendication

La signature numérique sur la revendication créée à l'aide de la clé privée détenue par un signataire. La signature de la revendication fait partie du manifeste C2PA.

2.3.4. Manifeste C2PA

Ensemble d'informations sur la *provenance* d'un actif basé sur la combinaison d'une ou plusieurs assertions (y compris les liaisons de contenu), d'une seule revendication et d'une signature de revendication. Un manifeste C2PA fait partie d'un magasin de manifestes C2PA.

REMARQUE Un manifeste C2PA peut faire référence à d'autres manifestes C2PA.

2.3.5. Magasin de manifestes C2PA

Une collection de manifestes C2PA qui peuvent être intégrés à un actif ou être externes à celui-ci.

2.3.6. Certificat de contenu

Il s'agit du terme non technique préféré pour désigner un manifeste C2PA. Le magasin de manifestes C2PA représente donc les informations d'identification de contenu d'un actif.

Les informations d'identification de contenu font également référence à la technologie C2PA dans son ensemble et sont donc essentiellement traitées comme un nom pluriel. Si un manifeste C2PA est une information d'identification de contenu, alors plusieurs manifestes C2PA ou le concept plus large et universel sont des informations d'identification de contenu.

2.3.7. Manifeste actif

Le dernier manifeste dans la liste des manifestes C2PA à l'intérieur d'un magasin de manifestes C2PA, qui est celui contenant l'ensemble des liaisons de contenu pouvant être validées.

2.3.8. Provenance

Concept logique permettant de comprendre l'historique d'un actif et ses interactions avec des acteurs et d'autres actifs, tel que représenté par les données de provenance.

2.3.9. Données de provenance

Ensemble des manifestes C2PA pour un actif et, dans le cas d'un actif composé, ses composants.

> **REMARQUE** Un manifeste C2PA peut faire référence à d'autres manifestes C2PA.

2.3.10. Authenticité

Propriété d'un contenu numérique comprenant un ensemble de faits (tels que les données de provenance et les liens physiques) dont l'authenticité peut être vérifiée par cryptographie.

2.3.11. Liaison de contenu

Informations qui associent un contenu numérique à un manifeste C2PA spécifique associé à un actif spécifique, sous forme de liaison forte ou de liaison faible.

2.3.12. Liaison forte

Un ou plusieurs hachages cryptographiques qui identifient de manière unique tout ou partie de l'actif.

2.3.13. Liaison souple

Identifiant de contenu qui est soit (a) statistiquement non unique, tel qu'une empreinte digitale, soit (b) intégré sous forme de filigrane invisible dans le contenu numérique identifié.

2.3.14. Signaux de confiance

Ensemble d'informations pouvant éclairer le jugement d'un consommateur manifeste sur la fiabilité d'un actif. Elles s'ajoutent au signataire sur lequel repose le modèle de confiance fondamental.

2.3.15. Liste de confiance C2PA

Liste gérée par la C2PA des ancres de confiance de certificats X.509 qui délivrent des certificats aux signataires matériels et logiciels qui les utilisent pour signer des revendications.

2.4. Conditions supplémentaires

2.4.1. Certificat de contenu durable

Un certificat de contenu durable est un certificat de contenu pour lequel il existe un ou plusieurs liens souples qui permettent sa découverte dans un référentiel manifeste.

2.4.2. Empreinte

Ensemble de propriétés inhérentes calculables à partir d'un contenu numérique qui identifie ce contenu ou ses quasi-duplicatas.

> **EXEMPLE :** Un actif peut être séparé de son manifeste C2PA en raison de la suppression ou de la corruption des métadonnées de l'actif. Une empreinte digitale du contenu numérique de l'actif pourrait être utilisée pour effectuer une recherche dans une base de données afin de récupérer l'actif avec un manifeste C2PA intact.

2.4.3. Filigrane invisible

Informations intégrées de manière pratiquement imperceptible pour l'œil humain dans le contenu numérique d'un actif, pouvant être utilisées, par exemple, pour identifier de manière unique l'actif ou pour stocker une référence à un manifeste C2PA.

2.4.4. Filigrane visible

Composante perceptible du contenu numérique contenant des informations exploitables par l'homme sur la provenance de l'actif.

2.4.5. Référentiel de manifestes

Référentiel dans lequel les manifestes C2PA et les magasins de manifestes C2PA peuvent être placés, et qui peut être consulté à l'aide d'une liaison de contenu.

2.5. Présentation

Cette image montre comment tous ces différents éléments s'assemblent pour représenter l'architecture C2PA.

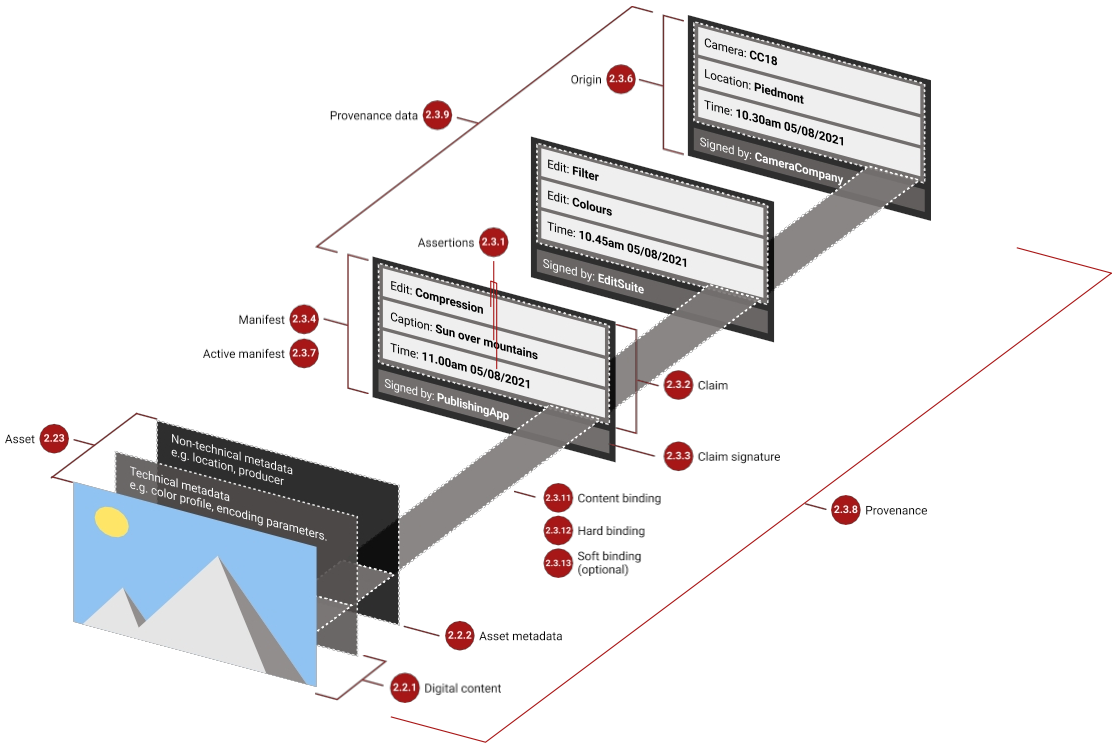

Figure 3. Éléments de la C2PA

Chapitre 3. Références normatives

3.1. Formats de base

- CBOR
- JSON
- JSON-LD
- Format universel de boîte de métadonnées JPEG (JUMBF)

3.2. Schémas

- CDDL
- Schéma JSON
- Initiative Dublin Core pour les métadonnées

3.3. Signatures numériques et électroniques

- Syntaxe des messages cryptographiques (CMS)
- Protocole d'horodatage PKI Internet X.509
- Algorithmes et identifiants pour l'infrastructure à clé publique Internet X.509 Profil de certificat et de liste de révocation de certificats (CRL)
- Infrastructure à clé publique Internet X.509 : algorithmes et identifiants supplémentaires pour DSA et ECDSA
- Algorithmes de hachage sécurisés américains
- Protocole OCSP (Online Certificate Status Protocol)
- Algorithmes Web JSON (JWA)
- PKCS #1 : spécifications cryptographiques RSA version 2.2
- Algorithme de signature numérique Edwards-Curve (EdDSA)
- Signature et chiffrement d'objets CBOR (COSE)
- Utilisation des algorithmes RSA avec les messages COSE
- Identificateurs d'algorithmes pour Ed25519, Ed448, X25519 et X448 à utiliser dans l'infrastructure à clé publique Internet X.509
- Certificat X.509 Utilisation étendue de la clé (EKU) à usage général pour la signature de documents
- Signature et chiffrement d'objets CBOR (COSE) : paramètres d'en-tête pour le transport et la référence des certificats X.509
- Infrastructure à clé publique Internet X.509 : logotypes dans les certificats X.509
- Signatures électroniques avancées JSON (JAdES)

3.4. Formats intégrables

- Format de fichier multimédia de base ISO (BMFF)
- PDF 1.7
- PDF 2.0
- JPEG 1
- JPEG XT, ISO/IEC 18477-3
- JPEG XL, ISO/IEC 18181-2:2024
- PNG
- SVG
- GIF
- ID3
- Négatif numérique ou DNG
- TIFF/EP
- TIFF v6)
- RIFF
- Format multi-images (MPF)
- Format de police ouvert
- OpenType

3.5. Autre

- Plateforme de métadonnées extensible (XMP)
- Sérialisation JSON-LD de XMP
- Norme IPTC pour les métadonnées photographiques
- Exif
- UUID
- Noms de ressources uniformes (URN)
- Identifiants universels uniques (UUID)
- ISO 8601
- RFC 3339
- RFC 2326
- Fragments multimédias

- Modèle de données d'annotation Web
- Format de données compressées Brotli
- RFC 5646, BCP 47

Chapitre 4. Termes standard

Les mots clés « DOIT », « NE DOIT PAS », « OBLIGATOIRE », « DOIT », « NE DOIT PAS », « DEVRAIT », « NE DEVRAIT PAS »,
Les termes « RECOMMANDÉ », « NON RECOMMANDÉ », « PEUT » et « FACULTATIF » utilisés dans le présent document doivent être interprétés conformément aux dispositions des documents BCP 14, RFC 2119 et RFC 8174, quelle que soit leur casse (majuscules, minuscules ou mixtes).

Chapitre 5. Gestion des versions

5.1. Compatibilité

Au fur et à mesure de l'évolution de la spécification Content Credentials, des constructions telles que les étiquettes de boîte, les assertions (et leurs champs), les revendications et les horodatages ont également évolué. De nouvelles assertions ont été ajoutées, et certaines assertions existantes ainsi que la revendication ont été mises à jour avec des champs supplémentaires. De plus, certaines constructions ont été dépréciées. Dans cette spécification, lorsqu'une construction est marquée comme dépréciée, cela signifie qu'un générateur de revendications ne doit pas écrire cette construction (ou valeur), mais qu'un validateur doit la lire.

Afin de faciliter l'interopérabilité entre les générateurs de revendications et les validateurs, un générateur de revendications déclare la version de la spécification qu'il utilise pour générer la revendication. Lorsqu'un générateur de revendications déclare qu'il utilise une version de la spécification, il déclare que le manifeste actif de l'actif est produit conformément à cette version de la spécification et ne contient donc aucune construction obsolète répertoriée sous cette version de la spécification dans le tableau 19, « Statut des constructions » de l'annexe C, *Considérations relatives à l'obsolescence.*

REMARQUE La présente spécification ne dicte pas la manière technique spécifique de cette déclaration, mais il est prévu que des orientations soient fournies par d'autres moyens.

Un validateur doit être compatible avec au moins une version de la spécification, mais peut être compatible avec d'autres versions. Un validateur compatible avec une version spécifique de la spécification doit prendre en charge toutes les constructions non obsolètes répertoriées pour cette version. Si le validateur rencontre un manifeste qui utilise des constructions provenant d'une version de la spécification qu'il ne prend pas en charge (soit parce qu'elles sont obsolètes, soit parce qu'elles sont inconnues), il peut ignorer la construction obsolète et traiter le reste du manifeste comme si cette construction n'était pas présente. Le validateur peut également traiter l'ensemble du manifeste comme ayant une provenance inconnue, en renvoyant le code d'état `ingredient.unknownProvenance` ou `manifest.unknownProvenance`, selon le cas.

5.2. Historique des versions

5.2.1. 2.2 - mai 2025

Cette version se concentre sur les modifications techniques et rédactionnelles apportées à la spécification afin de clarifier certaines des nouvelles fonctionnalités de la version 2.1, tout en répondant aux demandes des développeurs. La spécification a été mise à jour afin de refléter les dernières bonnes pratiques dans ce domaine.

- Ajout de nouvelles spécifications supplémentaires pour l'API Soft Binding Resolution
- Ajout de nouveaux champs à l'assertion d'ingrédient pour indiquer la récupération du manifeste à liaison souple
- Ajout de la prise en charge des ressources en plusieurs parties, telles que les photos animées Android
- Ajout de la prise en charge de l'ajout d'horodatages et d'informations de révocation dans un manifeste de mise à jour, remplaçant les manifestes d'horodatage
- Ajout de la prise en charge d'un « temps de création de signature revendiqué »

- Ajout de la prise en charge du nouvel EKU `c2pa-kp-claimSigning`
- Utilisation restreinte de la liste de confiance C2PA aux certificats avec l'EKU `c2pa-kp-claimSigning`
- Introduit `digitalSourceType` Valeurs d' `http://c2pa.org/digitalsourcetype/trainedAlgorithmicData` (remplaçant `c2pa.trainedAlgorithmicData`) et `http://c2pa.org/digitalsourcetype/empty`
- Remplacement des boîtes de données par des assertions de données intégrées
- Fourni des conseils supplémentaires sur la remise à zéro des assertions expurgées
- Clarification de l'utilisation de `created_assertions` et `gathered_assertions` en ce qui concerne le modèle de confiance
- Clarification de la terminologie relative aux termes « signataire » et « générateur de revendications », en ce qui concerne leurs rôles
- Modifications et améliorations apportées à diverses assertions contraignantes
    - Autoriser `c2pa.hash.data` à exclure les sections de métadonnées classiques d'un actif
    - Ajouter la prise en charge des exclusions dans l'assertion `c2pa.hash.boxes`
    - Ajouter la prise en charge de l'utilisation d'une assertion `c2pa.hash.bmff` dans un manifeste de mise à jour
- Clarification des boîtes JUMBF autorisées dans le magasin d'assertions
- Clarification du traitement de la révocation des certificats
- Clarification de la validation des horodatages
- Améliorations et clarifications apportées aux assertions d'action
- Améliorations apportées aux assertions de liaison souple
- Refonte des diagrammes de hachage BMFF pour plus de clarté et d'exactitude
- Suppression de l'obligation de référencer tous les manifestes dans un magasin de manifestes

5.2.2. 2.1 - Septembre 2024

Cette version se concentre sur les modifications techniques et rédactionnelles apportées à la spécification dans le but d'améliorer la sécurité et la fiabilité des informations d'identification de contenu. Toutes les vulnérabilités de sécurité rendues publiques ont été corrigées et la spécification a été mise à jour afin de refléter les dernières bonnes pratiques dans ce domaine.

- Définitions claires des états des manifestes et des actifs
    - Manifestes bien formés
    - Manifestes valides
    - Manifestes fiables
    - Actifs valides
- Définitions et processus clairs pour la gestion de la dépréciation et du versionnage
- Nouvel espace de noms URN `c2pa` pour l'étiquetage des manifestes !

    - incluant un ABNF entièrement spécifié
        - Nouvelle assertion `v3 sur les ingrédients`
    - Prend en charge des modèles plus riches de workflows basés sur les ingrédients.
    - Prise en charge des `types de données` et `des signatures de revendication`.
    - Champs renommés pour être plus cohérents avec les autres assertions.
    - Ajout de nouveaux champs de statut de validation pour accompagner les nouvelles informations de statut
    - `dc:title` et `dc:format` sont désormais facultatifs
- Nouvelle assertion `c2pa.hash.bmff.v3`
    - Prise en charge du hachage de tailles de blocs fixes et variables pour les actifs basés sur BMFF
- Nouveau manifeste d'horodatage
    - Établir une « durée d'existence » pour un actif donné.
    - Similaire à un manifeste de mise à jour, mais avec le signataire étant un TSA
- Modèle amélioré pour effectuer un horodatage standard RFC 3161.
    - Horodatage `sigTst2` & CTT
    - Introduction de la nouvelle liste de confiance C2PA TSA
- Améliorations de la validation
    - Instructions de validation détaillées pour toutes les assertions standard
    - La validation des ingrédients est désormais requise lors de l'utilisation de l'assertion `relative aux ingrédients`
    - Validation étendue des ingrédients pour fournir des informations plus détaillées sur leur statut
    - Prise en charge de la validation des allégations expurgées dans les ingrédients
    - Ajout d'exigences détaillées pour la validation des horodatages
    - Les URI hachés vers les boîtes de données et toutes les boîtes personnalisées sont désormais validés
    - Procédure définie pour le traitement des manifestes avec des identifiants uniques correspondants
    - Traitement des « manifestes orphelins » dans le processus de validation
    - De nombreux nouveaux codes d'état de validation, y compris un nouveau type de code « informatif »
- Améliorations de la documentation et de la sécurité des méthodes de hachage
    - Ressources basées sur BMFF
    - « Boîtes générales »
    - ZIP
- La section relative à l'intégration des formats a été déplacée dans une annexe distincte.
    - Ajout de la prise en charge du format JPEG-XL

- Améliorations apportées aux liaisons logicielles
- Améliorations apportées aux assertions d'action
    - Une balise `c2pa.created` ou `c2pa.opened` est désormais obligatoire dans un manifeste standard
    - De nouveaux types d'actions standard ont été ajoutés
    - Il est désormais possible d'avoir plusieurs assertions d'action dans un seul manifeste
    - Les modèles d'action sont désormais mieux expliqués à l'aide d'exemples supplémentaires.
    - Régions d'intérêt basées sur la RFC 3339
- Les différents types/formes d'identifiants uniques pour les actifs ont été clarifiés.
- Ajout d'une prise en charge de compatibilité manquante pour JPEG Trust
- Nettoyage de tous les CDDL, y compris la suppression de tout langage normatif
- Et diverses améliorations rédactionnelles.
    - Redéfinition des étiquettes personnalisées selon un schéma de nommage personnalisé.
    - Intégration dans les PDF.
    - NOMBREUSES améliorations rédactionnelles afin de préparer le document à la normalisation par l'ISO

5.2.3. 2.0 - janvier 2024

Cette version représente un changement significatif par rapport aux versions précédentes. Elle réduit l'utilisation du terme « acteur », qui ne désigne plus les personnes physiques et morales. Outre les listes de confiance configurées par le validateur, elle introduit également une nouvelle liste de confiance par défaut, la « liste de confiance C2PA », qui est destinée à couvrir les certificats délivrés au matériel et aux logiciels. Ce changement philosophique a entraîné les modifications fonctionnelles suivantes dans la spécification :

- Seuls les certificats X.509 peuvent être utilisés pour la signature.
- Améliorations apportées aux sections Validation et Modèle de confiance
    - Introduction des concepts de manifestes C2PA « bien formés » et « valides ».
    - Clarification de divers aspects du processus de validation
- Amélioration du traitement des métadonnées
    - suppression des assertions de métadonnées Exif, IPTC et Schema.org obsolètes
    - défini un nouveau concept général d'« assertion de métadonnées »
    - `c2pa.metadata` n'autorise qu'un ensemble fixe de schémas et de valeurs
    - le processus de création de `c2pa.metadata` est désormais documenté plus en détail
    - les sections de traitement XMP ont été remaniées pour refléter les changements pertinents
    - amélioration des recommandations concernant le hachage des emplacements de métadonnées standard en dehors du manifeste
- Suppression de la section « W3C Verifiable Credentials »

    - Suppression de toutes les références à celle-ci et au VC Store.
    - Suppression du champ « `actors` » de l'assertion « actions ».
    - Suppression des personnes identifiées des métadonnées d'assertion
- Suppression de l'assertion « Formation et exploration de données »
- Suppression de l'assertion « Recommandations »

En outre, les modifications suivantes ont été apportées afin d'améliorer divers aspects de la spécification :

- Version v2 de la revendication.
    - Suppression des champs obsolètes et inutilisés
    - Division `des assertions` en `created_assertions` et `gathered_assertions`
    - N'autorise qu'un seul générateur de revendication, qui doit être le signataire
    - `claim-generator-info` dispose désormais d'un champ `operating_system` spécifique
- Le hachage basé sur des boîtes est désormais fortement recommandé pour tout format qui le prend en charge.
- Suppression de l'assertion `c2pa.hash.bmff` obsolète
- Ajout d'une nouvelle action `c2pa.watermarked`
- Les actions `c2pa.font` sont désormais simplement des actions `de police`
    - De même, `c2pa.font.info` est désormais simplement `font.info`
- Nettoyage du rendu des schémas CDDL
- Mise à jour de certaines références normatives et suppression des notes concernant les versions futures
- Nombreuses améliorations rédactionnelles, y compris la correction de liens

5.2.4. 1.4 - Novembre 2023

- Ajout de la prise en charge de l'intégration d'un manifeste C2PA dans un format ZIP (par exemple, EPUB, OOXML, ODF, OpenXPS).
- Les manifestes peuvent désormais être compressés dans une boîte `brob` spéciale.
- Ajout de la prise en charge du hachage de plusieurs fichiers, également appelé collection.
- Ajout de nouvelles zones d'intérêt pour les formats textuels (par exemple, PDF, Office, EPUB, etc.)
- Ajout d'une nouvelle assertion `c2pa.metadata` pour prendre en charge Exif, IPTC, Schema.org et XMP
- Révision majeure de la prise en charge de l'intégration TIFF.
- Ajout de la prise en charge de l'intégration des manifestes C2PA dans les polices OpenType et TrueType
- Ajout de la prise en charge des manifestes au niveau des objets dans les fichiers PDF
- Extension de la prise en charge de l'en-tête Link pour les manifestes intégrés
- Clarification des problèmes liés au hachage des boîtes

- Clarification des problèmes liés à la signature, notamment l'horodatage, PKIStatus et la signature de documents EKU
- Alignement avec Exif 3.0
- Améliorations apportées aux schémas CDDL
- Nombreuses améliorations rédactionnelles

5.2.5. 1.3 - Avril 2023

- Nouvelle version v2 de l'assertion d'actions avec prise en charge de nombreuses nouvelles options
- Nouvelle version v2 de l'assertion d'ingrédients avec prise en charge des données intégrées
- Nouvelles assertions de référence d'actif et de type d'actif
- Nouvelles boîtes de données, pour stocker des données arbitraires dans le manifeste
- Nouvelle méthodologie générale de hachage des boîtes pour un hachage plus inclusif de la plage d'octets
- Nouvelles structures de données « Régions d'intérêt » pouvant être appliquées à diverses assertions
- Ajout de la signature de document EKU comme EKU par défaut alternative pour les signataires C2PA lorsqu'un validateur n'est pas configuré avec une liste EKU
- Ajout d'un nouveau champ `digitalSourceType` à utiliser par C2PA
- Ajout de la prise en charge de nombreux nouveaux formats : MPF, WebP, AIFF, AVI, GIF
- Mise à jour du diagramme d'entités pour refléter les ajouts depuis la `version` 1.0
- Mise à jour de la définition de l'en-tête COSE pour les certificats X.509 conformément à la norme RFC 9360
- Mise à jour des recommandations sur l'intégration de PDF et leur relation avec les signatures PDF
- Mise à jour des informations sur le hachage JUMBF et les boutons JUMBF
- Dépréciation de la version 1 du hachage BMFF
- Clarification de l'utilisation de la boîte de protection JUMBF dans un manifeste C2PA
- Clarification de l'exigence spécifique à la C2PA selon laquelle tous les certificats X.509 intermédiaires doivent être inclus dans les signatures COSE.
- Précision apportée sur la validité illimitée des horodatages
- De nombreuses améliorations rédactionnelles !

5.2.6. 1.2 - Octobre 2022

- Ajout d'informations détaillées sur la manière d'intégrer un manifeste C2PA dans un fichier DNG ou TIFF.
- Ajout d'un nouveau champ `digitalSourceType` à Actions
- Modification de `stds.iptc.photometadata` → `stds.iptc` pour prendre en charge les métadonnées vidéo IPTC
- Clarification de la gestion des versions des assertions lors de l'ajout de champs facultatifs

5.2.7. 1.1 - Septembre 2022

- Définition d'un mécanisme pour prendre en charge le hachage salting box
- Nouvelle assertion `c2pa.hash.bmff.v2`, avec des modifications apportées au modèle de hachage, afin d'améliorer la sécurité
- Activation des métadonnées d'assertion pour la revendication
- Remplacement de `claim_generator_hints` par `claim_generator_info`
- Ajout d'une nouvelle assertion pour prendre en charge le concept d'approbations
- Améliorations apportées à l'assertion `c2pa.actions`
- Tous les codes d'erreur et d'état sont désormais préfixés par `c2pa`
- Définition d'un mécanisme pour la rédaction des VC du W3C
- Clarification de la validation des EKU dans les certificats
- Révision de l'algorithme de validation afin de refléter les changements techniques
- Corrections apportées aux schémas CDDL et JSON afin de les aligner sur le texte normatif
- Révision des figures afin de refléter les modifications
- Diverses corrections rédactionnelles et typographiques
- Mise à jour des références normatives (y compris JUMBF et W3C VC Data Model)

5.2.8. 1.0 - Décembre 2021

- Version initiale

Chapitre 6. Affirmations

6.1. Généralités

On s'attend à ce que chaque générateur de revendications, utilisé par les acteurs du système qui créent ou traitent un actif, crée ou assemble une ou plusieurs assertions concernant le moment, le lieu et la manière dont l'actif a été créé ou transformé. Une assertion est une donnée étiquetée, généralement (mais pas obligatoirement) dans une structure basée sur CBOR, qui représente une déclaration concernant un actif. Certaines de ces assertions contiendront des informations générées par l'homme (par exemple, un texte alternatif pour l'accessibilité), tandis que d'autres proviendront de machines (logiciels/matériels) fournissant les informations qu'elles ont générées (par exemple, le type d'appareil photo).

Voici quelques exemples d'assertions :

- les métadonnées (par exemple, les informations sur l'appareil photo telles que le fabricant ou l'objectif) ;
- actions effectuées sur l'actif (par exemple, recadrage, correction des couleurs) ;
- miniature de l'actif ou de ses composants ;
- liens de contenu (par exemple, hachages cryptographiques).

Certaines assertions peuvent être expurgées par des revendications ultérieures (voir la section 6.8, « Expurgation des assertions »), mais elles ne peuvent pas être modifiées une fois qu'elles ont été intégrées à une revendication.

6.2. Étiquettes

6.2.1. Espaces de noms

Les valeurs de chaîne dans les structures de données C2PA peuvent être organisées en espaces de noms à l'aide d'un point (`.`) comme séparateur. L'espace de noms C2PA, `c2pa`, doit être le début de toute valeur de chaîne définie dans la présente spécification. Les espaces de noms spécifiques à une entité doivent commencer par le nom de domaine Internet de l'entité, de la même manière que les paquets Java sont définis (par exemple, `com.litware`, `net.fineartschool`).

Les composants séparés par des points d'un espace de noms spécifique à une entité doivent respecter la convention de nommage des variables (`[a-zA-Z0-9][a-zA-Z0-9_-]*`) spécifiée dans la locale POSIX ou C, telle que définie dans l'ABNF ci-dessous (ABNF pour les espaces de noms).

ABNF pour les espaces de noms

```
qualified-namespace = "c2pa" / entity
entité = composant-entité *( « . » composant-entité )
composant-entité = 1( CHIFFRE / ALPHA ) *( CHIFFRE / ALPHA / « - » / « _ » )
```

6.2.2. Dénomination des étiquettes

Chaque assertion possède une étiquette définie soit par les spécifications C2PA, soit par une entité externe. Ces étiquettes sont des chaînes de caractères qui sont nommées, comme décrit dans la clause précédente, ou par une entité. Les étiquettes les plus courantes seront définies dans l'espace de noms `c2pa`, mais les étiquettes peuvent utiliser n'importe quel espace de noms qui respecte les conventions. Les étiquettes sont également versionnées selon un schéma simple d'incrémentation d'entiers (par exemple, `c2pa.actions.v2)`. Si aucune version n'est fournie, elle est considérée comme `v1.` La liste des étiquettes connues du public se trouve au chapitre 18, *Assertions standard C2PA.*

REMARQUE Les versions précédentes de ce document prévoyaient également l'utilisation d'espaces de noms pour les normes bien établies. mais cela a été remplacé par le simple fait de les avoir via des espaces de noms spécifiques à l'entité (par exemple, `org.iso`, `org.w3`).

ABNF pour les étiquettes d'assertion

```
étiquette-avec-espace-de-noms = étiquette-avec-
espace-de-noms qualifié espace-de-noms qualifié = «
c2pa » / entité
entité = composant-entité *( « . » composant-entité )
composant-entité = 1( CHIFFRE / ALPHA ) *( CHIFFRE / ALPHA / « - » / « _ » )
étiquette = 1*( « . » composant-étiquette )
composant-étiquette = 1( CHIFFRE / ALPHA ) *( CHIFFRE / ALPHA / « - » / « _ » )
```

Les composants séparés par des points d'une étiquette suivent la convention de nommage des variables (`[a-zA-Z][a-zA-Z0-9_-]*`) spécifiée dans la locale POSIX ou C, avec la restriction que l'utilisation d'un caractère de soulignement répété (___) est réservée à l'étiquetage de plusieurs assertions du même type.

6.3. Gestion des versions

Lorsque le schéma d'une assertion est modifié, cela doit être fait de manière rétrocompatible. Cela signifie que de nouveaux champs peuvent être ajoutés et que les champs existants peuvent être marqués comme obsolètes (c'est-à-dire qu'ils peuvent être lus, mais jamais écrits). Les champs existants ne doivent pas être supprimés. L'étiquette serait alors composée d'un numéro de version incrémenté, par exemple en passant de `c2pa.action` (obsolète) à `c2pa.action.v2.`

Étant donné que l'ajout de champs facultatifs peut être effectué tout en conservant la rétrocompatibilité, ces champs peuvent être ajoutés au schéma d'une assertion existante sans modifier le numéro de version.

Les champs obsolètes pour les assertions standard C2PA doivent être indiqués au chapitre 18, *Assertions standard C2PA*. Les générateurs de revendications ne doivent pas insérer de données dans les champs d'assertion obsolètes lors de la création d'assertions.

Dans les situations où une modification non rétrocompatible est nécessaire, au lieu d'augmenter le numéro de version de l'étiquette, l'assertion doit recevoir une nouvelle étiquette.

REMARQUE Par exemple, `c2pa.ingredient` pourrait être remplacé par le nom fictif `c2pa.component`.

6.4. Instances multiples

Plusieurs assertions du même type peuvent apparaître dans le même manifeste, mais comme les assertions sont référencées par les revendications via

leur étiquette, les étiquettes d'assertion doivent être uniques. Pour ce faire, on ajoute un double trait de soulignement et un index croissant de manière monotone à l'étiquette. Par exemple, si un manifeste contient une seule assertion de type `c2pa.metadata`, le label de l'assertion sera `c2pa.metadata`. Si un manifeste contient trois assertions de ce type, les labels seront `c2pa.metadata`, `c2pa.metadata__1` et `c2pa.metadata__2`.

Lorsqu'une étiquette comprend un numéro de version, ce numéro fait partie intégrante de l'étiquette. Ainsi, lorsqu'il existe plusieurs instances, le numéro d'instance suit l'étiquette, par exemple `c2pa.ingredient.v2 2`.

6.5. Validation du schéma

Les schémas fournis dans ce document, ainsi que ceux lisibles par machine qui peuvent être téléchargés sur le site web de la C2PA, ne doivent être utilisés que pour aider à comprendre la syntaxe à lire ou à écrire. Il n'est pas nécessaire, ni recommandé, qu'un validateur effectue une quelconque forme de validation de schéma.

6.6. Magasin d'assertions

L'ensemble des assertions référencées par une revendication dans un manifeste sont regroupées dans une construction logique appelée *magasin d'assertions*. Les assertions et le magasin d'assertions doivent être stockés comme décrit dans la section 11.1, « Utilisation de JUMBF » ; en particulier, chaque assertion référencée dans `les assertions créées` ou `rassemblées` (mais pas `les assertions expurgées`) d'une revendication doit être présente dans le magasin d'assertions situé dans le même manifeste C2PA que la revendication.

Chaque manifeste contient un seul magasin d'assertions. Cependant, comme un actif peut être associé à plusieurs manifestes, chacun représentant une série spécifique d'assertions, il peut y avoir plusieurs magasins d'assertions associés à un actif.

6.7. Données intégrées ou stockées en externe

Certaines données d'assertion, en raison de leur taille ou de leur utilisation peu fréquente, peuvent être hébergées en externe. Ces données ne sont pas intégrées dans le magasin d'assertions, mais sont référencées par URI. Cela est réalisé grâce à une assertion de données cloud (voir section 18.11, « Données cloud »). Contrairement aux données d'assertion intégrées, les données cloud ne sont ni récupérées ni validées dans le cadre de la validation du manifeste, et ne sont récupérées et validées que lorsqu'une application en a spécifiquement besoin, selon un ensemble différent de règles de validation, comme décrit à la section 15.10, « Valider les assertions ».

6.8. Rédaction des assertions

Les assertions présentes dans un manifeste intégré à un actif peuvent être supprimées du manifeste de cet actif lorsque celui-ci est utilisé comme ingrédient. Ce processus est appelé rédaction.

La rédaction consiste soit à supprimer l'intégralité de l'assertion du magasin d'assertions du manifeste, soit à conserver le conteneur d'assertions étiqueté, mais en remplaçant les boîtes de contenu JUMBF dans cette assertion par une seule boîte de contenu UUID dont le champ `ID` a la valeur `CAA98EEE-9D4D-F80E-86AD-4DFFCA263973` (appelée UUID de rédaction C2PA)

et dont le champ `DATA` ne contient que des zéros (valeurs binaires `0x00`).

En outre, une mention indiquant qu'un élément a été supprimé doit être ajoutée à la revendication sous la forme d'une référence URI à l'assertion révisée dans le champ `redacted_assertions` de la revendication. Il est également fortement recommandé que le générateur de revendications ajoute une assertion d'action `c2pa.redacted` avec un champ `révisé,` comme décrit dans la section 18.14.4.7, « Paramètres ».

Lors de la rédaction d'une assertion d'ingrédient qui fait référence à un manifeste C2PA, le manifeste associé doit être supprimé du magasin de manifestes C2PA s'il ne reste aucune autre référence à celui-ci après la rédaction.

NOTE

Étant donné que la référence URI de chaque assertion inclut le label de l'assertion, on sait également de quel type d'assertion il s'agit.
(par exemple, vignette, métadonnées, etc.) a été supprimé. Cela permet aux humains et aux machines d'appliquer des règles pour déterminer si la suppression était acceptable.

À moins que la rédaction de l'assertion ne nécessite également une modification du contenu numérique, un manifeste de mise à jour doit être utilisé pour documenter la rédaction, car il atteste que le contenu n'a pas été modifié.

Les générateurs de revendications ne doivent pas expurger les assertions portant l'étiquette `c2pa.actions` ou `c2pa.actions.v2,` car ce type d'assertion représente des informations essentielles pour comprendre l'historique d'un actif. Ils ne doivent pas non plus expurger les liens contraignants vers les assertions de contenu, qu'il s'agisse de `c2pa.hash.data, c2pa.hash.boxes, c2pa.hash.collection.data, c2pa.hash.bmff.v2` (obsolète) ou `c2pa.hash.bmff.v3,` car ces assertions sont nécessaires pour déterminer l'intégrité de l'actif.

NOTE

Lorsque les assertions sont rédigées dans un manifeste d'ingrédients référencé via l'une des assertions d'ingrédients obsolètes (`c2pa.ingredient` ou `c2pa.ingredient.v2`), la validation de cette assertion échouera (comme décrit dans la section 15.11.3, « Validation des assertions d'ingrédients »), car seules les assertions `c2pa.ingredient.v3` prennent en charge la méthode de validation du hachage de signature de revendication, décrite dans la section 15.11.3.3.1, « Méthode de validation du hachage de signature de revendication ».

6.9. Spécifications de l'heure dans les assertions

La spécification par défaut pour une valeur de date et/ou d'heure dans une assertion est le format date/heure sérialisé en CBOR sous le numéro de balise `0` (RFC 8949, 3.4.1) et représenté en CDDL sous le type `tdate`.

Il existe un cas, décrit lors de l'ajout d'une heure de signature revendiquée, où l'heure est représentée comme un type spécial de date/heure CBOR.

Il existe également l'assertion d'horodatage, qui utilise les formats d'horodatage standard décrits dans le processus de signature.

La raison pour laquelle il existe différents types de représentations de date et d'heure est de permettre la représentation la plus appropriée, sur la base des normes existantes en vigueur, pour chaque cas d'utilisation spécifique.

Chapitre 7. Boîtes de données

IMPORTANT	Cette section est conservée à des fins historiques. Le concept de boîte de données a été Obsolète au profit d'une assertion standard qui utilise une boîte de type de contenu JUMBF Embedded File standard pour contenir les données. Pour plus d'informations, voir [_data_box].

7.1. Généralités

Les boîtes de données permettent d'inclure des données arbitraires dans le manifeste C2PA référencé à partir d'une assertion, au lieu de les intégrer directement dans un champ de l'assertion sous forme de chaîne binaire. Ces boîtes de données sont placées dans le magasin de boîtes de données et chacune d'entre elles sera une boîte de type de contenu CBOR (`cbor`) unique.

Les données d'une boîte de données sont fournies directement sous forme de valeur du champ `de données`, qui est un `bstr`, de sorte que toutes les données binaires peuvent être fournies. Le type des données doit être identifié à l'aide du champ `dc:format`, avec un type de média IANA standard.

REMARQUE	IANA suffixes structurés (https://www.iana.org/assignments/media-type-structured-suffix/media-type-structured-suffix.xhtml), tels que `+json` et `+zip`, sont également pris en charge comme valeurs du champ champ `dc:format`.

Parfois, il peut également être nécessaire de fournir un ou plusieurs types d'actifs comme valeur du champ `data_types` afin de clarifier le format et l'utilisation de ces données.

Une boîte de données doit porter l'étiquette `c2pa.data` et respecter les règles relatives aux étiquettes d'assertion en ce qui concerne les instances multiples.

7.2. Schéma et exemple

Le schéma de ce type est défini par la règle `data-box-map` dans la définition CDDL dans CDDL pour la boîte de données :

CDDL pour boîte de données

```
; boîte permettant le stockage de données arbitraires

data-box-map = {
  « dc:format » : format-string, ; type de média IANA des données « data » :
  bstr, ; données textuelles/binaires arbitraires
  ? « data_types » : [1* $asset-type-map], ; informations supplémentaires sur le type des données
}
```

Chapitre 8. Identifiants uniques

8.1. Identification unique des manifestes et des actifs C2PA

Chaque manifeste C2PA est identifié et référencé de manière unique par un nom de ressource uniforme RFC 8141, des URN provenant de l'espace de noms URN `c2pa`, et un actif C2PA est identifié de manière unique par la valeur URN `c2pa` de son manifeste actif. L'ABNF pour l'URN C2PA est décrit par l'ABNF pour l'URN C2PA.

Un URN `c2pa` doit comporter deux composants obligatoires et deux composants facultatifs, dans l'ordre suivant, avec des deux-points (`:`) entre chaque section.

- Identifiant URN (`urn:c2pa`) : OBLIGATOIRE.
- UUID v4, sous forme de chaîne de caractères (conformément à la section 4 de la RFC 9562) : OBLIGATOIRE.
- Chaîne d'identifiant du générateur de revendications : FACULTATIF.
- Chaîne de caractères Version et Reason (comme décrit ci-dessous) : FACULTATIF.

Lorsqu'elle est présente, la chaîne « Identifiant du générateur de revendications » doit comporter au maximum 32 caractères de la gamme ASCII (conformément à la RFC 20), mais qui ne sont pas des caractères de contrôle (RFC 20, 5.2) ou des caractères graphiques (RFC 20, 5.3).

Lorsqu'elle est présente, la chaîne « Version et raison » doit être composée d'un nombre entier positif, suivi d'un trait de soulignement (_) puis d'un autre nombre entier positif. Les détails de chacune de ces valeurs et leur utilisation sont décrits dans la section Manifestes de versionnement en cas de conflits. De plus, lorsqu'une chaîne « Version et raison » est présente, une chaîne « Identifiant du générateur de revendication » doit également être présente, mais elle peut être vide.

ABNF pour C2PA URN

```
c2pa_urn = c2pa-namespace UUID [claim-generator [version-reason]] c2pa-

namespace = « urn:c2pa: »

; cette définition est tirée de la RFC 9562 UUID
= 4hexOctet « - »
              2octets
              hexadécimaux   «
              -    »    2octets
              hexadécimaux   «
              -    »    2octets
              hexadécimaux   «
              -    »    6octets
              hexadécimaux
octet hexadécimal = HEXDIG
HEXDIG DIGIT  = %x30-39
HEXDIG     = DIGIT / « A » / « B » / « C » / « D » / « E » / « F »

; ASCII, mais pas les caractères de contrôle ni les caractères
graphiques visible-char-except-space = %x21-7E / %x80-FF

; l'identifiant du générateur de revendication est une chaîne de 0 à 32 caractères visibles à l'exception des
espaces
; cela signifie qu'une chaîne vide est valide
générateur-de-revendication = « : » identifiant-générateur-de-revendication
identifiant-générateur-de-revendication = 0*32caractères-visibles-à-l'exception-de-l'espace

; version-reason est une chaîne composée d'un nombre entier positif
; suivi d'un trait de soulignement et d'un entier positif
```

```
version-reason = ":" version "_" reason version =
1*DIGIT
raison = 1*DIGIT
```

Exemples :

- `urn:c2pa:F9168C5E-CEB2-4FAA-B6BF-329BF39FA1E4`
- `urn:c2pa:F9168C5E-CEB2-4FAA-B6BF-329BF39FA1E4:acme`
- `urn:c2pa:F9168C5E-CEB2-4FAA-B6BF-329BF39FA1E4:acme:2_1`
- `urn:c2pa:F9168C5E-CEB2-4FAA-B6BF-329BF39FA1E4::2_1`

REMARQUE	Les versions précédentes de cette spécification utilisaient la norme RFC 9562, les UUID URN et avaient l'identifiant de la revendication générateur au début de l'URN. Cependant, cela s'est avéré non conforme à la norme RFC 9562, UUIDs ou à la norme RFC 8141, URNs.

Cet identifiant URN `c2pa` est utilisé dans différentes parties d'un workflow compatible C2PA, par exemple pour identifier un actif en tant qu'ingrédient d'un actif dérivé ou composé.

8.2. Manifestes de versionnement en raison de conflits

Il peut arriver qu'il soit nécessaire de réétiqueter un manifeste C2PA en raison d'un conflit d'identifiants. Par exemple, si un générateur de revendications avait déjà ajouté un manifeste d'ingrédients dans le magasin de manifestes C2PA de l'actif, puis ajouté ultérieurement un autre ingrédient dont le manifeste portait la même étiquette dans son magasin de manifestes, mais que cette dernière version du manifeste était différente en raison, par exemple, d'une manipulation de l'une de ses valeurs d'assertion. Dans un tel cas, la version modifiée du manifeste des ingrédients doit être copiée dans le magasin de manifestes C2PA de l'actif et doit être réétiquetée.

Pour renommer un manifeste :

- Si l'URN actuel ne contient pas de « chaîne d'identifiant du générateur de revendications », le générateur de revendications doit ajouter un :.
- Dans tous les cas, le générateur de revendications doit ajouter un `:` à l'URN, suivi d'un entier croissant de manière monotone, commençant par 1, suivi d'un trait de soulignement (_) puis d'un entier de la liste ci-dessous représentant la raison du réétiquetage.
    - 1 : Conflit avec un autre manifeste C2PA

Par exemple, si le générateur de revendications doit réétiqueter un manifeste C2PA pour la deuxième fois en raison d'un conflit, la chaîne ajoutée serait `:2_1`.

8.3. Identification des actifs non C2PA

Lorsque vous travaillez avec des actifs qui ne contiennent pas de manifeste C2PA, mais qui contiennent des XMP intégrés incluant des valeurs pour `xmpMM:DocumentID` et/ou `xmpMM:InstanceID` telles que définies dans la spécification XMP, partie 2, 2.2, ces valeurs doivent être utilisées comme identifiants pour l'actif.

Lorsqu'il travaille avec des ressources qui ne contiennent pas de manifeste C2PA ni de XMP intégré, le générateur de revendications peut utiliser la méthode de son choix pour lui attribuer un identifiant unique.

8.4. Références URI

8.4.1. URI standard

Toutes les références aux informations contenues dans le manifeste, qu'elles soient stockées en interne dans l'actif (c'est-à-dire intégrées) ou en externe (par exemple dans le cloud), doivent être référencées via des références URI JUMBF telles que définies dans la norme ISO 19566-5:2023, C.2. Ces URI sont normalement utilisées dans le cadre d'une structure de données `hashed_uri` ou `hashed_ext_uri`.

Lorsque la référence concerne un manifeste compressé, l'URI JUMBF ne doit contenir aucune information sur la boîte `brob`, mais l'URI du manifeste est traité comme si le manifeste n'était pas compressé. Cela signifie que l'URI inclurait l'étiquette de la boîte `c2ma` ou `c2um`, mais pas celle de la boîte `c2cm`. De plus, la référence URI à un manifeste compressé ne doit pas inclure l'étiquette de la boîte `brob`, mais uniquement l'étiquette du manifeste compressé lui-même.

Lors de la résolution d'une référence URI JUMBF interne, si une étiquette du chemin est ambiguë en raison de la présence de plusieurs boîtes enfants portant la même étiquette, le validateur doit traiter la référence comme non résolue.

8.4.2. URI hachés

8.4.2.1. Intégré

Un `hashed_uri` est utilisé lorsque l'URI concerne un élément intégré dans le même magasin de manifestes C2PA.

Cette spécification fournit une structure de données `hashed-uri-map` équivalente (en CDDL pour les URI hachés) pour les schémas utilisant une définition CDDL :

CDDL pour les URI hachés

```
; La structure de données utilisée pour stocker une référence à une URL dans le même JUMBF et son hachage. Nous utilisons ici un socket/plug pour permettre à hashed-uri-map d'être utilisé dans des fichiers individuels sans que la carte soit définie dans le même fichier
$hashed-uri-map /= {
  « url » : jumbf-uri-type, ; Référence URI JUMBF
  ? « alg » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme de hachage cryptographique utilisé pour calculer tous les hachages dans cette revendication, tirée de la liste des identifiants d'algorithmes de hachage C2PA. Si ce champ est absent, l'algorithme de hachage est tiré d'une structure englobante telle que définie par cette structure. Si les deux sont présents, le champ de cette structure est utilisé. Si aucune valeur n'est présente à l'un de ces emplacements, cette structure n'est pas valide ; il n'y a pas de valeur par défaut.
  « hash » : bstr, ; chaîne d'octets contenant la valeur de hachage
}

; avec CBOR Head (#) et tail ($) sont introduits dans regexp, donc pas besoin de les mentionner explicitement jumbf-uri-type /= tstr .regexp "self#jumbf=[\\w\\d\/][\\w\\d\\.\/:-]+[\\w\\d]"
```

Étant donné que les magasins d'assertions doivent se trouver dans la même boîte C2PA Manifest que la revendication qui y fait référence, seules les

les URI `self#jumbf` sont autorisées. Ces URI `self#jumbf` peuvent être relatives à l'ensemble du magasin de manifestes C2PA, auquel cas

auquel cas elles doivent commencer par un `/` (U+002F, barre oblique), ou relatives au manifeste C2PA actuel. Les URI ne doivent pas contenir la séquence `..` (une paire de U+002E, point).

Exemple 1. Exemples d'URI `self#jumbf`

Voici des exemples d'URI `self#jumbf` valides :

- `self#jumbf=/c2pa/urn:c2pa:F095F30E-6CD5-4BF7-8C44-CE8420CA9FB7/c2pa.assertions/c2pa.thumbnail.claim` est relatif à l'ensemble du magasin (puisqu'il commence par `/`) ;
- `self#jumbf=c2pa.assertions/c2pa.thumbnail.claim` serait relatif au manifeste de la boîte contenant l'URI.

8.4.2.2. Externe

Lorsqu'il est fait référence à une ressource qui existe en dehors du magasin de manifestes C2PA, une structure de données `hashed-ext-uri-map` est utilisée. Il s'agit d'une variante de `hashed-uri`, dans la mesure où elle fait référence à un URI externe plutôt qu'à un `self#jumbf`. La structure de données `hashed-ext-uri` est définie par la règle `hashed-ext-uri-map` dans le CDDL suivant dans CDDL pour les URI externes hachés :

CDDL pour les URI externes hachés

```
; Structure de données utilisée pour stocker une référence à une URL externe et son hachage.
; Nous utilisons ici une prise/fiche pour permettre l'utilisation de hashed-ext-uri-map dans des fichiers individuels
; sans que la carte soit définie dans le même fichier
$hashed-ext-uri-map /= {
  « url » : ext-url-type, ; Référence URI http/https
  « alg » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme de hachage cryptographique utilisé pour calculer le hachage des données de cette URI, tirée de la liste des identifiants d'algorithmes de hachage C2PA. Contrairement aux champs alg dans d'autres types, ce champ est obligatoire ici.
  « hash » : bstr, ; chaîne d'octets contenant la valeur de hachage
  ? « dc:format » : format-string, ; Type de média IANA des données
  ? « size » : type de taille, ; nombre d'octets de données
  ? « data_types » : [1* $asset-type-map], ; informations supplémentaires sur le type de données
}

; avec CBOR Head (#) et tail ($) sont introduits dans regexp, donc pas besoin de les mentionner explicitement
ext-url-type /= tstr .regexp « https?:\/\/[-a-zA-Z0-9@:%._\\+~#=]{2,256}\\.[a-z]{2,6}\\b[-a-zA-Z0-9@:%_\\+.~#?&//=]* »
```

IMPORTANT

Conformément à la pratique courante, il est recommandé d'utiliser le schéma `https` pour récupérer les données d'assertion afin de protéger la confidentialité des données en transit, mais `le schéma http` est également autorisé car l'intégrité des données est protégée par le champ `de hachage` et cette confidentialité n'est pas nécessaire dans toutes les circonstances. Les auteurs de manifestes avec des URI externes doivent choisir le schéma qui correspond à leurs besoins.

Le champ facultatif `dc:format`, lorsqu'il est présent, offre une alternative au champ `Content-Type` des en-têtes http(s). S'il est présent, ce champ doit être utilisé comme format requis récupéré lors de toute négociation/demande de contenu.

Il peut parfois être nécessaire de fournir un ou plusieurs types d'actifs comme valeur du champ `data_types` afin de clarifier le format et l'utilisation de ces données.

Un champ `de taille` facultatif est également fourni pour spécifier la taille des données à récupérer. Cela peut être utile à un validateur comme indice en plus du hachage.

REMARQUE Il peut être utilisé pour fournir des informations sur la nécessité d'effectuer un téléchargement, une validation ou les deux.

8.4.2.3. Hachage des boîtes JUMBF

Lors de la création d'une référence URI vers une boîte JUMBF (par exemple, des boîtes d'assertions et de données), le hachage doit être effectué sur le contenu de la superboîte JUMBF de la structure, qui comprend à la fois la boîte de description JUMBF et toutes les boîtes de contenu qu'elle contient (mais n'inclut pas l'en-tête de la superboîte JUMBF de la structure).

REMARQUE Pour plus de détails sur le hachage, voir la section 13.1, « Hachage ».

Comme décrit dans la dernière version de JUMBF (ISO 19566-5:2023) et illustré à la figure 4, « Exemple d'assertion `c2pa.actions` », un nouveau champ `privé` peut être présent dans n'importe quelle boîte de description JUMBF. Cette spécification C2PA définit le sel C2PA comme un champ `privé` dont la valeur est une boîte standard composée de :

- une longueur de boîte (LBox, sous forme d'entier non signé big-endian de 4 octets) ;
- un type de boîte (TBox, entier non signé big-endian de 4 octets, avec une valeur de `c2sh` (pour le hachage du sel C2PA)) ;
- et les données utiles (composées de données binaires générées aléatoirement d'une longueur de 16 ou 32 octets).

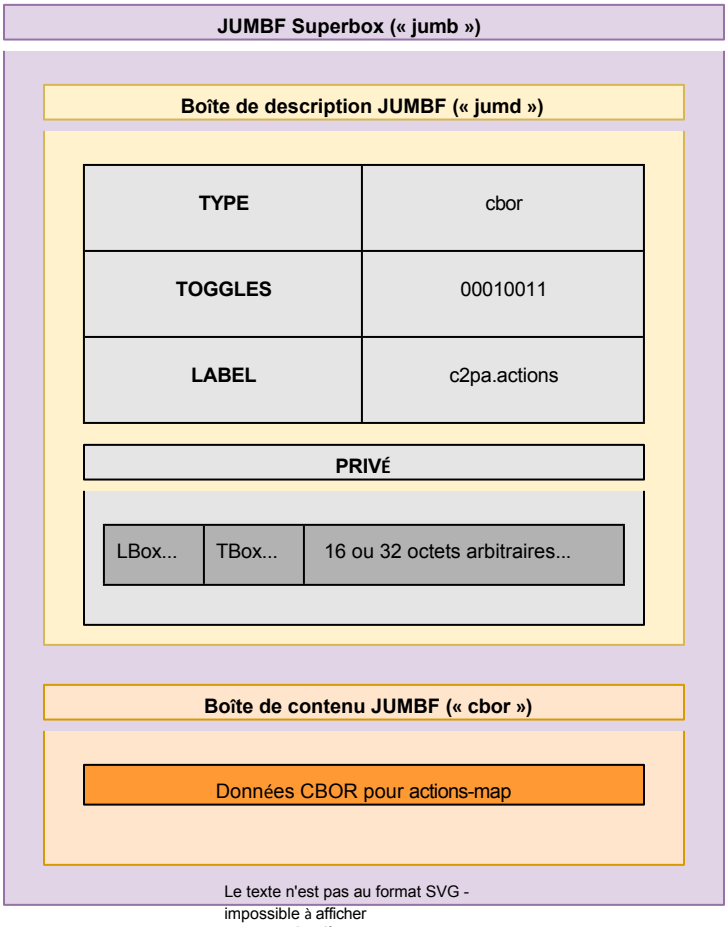

Figure 4. Exemple d'assertion `c2pa.actions`

Chapitre 9. Liaison au contenu

9.1. Présentation

Un aspect essentiel du manifeste C2PA standard est la présence d'une ou plusieurs structures de données, appelées liaisons de contenu, qui permettent d'identifier de manière unique certaines parties de l'actif. Il existe deux types de liaisons prises en charge par C2PA : les liaisons fortes et les liaisons faibles. Une liaison forte (également appelée liaison cryptographique) permet au validateur de s'assurer que (a) ce manifeste appartient à cet actif et (b) que l'actif n'a pas été modifié, en déterminant des valeurs qui ne peuvent correspondre qu'à cet actif et à aucun autre, pas même à d'autres actifs dérivés de celui-ci ou à des rendus produits à partir de celui-ci. Une liaison faible est calculée à partir du contenu numérique d'un actif, plutôt que de ses bits bruts. Une liaison souple est utile pour identifier les actifs dérivés et les rendus d'actifs.

Un manifeste unique ne doit pas contenir plus d'une assertion définissant une liaison forte, mais peut contenir zéro ou plusieurs assertions définissant des liaisons souples.

9.2. Liaisons fermes

9.2.1. Hachage à l'aide de plages d'octets

Le type de liaison forte le plus simple pouvant être utilisé pour détecter une altération est un algorithme de hachage cryptographique, tel que décrit à la section 13.1, « Hachage », sur tout ou partie des octets d'un actif. Cette approche peut être utilisée sur tout type d'actif, mais ne doit être envisagée que pour les formats qui ne prennent pas en charge l'une des formes de hachage basé sur des boîtes.

Lorsque vous utilisez ce type de liaison forte, une assertion de hachage de données est utilisée pour définir la plage d'octets qui sont hachés (et ceux qui ne le sont pas). Étant donné qu'une assertion de hachage de données définit une plage d'octets, elle est suffisamment flexible pour être utilisable, que l'actif soit un fichier binaire unique ou représenté en plusieurs morceaux ou portions.

9.2.2. Hachage à l'aide d'un hachage général

Lorsque le format d'un actif n'est pas basé sur BMFF, tel que JPEG, PNG, GIF ou d'autres formats répertoriés ici, une assertion de hachage de boîte générale doit être utilisée. Cette assertion se compose d'un ensemble de structures, chacune répertoriant une ou plusieurs boîtes (par leur nom/identifiant) et un hachage qui couvre les données de ces boîtes (et toutes les données éventuellement présentes dans le fichier entre elles), ainsi que l'algorithme utilisé pour le hachage.

9.2.3. Hachage d'un élément au format BMFF

Si la ressource est basée sur ISO BMFF, un lien dur optimisé pour le format basé sur des boîtes (appelé assertions de hachage basées sur BMFF) peut être utilisé à la place.

Pour un fichier MP4 monolithique dont la boîte `mdat` est validée en tant qu'unité, l'assertion est validée de manière presque identique à une assertion de hachage de données. Elle utilise simplement une liste d'exclusion de boîtes au lieu de plages d'octets pour définir la plage d'octets qui sont hachés (et ceux qui ne le sont pas).

Pour les fichiers MP4 fragmentés (fMP4), l'assertion elle-même doit être combinée avec des informations de hachage spécifiques aux morceaux, qui se trouvent à l'emplacement indiqué dans la section A.5, « Intégration de manifestes dans des fichiers basés sur BMFF ».

9.2.4. Hachage d'une collection

Dans les flux de travail où le manifeste C2PA fait référence à un ensemble d'actifs plutôt qu'à un seul actif, l'assertion de hachage des données de l'ensemble doit être utilisée comme méthode pour spécifier les liaisons fixes pour les actifs de l'ensemble.

REMARQUE Par exemple, une assertion de hachage des données de collection peut être utilisée pour décrire chaque dossier d'un ensemble de données d'entraînement pour un modèle d'IA/ML.

9.2.5. Liaisons de métadonnées d'actifs

Le générateur de revendications peut exclure les métadonnées des actifs (c'est-à-dire les métadonnées ne figurant pas dans un manifeste C2PA, telles que les métadonnées EXIF ou XMP) de la liaison de contenu. Pour ce faire, il doit utiliser les mécanismes d'exclusion applicables aux assertions de hachage de données, aux assertions de hachage de boîte générale ou aux assertions de hachage basées sur BMFF.

REMARQUE Les métadonnées d'actifs exclues ne sont pas attribuées au signataire.

Toutes les valeurs de métadonnées d'actifs prises en charge par l'assertion de métadonnées communes, telles que décrites dans l'annexe B, *Détails de mise en œuvre pour* `c2pa.metadata`, et pouvant être affirmées par le signataire doivent être copiées dans une telle assertion et incluses dans le manifeste C2PA.

9.3. Liens souples

9.3.1. Général

Les liaisons souples sont décrites à l'aide d'assertions de liaison souple telles qu'une empreinte digitale calculée à partir du contenu numérique ou un filigrane invisible intégré dans le contenu numérique. Ces liaisons souples permettent de faire correspondre le contenu numérique même si les bits sous-jacents diffèrent.

REMARQUE Par exemple, une représentation d'un actif dans une résolution ou un format d'encodage différent.

De plus, si un manifeste C2PA est supprimé d'un actif, mais qu'une copie de ce manifeste reste dans un magasin de provenance ailleurs, le manifeste et l'actif peuvent être mis en correspondance à l'aide des liaisons souples disponibles.

Étant donné qu'ils ont des objectifs différents, un lien souple ne doit pas être utilisé comme un lien fort.

9.3.2. Liste des algorithmes de liaison souple autorisés

Toutes les liaisons souples doivent être générées à l'aide de l'un des algorithmes répertoriés dans la liste des algorithmes de liaison souple pris en charge par la présente spécification.

Chapitre 10. Revendications

10.1. Aperçu

Une **revendication** rassemble toutes les assertions relatives à un actif à un moment donné, y compris l'ensemble des assertions liées au contenu. La revendication est ensuite hachée cryptographiquement et signée comme décrit dans la section 10.3.2.4, « Signature d'une revendication ». Une revendication possède toutes les mêmes propriétés qu'une assertion, y compris l'attribution du label (`c2pa.claim.v2`), mais elle ne prend pas en charge l'utilisation des métadonnées d'assertion. Une revendication est encodée sous forme de données CBOR et, à ce titre, doit être conforme aux exigences d'encodage déterministe de base du CBOR (voir RFC 8949, clause 4.2.1).

REMARQUE Les versions précédentes prenaient en charge l'utilisation de métadonnées d'assertion avec les revendications, mais cette fonctionnalité est désormais obsolète.

Une version précédente de cette spécification utilisait le label `c2pa.claim` et `la carte de revendication` associée pour la revendication, mais ceux-ci sont désormais obsolètes. Les validateurs doivent toujours accepter ce label (et `la carte de revendication` associée), mais les générateurs de revendications ne doivent pas produire une telle revendication.

10.2. Syntaxe

10.2.1. Schéma

Le schéma pour ce type est défini par les règles `claim-map-v2` et `claim-map` dans la définition CDDL suivante pour les revendications avec les étiquettes `c2pa.claim.v2` et `c2pa.claim`, respectivement :

```
; Schéma CDDL pour une carte de
revendication dans C2PA claim-map = {
  « claim_generator » : tstr, ; Chaîne User-Agent formatée conformément à
http://tools.ietf.org/html/rfc7231#section-5.5.3, pour inclure le nom et la version du générateur de
revendications qui a créé la revendication
  « claim_generator_info » : [1* generator-info-map],
  « signature » : jumbf-uri-type, ; Référence URI JUMBF à la signature de cette revendication « assertions » :
  [1* $hashed-uri-map],
  « dc:format » : tstr, ; type de média de l'actif
  « instanceID » : tstr .size (1..max-tstr-length), ; identifie de manière unique une version spécifique d'un
actif
  ? « dc:title » : tstr .size (1..max-tstr-length), ; nom de l'actif,
  ? « redacted_assertions » : [1* jumbf-uri-type], ; liste des références URI JUMBF aux assertions des
manifestes d'ingrédients en cours de rédaction
  ? « alg » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme de hachage cryptographique
utilisé pour calculer toutes les assertions de hachage de données répertoriées dans cette revendication, sauf
indication contraire, tirée du registre des identifiants d'algorithmes de hachage de données C2PA. Cela fournit
la valeur du champ « alg » dans les structures data-hash et hashed-uri contenues dans cette revendication
  ? « alg_soft » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme utilisé pour calculer
toutes les assertions de liaison souple répertoriées dans cette revendication, sauf indication contraire,
tirée du registre des identifiants d'algorithmes de liaison souple C2PA.
  ? « metadata » : $assertion-metadata-map, ; informations supplémentaires sur l'assertion
}

; Schéma CDDL pour une carte de
revendications dans C2PA claim-map-v2 = {
```

```
  « instanceID » : tstr .size (1..max-tstr-length), ; identifie de manière unique une version spécifique d'un
actif
  « claim_generator_info » : $generator-info-map, ; générateur de revendications de cette revendication «
  signature » : jumbf-uri-type, ; référence URI JUMBF à la signature de cette revendication «
  created_assertions » : [1* $hashed-uri-map],
  ? « gathered_assertions » : [1* $hashed-uri-map],
  ? « dc:title » : tstr .size (1..max-tstr-length), ; nom de l'actif,
  ? « redacted_assertions » : [1* jumbf-uri-type], ; liste des références URI JUMBF aux assertions des
manifestes d'ingrédients en cours de rédaction
  ? « alg » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme de hachage cryptographique
utilisé pour calculer toutes les assertions de hachage de données répertoriées dans cette revendication, sauf
indication contraire, tirée du registre des identifiants d'algorithmes de hachage de données C2PA. Cela fournit
la valeur du champ « alg » dans les structures data-hash et hashed-uri contenues dans cette revendication
  ? « alg_soft » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme utilisé pour calculer
toutes les assertions de liaison souple répertoriées dans cette revendication, sauf indication contraire,
tirée du registre des identifiants d'algorithmes de liaison souple C2PA.
  ? « metadata » : $assertion-metadata-map, ; (OBSOLÈTE) informations supplémentaires sur l'assertion
}

generator-info-map = {
  « name » : tstr .size (1..max-tstr-length), ; Chaîne lisible par l'utilisateur nommant le générateur de
revendications
  ? « version » : tstr, ; Chaîne lisible par l'utilisateur indiquant la version du produit
  ? « icon » : $hashed-uri-map / $hashed-ext-uri-map, ; URI haché vers l'icône (intégrée ou distante)
  ? « operating_system » : tstr, ; Chaîne lisible par l'utilisateur indiquant le système d'exploitation sur lequel
le générateur de revendications est exécuté
  * tstr => any
}
```

Exemple de structure `claim-map-v2` en notation diagnostique CBOR (RFC 8949, clause 8) :

```
{
  « alg » : « sha256 », «
  claim_generator_info » : {
    « nom » : « Joe's Photo Editor », «
    version » : « 2.0 »,
    « operating_system » : « Windows 10 »
  },
  « signature » : « self#jumbf=c2pa.signature », «
  created_assertions » : [
    {
      « url » : « self#jumbf=c2pa.assertions/c2pa.hash.data », « hash » :
      b64'U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8='
    },
    {
      « url » : « self#jumbf=c2pa.assertions/c2pa.thumbnail.claim », « hash » :
      b64'G5hfJwYeWTlflxOhmfCO9xDAK52aKQ+YbKNhRZeq92c='
    },
    {
      « url » : « self#jumbf=c2pa.assertions/c2pa.ingredient.v3 », « hash »
      : b64'Yzag4o5jO4xPyfANVtw7ETlbFSWZNfeM78qbSi8Abkk='
    }
  ],
  « assertions_expurgées » : [ «
    self#jumbf=/c2pa/urn:c2pa:5E7B01FC-4932-4BAB-AB32-
D4F12A8AA322/c2pa.assertions/c2pa.metadata"
  ]
}
```

10.2.2. Champs

Si présent, la valeur de `dc:title` doit être un nom lisible par l'utilisateur pour l'actif.

REMARQUE Le champ `dc:format` présent dans `c2pa.claim` n'existe plus dans `c2pa.claim.v2`.

Si l'actif contient du XMP, alors le `xmpMM:InstanceID` de l'actif doit être utilisé comme `instanceID`. Si aucun XMP n'est disponible, alors un autre identifiant unique pour l'actif doit être utilisé comme valeur pour `instanceID`.

REMARQUE Certains noms de champs, tels que `dc:title`, ont des préfixes d'espace de noms, car leurs noms et leurs définitions sont directement tirés de la norme XMP. Cependant, leur utilisation dans C2PA ne nécessite pas l'utilisation de XMP.

Le champ `signature` doit être présent et contenir une référence URI vers une signature de revendication.

Le champ `created_assertions` doit être présent et contenir une ou plusieurs références URI aux assertions faites par cette revendication. Dans un manifeste standard, il doit contenir au minimum une référence à une assertion représentant une liaison forte et une référence à une assertion d'action.

REMARQUE Toutes `les assertions créées` sont attribuées au signataire, car le modèle de confiance repose sur la confiance accordée au signataire.

Lorsqu'il est présent, le champ `gathered_assertions` doit contenir une ou plusieurs références URI vers des assertions qui ont été fournies au générateur de revendications par d'autres composants du flux de travail.

REMARQUE En ajoutant une assertion à cette liste, le générateur de revendications déclare que l'assertion fait partie du , mais qu'elle ne provient pas du générateur de revendications et n'est pas attribuée au signataire. Par exemple, les assertions contenant des informations saisies par un acteur humain seraient répertoriées dans `gathered_assertions`.

Lorsqu'il est présent, le champ `redacted_assertions` doit contenir une ou plusieurs références URI vers des assertions expurgées.

10.2.3. Informations sur le générateur de revendications

10.2.3.1. Général

Les informations détaillées concernant le générateur de revendications doivent être présentes sous la forme de la valeur `claim_generator_info`. Un consommateur de manifeste doit utiliser la valeur `claim_generator_info` pour déterminer les informations concernant le générateur de revendications pour lui-même ou pour les présenter dans une expérience utilisateur.

REMARQUE Le `c2pa.claim` comporte un champ `claim_generator`, dont la valeur est une simple chaîne de caractères, qui n'est plus présente dans `c2pa.claim.v2`.

10.2.3.2. Carte d'informations sur le générateur

Lors de l'ajout d'un champ `claim_generator_info`, sa valeur est un objet `generator-info-map` qui doit contenir un champ `name`. Il peut également contenir un champ `version` ou un champ `icon`, ou les deux. En outre, tout autre champ est

autorisé, en utilisant l'espace de noms spécifique à l'entité standard décrit à la section 6.2.1, « Espaces de noms ». Les données de cet objet doivent représenter l'acteur non humain (matériel ou logiciel) qui a réellement généré la revendication (c'est-à-dire le générateur de revendication lui-même).

Un générateur de revendications peut souhaiter fournir une représentation graphique de lui-même, appelée ici `icône`, à un consommateur de manifeste qui présente une expérience utilisateur. La valeur du champ `icône`, s'il est présent, doit être un URI haché. Cet URI haché doit renvoyer à une assertion de données intégrée dont l'étiquette est `c2pa.icon` et qui respecte les règles relatives aux étiquettes d'assertion en ce qui concerne les instances multiples. Les consommateurs de manifeste doivent également prendre en charge l'approche de la boîte de données recommandée par les versions antérieures de cette spécification.

REMARQUE Comme pour le tableau d'assertions, l'algorithme de hachage utilisé pour un URI haché est déterminé par le champ `alg` présent dans l'URI haché ou, en son absence, par un champ `alg` dans la revendication.

Exemple utilisant les informations du générateur de revendications

```
{
     « claim_generator_info » : { « name » :
          « Joe's Photo Editor », « version »
          : « 2.0 »,
          "operating_system" : "Windows 10", "icon" :
          {
               "url" : "http://cdn.examplephotoagency.com/logo.svg", "hash" :
               "5bdec8169b4e4484b79aba44cee5c6bd"
          }
     }
}
```

10.3. Création d'une revendication

10.3.1. Création d'assertions

Avant que la revendication puisse être finalisée, toutes les assertions doivent être créées et stockées dans un magasin d'assertions C2PA nouvellement créé, comme décrit plus loin dans ce document.

Lors de la création d'un manifeste standard, il peut être impossible de connaître toutes les informations de liaison requises au moment de la création de la revendication. Dans ce cas, utilisez la méthode de traitement en plusieurs étapes pour configurer puis remplir les informations ultérieurement.

10.3.2. Préparation de la revendication

10.3.2.1. Ajout d'assertions et de rédactions

La revendication doit contenir un champ `created_assertions` et peut contenir un champ `gathered_assertions`. Les valeurs combinées de ces deux champs représentent une liste de toutes les références URI pour toutes les assertions qui ont été ajoutées au magasin d'assertions et qui sont « revendiquées » par cette revendication. Dans un manifeste standard, la valeur du champ `created_assertions` doit inclure au moins une assertion qui représente une liaison forte.

Si certaines assertions relatives aux allégations sur les ingrédients sont supprimées, leurs références URI doivent être ajoutées à la liste qui correspond à la valeur du champ `redacted_assertions`.

10.3.2.2. Ajout d'ingrédients

Dans de nombreux scénarios de création, un acteur ne crée pas un élément entièrement nouveau, mais utilise plutôt d'autres éléments existants sur lesquels il s'appuie pour créer son travail, que ce soit sous la forme d'un élément dérivé, d'un élément composé ou d'une représentation d'élément. Ces éléments existants sont appelés « ingrédients » et leur utilisation est documentée dans les données de provenance à l'aide d'une assertion d'ingrédient.

Lorsqu'un ingrédient contient un ou plusieurs manifestes C2PA, ces manifestes doivent être insérés dans le magasin de manifestes C2PA de cet actif afin de garantir que les données de provenance restent intactes. Ces manifestes d'ingrédients sont ajoutés au JUMBF comme décrit dans la section 11.1.4, « Détails de la boîte C2PA ». Si un manifeste portant le même identifiant unique est déjà présent dans le magasin de manifestes C2PA, les deux doivent être comparés (via hachage). S'ils sont identiques, le nouveau manifeste doit être ignoré. S'ils sont différents, le nouveau manifeste doit être ajouté au magasin après avoir changé son identifiant unique pour une nouvelle valeur, comme décrit au chapitre 8, *Identifiants uniques.*

Si le manifeste d'un ingrédient est distant et que le générateur de déclaration n'est pas en mesure de le récupérer, il doit utiliser le code d'erreur `manifest.inaccessible` pour l'indiquer.

10.3.2.3. Connexion de la signature

La signature ne peut pas faire partie de la charge utile signée, mais comme son étiquette est prédéfinie, la référence URI complète est également connue. Nous pouvons donc l'inclure dans la revendication en définissant la valeur du champ `signature` de la revendication sur cette référence URI.

REMARQUE Cela permet de lier explicitement la revendication à sa signature.

10.3.2.4. Signature d'une revendication

La production de la signature est spécifiée dans la section 13.2, « Signatures numériques ».

Pour les deux types de manifestes, standard et mise à jour, le champ `de charge utile` de `Sig_structure` doit être le CBOR sérialisé du document de réclamation et doit utiliser le mode de contenu détaché.

La structure `COSE_Sign1_Tagged` sérialisée résultant de la procédure de signature numérique est écrite dans la case « Signature de la revendication C2PA ».

10.3.2.5. Horodatage

10.3.2.5.1. Utilisation de la norme RFC 3161

Si possible, le générateur de revendication doit utiliser une autorité d'horodatage (TSA) conforme à la norme RFC 3161 (RFC 3161) afin d'obtenir un horodatage fiable prouvant que la signature elle-même existait réellement à une date et une heure données, et l'intégrer dans la structure `COSE_Sign1_Tagged` en tant que contresignature.

Les générateurs de revendications sont encouragés à obtenir et à inclure des horodatages afin de garantir la validité de leurs manifestes. Comme

décrit au chapitre 15, *Validation*, les manifestes sans horodatage cessent d'être valides lorsque le certificat de signature expire ou est révoqué. Un manifeste ne doit contenir qu'un seul horodatage.

REMARQUE Les versions précédentes de cette spécification autorisaient l'inclusion de plusieurs horodatages dans un manifeste.

10.3.2.5.2. Choix de la charge utile

Une version précédente de cette spécification utilisait la même valeur pour le champ `de charge utile` dans l'horodatage que celle utilisée dans la `signature Sig_signature`, comme décrit à la section 10.3.2.4, « Signature d'une revendication ». Cette charge utile est désormais appelée « charge utile v1 » dans un « horodatage v1 » et est considérée comme obsolète. Un générateur de revendications ne doit pas en créer, mais un validateur doit en traiter une si elle est présente.

La « charge utile v2 » de l'« horodatage v2 » est la valeur du champ `de signature` de la structure `COSE_Sign1_Tagged` créée dans le cadre de la section 10.3.2.4, « Signature d'une revendication ». Une « charge utile v2 » doit être utilisée par les générateurs de revendications effectuant une opération d'horodatage.

REMARQUE La valeur du champ `de signature` comprend l'ensemble `du bstr` sérialisé, y compris les octets qui indiquent le type principal et la longueur (et pas seulement la chaîne elle-même).

10.3.2.5.3. Obtention de l'horodatage

Tous les horodatages doivent être obtenus comme décrit dans la RFC 3161, avec les exigences supplémentaires suivantes :

- Le `MessageImprint` de la structure `TimeStampReq` (RFC 3161, section 2.4.1) doit être calculé en créant la valeur `ToBeSigned` dans la RFC 8152, section 4.4, avec les valeurs suivantes pour les éléments de `Sig_structure` :
  - L'élément `context` doit être `CounterSignature`.
  - L'élément `payload` doit être la valeur décrite à la section 10.3.2.5.2, « Choix de la charge utile ».
  - Les éléments restants de `Sig_structure` sont décrits dans la section 13.2.3, « Calcul de la signature ».
- La valeur `ToBeSigned` est ensuite hachée à l'aide d'un algorithme de hachage figurant dans la liste autorisée à la section 13.1, « Hachage », pris en charge par la TSA, et cet algorithme de hachage et cette valeur sont placés dans le `MessageImprint`. Si la TSA ne prend en charge aucun algorithme de hachage figurant dans la liste autorisée, elle ne peut pas être utilisée pour l'horodatage.
  - Dans la mesure du possible, l'algorithme de hachage doit utiliser le même algorithme de hachage que celui utilisé dans la signature numérique de la revendication.
- La valeur booléenne `certReq` de la structure `TimeStampReq` doit être affirmée dans la demande adressée à la TSA, afin de garantir que sa chaîne de certificats est fournie dans la réponse.

10.3.2.5.4. Stockage de l'horodatage

Les horodatages v1 (obsolètes) sont stockés dans un en-tête COSE non protégé dont l'étiquette est la chaîne `sigTst`. S'il est présent, la valeur de cet en-tête doit être un `tstContainer` défini par l'exemple 2, « CDDL pour `tstContainer` ». Le contenu de la structure `TimeStampResp` reçue en réponse du TSA doit être stocké en tant que valeur de la propriété `val` d'un élément de `tstTokens`.

Les horodatages v2 doivent être stockés dans un en-tête COSE non protégé dont l'étiquette est la chaîne `sigTst2`. Lorsqu'il est présent, la valeur de cet en-tête doit être un `tstContainer` défini par l'exemple 2, « CDDL pour `tstContainer` ». La valeur du champ `timeStampToken` de la structure `TimeStampResp` reçue en réponse de la TSA doit être stockée en tant que valeur de la propriété `val` d'un élément de `tstTokens`. Elle doit être formatée en tant que `TimeStampToken` RFC 3161 codé en DER et encapsulé dans une chaîne d'octets CBOR.

NOTE

Un horodatage v2 est équivalent au modèle « CTT » du paramètre d'en-tête COSE pour RFC 3161 Time-
Il nécessite que la structure de signature complète soit achevée avant l'horodatage, permettant ainsi à l'horodatage de servir de contresignature sur l'ensemble de la structure de signature, y compris le certificat proprement dit.

Si aucun horodatage n'est inclus, aucun en-tête (`sigTst` ou `sigTst2`) ne doit être présent dans l'en-tête non protégé COSE.

Exemple 2. CDDL pour `tstContainer`

```
; Version CBOR de tstContainer et des structures associées basée sur le schéma JSON disponible à l'adresse
; https://forge.etsi.org/rep/esi/x19_182_JAdES/raw/v1.1.1/19182-jsonSchema.json tstContainer = {
  « tstTokens » : [1* tstToken]
}

tstToken = { « val »
   : bstr
}
```

REMARQUE

La définition ci-dessus est une adaptation CBOR d'un sous-ensemble du schéma de JAdES, section 5.3.4 et de son schéma JSON, à l'exception de la modification selon laquelle le contenu de `val` est une chaîne d'octets et non une chaîne codée en Base64.

10.3.2.6. Informations sur la révocation des informations d'identification

Si les informations d'identification du signataire permettent d'interroger leur statut en ligne et qu'elles contiennent un pointeur vers un service fournissant des informations horodatées sur leur statut, le générateur de revendications doit interroger le service, capturer la réponse et la stocker de la manière décrite pour les informations d'identification dans le modèle de confiance. Si les informations de révocation des informations d'identification sont jointes de cette manière, un horodatage fiable doit également être obtenu après la signature, comme décrit dans la section 10.3.2.5, « Horodatages ».

10.3.3. Exemples de réclamations

10.3.3.1. Revendication unique

Voici une représentation visuelle d'une image contenant une revendication unique avec plusieurs assertions qui y ont été intégrées.

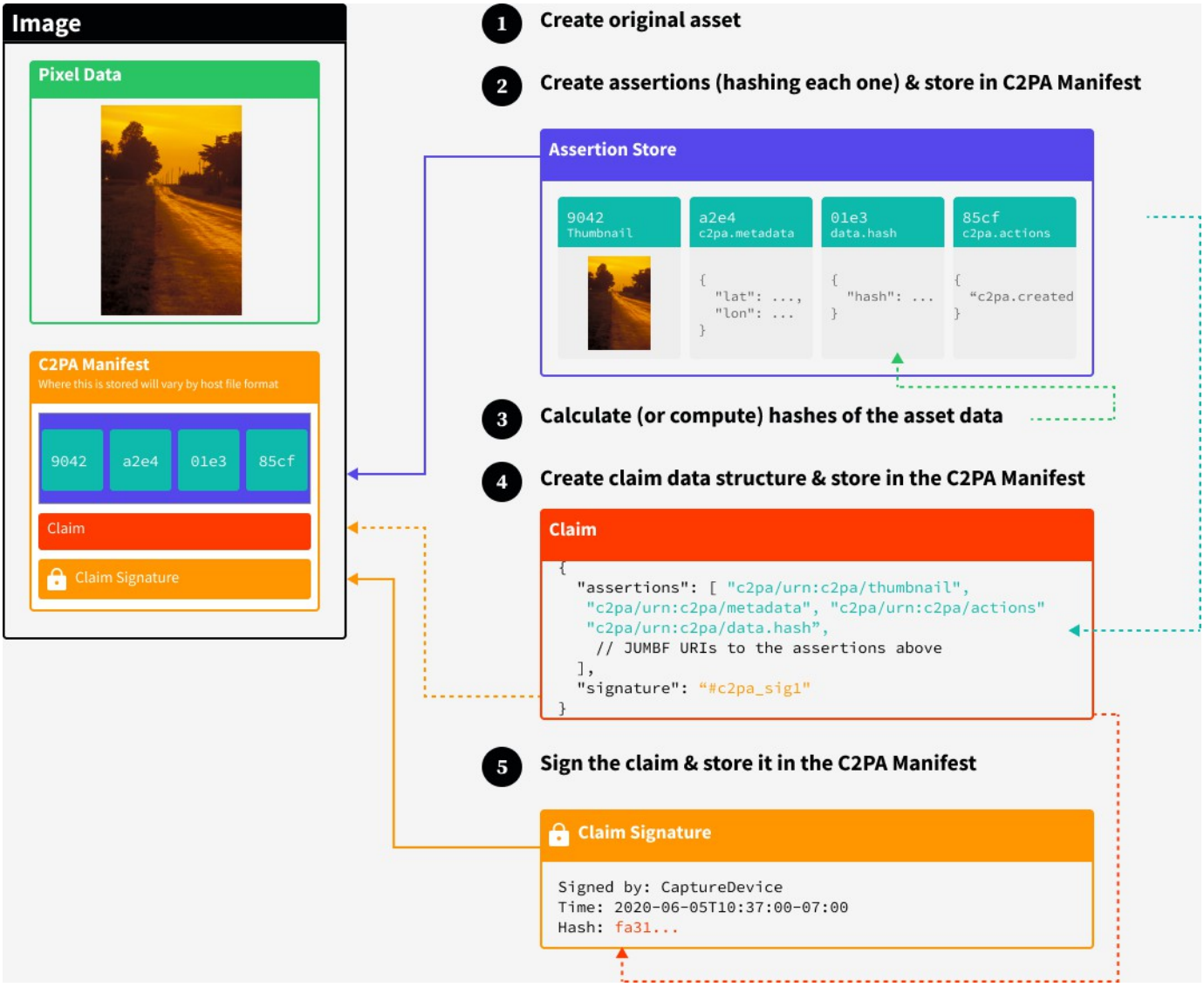

Figure 5. Une revendication unique avec des assertions

10.3.3.2. Revendications multiples

Dans cet exemple de création d'une deuxième revendication pour l'exemple précédent, l'une des assertions originales a été supprimée de la revendication précédente. La représentation visuelle de ce scénario ressemblerait à ceci :

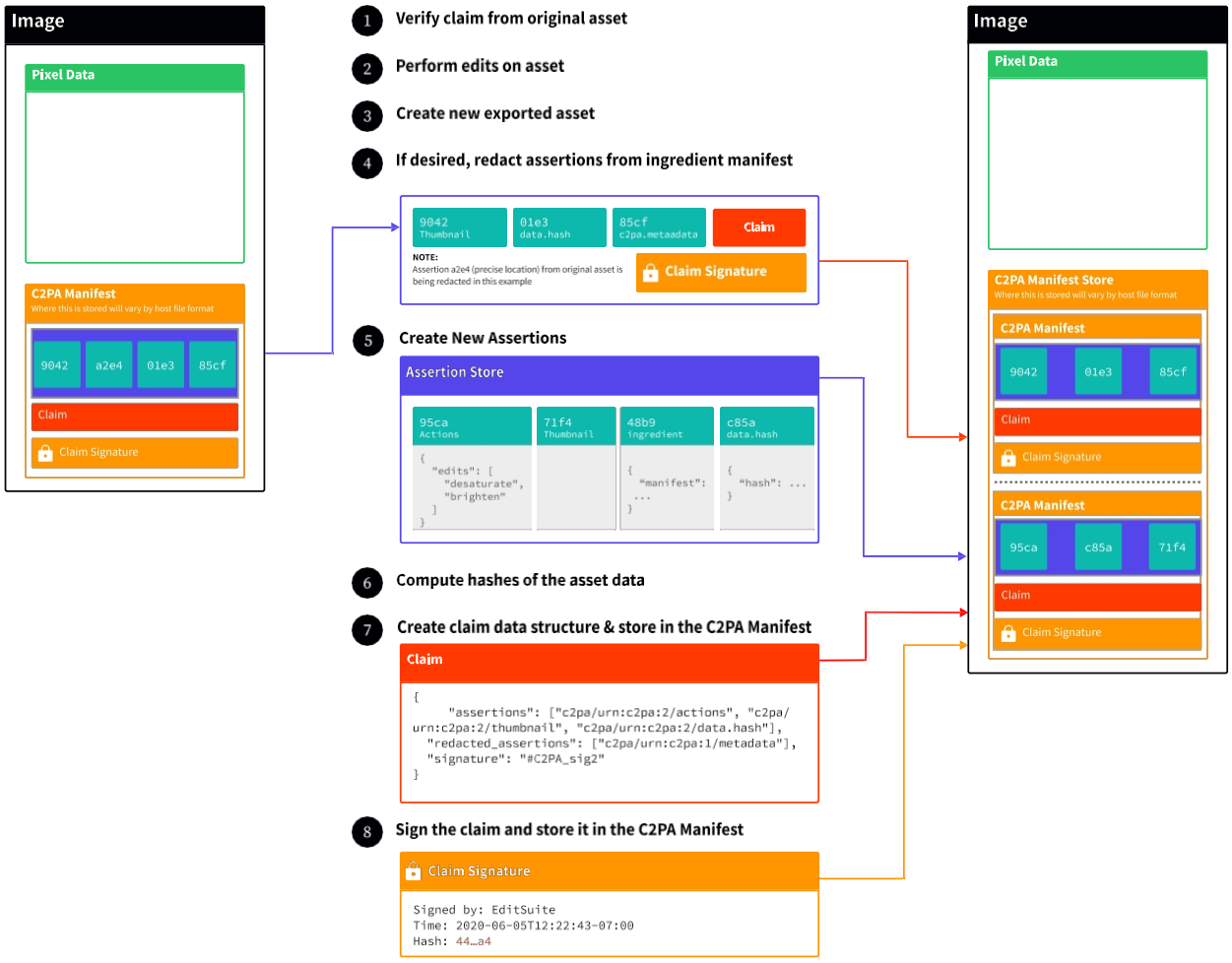

Figure 6. Suppression d'assertions dans une revendication secondaire

10.4. Traitement en plusieurs étapes

Certains formats de fichiers d'actifs exigent que les décalages de fichiers du magasin de manifestes C2PA et le contenu des actifs soient corrigés avant la signature du manifeste, afin que les liaisons de contenu s'alignent correctement avec le contenu qu'elles authentifient. Malheureusement, la taille d'un manifeste et sa signature ne peuvent être connues avec précision qu'après la signature, ce qui peut entraîner une modification des décalages de fichiers.

Par exemple, dans les fichiers JPEG 1, l'ensemble du magasin de manifestes C2PA doit apparaître dans le fichier avant les données d'image, et sa taille aura donc une incidence sur les décalages de fichiers du contenu authentifié.

Pour ce faire, une approche en plusieurs étapes sera adoptée, similaire à celle utilisée pour les signatures dans les fichiers PDF.

10.4.1. Créer des liens de contenu

Lors de la création d'un manifeste standard, sa revendication doit inclure une ou plusieurs assertions de liaison de contenu dans sa liste d'assertions afin de garantir que l'actif est inviolable.

Créer l'assertion de hachage des données et l'ajouter au magasin d'assertions en tenant compte des considérations suivantes.

Dans de nombreux cas, comme avec JPEG 1, il n'est pas possible de hacher l'intégralité de la ressource, car le manifeste sera intégré au milieu du fichier. La taille ou l'emplacement des données du manifeste ne seront donc pas connus au moment du calcul du hachage de la ressource. Cette dépendance circulaire est évitée en autorisant la spécification de plages d'exclusion pendant le hachage. Lorsque des plages d'exclusion sont spécifiées, un seul hachage est effectué, mais uniquement sur les plages de ressources qui ne font partie d'aucune des exclusions.

Si un manifeste est intégré au centre d'un fichier JPEG 1 dans un segment APP11, le créateur de la revendication peut exclure le ou les segments APP11 du calcul du hachage.

Afin d'empêcher les attaques par insertion, il est préférable de n'avoir qu'une seule plage d'exclusion lorsque cela est possible. Lorsque la taille ou l'emplacement (ou les deux) du manifeste dans l'actif n'est pas connu, les valeurs `de début` et `de longueur` dans l'assertion de hachage des données doivent toutes deux être égales à zéro et la taille de la valeur `de remplissage` doit être suffisamment grande pour permettre l'écriture des valeurs lors du deuxième passage. Une taille minimale de 16 octets est recommandée. La valeur de la clé `de remplissage` doit être composée uniquement de 0x00.

Si un remplissage est utilisé, il est possible que les données de remplissage soient modifiées sans entraîner d'échec de validation. Les générateurs de revendications doivent s'assurer que les modifications apportées aux données de remplissage (ou à toute autre donnée d'actif exclue) ne peuvent pas modifier la façon dont l'actif est interprété.

REMARQUE

Dans le cas des fichiers JPEG 1, cela peut être réalisé soit en éliminant le remplissage, soit en s'assurant que les segments segments `JFIF APP11/C2PA` ne peuvent pas être raccourcis ou modifiés en un type de segment différent. Il s'agit de la réalisé en incluant tous les en-têtes de segment de manifeste C2PA (APP11) et les champs de longueur de 2 octets dans la table de hachage des données pour tous les segments contenant un manifeste. Cela garantit que les données modifiées dans la région d'exclusion ne seront pas mal interprétées par les processeurs JPEG.

10.4.2. Créer une revendication et une signature temporaires

Ajoutez la référence d'assertion de hachage de données nouvellement créée à la liste d'assertions de la revendication en fournissant une valeur de hachage temporaire, telle que des espaces vides.

À ce stade, la revendication temporaire est terminée et peut être ajoutée au manifeste C2PA en cours de création.

Comme la revendication n'est pour l'instant que temporaire, il n'est pas possible de la signer. Pour vous assurer que la zone de signature de la revendication contient une structure CBOR valide, créez une structure `COSE_Sign1_Tagged` temporaire comme décrit dans la section 4.2 de la RFC 8152. La `structure COSE_Sign1_Tagged` est un octet de balise suivi d'une structure `COSE_Sign1`, qui est un tableau CBOR à quatre éléments. Construisez le tableau comme suit :

- Le premier élément est le compartiment d'en-tête `protégé` (RFC 8152, section 3). Créez un compartiment vide en plaçant un `bstr` de taille 0 à cette position.
- Le deuxième élément est le compartiment d'en-tête `non protégé`, qui est une carte CBOR. Créez une carte d'une paire. Utilisez le `tampon` de chaîne comme étiquette et placez un `bstr` de la taille de remplissage souhaitée rempli d'octets zéro (0x00) comme valeur. Une taille de 25 kilo-octets est recommandée pour la taille initiale de ce remplissage.
- Le troisième élément est la `charge utile`. Placez ici la valeur `nil` (type majeur CBOR 7, valeur 22).

- Le quatrième élément est `la signature`. Placez ici un `bstr` de taille 0.

10.4.3. Complétez le manifeste C2PA

À ce stade, toutes les cases qui composent le manifeste C2PA complet pour l'actif sont remplies et peuvent être (si ce n'est déjà fait) assemblées pour former leur version finale. Le manifeste C2PA de l'actif, ainsi que les manifestes de tous les ingrédients, sont combinés pour former le magasin de manifestes C2PA complet. Le manifeste actif doit être la dernière superboîte de manifeste C2PA dans la superboîte du magasin de manifestes C2PA. Le magasin de manifestes C2PA peut ensuite être intégré à l'actif comme indiqué dans la section 11.3, « Intégration de manifestes dans divers formats de fichiers ».

10.4.4. Revenir en arrière et remplir

Maintenant que le magasin de manifestes C2PA a été intégré à l'actif, le décalage de départ et la longueur du manifeste actif peuvent être mis à jour dans son assertion de hachage de données. Il est nécessaire, lorsque vous effectuez cette opération, de ne pas modifier la taille de la boîte de l'assertion, mais uniquement ses données. Pour ce faire, ajustez la valeur du champ `de remplissage` afin qu'il ait la longueur nécessaire pour « remplir » les octets restants.

NOTE

La sérialisation CBOR préférée/déterministe du `pad` utilise un entier de longueur variable pour spécifier la longueur des données binaires encodées. Lorsque la longueur passe de zéro à 1 octet, ou de 1 à 2 octets (etc.), la longueur du pad résultant augmente de deux octets. Cela signifie que tous les paddings ne peuvent pas être exprimés à l'aide d'un champ de remplissage unique. Par exemple, il est possible de créer des remplissages de 24 et 26 octets, mais pas de 25 octets. Si cette situation se présente, le remplissage souhaité peut être réparti entre `pad` et `pad2`. Par exemple, pour créer un remplissage de 25 octets, un générateur de revendication peut encoder 19 octets dans `pad` (ce qui donne une longueur encodée de 20 octets) et 4 octets dans `pad2` (ce qui donne 5 octets).

Une fois l'assertion de hachage des données mise à jour, elle peut être hachée et le hachage écrit sur les espaces vides qui étaient auparavant utilisés pour contenir l'emplacement.

La revendication est désormais complète et peut être hachée et signée comme décrit dans la section 10.3.2.4, « Signature d'une revendication », la signature résultante remplissant l'espace pré-alloué. L'en-tête `du remplissage` peut alors être réduit selon les besoins afin que la boîte de signature de la revendication reste de la même taille ; comme cet en-tête n'est pas protégé, le modifier n'invalide pas la signature de la revendication.

Si la structure sérialisée `COSE_Sign1_Tagged` dépasse la taille réservée de la zone C2PA Claim Signature, le traitement en plusieurs étapes doit être répété avec une taille de remplissage plus grande choisie dans la section 10.4.2, « Créer une revendication et une signature temporaires ». Les informations de révocation récupérées lors de la tentative précédente doivent être réutilisables si elles sont toujours dans leur intervalle de validité (RFC 6960, section 4.2.2.1), mais un nouvel horodatage sera nécessaire sur la nouvelle revendication avec les décalages de fichier modifiés à la suite de l'ajout de remplissage.

Un manifeste C2PA peut contenir des assertions définies en dehors de cette spécification, et celles-ci peuvent dépendre de la disposition des fichiers. De ce fait, le générateur de revendications peut ne plus être en mesure de modifier la disposition des fichiers et/ou les décalages dans une assertion de hachage de données. Dans ce cas, les générateurs de revendications doivent utiliser un remplissage avant la création de l'assertion afin de s'assurer que la disposition des fichiers n'a pas besoin d'être modifiée une fois l'assertion finalisée.

Chapitre 11. Manifestes

11.1. Utilisation de JUMBF

11.1.1. Justification

Afin de répondre à bon nombre des exigences de la norme C2PA, les manifestes C2PA devaient être stockés (sérialisés) dans un magasin de données binaires structuré permettant certaines fonctionnalités spécifiques, notamment :

- Possibilité de stocker plusieurs manifestes (par exemple, parents et ingrédients) dans un seul conteneur.
- La possibilité de faire référence à des éléments individuels (à la fois au sein d'un même manifeste et entre différents manifestes) via des URI.
- Possibilité d'identifier clairement les parties d'un élément à hacher.
- Possibilité de stocker des types de données prédéfinis utilisés par la C2PA (par exemple, JSON et CBOR).
- Possibilité de stocker des formats de données arbitraires (par exemple, XML, JPEG, etc.).

En plus de prendre en charge toutes les exigences ci-dessus, le format de conteneur que nous avons choisi, ISO 19566-5:2023 (JUMBF), est également pris en charge de manière native par la famille de formats JPEG et est compatible avec le modèle basé sur des boîtes (c'est-à-dire ISOBMFF, ISO 14496-12) utilisé par de nombreux formats de fichiers image et vidéo courants. L'utilisation de JUMBF offre les mêmes avantages (et quelques extras, tels que les références URI) tout en permettant de travailler avec des formats d'image classiques, tels que JPEG/JFIF et PNG, ainsi que des formats 3D et de documents (par exemple, PDF). Ce format sérialisé doit également être utilisé dans les formats qui ne prennent pas en charge nativement JUMBF, ou lorsque les magasins de manifestes C2PA sont stockés séparément de la ressource, par exemple dans un fichier séparé ou à un emplacement URI distinct.

REMARQUE

Étant donné que la plupart des assertions standard, ainsi que la signature de la revendication, sont sérialisées en CBOR, l'utilisation du CBOR pour l'ensemble du manifeste C2PA a été envisagée, mais n'a pas été retenue car le CBOR n'est pas un format conteneur.

Par exemple, pour stocker un « blob JSON » dans CBOR et savoir qu'il s'agit bien de JSON (et non d'un autre format), il faudrait concevoir une structure de données permettant de stocker ce type d'éléments. Il faudrait ensuite définir la structure parent afin de déterminer comment transporter cette structure. Ce même concept devrait également être appliqué à chacune des fonctionnalités natives de JUMBF.

Bien qu'il soit certainement possible de réimplémenter toutes les fonctionnalités requises entièrement en CBOR, cela représenterait beaucoup de travail et ne supprimerait pas complètement le besoin d'un analyseur JUMBF/BMFF dans toutes les implémentations.

11.1.2. Règles de traitement

Un consommateur de manifeste C2PA ne doit jamais traiter une assertion, un magasin d'assertions, une revendication, une signature de revendication ou un manifeste C2PA qui ne figure pas dans un magasin de manifestes C2PA. De plus, lorsqu'un consommateur de manifeste C2PA rencontre une boîte ou une superboîte JUMBF dont il ne reconnaît pas l'UUID de type JUMBF, il doit sauter (et ignorer) son contenu.

REMARQUE	Cela signifie que le consommateur de manifeste C2PA peut traiter les boîtes privées qu'il connaît, mais ignorer celles qu'il ne connaît pas.

Si les boutons *« Requestable » (Demande possible)* et *« Label Present » (Étiquette présente)* sont tous deux activés dans la zone « JUMBF Description » (Description JUMBF) d'une zone JUMBF ou d'une superzone, cette zone ou superzone doit être conservée dans tout magasin de manifeste C2PA mis à jour.

REMARQUE	Les boîtes dont ces boutons sont activés sont destinées à être référencées via des URI JUMBF, et leur suppression pourrait entraîner l'échec des workflows en aval.

11.1.3. Extensions

11.1.3.1. Général

Cette section décrit les extensions de la spécification JUMBF (ISO 19566-5:2023) requises par la présente spécification.

11.1.3.2. Boîtes compressées

Afin de prendre en charge la compression des manifestes, une nouvelle boîte de contenu `brob` est prise en charge par C2PA. Basée sur une boîte similaire dans JPEG-XL (ISO/IEC 18181-2:2024), la boîte `brob` est une boîte de contenu dont le contenu est constitué des octets compressés par Brotli d'un manifeste standard ou d'un manifeste de mise à jour, comme décrit dans la clause sur les manifestes compressés. La boîte `brob` doit avoir l'ID de boîte `0x62726F62` (`brob`).

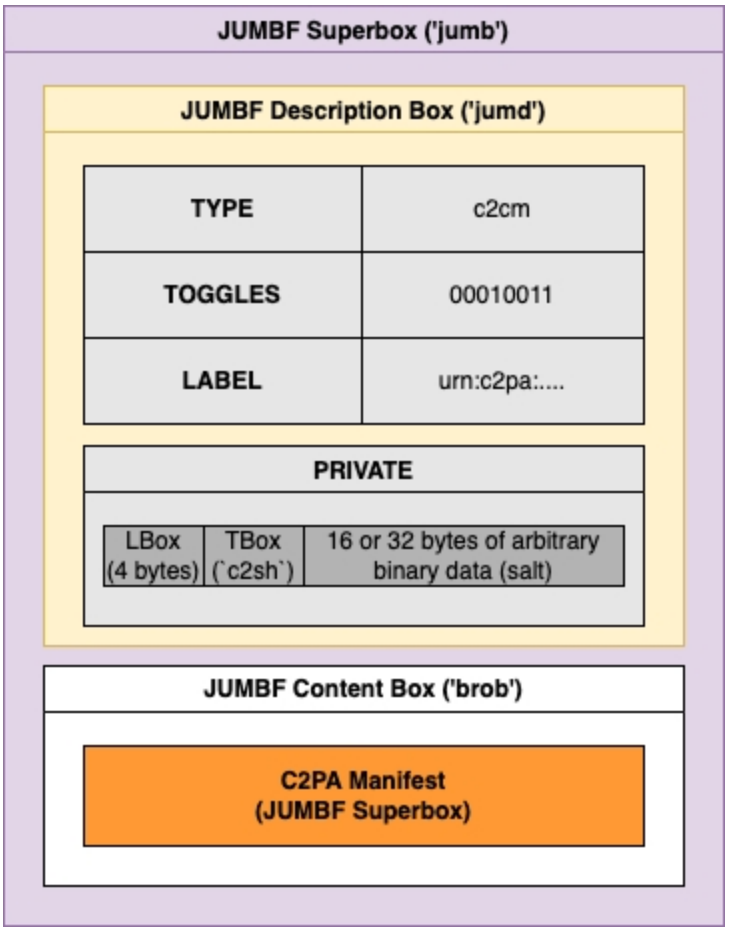

Figure 7. Exemple de manifeste compressé

Le hachage d'une boîte compressée s'effectue de la même manière que pour toute autre boîte, comme décrit dans la section 8.4.2.3, « Hachage des boîtes JUMBF ».

REMARQUE

Cela signifie que, étant donné une référence `hashed_uri` provenant d'une assertion d'ingrédient vers un manifeste C2PA via le champ `activeManifest`, le hachage est calculé en utilisant le même processus que pour toute autre superbox JUMBF : sur la boîte de description JUMBF et la boîte `brob` avec sa charge utile compressée, mais en excluant l'en-tête de la superbox. Le contenu de la boîte `brob` n'est pas décompressé au préalable pour calculer le hachage.

11.1.4. Détails de la boîte C2PA

11.1.4.1. Boîtes de description JUMBF

11.1.4.1.1. Étiquettes

Comme décrit dans la spécification JUMBF (ISO 19566-5:2023, A.3), une étiquette doit être stockée sous forme de caractères ISO/IEC 10646 dans le codage UTF-8. Les caractères compris entre U+0000 et U+001F inclus et entre U+007F et U+009F inclus, ainsi que les caractères spécifiques « / », « ; », « ? » et « # », ne sont pas autorisés dans l'étiquette. L'étiquette doit être terminée par un caractère nul.

Comme pour les étiquettes utilisées dans le cadre des URI JUMBF, les caractères U+FEFF, U+FFFF et U+D800-U+DFFF ne doivent pas non plus être utilisés.

11.1.4.1.2. Commutateurs

Toutes les boîtes de description JUMBF (ISO 19566-5:2023, A.3) utilisées dans un manifeste C2PA nécessitent une étiquette, le commutateur *Label Present* (`xxxxxx1x`) doit être activé. De plus, comme les URI JUMBF sont utilisées pour faire référence à des boîtes dans tout le système (par exemple, listes d'assertions, références aux ingrédients, etc.), le commutateur *« Requestable »* (`xxxxxx11`) doit être activé.

Lorsqu'un sel est inclus dans une boîte *PRIVATE* comme décrit à la section 8.4.2.3, « Hachage des boîtes JUMBF », le commutateur *Private* (`xxx1xxxx`) doit également être activé.

11.1.4.2. Magasin de manifestes

Les données C2PA sont sérialisées dans une structure de boîte compatible JUMBF. La boîte la plus externe est appelée « C2PA Manifest Store » (magasin de manifestes C2PA), également connue sous le nom de « Content Credentials » (informations d'identification de contenu). La figure 8, « C2PA Manifest Store », est un exemple de magasin de manifestes C2PA contenant un seul manifeste C2PA :

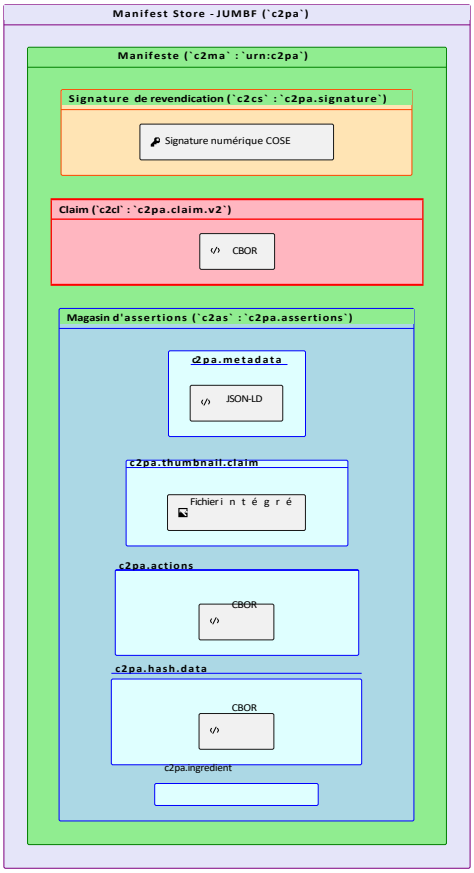

Figure 8. C2PA Manifest Store

Le magasin de manifestes C2PA est une superboîte JUMBF composée d'une série d'autres boîtes et superboîtes JUMBF, chacune identifiée par son propre UUID de type JUMBF et son étiquette dans sa boîte de description JUMBF. Le magasin de manifestes C2PA doit avoir une étiquette `c2pa`, un UUID de type JUMBF de `63327061-0011-0010-8000-00AA00389B71` (`c2pa`) et doit contenir une ou plusieurs superboîtes de manifestes C2PA, également appelées manifestes C2PA. Le magasin de manifestes C2PA peut également contenir des boîtes et des superboîtes JUMBF dont les UUID de type JUMBF ne sont pas définis dans la présente spécification.

REMARQUE L'autorisation d'autres boîtes et superboîtes permet des extensions personnalisées à C2PA ainsi que l'ajout de nouvelles boîtes dans les versions futures de cette spécification sans rompre la compatibilité.

Chaque manifeste C2PA doit contenir les données créées au moment où une revendication est émise, y compris le magasin d'assertions C2PA, une revendication C2PA et une signature de revendication C2PA. Un manifeste C2PA peut également contenir des boîtes et des superboîtes JUMBF dont les UUID de type JUMBF ne sont pas définis dans cette spécification.

L'UUID de type JUMBF pour chaque manifeste C2PA doit être soit `63326D61-0011-0010-8000-00AA00389B71` (`c2ma`), ou `6332756D-0011-0010-8000-00AA00389B71` (`c2um`) selon le type de manifeste. La case « Manifeste C2PA » doit être étiquetée avec une valeur `urn:c2pa` calculée comme décrit dans la section Identificateurs uniques.

11.1.4.3. Magasin d'assertions

Le magasin d'assertions C2PA est une superbox qui doit porter l'étiquette `c2pa.assertions` et un UUID de type JUMBF de `63326173-0011-0010-8000-00AA00389B71` (`c2as`). Il doit contenir une ou plusieurs superboîtes JUMBF (appelées

Boîtes d'assertion C2PA) dont le type JUMBF définit le type des sous-boîtes qui contiennent les données d'assertion (ISO 19566-5:2023, annexe B). Ces superboîtes doivent chacune avoir une étiquette telle que définie dans les assertions standard et doivent contenir une boîte de description JUMBF, une ou plusieurs boîtes de contenu JUMBF et éventuellement une boîte de remplissage (ISO 19566-5:2023, A.4).

Le type de contenu JUMBF (ISO 19566-5:2023, annexe B) contenues dans chaque superbox d'assertion doivent être de type CBOR (`cbor)`, JSON (`json)`, fichier intégré (`bfdb` & `bidb)` ou UUID (`uuid),` bien que tout type de contenu défini dans JUMBF (ISO 19566-5:2023) et ses amendements soit autorisé. En outre, une boîte de protection JUMBF telle que décrite dans la norme ISO 19566-4:2020 peut également être utilisée.

REMARQUE	Les assertions personnalisées contenant d'autres formats/sérialisations de données, telles que des données cryptées, sont pris en charge grâce à l'utilisation d'une boîte de contenu UUID contenant l'UUID personnalisé suivi des données (ISO 19566-5:2023, B.5).

11.1.4.4. Réclamation et signature de réclamation

La boîte de revendication C2PA doit porter l'étiquette `c2pa.claim.v2`, un UUID de type JUMBF de `6332636C-0011-0010-8000-00AA00389B71` (`c2cl`) et doit être composée d'une seule boîte de type de contenu CBOR (`cbor`).

La boîte C2PA Claim Signature doit porter l'étiquette `c2pa.signature`, avoir un UUID de type JUMBF de `63326373-0011-0010-8000-00AA00389B71` (`c2cs`) et se composer d'une seule boîte CBOR Content Type (`cbor`).

11.1.4.5. Stockage des ingrédients

Lorsqu'un manifeste C2PA comprend des assertions d'ingrédients et qu'un ingrédient contient un manifeste C2PA, ce manifeste C2PA doit être inclus afin de garantir que les données de provenance restent intactes. Ces manifestes d'ingrédients sont ajoutés au magasin de manifestes C2PA en tant que pairs du manifeste C2PA pour l'actif lui-même.

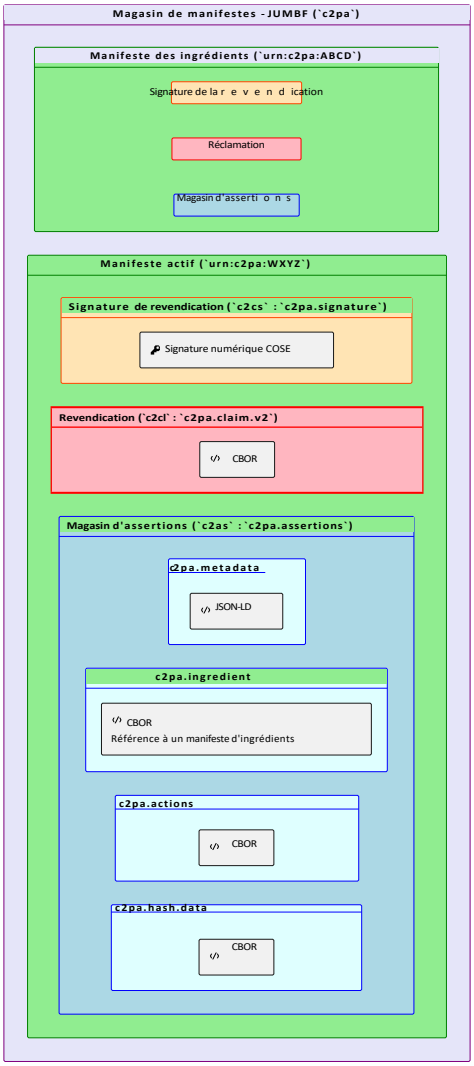

Figure 9. Magasin de manifestes C2PA avec un ingrédient

11.1.4.6. Stockage des données

IMPORTANT

Cette section est conservée à des fins historiques. Le concept de boîte de données a été
au profit d'une assertion standard qui utilise une boîte de type de contenu JUMBF Embedded File standard pour contenir les données. Pour plus d'informations sur l'assertion de données intégrées, voir la section 18.12, « Données intégrées ».

Un magasin de boîtes de données C2PA est une superboîte JUMBF qui ne doit contenir qu'une ou plusieurs boîtes de type de contenu CBOR (`cbor`). Elle ne doit contenir aucun autre type de boîte ou de superboîte JUMBF. Elle doit avoir une étiquette `c2pa.databoxes` et un UUID de type JUMBF de `63326462-0011-0010-8000-00AA00389B71` (`c2db`).

Les boîtes de type de contenu CBOR doivent avoir une étiquette `c2pa.data` (pour les données intégrées).

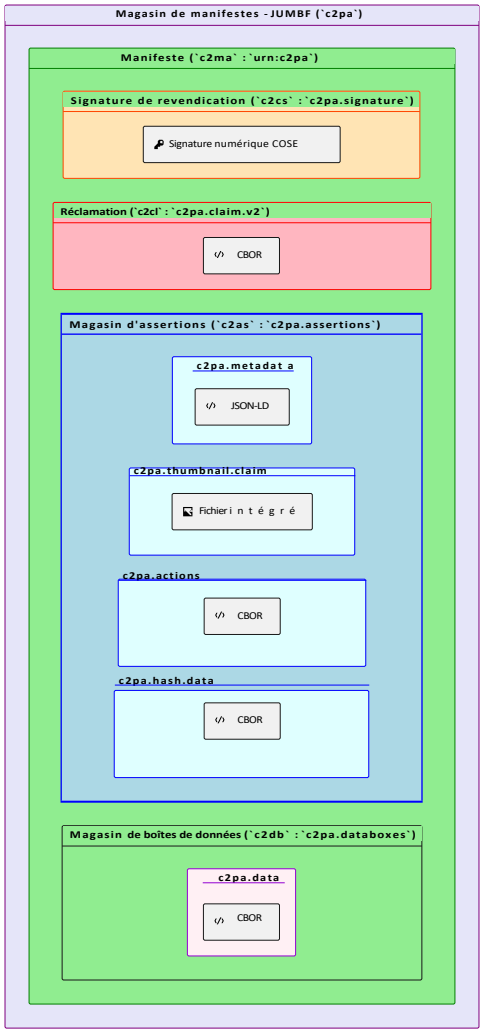

Figure 10. Magasin de manifestes C2PA avec boîtes de données

11.2. Types de manifestes

11.2.1. Points communs

Tous les manifestes C2PA doivent contenir un magasin d'assertions comprenant au moins une assertion, une revendication et une signature de revendication.

11.2.2. Manifestes standard

Un manifeste C2PA standard (UUID de type JUMBF : `63326D61-0011-0010-8000-00AA00389B71 (c2ma)`) doit contenir exactement une liaison forte à l'assertion de contenu, soit un `c2pa.hash.data`, `c2pa.hash.boxes`, `c2pa.hash.collection.data`, `c2pa.hash.bmff.v2` (obsolète) ou `c2pa.hash.bmff.v3` en fonction du type d'actif et de la version auxquels le manifeste est destiné. En raison de cette exigence, il s'agit du type de manifeste prédominant qui sera présent dans les données de provenance C2PA.

Les consommateurs de manifestes doivent également accepter les manifestes C2PA standard spécifiés avec le type JUMBF UUID `63326D64-0011-0010-8000-00AA00389B71` (c2md), mais les générateurs de revendications ne doivent pas créer de manifestes avec ce type JUMBF UUID.

REMARQUE Un manifeste C2PA standard peut être localisé soit comme manifeste actif, soit comme manifeste d'ingrédients.

11.2.3. Mettre à jour les manifestes

Il existe toutefois des flux de travail liés à la provenance dans lesquels des assertions supplémentaires doivent être ajoutées sans que le contenu numérique ne soit modifié. Dans ces flux de travail, un manifeste de mise à jour (JUMBF type UUID : `6332756D-0011-0010-8000-00AA00389B71` (`c2um`)) doit être utilisé.

Un manifeste de mise à jour ne doit pas contenir d'assertions de type `c2pa.hash.data`, `c2pa.hash.boxes`, `c2pa.hash.collection.data`, `c2pa.hash.bmff.v2` (obsolète) ou `c2pa.hash.bmff.v3`, car le contenu n'a pas été modifié et les liaisons n'ont donc pas besoin d'être mises à jour. Dans le cas d'un hachage de décalage de fichier (`c2pa.hash.data`), le magasin de manifestes C2PA doit continuer à démarrer au même décalage de fichier après la mise à jour - seule sa longueur peut changer. En outre, il ne doit pas contenir d'assertion `c2pa.hash.multi-asset`.

Un manifeste de mise à jour peut contenir des assertions de type `c2pa.actions` ou `c2pa.actions.v2`, à condition que la valeur du champ `action` de chaque action présente dans le tableau `actions` de ces assertions ne soit que l'une des valeurs suivantes :

- `c2pa.edited.metadata`
- `c2pa.opened`
- `c2pa.published`
- `c2pa.redacted`

Un manifeste de mise à jour ne doit pas contenir d'assertion de type `c2pa.actions` ou `c2pa.actions.v2` qui contient un champ `d'action` en dehors de cette liste.

Un manifeste de mise à jour peut contenir soit une assertion d'horodatage, soit une assertion d'état de certificat, soit les deux.

REMARQUE Il s'agit de l'approche de remplacement de la fonctionnalité obsolète des manifestes d'horodatage.

Un manifeste de mise à jour ne doit pas contenir d'assertion de vignette.

REMARQUE Ces exigences s'expliquent par le fait qu'un champ `d'action` ne figurant pas dans cette liste ou une vignette implique des modifications du contenu numérique.

Le manifeste de mise à jour doit contenir exactement une assertion `c2pa.ingredient.v3` qui (a) inclut à la fois les champs `activeManifest` et `claimSignature` avec des valeurs qui sont respectivement les références URI au manifeste C2PA et à la signature de revendication (ou un `c2pa.ingredient.v2` ou `c2pa.ingredient` qui inclut un champ `c2pa_manifest`) de l'actif en cours de mise à jour et (b) a la valeur `parentOf` pour le champ `de relation`.

REMARQUE Le manifeste C2PA de l'ingrédient peut être soit un manifeste standard, soit un manifeste de mise à jour.

11.2.4. Manifestes compressés

Les manifestes standard et de mise à jour peuvent être compressés dans leur intégralité à l'aide de l'algorithme de compression Brotli décrit ci-dessus. Pour les deux types de manifeste, la valeur du champ `TYPE` doit être `c2cm`, la valeur du champ label doit être identique à celle du label de la superbox du manifeste compressé, et le contenu de la boîte de contenu `brob` doit être constitué des octets compressés de l'ensemble de la superbox du manifeste. Voir la figure 7, « Exemple de manifeste compressé », pour un exemple de manifeste standard compressé.

IMPORTANT Partout dans cette spécification où il est fait référence à un manifeste standard ou de mise à jour, un manifeste standard ou de mise à jour compressé est également valide.

11.2.5. Manifestes horodatés (HISTORIQUES)

IMPORTANT Cette fonctionnalité a été abandonnée au profit de l'assertion d'horodatage et ne doit pas être écrite. par les générateurs de revendications ni lue par les consommateurs de manifestes. À la place, une assertion d'horodatage est utilisée pour atteindre les mêmes objectifs.

REMARQUE Les informations ci-dessous sont conservées à des fins historiques.

Dans certains flux de travail liés à la provenance, un manifeste standard ou de mise à jour est créé hors ligne, où il n'est pas possible d'obtenir un horodatage fiable (conformément à la norme RFC 3161) auprès d'un TSA au moment de la signature. Pour pallier cela, il est possible d'utiliser un manifeste d'horodatage (JUMBF type UUID : `6332746D-0011-0010-8000-00AA00389B71 (c2tm)`) afin d'ajouter l'horodatage lors d'une opération ultérieure, lorsqu'il est possible de contacter une TSA.

11.3. Intégration des manifestes dans divers formats de fichiers

Un manifeste C2PA peut être intégré dans divers formats de fichiers couvrant différents types de médias, notamment des images, des vidéos, des fichiers audio, des polices et des documents. L'annexe A, *Intégration des manifestes*, fournit les détails techniques sur la manière d'intégrer les manifestes C2PA dans chaque format de fichier spécifiquement pris en charge.

REMARQUE De nombreux formats d'image classiques tels que BMP ne prennent pas en charge l'intégration de données arbitraires, ce qui nécessite l'utilisation d'un manifeste externe.

11.4. Manifestes externes

Dans certains cas, il peut être impossible (ou peu pratique) d'intégrer un magasin de manifestes C2PA dans une ressource. Dans ces cas, le stockage des manifestes C2PA en dehors de la ressource est un modèle acceptable pour fournir la provenance des ressources. Le manifeste C2PA doit être stocké dans un emplacement, appelé référentiel de manifestes, facilement localisable par un consommateur de manifestes travaillant avec l'actif, par exemple par référence ou URI. Le magasin de manifestes C2PA étant une boîte JUMBF, il doit être fourni avec le type de média JUMBF, `application/c2pa`.

REMARQUE Les versions précédentes de cette spécification utilisaient le type de média `application/x-c2pa-manifest-store` pour le magasin de manifestes C2PA. Ce type de média est désormais obsolète.

Voici quelques raisons courantes d'utiliser un manifeste externe :

- Cela peut être techniquement impossible, comme avec un fichier `.txt`.
- Cela peut ne pas être pratique, par exemple lorsque la taille du magasin de manifestes C2PA est supérieure à celle du contenu numérique de la ressource.
- Cela peut ne pas être approprié, par exemple lorsque cela modifierait un actif qui ne devrait pas être modifié.

REMARQUE Un bon exemple de cela est la création d'un manifeste pour un élément préexistant.

11.5. Intégration d'une référence à un manifeste externe

Si l'actif comporte un XMP intégré et que le manifeste C2PA sera stocké en externe, il est recommandé que le générateur de revendications ajoute une clé `dcterms:provenance` au XMP, dont la valeur (une référence URI) correspond à l'emplacement du manifeste actif.

REMARQUE Une version précédente de cette spécification recommandait également d'utiliser cette méthode pour les références aux manifestes intégrés. Désormais, ce mécanisme est réservé aux manifestes externes.

Les polices ne prenant pas en charge le XMP, une méthode équivalente pour spécifier un URI vers un magasin de manifestes C2PA distant est décrite dans cette clause sur les polices.

Chapitre 12. Diagramme d'entités

La figure 11, « Diagramme d'entités C2PA », montre comment tous les éléments du système C2PA s'intègrent et interagissent les uns avec les autres.

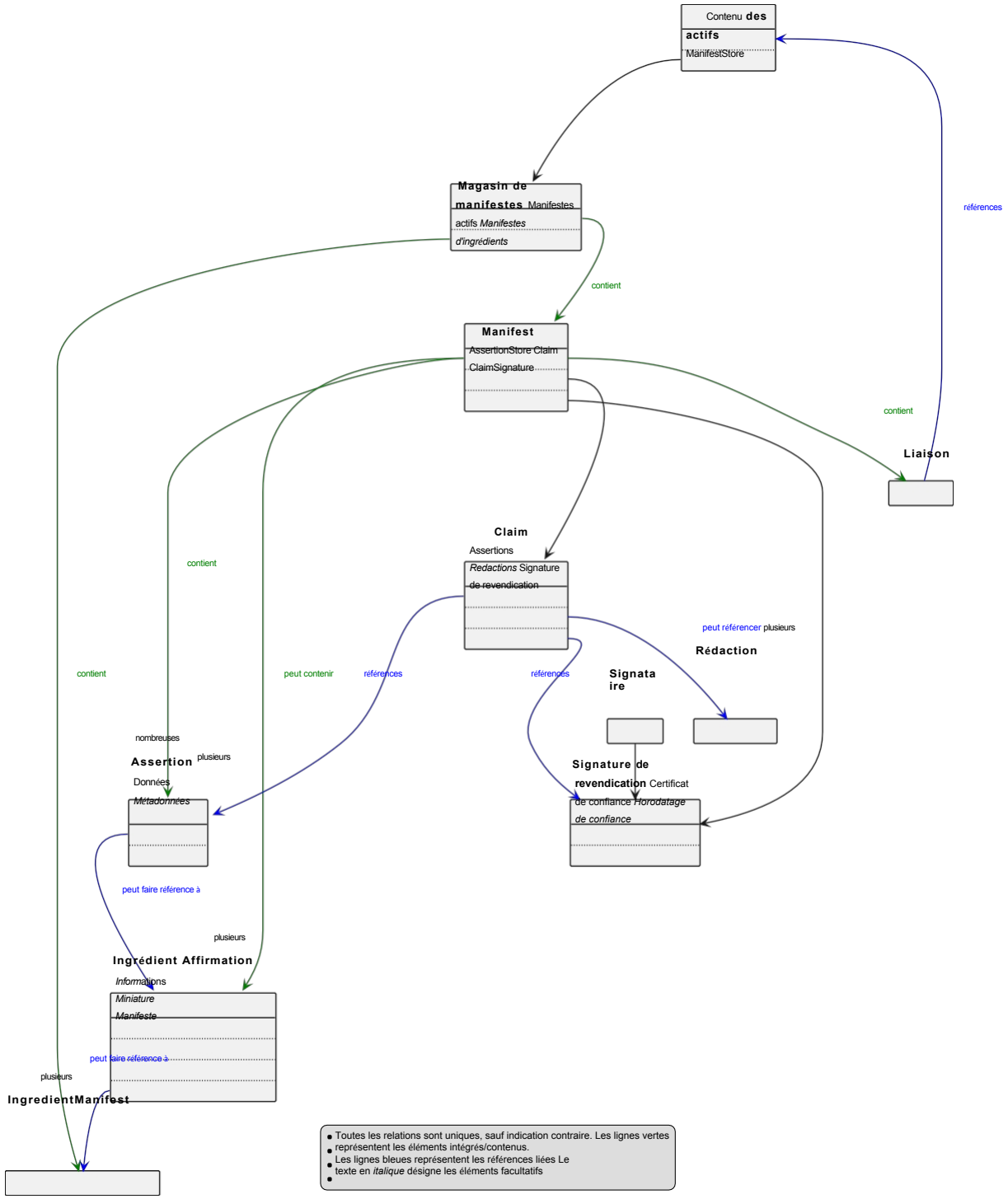

Figure 11. Diagramme d'entités C2PA

Chapitre 13. Cryptographie

13.1. Hachage

Tous les hachages cryptographiques appliqués conformément aux exigences techniques de la présente spécification doivent être générés à l'aide de l'un des algorithmes de hachage décrits dans cette section. Cette section définit à la fois :

- Une liste des algorithmes de hachage autorisés pour générer des hachages de nouveaux contenus et requis pour valider les hachages de contenus existants (la liste autorisée) ;
- Une liste des algorithmes de hachage qui doivent être pris en charge pour valider les hachages de contenu existant, mais qui ne sont pas autorisés pour générer des hachages de nouveau contenu (la liste obsolète).

REMARQUE Cette section ne régit pas les algorithmes utilisés pour les liaisons souples, comme décrit dans la section 18.10, « Liaison souple ».

REMARQUE Cette section ne régit pas les algorithmes utilisés par les assertions personnalisées qui sont définies en dehors de cette spécification.

Un algorithme ne doit apparaître que dans une seule liste. Un algorithme instancié sur plusieurs longueurs de sortie (telles que les différentes longueurs de SHA2) sera considéré comme un algorithme différent, et chaque instanciation doit être répertoriée séparément. Si un algorithme n'apparaît dans aucune des deux listes, il est interdit et ne doit pas être utilisé ou pris en charge. Les algorithmes peuvent être supprimés des listes afin d'interdire leur utilisation. Pour cette raison, les implémentations ne doivent pas prendre en charge d'algorithmes supplémentaires sur une base facultative.

Les développeurs doivent consulter cette section dans la version actuelle de la spécification lorsqu'ils publient des mises à jour logicielles et s'assurer que les algorithmes pris en charge sont conformes à celle-ci.

Ces listes établissent les algorithmes autorisés pour créer des hachages et un identifiant d'algorithme de chaîne à utiliser comme identifiant d'algorithme (généralement appelé `alg`) dans le champ correspondant des structures de données C2PA. Les sorties des fonctions de hachage doivent être stockées sous forme de valeurs binaires encodées dans CBOR sous forme de chaînes d'octets (type majeur 2) avec une longueur déclarée. Chaque fois qu'un champ contient le résultat d'une fonction de hachage, un champ de chaîne d'identifiant d'algorithme doit être présent dans la même structure, ou dans une structure englobante, ou dans la structure `claim-map` ou `claim-map-v2` afin de déclarer quel algorithme a été utilisé. Un champ d'identifiant d'algorithme de hachage doit être présent à exactement un de ces emplacements, mais si plusieurs sont présents dans la structure et ses structures englobantes, l'identifiant le plus proche doit être utilisé. Le plus proche est d'abord défini comme un identifiant qui est un champ frère de la valeur de hachage, puis la structure englobante immédiate, jusqu'à la structure racine. Si aucun identifiant n'est présent à l'un de ces emplacements, le champ `alg` de la structure `claim-map` ou `claim-map-v2` doit être utilisé.

La liste autorisée est la suivante :

- SHA2-256 (« sha256 ») ;
- SHA2-384 (« sha384 ») ;
- SHA2-512 (« sha512 »).

REMARQUE La famille d'algorithmes de hachage SHA-3 ne figure pas dans la liste autorisée afin d'assurer la cohérence avec la liste autorisée des algorithmes de signature numérique, car le COSE n'a pas encore établi d'algorithmes de signature numérique utilisant un algorithme SHA-3 comme algorithme de hachage.

La liste des algorithmes obsolètes est vide.

13.2. Signatures numériques

Toutes les signatures numériques appliquées conformément aux exigences techniques de la présente spécification doivent être générées à l'aide de l'un des algorithmes de signature numérique et des types de clés répertoriés dans la présente section. Cette section définit à la fois :

- Une liste des algorithmes de signature numérique et des types de clés autorisés pour générer des signatures pour les nouvelles signatures de revendication, ainsi que ceux requis pour valider les signatures de revendication existantes (la liste autorisée) ;
- Une liste des algorithmes de signature numérique et des types de clés qui doivent être pris en charge pour valider les signatures de revendications existantes, mais qui ne sont pas autorisés pour générer de nouvelles signatures de revendications (la liste obsolète).

REMARQUE Cette section ne régit pas les signatures numériques utilisées par les assertions personnalisées qui sont définies en dehors de cette spécification.

Ces listes établissent les algorithmes et les types de clés autorisés en faisant référence à un identifiant d'algorithme provenant des normes pertinentes qui définissent les algorithmes pour COSE et leurs correspondances avec les identifiants CBOR, y compris, mais sans s'y limiter, RFC 8152 et RFC 8230. Ces normes spécifient également l'algorithme de hachage utilisé dans le schéma de signature. Aucune disposition de la section 13.1, « Hachage », ne s'applique à cette utilisation des algorithmes de hachage ; si un algorithme de signature numérique est présent dans l'algorithme de signature numérique et le type de clé ci-dessous, l'utilisation de son algorithme de hachage spécifié dans le schéma de signature est autorisée et doit être respectée.

REMARQUE Les notes entre parenthèses dans les listes ci-dessous sont fournies à titre explicatif uniquement pour aider le lecteur.

13.2.1. Algorithmes de signature

La liste autorisée est la suivante :

- ES256 (ECDSA avec SHA-256) ;
- ES384 (ECDSA avec SHA-384) ;
- ES512 (ECDSA avec SHA-512) ;
- PS256 (RSASSA-PSS utilisant SHA-256 et MGF1 avec SHA-256) ;
- PS384 (RSASSA-PSS utilisant SHA-384 et MGF1 avec SHA-384) ;
- PS512 (RSASSA-PSS utilisant SHA-512 et MGF1 avec SHA-512) ;
- EdDSA (Edwards-Curve DSA).
    - Instance Ed25519 uniquement. Aucune autre instance EdDSA n'est autorisée.

La liste obsolète est vide.

Les implémentations doivent vérifier que les clés fournies pour les opérations de signature ou de vérification sont correctes pour l'algorithme choisi, comme l'exigent la RFC 8152, section 8.1 pour ECDSA, la RFC 8152, section 8.2 pour EdDSA, et la RFC 8230, sections 2 et 4 pour RSASSA-PSS.

Ces exigences sont résumées ici pour plus de commodité :

- ECDSA nécessite des clés de courbes elliptiques sur les courbes elliptiques P-256, P-384 ou P-521.
    - Bien qu'il soit recommandé d'utiliser des clés P-256 avec `ES256`, des clés P-384 avec `ES384` et des clés P-521 avec `ES512`, cela n'est pas obligatoire. Les implémentations doivent accepter les clés sur n'importe laquelle de ces courbes pour tous les choix d'algorithmes ECDSA.
- Ed25519 nécessite des clés à courbe elliptique sur la courbe elliptique edwards25519.
- RSASSA-PSS nécessite des clés RSA d'une longueur de module d'au moins 2048 bits.

Les implémentations doivent refuser de générer ou de vérifier des signatures avec des clés qui ne sont pas adaptées au choix de l'algorithme. Les implémentations peuvent refuser les clés RSA dont la longueur du module est supérieure à 16384 bits.

13.2.2. Utilisation de COSE

La signature pour la revendication codée CBOR est produite par CBOR Object Signing and Encryption (COSE) comme décrit dans la RFC 8152, sections 4.2 et 4.4.

NOTE

Les charges utiles peuvent être présentes à l'intérieur d'une signature COSE ou transportées séparément (« détachées »). contenu » tel que décrit dans la RFC 8152, section 4.1). En mode « contenu détaché », les données signées sont stockées en dehors de la structure `COSE_Sign1_Tagged`, et le champ `de charge utile` de la structure `COSE_Sign1_Tagged` est toujours `nul`.

Que la charge utile soit présente dans la signature `COSE_Sign1_Tagged` ou détachée de celle-ci, le contenu du champ `de charge utile` de `Sig_structure` en mémoire, lorsqu'il est construit pour calculer ou vérifier une signature numérique, doit être rempli avec ces données externes, comme décrit par l'utilisation particulière de la signature numérique dans cette spécification. Le champ `de charge utile` de `Sig_structure` ne doit jamais être `nul`.

Lors du calcul ou de la vérification de la signature d'un manifeste standard ou de mise à jour, le champ `de charge utile` de la `structure Sig_structure` contiendra le contenu de la boîte JUMBF de la revendication, comme décrit dans la section 10.3.2.4, « Signature d'une revendication » et la section 11.1, « Utilisation de JUMBF ».

13.2.3. Calcul de la signature

La signature est calculée ou vérifiée comme décrit dans la RFC 8152, section 4.4. Les exigences supplémentaires suivantes s'appliquent à la construction de `Sig_structure` :

- La valeur de l'élément `context` doit être `Signature1,` sauf si une utilisation particulière des signatures numériques dans la présente spécification spécifie d'utiliser `CounterSignature` à la place. `Signature` ne doit pas être utilisé.
- La valeur de l'élément `payload` sera spécifiée par chaque utilisation des signatures numériques dans la présente spécification.

- L'élément `external_aad` doit être un `bstr` de longueur zéro. Les données authentifiées externes ne doivent pas être utilisées.
- L'en-tête `alg` spécifiant l'algorithme de signature doit être présent dans l'élément `body_protected` tel que défini dans la RFC 8152, section 3.1.

REMARQUE

L'en-tête `alg` est un en-tête COSE standard et est donc toujours inclus dans le protégé. mappe d'en-tête avec l'entier `1` comme étiquette, tel qu'établi dans le registre des paramètres d'en-tête COSE de l'IANA. La chaîne littérale `alg` n'est jamais utilisée comme étiquette. L'élément `sign_protected` est toujours omis lors de l'utilisation de `COSE_Sign1`.

Toutes les signatures numériques dans les structures C2PA doivent être une structure `COSE_Sign1_Tagged` telle que définie dans la RFC 8152, section
4.2. `COSE_Sign1_Tagged` contient une structure `COSE_Sign1`. Les exigences supplémentaires suivantes s'appliquent à la construction de `COSE_Sign1_Tagged` :

- Le même en-tête `alg` dans la `structure Sig_structure` ci-dessus doit être présent dans le compartiment d'en-tête `protégé`.
- La valeur du champ `payload` et la présence ou non de la charge utile dans la signature ou détachée seront spécifiées par chaque utilisation des signatures numériques dans cette spécification. Lorsque la `charge utile` est spécifiée comme détachée, sa valeur ici doit être `nulle`. À l'inverse, lorsque la charge utile est présente dans la signature, le contenu binaire de la charge utile est stocké dans ce champ sous forme de `bstr`.

REMARQUE

COSE définit `nil` comme étant de type majeur 7, valeur 22 dans la RFC 8152, section 1.3, et utilise cette valeur exclusivement pour le contenu détaché. Un tableau d'octets (type majeur 2) de longueur zéro ne peut pas être utilisé pour indiquer un contenu détaché.

13.2.4. Ajout d'une heure de signature revendiquée

Un générateur de revendication peut également souhaiter établir une « heure de signature revendiquée » en ajoutant un en-tête protégé `iat`, dont la valeur est une `date numérique`. S'il est présent, il doit représenter l'heure à laquelle la signature a été générée.

REMARQUE

`Une` date numérique `(NumericDate)` est une date numérique CBOR (telle que décrite dans la RFC 8949, section 3.4.2), mais sans le tag initial 1 (date/heure basée sur l'époque). Elle n'est utilisée nulle part ailleurs dans cette spécification.

REMARQUE

Cette recommandation est basée sur des mises à jour en cours de JAdES visant à fournir un horodatage non fiable qui n'est pas utilisé pour vérifier la validité des certificats, mais qui pourrait être utilisé dans le cadre d'une expérience utilisateur. Elle pourrait être utile dans les cas où le générateur de revendications n'est pas en mesure d'accéder à une source de temps fiable, mais souhaite tout de même fournir une heure de signature.

13.2.5. Validation de la signature

Lors de la production d'une signature, si le générateur de revendications peut également agir en tant que validateur, il doit vérifier que les informations d'identification de signature sont acceptables conformément au chapitre 14, *Modèle de confiance*, et émettre un avertissement si ce n'est pas le cas. Le générateur de revendications peut néanmoins autoriser la signature avec ces informations d'identification s'il le souhaite. Cela peut être souhaitable s'il est connu que le validateur du générateur de revendications local a une configuration différente de celle des validateurs utilisés par le public visé par l'actif.

13.2.6. Validation cryptographique

Lors de la vérification d'une signature, une `structure Sig_structure` en mémoire est générée. Son champ `body_protected` est rempli avec le contenu du compartiment d'en-tête `protégé` de la structure `COSE_Sign1_Tagged` (RFC 8152, section 4.4). Pour le champ `payload`, si la charge utile a été spécifiée comme présente dans la signature, elle est remplie à partir du champ `payload` de la structure `COSE_Sign1_Tagged`. Si la charge utile a été spécifiée comme détachée, le champ `payload` de la structure `COSE_Sign1_Tagged` sera `nul.` Dans ce cas, le contenu du champ `payload` de `la structure Sig_structure` doit être renseigné à partir de la même source externe que celle utilisée pour la génération de la signature. Ceux-ci sont définis aux endroits où la signature numérique est utilisée dans cette spécification.

13.2.7. Inclusion d'icônes de signataires

Un consommateur de manifeste C2PA peut souhaiter afficher une icône ou un logo pour le signataire. Pour localiser un tel graphique, il doit rechercher dans le certificat intégré un logotype tel que défini dans la RFC 9399. Si aucun logotype n'est présent, le consommateur de manifeste peut utiliser des icônes ou des logos provenant d'autres sources, selon la manière prévue par l'implémentation.

Chapitre 14. Modèle de confiance

REMARQUE Dans cette section, le terme « utilisateur » désigne les acteurs humains qui utilisent des validateurs conformes à la norme C2PA dans des scénarios de consommation et de création.

14.1. Présentation

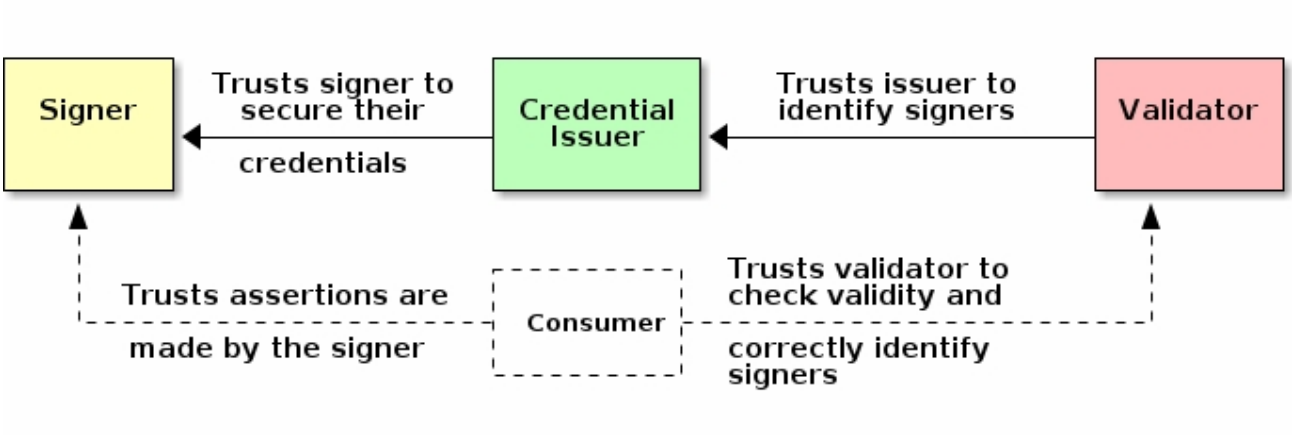

Figure 12. Diagramme du modèle de confiance C2PA

La figure 12, « Schéma du modèle de confiance C2PA », montre, en jaune, vert et rouge, les trois entités spécifiées dans le modèle de confiance, qui concerne la confiance dans l'identité d'un signataire. En pointillés, ci-dessous, se trouve le consommateur (qui n'est pas spécifié dans le modèle de confiance), qui utilise l'identité du signataire, ainsi que d'autres signaux de confiance, pour décider si les assertions faites au sujet d'un actif sont vraies.

14.2. Identité des signataires

Dans le modèle de confiance, l'identité est le moyen par lequel une clé de signature cryptographique (ou identifiant) est associée au signataire afin de prendre des décisions de confiance basées sur la signature de la revendication ou toute structure (y compris, mais sans s'y limiter, les assertions et les revendications) signée avec cette clé.

Les informations d'identification doivent être répertoriées dans les en-têtes protégés COSE de la structure `COSE_Sign1_Tagged` utilisée pour les signatures numériques dans tous les manifestes C2PA. Une seule instance d'informations d'identification doit apparaître dans l'union des en-têtes protégés et non protégés. Les structures `COSE_Sign1_Tagged` sans identifiant, ou comportant deux identifiants ou plus, doivent être rejetées. La répétition du même identifiant plus d'une fois, y compris séparément dans les en-têtes protégés et non protégés, constitue également un cas de deux identifiants ou plus et doit être rejetée.

REMARQUE Les versions antérieures de cette spécification autorisaient également l'apparition de l'identifiant dans les en-têtes non protégés COSE.

La manière dont l'identifiant est stocké dans la valeur de l'en-tête, la manière dont les chaînes de confiance sont construites sont spécifiées, et des informations supplémentaires peuvent être trouvées dans la section 14.5, « Certificats X.509 ».

14.3. États de validation

14.3.1. Général

Un validateur est un consommateur de manifeste qui émettra certaines déclarations de validation concernant cet actif et son manifeste actif associé. Le processus de récupération de ces déclarations est décrit dans la section validation. L'acteur qui consomme l'actif, généralement par l'intermédiaire de son agent utilisateur et de son interface utilisateur, doit ensuite interpréter ces déclarations afin de parvenir à ses propres conclusions quant à la provenance de l'actif qu'il consomme. Ces conclusions seront tirées de ces déclarations et du contenu de l'actif lui-même.

14.3.2. États du manifeste

Sur la base de ces déclarations, un manifeste C2PA peut être l'un des suivants :

- Bien formé
- Valide
- Fiable

REMARQUE Tout manifeste fiable est également valide, et tout manifeste valide est également bien formé.

14.3.3. États des actifs

Si un validateur signale que les parties de l'actif couvertes par des liaisons de contenu n'ont pas été modifiées depuis la production du manifeste actif [Section 15.12, « Valider le contenu de l'actif »] et que son manifeste actif est valide ou fiable, alors l'actif lui-même est un actif valide.

14.3.4. Manifeste bien formé

Un manifeste C2PA est bien formé si la validation détermine que chacune des conditions suivantes est vraie :

- Le contenu du manifeste respecte les exigences normatives de la présente spécification, qui sont validées via le processus de validation.
- Seules les assertions autorisées pour le type spécifique du manifeste sont présentes [Section 15.10.1, « Valider les assertions correctes pour le type de manifeste »].
- Les assertions du manifeste satisfont à toutes les exigences relatives aux assertions [Section 15.10.3, « Validation des assertions »].
- Tous les ingrédients présents dans le manifeste répondent à toutes les exigences relatives aux ingrédients [Section 15.11, « Validation des ingrédients »].

14.3.5. Manifeste valide

Un manifeste C2PA est valide si la validation détermine que chacune des conditions suivantes est remplie :

- Le manifeste est bien formé [Section 14.3.4, « Manifeste bien formé »].
- Le manifeste n'a pas été modifié depuis sa signature [Section 13.2.6, « Validation cryptographique »].
- La signature de la revendication reçoit un code de réussite `claimSignature.validated` [Section 15.7, « Valider la signature »].
- Validation de la revendication signature validité période reçoit le succès code de `claimSignature.insideValidity` [Section 15.8, « Valider l'horodatage »].
- Les informations d'identification du signataire du manifeste C2PA ne sont pas rejetées avec un code d'échec `signingCredential.ocsp.revoked` ou `signingCredential.ocsp.unknown` [Section 15.9, « Valider les informations de révocation des informations d'identification »].

Si un manifeste C2PA est valide, la revendication du manifeste peut être attribuée au générateur de revendication identifié par le champ champ `claim_generator_info` de la revendication [Section 10.2.3, « Informations sur le générateur de revendications »].

14.3.6. Manifeste fiable

Un manifeste C2PA est considéré comme fiable si la validation détermine que chacune des conditions suivantes est remplie :

- Le manifeste est valide [Section 14.3.5, « Manifeste valide »].
- Les informations d'identification de signature du manifeste C2PA reçoivent le code de réussite `signingCredential.trusted.` [Section 15.7, « Valider la signature »].

14.4. Listes de confiance

14.4.1. Signataires C2PA

Un validateur doit conserver les informations suivantes pour évaluer les signataires C2PA :

- Une liste des valeurs EKU (Extended Key Usage) acceptées.
- Pour chaque valeur EKU acceptée, une liste des ancres de confiance des certificats X.509.

Pour l'EKU `c2pa-kp-claimSigning` (1.3.6.1.4.1.62558.2.1), la liste des ancres de confiance doit inclure, sans s'y limiter, les ancres de confiance des signataires fournies par C2PA (c'est-à-dire la liste de confiance C2PA).

REMARQUE Certaines de ces listes peuvent être vides.

En plus de la liste des ancres de confiance pour l'EKU `c2pa-kp-claimSigning` fournie dans la liste de confiance C2PA, un validateur doit permettre à un utilisateur de configurer des ancres de confiance supplémentaires pour cet EKU et/ou pour d'autres EKU (par exemple, `id-kp-emailProtection` (1.3.6.1.5.5.7.3.4) ou `id-kp-documentSigning` (1.3.6.1.5.5.7.3.36)). Un validateur doit fournir des options par défaut ou proposer des listes gérées par des parties externes auxquelles l'utilisateur peut adhérer afin de remplir le magasin d'ancres de confiance du validateur pour les signataires C2PA.

REMARQUE Les versions précédentes de cette spécification exigeaient la présence d'`id-kp-emailProtection` ou d'`id-`

`kp-documentSigning`, de sorte que l'inclusion d'au moins l'un de ces deux EKU dans le certificat d'un signataire, avec `c2pa-kp-claimSigning`, peut améliorer la compatibilité avec les validateurs plus anciens.

14.4.2. Autorités d'horodatage

Un validateur doit tenir à jour une liste des ancres de confiance des certificats X.509 pour les autorités d'horodatage (TSA), qui doit être distincte des listes pour les signataires C2PA. Cette liste doit inclure, sans s'y limiter, les ancres de confiance des autorités d'horodatage fournies par la C2PA (c'est-à-dire la liste de confiance TSA de la C2PA).

REMARQUE Cette liste peut être vide.

En plus de la liste des ancres de confiance fournie dans la liste de confiance TSA de la C2PA, un validateur doit permettre à un utilisateur de configurer des magasins d'ancres de confiance TSA supplémentaires et doit fournir des options par défaut ou proposer des listes gérées par des parties externes que l'utilisateur peut choisir d'utiliser pour remplir le magasin d'ancres de confiance du validateur pour les autorités d'horodatage.

14.4.3. Stockage privé des informations d'identification

Un validateur peut également autoriser l'utilisateur à créer et à gérer un magasin privé de certificats de signature. Ce magasin est conçu comme un « carnet d'adresses » des certificats auxquels l'utilisateur a choisi de faire confiance sur la base d'une relation hors bande. S'il existe, le magasin privé de certificats ne s'applique qu'à la validation des manifestes C2PA signés et ne s'applique pas à la validation des horodatages. S'il existe, le magasin privé de certificats ne permet que la confiance directe dans les certificats du signataire ; les entrées du magasin privé de certificats ne peuvent pas émettre de certificats et ne doivent pas être incluses comme ancres de confiance lors de la validation.

Un validateur ne doit pas être préconfiguré avec des entrées dans un magasin d'informations d'identification privé.

Un validateur ne doit ajouter des entrées à un magasin d'informations d'identification privé qu'en réponse à une demande de l'utilisateur visant à faire confiance à l'information d'identification. De même, un validateur ne doit supprimer des entrées d'un magasin d'informations d'identification privé qu'en réponse à une demande de l'utilisateur visant à ne plus faire confiance à l'information d'identification.

14.5. Certificats X.509

Les certificats X.509 sont stockés conformément à la norme RFC 9360 (*CBOR Object Signing and Encryption (COSE) : paramètres d'en-tête pour le transport et la référence des certificats X.509*). Pour plus de commodité, la définition de `x5chain` est reproduite ci-dessous.

IMPORTANT Cette spécification ajoute des exigences supplémentaires à celles de la RFC 9360, qui sont énumérées après le texte cité. En particulier, cette spécification exige que tous les certificats intermédiaires de la chaîne de certificats du signataire soient inclus dans l'en-tête `x5chain`, et exige que les générateurs de revendications placent toujours l'en-tête `x5chain` dans le compartiment d'en-têtes protégé.

> x5chain : ce paramètre d'en-tête contient un tableau ordonné de certificats X.509. Les certificats doivent être classés en commençant par le certificat contenant la clé de l'entité finale, suivi du certificat qui l'a signé, et ainsi de suite. Il n'y a aucune exigence pour l'ensemble du

chaîne doit être présente dans l'élément s'il y a lieu de croire que la partie utilisatrice dispose déjà des certificats manquants ou peut les localiser. Cela signifie que la partie utilisatrice est toujours tenue d'établir le chemin, mais qu'un chemin candidat est proposé dans ce paramètre d'en-tête.

Le mécanisme de confiance DOIT traiter tous les certificats de ce paramètre comme des entrées non fiables. La présence d'un certificat auto-signé dans le paramètre NE DOIT PAS entraîner la mise à jour de l'ensemble des ancres de confiance sans confirmation hors bande. Comme le contenu de ce paramètre d'en-tête est une entrée non fiable, le paramètre d'en-tête peut se trouver dans le compartiment d'en-tête protégé ou non protégé. L'envoi du paramètre d'en-tête dans le compartiment d'en-tête non protégé permet à un intermédiaire de supprimer ou d'ajouter des certificats.

Le certificat d'entité finale DOIT être protégé contre les altérations par COSE. Cela peut, par exemple, être réalisé en envoyant le paramètre d'en-tête dans l'en-tête protégé, en envoyant un « x5chain » dans l'en-tête non protégé combiné à un « x5t » dans l'en-tête protégé, ou en incluant le certificat d'entité finale dans l'external_aad.

Ce paramètre d'en-tête permet de transporter un seul certificat X.509 ou une chaîne de certificats X.509 dans le message.

- Si un seul certificat est transmis, il est placé dans une chaîne d'octets CBOR.
- Si plusieurs certificats sont transmis, un tableau CBOR de chaînes d'octets est utilisé, chaque certificat se trouvant dans sa propre chaîne d'octets.

e validateur ne doit disposer que des certificats de ses ancres de confiance. Par conséquent, lors de la création de l'en-tête `x5chain` dans e cadre de la signature, le générateur de revendications doit inclure le certificat du signataire et toutes les autorités de certification ntermédiaires dans la valeur de l'en-tête. Le certificat de l'ancre de confiance (également appelé certificat racine) ne doit pas être inclus.

L élément `subjectPublicKeyInfo` du premier ou du seul certificat sera la clé publique utilisée pour valider la signature. L'élément `validity` de la séquence `tbsCertificate` fournit la période de validité du certificat.

Une version précédente de cette spécification exigeait que les générateurs de revendications écrivent uniquement la chaîne `x5chain` afin d'éviter la possibilité improbable que l'étiquette entière `33` ne soit pas normalisée.

L'étiquette entière `33` a désormais été normalisée, et cette spécification l'adopte désormais comme norme et déconseille l'utilisation de l'étiquette sous forme de chaîne. Par conséquent :

- Les générateurs de revendications ne doivent utiliser que l'étiquette entière `33` lorsqu'ils insèrent cet en-tête dans une signature COSE. Les générateurs de revendications peuvent continuer à écrire l'étiquette de chaîne `x5chain,` mais cette pratique est désormais dépréciée et les générateurs de revendications

générateurs de revendications doivent être mis à jour pour n'utiliser que l'étiquette entière. Les générateurs de revendications doivent placer cet en-tête uniquement dans le compartiment d'en-tête protégé de la signature COSE, comme requis ci-dessus.

- Les validateurs doivent accepter soit la chaîne `x5chain`, soit l'entier `33` comme étiquette pour cet en-tête. Si les deux étiquettes sont présentes, les validateurs doivent utiliser l'en-tête avec l'étiquette entière `33` et ignorer l'en-tête avec la chaîne `x5chain` comme étiquette. Les validateurs doivent accepter l'en-tête provenant du compartiment protégé ou non protégé, afin de maintenir la compatibilité avec les versions précédentes de cette spécification. Conformément à la section 14.2, « Identité des signataires », si cet en-tête apparaît à la fois dans les compartiments protégés et non protégés avec la même étiquette, un validateur doit rejeter la signature de la revendication comme étant mal formée en raison de la présence de plusieurs informations d'identification.

14.5.1. Profils de certificats

14.5.1.1. Exigences générales

Cette section définit les exigences permettant de valider qu'un certificat X.509 est acceptable en tant qu'identifiant de signature, comme décrit dans la section 15.7, « Valider la signature ».

Tous les certificats doivent satisfaire aux exigences suivantes.

- Le champ « `signatureAlgorithm` » doit contenir l'une des valeurs suivantes :

`ecdsa-with-SHA256`

RFC 5758, section 3.2

`ecdsa-with-SHA384`

RFC 5758, section 3.2

`ecdsa-with-SHA512`

RFC 5758, section 3.2

`sha256WithRSAEncryption`

RFC 8017, annexe A.2.4

`sha384WithRSAEncryption`

RFC 8017, annexe A.2.4

`sha512WithRSAEncryption`

RFC 8017, annexe A.2.4

`id-RSASSA-PSS`

RFC 8017, annexe A.2.3

`id-Ed25519`

RFC 8410 section 3

- Si le champ `algorithm` du champ `signatureAlgorithm` est `id-RSASSA-PSS`, le champ `parameters` est de type `RSASSA-PSS-params`. Ses champs doivent répondre aux exigences suivantes, telles que définies dans la RFC 8017, annexe A.2.3 :
  - Le champ `hashAlgorithm` doit être présent.
  - Le champ `algorithm` du champ `hashAlgorithm` doit avoir l'une des valeurs suivantes, telles que définies dans la RFC 8017, annexe B.1 :
    - `id-sha256`.
    - `id-sha384`.
    - `id-sha512`.
  - Le champ `maskGenAlgorithm` doit être présent.
  - Le champ `algorithm` du champ `parameters` du champ `maskGenAlgorithm` doit être égal au champ `algorithm` du champ `hashAlgorithm`.
- Si le champ « `algorithm` » du champ « `subjectPublicKeyInfo` » du certificat est « `id-ecPublicKey` », le champ « `parameters` » doit être l'une des courbes nommées suivantes, issues de la section 2.1.1.1 de la norme RFC 5480 :
  - `prime256v1`.
  - `secp384r1`.
  - `secp521r1`.
- Si le champ `algorithme` du champ `algorithme` du `subjectPublicKeyInfo` du certificat est `rsaEncryption` ou `id-RSASSA-PSS`, le champ `module` du champ `paramètres` doit avoir une longueur d'au moins 2048 bits.

Tous les certificats, à l'exception de ceux qui se trouvent dans le magasin d'informations d'identification privées pour les certificats X.509, doivent remplir les exigences supplémentaires suivantes pour être acceptables.

- La version doit être `v3` conformément à la section 4.1.2.1 de la RFC 5280.
- Les champs facultatifs `issuerUniqueID` et `subjectUniqueID` de la séquence `TBSCertificate` ne doivent pas être présents, conformément à la section 4.1.2.8 de la RFC 5280.
- L'extension Basic Constraints doit respecter la norme RFC 5280, section 4.2.1.9. En particulier, l'une des conditions suivantes doit être remplie :
  - Si la clé publique certifiée peut être utilisée pour vérifier les signatures de certificats, l'extension Basic Constraints doit être présente avec la valeur booléenne `cA` affirmée.
  - Si la clé publique certifiée ne peut pas être utilisée pour vérifier les signatures de certificats, soit l'extension Basic Constraints doit être absente, soit la valeur booléenne `cA` dans l'extension ne doit pas être affirmée et le bit `keyCertSign` dans l'extension d'utilisation de la clé ne doit pas être affirmé.
- L'extension « Authority Key Identifier » doit être présente dans tout certificat qui n'est pas auto-signé, conformément à la section 4.2.1.1 de la RFC 5280.

- Conformément à la section 4.2.1.2 de la RFC 5280, l'extension Subject Key Identifier doit être présente dans tout certificat agissant en tant qu'autorité de certification. Elle doit être présente dans les certificats d'entité finale.
- Conformément à la section 4.2.1.3 de la RFC 5280, l'extension Key Usage doit être présente et marquée comme critique. Les certificats utilisés pour signer les manifestes C2PA doivent affirmer le bit `digitalSignature`. Le bit `keyCertSign` ne doit être affirmé que si le booléen `cA` est affirmé dans l'extension Basic Constraints.
- L'extension Extended Key Usage (EKU) doit être présente et non vide dans tout certificat où l'extension Basic Constraints est absente ou où la valeur booléenne `cA` n'est pas affirmée, conformément à la section 4.2.1.12 de la RFC 5280. Ces certificats sont communément appelés certificats « d'entité finale » ou « feuilles ».
    - L'EKU `anyExtendedKeyUsage` (2.5.29.37.0) ne doit pas être présente.
    - Un certificat qui signe des contre-signatures d'horodatage doit être valide pour l'objectif `id-kp-timeStamping` (1.3.6.1.5.5.7.3.8).
    - Un certificat qui signe les réponses OCSP pour les certificats doit être valide pour l'objectif `id-kp-OCSPSigning` (1.3.6.1.5.5.7.3.9).
    - Si un certificat est valide pour l'une ou l'autre des finalités `id-kp-timeStamping` ou `id-kp-OCSPSigning`, il doit être valide pour exactement l'une de ces deux finalités, et non valide pour toute autre finalité.
    - Un certificat ne doit pas être valide à d'autres fins que celles énumérées ci-dessus, mais la présence d'EKU non mentionnés dans ce profil et ne figurant pas dans la liste des EKU du magasin de configuration ne doit pas entraîner le rejet du certificat.

14.5.1.2. Chaîne de confiance des certificats

Lors de la validation d'un certificat en tant qu'identifiant de signature, si le certificat est présent dans le magasin d'identifiants privés pour les X.509, le certificat est accepté. Le magasin d'informations d'identification privées n'est pas consulté lors de la validation des horodatages.

Si le certificat n'est pas présent dans le magasin d'informations d'identification privé, ou si le validateur n'en implémente pas, la chaîne de confiance doit être construite et validée conformément à la procédure décrite dans la section 6 de la RFC 5280 pour l'objectif particulier requis (signature, horodatage ou signature OCSP) et pour le magasin d'ancrage de confiance approprié à cet objectif. Tout échec de cet algorithme de validation entraîne le rejet de la chaîne. Le magasin d'informations d'identification privées n'est jamais inclus lors de la création de chaînes de certificats ; les certificats du magasin d'informations d'identification privées ne peuvent pas servir d'autorités de certification.

Seuls les certificats d'entité finale doivent être utilisés pour signer les revendications C2PA ou les horodatages. Un certificat CA ne doit pas être utilisé à ces fins. Tout certificat CA (lorsque la valeur booléenne `cA` dans l'extension Basic Constraints est affirmée) utilisé pour valider une signature sur une revendication C2PA, un horodatage ou une réponse OCSP doit être rejeté avec un code d'échec `signingCredential.untrusted`.

Un validateur doit s'assurer qu'un certificat de signature est autorisé pour l'usage auquel il est destiné et rejeter les certificats utilisés à des fins non autorisées. Un certificat est autorisé pour un usage particulier si l'identifiant d'objet (OID) EKU de cet usage est présent dans l'extension Extended Key Usage du certificat (RFC 5280, section 4.2.1.12).

Lors de la validation d'un certificat utilisé pour signer une revendication C2PA, le certificat de signature doit comporter au moins l'un des EKU pour lesquels le validateur dispose d'une liste associée d'ancres de confiance (voir section 14.4.1, « Signataires C2PA »), et le validateur doit

utiliser uniquement les ancres de confiance qu'il associe aux EKU présentes dans le certificat.

Lors de la validation d'une chaîne de certificats utilisée pour signer un horodatage, le certificat de signature doit disposer de l'EKU `id-kp-timeStamping` (1.3.6.1.5.5.7.3.8).

Lors de la validation d'une chaîne de certificats utilisée pour signer une réponse OCSP, le certificat de signature doit avoir l'EKU `id-kp-OCSPSigning` (1.3.6.1.5.5.7.3.9).

À l'exception des certificats acceptés via le magasin de certificats privé pour les certificats X.509, un validateur doit vérifier la conformité d'un certificat avec le profil de certificat et rejeter les certificats non conformes. Cela inclut l'exigence de la présence de l'extension Extended Key Usage, ainsi que l'autorisation d'un certificat pour un seul des trois objectifs énumérés dans cette section : signature C2PA, signature d'horodatage ou signature de réponse OCSP.

Comme décrit dans le profil de certificat, les certificats d'autorité de certification (CA) qui émettent des certificats ne sont pas tenus d'avoir une extension EKU et n'en ont généralement pas. Si une telle extension est présente, elle doit être ignorée. Cette exigence s'applique uniquement aux certificats d'entité finale signant des manifestes C2PA, des horodatages ou des réponses OCSP. Les certificats CA ne doivent pas être utilisés pour signer des manifestes C2PA, des horodatages ou des réponses OCSP.

14.5.2. Révocation de certificat

Les certificats X.509 prennent en charge les requêtes d'état de révocation. Un générateur de revendications doit utiliser le protocole OCSP (Online Certificate Status Protocol, RFC 6960) et l'agrafage OCSP (tel que conceptualisé à l'origine dans la RFC 6066, section 8, mais mis en œuvre comme décrit dans la présente clause) pour mettre en œuvre la révocation. Le générateur de revendications ne doit pas utiliser les listes de révocation de certificats (CRL, RFC 5280). ``

REMARQUE	L'utilisation des CRL nécessite le téléchargement de la liste complète des certificats révoqués pour chaque autorité de certification. rencontrée, ce qui peut prendre beaucoup de temps. Bien qu'une CRL puisse être incluse de la même manière qu'une réponse OCSP est agrafée, la taille potentielle d'une CRL par rapport à une réponse OCSP rend également cette solution peu souhaitable.

Une autorité de certification conforme doit inclure une extension AuthorityInfoAccess (AIA) (RFC 5280, section 4.2.2.1) dans les certificats qu'elle délivre afin de fournir des informations d'accès au service OCSP exploité par l'autorité de certification.

Si le certificat comporte une extension AIA, les informations de révocation doivent être stockées dans un en-tête non protégé de la structure `COSE_Sign1` avec l'étiquette de chaîne `rVals` et le schéma de la valeur doit suivre la règle `rVals` de l'exemple 3, « CDDL pour `rVals` » :

Exemple 3. CDDL pour `rVals`

```
; Version CBOR de rVals et structures associées basées sur le schéma JSON dans
https://www.etsi.org/deliver/etsi_ts/119100_119199/11918201/01.01.01_60/ts_11918201v010 101p.pdf section
5.3.5.2
rVals = {
   « ocspVals » : [1* bstr]
}
```

REMARQUE

La définition ci-dessus est une adaptation CBOR d'un sous-ensemble du schéma de JAdES, section 5.3.5.2, qui stocke uniquement les réponses OCSP sous forme de chaînes binaires.

Avant de signer une revendication, si le certificat d'un signataire comporte l'extension AIA, un générateur de revendications doit interroger le service OCSP qui y est indiqué, capturer la réponse et la stocker dans un élément du tableau `ocspVals` de l'en-tête `rVals`. Le générateur de revendications doit faire de même pour tous les certificats CA intermédiaires qu'il inclut dans la signature de la revendication.

À la réception de la revendication, les réponses OCSP agrafées doivent être validées conformément à la section 3.2 de la RFC 6960.

Le processus de validation du statut de révocation d'un certificat après la signature d'une revendication est décrit plus en détail dans la section Valider les informations de révocation des informations d'identification.

Chapitre 15. Validation

15.1. Processus de validation

15.1.1. Description

La validation d'un manifeste C2PA est un processus en plusieurs étapes qui consiste à valider les assertions, les revendications et les signatures associées contenues dans celui-ci, ainsi que (pour les manifestes actifs uniquement) à valider toutes les liaisons matérielles associées. Ce processus de validation est effectué par un validateur, qui est un acteur matériel ou logiciel qui met en œuvre les algorithmes de validation décrits dans la présente clause.

15.1.2. Phases de validation

Ces phases, qui ne sont pas classées dans un ordre particulier, sont décrites dans les clauses suivantes :

- Section 15.10, « Valider les assertions » : Validation des assertions.
- Section 15.11, « Valider les ingrédients » : validation des ingrédients, le cas échéant.
- Section 15.8, « Valider l'horodatage » : validation de l'horodatage.
- Section 15.9, « Valider les informations de révocation des informations d'identification » : validation des informations de révocation des informations d'identification.
- Section 15.7, « Valider la signature » : validation de la signature de la revendication.
- Section 15.12, « Valider le contenu de l'actif » : validation du contenu de l'actif.

Comme décrit dans la section 14.3, « États de validation », un manifeste C2PA peut être considéré comme bien formé, valide ou fiable en fonction des résultats de ces étapes.

15.1.3. Représentation visuelle

La figure 13, « Validation d'une revendication », est une représentation visuelle du processus de validation d'un manifeste C2PA.

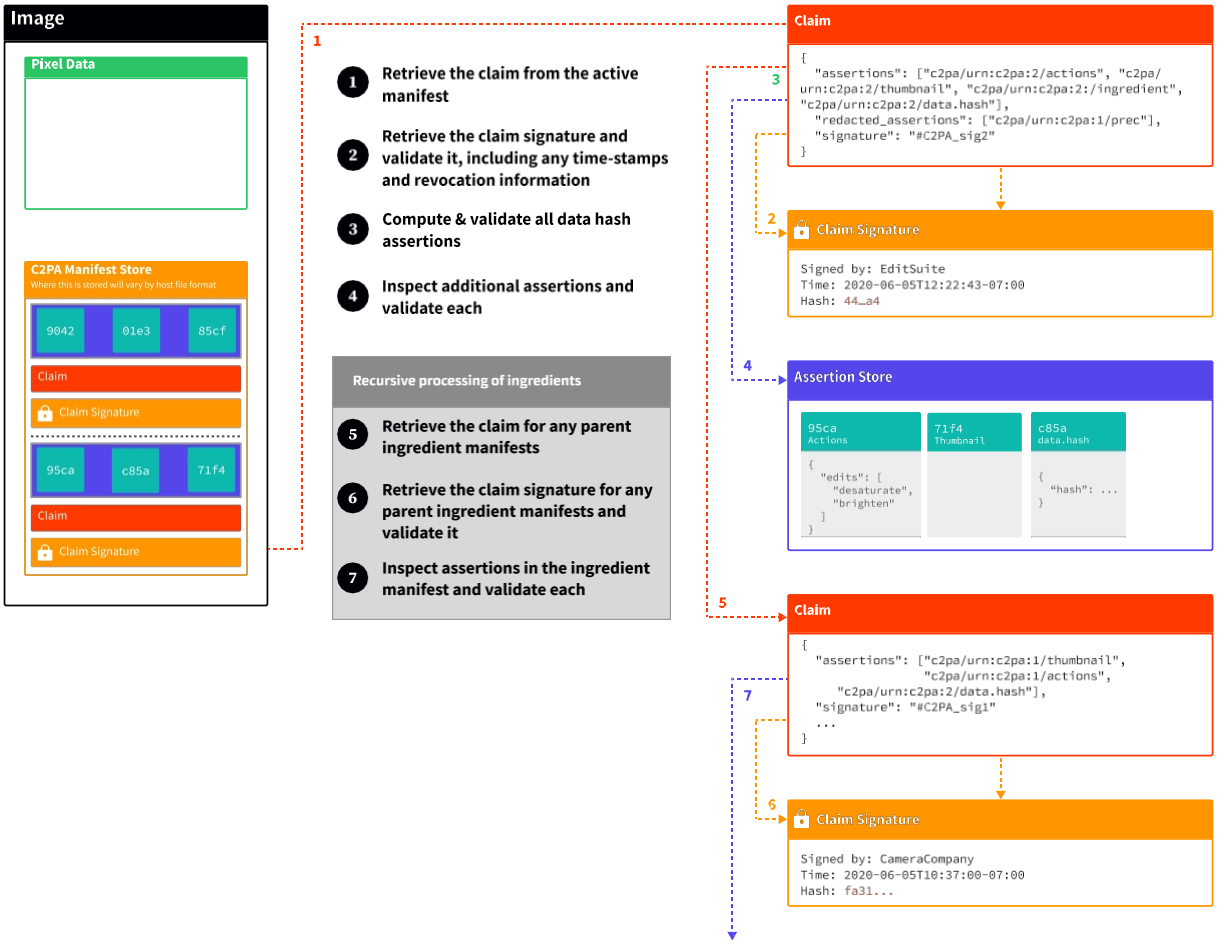

Figure 13. Validation d'une revendication

REMARQUE

En cas de divergence entre la représentation visuelle et le texte, le texte fait foi.

15.2. Renvoi des résultats de validation

15.2.1. Général

L'algorithme de validation doit renvoyer un ensemble consolidé de résultats de validation pour tous les manifestes du magasin de manifestes C2PA de l'actif, y compris le manifeste actif ainsi que tous les autres manifestes du magasin de manifestes C2PA référencés via des assertions d'ingrédients.

Les résultats de la validation sont exprimés à l'aide d'un ensemble standard de codes de réussite, d'information et d'échec, tels que définis ci-dessous dans la section 15.2.2, « Codes d'état standard ». Des codes d'état personnalisés sont également autorisés lorsqu'un générateur de réclamations a besoin d'enregistrer certaines informations d'état spécifiques au processus. Les codes personnalisés doivent respecter la même syntaxe que les espaces de noms spécifiques à une entité, par exemple `com.litware`.

Lorsqu'un générateur d'allégations ajoute un élément constitutif via une assertion d'ingrédient, il doit agir en tant que validateur et appliquer à cet ingrédient l'algorithme de validation complet décrit dans cette section. Le générateur d'allégations doit enregistrer les

résultats de l' de validation de l' ingrédient, par le suivant CDDL Définition schéma, comme la valeur de le champ `validationResults` dans l'assertion d'ingrédient.

CDDL pour les résultats de validation

```
; Codes de validation

; Codes de réussite
$status-code /= « assertion.accessible »
$status-code /= « assertion.bmffHash.match »
$code-statut /= « assertion.boxesHash.match »
$status-code /= « assertion.collectionHash.match »
$code-statut /= « assertion.dataHash.match »
$code-statut /= « assertion.hashedURI.match »
$code-statut /= « signature-revendication.validité-interne »
$code-statut /= « signature-revendication.validated »
$code-statut /= « ingrédient.signature-revendication.validée »
$code-statut /= « ingrédient.manifeste.validé »
$code-statut /= « signingCredential.ocsp.notRevoked »
$code-statut /= « signingCredential.trusted »
$code-statut /= « timeStamp.trusted »
$code-statut /= « timeStamp.validated »

; Codes d'information
$code-statut /= « ingrédient.provenance-inconnue »
$code-statut /= « manifest.unknownProvenance »
$code-statut /= « signingCredential.ocsp.inaccessible »
$code-statut /= « signingCredential.ocsp.skipped »
$code-statut /= « timeOfSigning.insideValidity »
$code-statut /= « heure-de-signature.hors-validité »
$code-statut /= « timeStamp.malformed »
$code-statut /= « horodatage.incompatibilité »
$code-statut /= « horodatage.hors-validité »
$code-statut /= « horodatage.non fiable »
$code-statut /= « assertion.dataHash.additionalExclusionsPresent »

; Codes d'échec
$code-statut /= « algorithm.deprecated »
$code-statut /= « algorithme.non-pris-en-charge »
$code-statut /= « assertion.action.ingredientMismatch »
$code-statut /= « assertion.action.malformed »
$code-statut /= « assertion.action.redacted »
$code-statut /= « assertion.action.redactionMismatch »
$code-statut /= « assertion.bmffHash.malformé »
$code-statut /= « assertion.bmffHash.mismatch »
$code-statut /= « assertion.boxesHash.mismatch »
$code-statut /= « assertion.boxesHash.malformed »
$code-statut /= « assertion.boxesHash.unknownBox »
$code-statut /= « assertion.cloud-data.hardBinding »
$code-statut /= « assertion.cloud-data.actions »
$code-statut /= « assertion.cloud-data.hardBinding »
$code-statut /= « assertion.cloud-data.malformed »
$code-statut /= « assertion.collectionHash.incorrectFileCount »
$code-statut /= « assertion.collectionHash.invalidURI »
$code-statut /= « assertion.collectionHash.malformé »
$code-statut /= « assertion.collectionHash.mismatch »
$code-statut /= « assertion.dataHash.malformed »
$code-statut /= « assertion.dataHash.incompatibilité »
$code-statut /= « assertion.hashedURI.incompatibilité »
$code-statut /= « assertion.inaccessible »
$code-statut /= « assertion.ingrédient.malformé »
```

```
$code-statut /= « assertion.json.invalide »
$code-statut /= « assertion.manquant »
$code-statut /= « assertion.multipleHardBindings »
$code-statut /= « assertion.notRedacted »
$code-statut /= « assertion.outsideManifest »
$code-statut /= « assertion.auto-expurgé »
$code-statut /= « assertion.non-déclaré »
$code-statut /= « revendication.cbor.invalide »
$code-statut /= « revendication.liaisons-rigides.manquantes »
$code-statut /= « claim.malformed »
$code-statut /= « claim.manquant »
$code-statut /= « revendication.multiple »
$code-statut /= « signature-réclamation.manquante »
$code-statut /= « signature-réclamation.incompatibilité »
$code-statut /= « signature-réclamation.hors-validité »
$code-statut /= « general.error » ; lorsque rien d'autre ne s'applique
$code-statut /= « hashedURI.manquant »
$code-statut /= « hashedURI.mismatch »
$code-statut /= « ingrédient.signature-revendication.manquante »
$code-statut /= « ingrédient.signature-revendication.incompatibilité »
$code-statut /= « ingrédient.manifeste.manquant »
$code-statut /= « ingrédient.manifeste.incompatibilité »
$code-statut /= « manifeste.compressé.invalide »
$code-statut /= « manifeste.inaccessible »
$code-statut /= « manifeste.parents-multiples »
$code-statut /= « manifest.timestamp.invalid »
$code-statut /= « manifest.timestamp.wrongParents »
$status-code /= « manifest.update.invalid »
$code-statut /= « manifest.update.wrongParents »
$code-statut /= « signingCredential.invalid »
$code-statut /= « signingCredential.ocsp.revoked »
$code-statut /= « signingCredential.ocsp.unknown »
$code-statut /= « signingCredential.untrusted »

; codes d'état personnalisés
$code-statut /= tstr .regexp « ([\\da-zA-Z_-]+\\.)+[\\da-zA-Z_-]+ »

status-map = {
  « code » : $code-statut,             ; Chaîne au format étiquette décrivant le statut
  ? « url » : jumbf-uri-type,         ; Référence URI JUMBF à la boîte JUMBF à laquelle s'applique ce code
d'état
  ? « explanation » : tstr .size (1..max-tstr-length), ; Chaîne lisible par l'utilisateur expliquant le statut
  ? « success » : bool                 ; OBSOLÈTE. Le code reflète-t-il une réussite (vrai) ou un échec
(faux)
}

status-codes-map = {
    « success » : [* $status-map],              ; tableau des codes de réussite de la validation. Peut
être vide.
    « informationnel » : [* $status-map],       ; un tableau de codes d'information de validation. Peut
être vide.
    « failure » : [* $status-map]               ; un tableau de codes d'échec de validation. Peut être
vide.
}

; Objets contenant les résultats de validation d'un manifeste et de ses ingrédients

validation-results-map = {
    ? « activeManifest » : $status-codes-map, ; Codes d'état de validation pour le manifeste actif de
l'ingrédient. Présent si l'ingrédient est un actif C2PA. Absent si l'ingrédient n'est pas un actif C2PA.
    ? « ingredientDeltas » : [* $ingredient-delta-validation-result-map] ; Liste de toutes les
```

```
changements/deltas entre les résultats de validation actuels et précédents pour le manifeste de chaque
ingrédient. Présent si l'ingrédient est un actif C2PA.
}

ingredient-delta-validation-result-map = {
     « ingredientAssertionURI » : jumbf-uri-type, ; Référence URI JUMBF à l'assertion de l'ingrédient
     « validationDeltas » : $status-codes-map             ; Résultats de validation pour le manifeste actif
de l'ingrédient
}
```

15.2.2. Codes d'état standard

15.2.2.1. Codes de réussite

Tableau 2. Codes de réussite de validation

Valeur	Signification	Utilisation de `l'URL`
`assertion.accessible`	Une assertion non intégrée (à distance) était accessible au moment de la validation.	Assertion C2PA
`assertion.bmffHash.match`	Le hachage d'un actif basé sur une boîte correspond au hachage déclaré dans l'assertion de hachage BMFF.	Assertion C2PA
`assertion.boxesHash.match`	Le hachage d'un actif basé sur une boîte correspond au hachage déclaré dans l'assertion de hachage de boîte générale.	Assertion C2PA
`assertion.collectionHash.matc h`	Les hachages de tous les actifs contenus dans une collection correspondent aux hachages déclarés dans l'assertion de hachage des données de collection.	Assertion C2PA
`assertion.dataHash.match`	Le hachage d'une plage d'octets de l'actif correspond au hachage déclaré dans l'assertion de hachage des données.	Assertion C2PA
`assertion.hashedURI.match`	Le hachage de l'assertion référencée correspond au hachage correspondant dans l'URI haché de l'assertion dans la revendication.	Assertion C2PA
`assertion.multiAssetHash.matc h`	Le hachage d'une partie d'une assertion de hachage multi-actifs correspond au hachage correspondant dans la carte de hachage multi-actifs de l'assertion.	Assertion C2PA
`claimSignature.insideValidity`	La signature de la revendication référencée dans la revendication a été créée pendant la période de validité du certificat de signature.	Boîte de signature de revendication C2PA
`claimSignature.validated`	La signature de la revendication mentionnée dans la revendication a été validée.	Boîte de signature de revendication C2PA
`ingredient.claimSignature.val idated`	Le hachage de la boîte de signature de revendication C2PA de l'ingrédient a été validé avec succès.	Assertion C2PA

Valeur	Signification	`url` Utilisation
`ingredient.manifest.validated`	Le hachage de la case C2PA Manifest de l'ingrédient a été validé avec succès.	Assertion C2PA
`signingCredential.ocsp.notRev oked`	Le certificat de signature n'était pas révoqué au moment de la signature.	Boîte de signature de revendication C2PA
`signingCredential.trusted`	Le certificat de signature est fiable	Boîte de signature de revendication C2PA
`timeStamp.trusted`	Les informations d'identification de l'horodatage figurent sur la liste des ancres de confiance du validateur pour les autorités d'horodatage.	Boîte de signature de revendication C2PA
`timeStamp.validated`	L'horodatage est bien formé, comporte une empreinte de message qui correspond à la signature de la revendication et a été créé pendant la période de validité de l'identifiant d'horodatage.	Boîte de signature de revendication C2PA

15.2.2.2. Codes d'information

Tableau 3. Codes d'information de validation

Valeur	Signification	`url` Utilisation
`algorithm.deprecated`	L'algorithme est obsolète.	Boîte de revendication C2PA ou assertion C2PA
`ingredient.unknownProvenance`	L'ingrédient ne dispose pas d'un manifeste C2PA.	Assertion C2PA
`signingCredential.ocsp.inacce ssible`	Le validateur a tenté d'effectuer une vérification OCSP en ligne, mais n'a pas reçu de réponse.	Boîte de signature de revendication C2PA
`signingCredential.ocsp.skippe d`	Le validateur a choisi de ne pas effectuer de vérification OCSP en ligne.	Boîte de signature de revendication C2PA
`timeOfSigning.insideValidity`	L'heure de signature indiquée (dans l'en-tête `iat` de la signature) se situe dans la période de validité de la chaîne de certificats du signataire et avant l'heure indiquée dans tout horodatage de confiance correspondant.	Boîte de signature C2PA
`timeOfSigning.outsideValidity`	L'heure de signature déclarée (dans l'en-tête `iat` de la signature) se situe en dehors de la période de validité de la chaîne de certificats du signataire ou après l'heure indiquée dans un horodatage fiable correspondant.	Boîte de signature de revendication C2PA
`timeStamp.malformed`	La réponse d'horodatage incluse dans l'en-tête de signature de la revendication n'est pas correctement formée, conformément à la norme RFC 3161.	Boîte de signature de revendication C2PA

Valeur	Signification	`url` Utilisation
`timeStamp.mismatch`	L'horodatage ne correspond pas au contenu de la revendication.	Boîte de signature de revendication C2PA
`timeStamp.outsideValidity`	L'attribut d'horodatage signé dans la signature a été créé en dehors de la période de validité du certificat TSA.	Boîte de signature de revendication C2PA
`timeStamp.untrusted`	Les informations d'identification de l'horodatage ne figurent pas sur les listes de confiance TSA du validateur.	Boîte de signature de revendication C2PA

15.2.2.3. Codes d'échec

Tableau 4. Codes d'échec de validation

Valeur	Signification	Utilisation de `l'URL`
`algorithm.unsupported`	L'algorithme n'est pas spécifié ou n'est pas pris en charge.	Boîte de revendication C2PA ou assertion C2PA
`assertion.action.ingredientMi smatch`	Une action qui nécessite un ingrédient associé n'en possède pas ou celui spécifié est introuvable.	Assertion C2PA
`assertion.action.malformed`	Une assertion d'action est mal formée.	Assertion C2PA
`assertion.action.redacted`	Une assertion d'actions a été expurgée lors de la création de la revendication.	Assertion C2PA
`assertion.action.redactionMis match`	Une action qui nécessite une rédaction associée n'en a pas ou celle qui est spécifiée est introuvable.	Assertion C2PA
`assertion.bmffHash.malformed`	Une assertion de hachage BMFF est mal formée.	Assertion C2PA
`assertion.bmffHash.mismatch`	Le hachage d'un actif basé sur une boîte ne correspond pas au hachage déclaré dans une assertion de hachage BMFF.	Assertion C2PA
`assertion.boxesHash.malformed`	L'assertion de hachage de boîte générale est mal formée.	Assertion C2PA
`assertion.boxesHash.mismatch`	Le hachage d'un format d'actif général de type boîte ne correspond pas au hachage déclaré dans une assertion de hachage de boîte générale.	Assertion C2PA
`assertion.boxesHash.unknownBo x`	Une boîte autre que celles attendues a été trouvée	Assertion C2PA
`assertion.cloud-data.actions`	Un manifeste de mise à jour contient une assertion de données cloud faisant référence à une assertion d'actions.	Assertion C2PA
`assertion.cloud-data.hardBinding`	Une assertion de liaison forte se trouve dans une assertion de données cloud.	Assertion C2PA

Valeur	Signification	Utilisation de `l'URL`
`assertion.cloud-data.malformed`	L'assertion cloud-data était incomplète	Assertion C2PA
`assertion.cbor.invalid`	Le cbor d'une assertion n'est pas valide	Assertion C2PA
`assertion.collectionHash.inco rrectFileCount`	Un élément répertorié dans l'assertion de hachage des données de la collection est manquant dans la collection.	Assertion C2PA
`assertion.collectionHash.inva lidURI`	L'URI d'un élément dans l'assertion de hachage des données de la collection contient la partie de fichier « .. » ou « . ».	Assertion C2PA
`assertion.collectionHash.malf ormed`	L'assertion de hachage de la collection était incomplète.	Assertion C2PA
`assertion.collectionHash.mismatch`	Le hachage d'un actif de la collection ne correspond pas au hachage déclaré dans l'assertion de hachage des données de la collection.	Assertion C2PA
`assertion.dataHash.malformed`	Une assertion de hachage de données est mal formée.	Assertion C2PA
`assertion.dataHash.mismatch`	Le hachage d'une plage d'octets de l'actif ne correspond pas au hachage déclaré dans l'assertion de hachage des données.	Assertion C2PA
`assertion.dataHash.redacted`	Une assertion contraignante a été expurgée lors de la création de la revendication.	Assertion C2PA
`assertion.hashedURI.mismatch`	Le hachage de l'assertion référencée dans le manifeste ne correspond pas au hachage correspondant dans l'URI haché de l'assertion dans la revendication.	Assertion C2PA
`assertion.inaccessible`	Une assertion non intégrée (à distance) était inaccessible au moment de la validation.	Assertion C2PA
`assertion.ingredient.malforme d`	L'assertion relative aux ingrédients était incomplète.	Assertion C2PA
`assertion.json.invalid`	Le JSON(-LD) d'une assertion n'est pas valide	Assertion C2PA
`assertion.missing`	Une assertion répertoriée dans la revendication du manifeste est manquante dans le manifeste de l'actif.	Boîte de revendication C2PA
`assertion.multiAssetHash.malf ormed`	Une assertion de hachage multi-actifs est mal formée.	Assertion C2PA
`assertion.multiAssetHash.miss ingPart`	Une partie requise de l'actif en plusieurs parties est introuvable.	Assertion C2PA
`assertion.multiAssetHash.mismatch`	Le hachage d'une partie d'un actif en plusieurs parties ne correspond pas au hachage déclaré dans l'assertion de hachage multi-actifs.	Assertion C2PA
`assertion.multipleHardBinding s`	Le manifeste contient plusieurs assertions de liaison forte.	Boîte de stockage des assertions C2PA

Valeur	Signification	`url` Utilisation
`assertion.notRedacted`	Une assertion a été déclarée comme expurgée dans la revendication, mais elle est toujours présente dans le manifeste.	Assertion C2PA
`assertion.outsideManifest`	Une assertion répertoriée dans la revendication ne figure pas dans le même manifeste C2PA que la revendication.	Boîte de revendication C2PA
`assertion.selfRedacted`	Une assertion a été déclarée comme ayant été expurgée par sa propre revendication.	Boîte de revendication C2PA
`assertion.timestamp.malformed`	L'assertion d'horodatage est mal formée.	Assertion C2PA
`assertion.undeclared`	Une assertion a été trouvée dans le manifeste qui n'était pas explicitement déclarée dans la revendication.	Affirmation C2PA
`claim.cbor.invalid`	Le cbor de la revendication n'est pas valide.	Boîte de revendication C2PA
`claim.hardBindings.missing`	Aucune liaison forte n'est présente.	Boîte de revendication C2PA
`claim.malformed`	Les données/champs de la revendication référencée dans le manifeste ne sont pas corrects.	Boîte de revendication C2PA
`claim.missing`	La revendication référencée dans le manifeste est introuvable.	Boîte de revendication C2PA
`claim.multiple`	Plusieurs boîtes de réclamation sont présentes dans le manifeste.	Boîte de revendication C2PA
`signatureDeRéclamationManquante`	La signature de la revendication référencée dans la revendication est introuvable dans son manifeste.	Boîte de signature de revendication C2PA
`claimSignature.mismatch`	La signature de la réclamation mentionnée dans la réclamation n'a pas pu être validée.	Boîte de signature de revendication C2PA
`claimSignature.outsideValidit y`	La signature de la réclamation mentionnée dans la réclamation a été créée en dehors de la période de validité du certificat de signature.	Boîte de signature de revendication C2PA
`general.error`	Valeur à utiliser lorsqu'une erreur non spécifiquement répertoriée ici s'est produite.	Boîte de revendication C2PA ou assertion C2PA
`hashedURI.missing`	Les données indiquées par un `hashed_uri` sont introuvables	Assertion C2PA
`hashedURI.mismatch`	Le hachage d'un `hashed_uri` donné ne correspond pas au hachage correspondant des données de l'URI de destination	Assertion C2PA
`ingredient.claimSignature.mis sing`	La signature de revendication C2PA de l'ingrédient référencé n'a pas été trouvée.	Assertion C2PA

Valeur	Signification	`url` Utilisation
`ingredient.claimSignature.mis match`	Le hachage de la signature de revendication C2PA d'un manifeste C2PA intégré ne correspond pas au hachage déclaré dans la valeur `hashed_uri` du champ `claimSignature` dans l'assertion sur les ingrédients.	Assertion C2PA
`ingredient.manifest.missing`	Le manifeste C2PA de l'ingrédient référencé est introuvable.	Assertion C2PA
`ingredient.manifest.mismatch`	Le hachage d'un manifeste C2PA intégré ne correspond pas au hachage déclaré dans la valeur `hashed_uri` du champ `activeManifest` dans l'assertion d'ingrédient.	Assertion C2PA
`manifeste.compressé.invalide`	Le manifeste compressé n'était pas valide.	Boîte de revendication C2PA
`manifest.inaccessible`	Un manifeste non intégré (à distance) était inaccessible au moment de la validation.	Boîte de revendication C2PA
`manifest.multipleParents`	Le manifeste contient plusieurs ingrédients dont `la relation` est `parentOf`.	Boîte de revendication C2PA
`manifest.timestamp.invalid`	Le manifeste est un manifeste horodaté, mais il contient une assertion non autorisée (non ingrédient).	Boîte de revendication C2PA
`manifest.timestamp.wrongParen ts`	Le manifeste est un manifeste horodaté, mais il contient soit zéro, soit plusieurs ingrédients `parentOf`.	Boîte de revendication C2PA
`manifest.update.invalid`	Le manifeste est un manifeste de mise à jour, mais il contient une assertion non autorisée, telle qu'une assertion de liaison forte ou d'actions.	Boîte de revendication C2PA
`manifest.update.wrongParents`	Le manifeste est un manifeste de mise à jour, mais il contient soit zéro, soit plusieurs éléments `parentOf`.	Boîte de revendication C2PA
`signingCredential.invalid`	Les informations d'identification de signature ne sont pas valides pour la signature.	Boîte de signature de revendication C2PA
`signingCredential.ocsp.revoke d`	La réponse OCSP indique que l'identifiant de signature a été révoqué par l'émetteur.	Boîte de signature de revendication C2PA
`signingCredential.ocsp.unknow n`	La réponse OCSP contient un statut `inconnu` pour le certificat de signature.	Boîte de signature C2PA
`signingCredential.untrusted`	Les informations d'identification de signature ne figurent sur aucune des listes de confiance applicables du validateur.	Boîte de signature de revendication C2PA

15.3. Affichage des informations du manifeste

Les consommateurs de manifestes ne doivent pas afficher les données provenant de manifestes qui ne sont pas valides ni d'actifs qui ne sont pas valides. Si le consommateur de manifestes choisit d'afficher ces données, il doit inclure dans l'affichage :

- un avertissement concernant le manque de validité,
- un avertissement indiquant que les données ne doivent pas être attribuées au signataire du manifeste et, dans le cas d'un manifeste d'ingrédients, un avertissement supplémentaire indiquant qu'elles ne doivent pas être attribuées au signataire du manifeste de l'actif.

REMARQUE Dans les scénarios de création, il est souhaitable de mettre davantage en évidence les avertissements afin que le créateur puisse prendre une décision éclairée sur la manière de procéder avec un actif qui n'est pas valide ou dont l'historique de provenance est erroné .

15.4. Détermination de l'algorithme de hachage

15.4.1. Pour les URI hachés

Diverses parties du manifeste C2PA utilisent une structure `hashed_uri` pour encapsuler une URI, son hachage et (facultativement) l'algorithme utilisé pour calculer le hachage. Si la structure `hashed_uri` contient un champ `alg`, celui-ci doit être utilisé comme algorithme de hachage. Si le champ `alg` n'est pas présent dans la structure `hashed_uri`, l'algorithme de hachage doit être déterminé en évaluant la structure englobante la plus proche qui contient un champ `alg`. Si aucun champ `alg` n'est trouvé à aucun de ces emplacements, la valeur du champ `alg` dans la revendication doit être utilisée comme algorithme de hachage. Si aucun champ `alg` n'est présent à aucun de ces emplacements, la revendication doit être rejetée avec un code d'échec `algorithm.unsupported.`

15.4.2. Pour les URI ext hachées

Certaines parties d'un manifeste C2PA utilisent une structure `hashed_ext_uri` pour encapsuler une URI externe, son hachage et l'algorithme utilisé pour calculer le hachage. S'il existe un champ `alg` dans la structure `hashed_ext_uri`, il doit être utilisé comme algorithme de hachage. Si le champ `alg` n'est pas présent dans la structure `hashed_ext_uri`, le code d'échec `algorithm.unsupported` doit être utilisé.

REMARQUE Le champ `alg` est obligatoire dans `hashed_ext_uri`, aucune procédure récursive n'est donc nécessaire pour déterminer l'algorithme de hachage.

15.4.3. Validation de l'algorithme

Une fois l'algorithme de hachage déterminé, il doit être comparé aux valeurs figurant dans la liste autorisée ou la liste obsolète de la section 13.1, « Hachage ». S'il ne figure dans aucune des deux listes, la revendication doit être rejetée avec un code d'échec `algorithm.unsupported`. Si l'algorithme figure dans la liste obsolète, la revendication doit recevoir un code d'information `algorithm.deprecated`.

15.5. Localisation du manifeste actif

15.5.1. Généralités

La dernière superbox C2PA Manifest dans la superbox C2PA Manifest Store doit être considérée comme le manifeste actif, mais la localisation du C2PA Manifest Store peut nécessiter de rechercher dans plusieurs emplacements possibles.

15.5.2. Intégré

15.5.2.1. Général

Le magasin de manifestes C2PA doit être localisé par le validateur intégré à l'actif aux emplacements standard pour l'intégration des manifestes. Cependant, si un actif a été récupéré via une connexion HTTP, un validateur peut rechercher un en-tête `Link`, comme décrit dans la clause En-tête Link ci-dessous, afin de déterminer si un magasin de manifestes C2PA est présent.

NOTE La vérification de l'en-tête `Link`, s'il est présent, permet à un validateur de déterminer si un magasin de manifestes C2PA est présente sans avoir à télécharger l'intégralité de la ressource. Cela est utile pour les ressources volumineuses ou diffusées en continu.

Si plusieurs magasins de manifestes C2PA sont présents dans un actif, ils doivent tous être considérés comme non valides et la validation doit traiter cela comme si aucun manifeste n'avait été trouvé. Dans le cas où cet actif est ajouté en tant qu'ingrédient, aucun de ces manifestes C2PA intégrés ne doit être inclus dans l'assertion d'ingrédient.

15.5.2.2. Considérations particulières pour les fichiers PDF

Les fichiers PDF prennent en charge une technologie appelée « mise à jour incrémentielle », dans laquelle les informations sont ajoutées à la fin du document au lieu de modifier l'original. Cela nécessite que les fichiers PDF prennent en charge plusieurs magasins de manifestes C2PA, bien qu'il ne doive y en avoir qu'un seul par section de mise à jour.

Si plusieurs magasins de manifestes C2PA sont présents dans une même section de mise à jour, ils doivent tous être considérés comme non valides et la validation doit les traiter comme s'il n'y avait aucun manifeste. Cependant, tout magasin de manifestes C2PA présent dans les premières mises à jour du PDF ou du PDF original doit toujours être considéré comme valide et traité en conséquence.

15.5.3. Par référence ou URI

15.5.3.1. Par référence

S'il n'y a pas de magasin de manifestes C2PA intégré, les tentatives suivantes doivent être effectuées pour en localiser un à distance.

- Si l'élément a été récupéré via une connexion HTTP, la clause Link Header ci-dessous décrit comment trouver un manifeste via l'en-tête `Link`.
- Si l'actif contient des données XMP dans les emplacements standard (c'est-à-dire en dehors du manifeste C2PA) et que ces données XMP contiennent une clé
  clé `dcterms:provenance`, l'URI fourni doit être utilisé pour localiser le manifeste actif.

- Si l'élément est une police avec une table `C2PA` et que sa `valeur activeManifestUriLength` est différente de zéro, l'URI indiqué doit être utilisé pour localiser le manifeste actif.
- Si aucun magasin de manifestes C2PA n'a été localisé, le validateur doit rechercher les fichiers au même chemin d'accès ou à la même URI, mais avec une extension de nom de fichier `.c2pa`. Si le magasin de manifestes C2PA n'est pas trouvé, un validateur peut rechercher dans tout autre emplacement qu'il juge le plus approprié pour le localiser. Par exemple, un sous-dossier d'un système de fichiers.

REMARQUE	Un validateur n'est pas limité aux emplacements ci-dessus, il peut également choisir de rechercher dans d'autres emplacements.

Si un manifeste est censé exister à un emplacement distant, mais qu'il n'y est pas présent ou que l'emplacement n'est pas disponible (par exemple, dans un scénario hors ligne), le code d'erreur `manifest.inaccessible` doit être utilisé pour signaler la situation.

Les informations relatives au type de média IANA pour un magasin de manifestes C2PA sont disponibles dans la section consacrée aux manifestes externes.

15.5.3.2. Par en-tête `Link`

Si la ressource a été récupérée via une connexion HTTP, le validateur doit rechercher dans l'en-tête de la réponse HTTP un en-tête `Link`, tel que défini dans la RFC 8288, contenant un paramètre `rel=c2pa-manifest`. S'il est présent, un magasin de manifestes C2PA peut être récupéré à partir de cette référence URI. L'URI sera une URI `http` ou `https` standard, telle que https://c2pa.org/image.c2pa.

Il est également possible d'utiliser la relation de lien pour faire référence au magasin de manifestes C2PA intégré à une ressource à l'aide d'un fragment d'URI JUMBF. L'URI inclurait un fragment d'URI JUMBF, vers la superbox du magasin de manifestes C2PA https://c2pa.org/image.jpg#jumbf=c2pa. Les références à des manifestes C2PA spécifiques dans le magasin de manifestes C2PA ne sont pas autorisées et le validateur doit ignorer toute partie `childlabel` du fragment d'URI JUMBF.

REMARQUE	HTTP fait référence au *protocole de transfert hypertexte* défini dans la RFC 7230, et non au schéma URL spécifique `http://`.

15.5.4. Décompression

Comme décrit précédemment, les manifestes standard et de mise à jour peuvent être compressés. Lorsqu'un manifeste compressé est rencontré, le validateur doit le décompresser avant de poursuivre le processus de validation standard. Si les données contenues dans la `brob` box d'un manifeste compressé ne sont pas un manifeste standard ou de mise à jour, ou si la décompression échoue, le validateur doit rejeter le manifeste avec un code d'échec `manifest.compressed.invalid`.

15.5.5. Validation d'une correspondance

Un validateur peut souhaiter vérifier que le magasin de manifestes C2PA localisé est bien celui associé à l'actif.

Si le magasin de manifestes C2PA a été localisé, l'assertion de liaison forte présente dans son manifeste actif doit être utilisée pour vérifier qu'il s'agit bien du manifeste correspondant et que l'actif n'a pas été modifié sans mise à jour du manifeste. Si la liaison forte ne correspond pas, il est impossible de savoir si cela est dû (a) à une modification de l'actif ou (b) à une erreur dans le magasin de manifestes C2PA

Manifest Store incorrect a été localisé. En conséquence, le validateur doit traiter cela comme une liaison forte non correspondante et rejeter le manifeste avec un code d'échec `assertion.dataHash.mismatch` si une assertion de hachage de données est utilisée, `assertion.boxesHash.mismatch` si une assertion de hachage de boîte générale est utilisée, `assertion.collectionHash.mismatch` si une assertion de hachage de données de collection est utilisée, ou `assertion.bmffHash.mismatch` si une assertion de hachage BMFF est utilisée.

15.6. Localisation et validation de la revendication

15.6.1. Localisation

Une fois que le manifeste à valider a été localisé (ci-après dénommé « manifeste actuel »), la revendication est trouvée en localisant, dans le manifeste actuel, la superbox JUMBF avec une étiquette `c2pa.claim.v2` (ou `c2pa.claim` pour les fichiers avec des structures de revendication plus anciennes) et un UUID de type JUMBF de `6332636C-0011-0010-8000-00AA00389B71` (`c2cl`). Notez que l'UUID de type JUMBF est le même pour les nouveaux formats de revendication (avec l'étiquette `c2pa.claim.v2`) et les anciens formats (avec l'étiquette `c2pa.claim`). Il ne doit y avoir qu'une seule case de ce type dans le manifeste actuel. Si plusieurs cases sont détectées, le manifeste C2PA sera rejeté avec un code d'échec `claim.multiple.`

15.6.2. Validation

Si le contenu de la revendication n'est pas correctement formé en CBOR, la revendication sera rejetée avec un code d'échec de `claim.cbor.invalid`.

REMARQUE Le format CBOR correct est défini dans la RFC 8949, annexe C.

Pour un « c2pa.claim.v2 », les champs suivants doivent être présents dans l'objet CBOR. Si l'un d'entre eux est absent, la revendication sera rejetée avec un code d'échec `claim.malformed`.

- `instanceID`
- `signature`
- `created_assertions`
- `claim_generator_info`

Si le champ `claim_generator_info` ne contient pas de champ `name`, la demande sera rejetée avec un code d'échec `claim.malformed`.

S'il existe un champ `icône` dans le `générateur-info-map` référencé par le champ `claim_generator_info` du `claim-map` ou du `claim-map-v2`, alors sa valeur doit être validée comme décrit dans la section 15.10.3.3, « Validation des références ».

15.7. Valider la signature

Récupérer la référence URI pour la signature à partir de la valeur du champ `signature` de la revendication et résoudre l'URI.

référence pour obtenir la signature COSE. La signature doit être intégrée dans le même manifeste que celui décrit à la section 11.1.4, « Détails de la boîte C2PA ». Si l'URI de la signature ne renvoie pas à un emplacement dans la même boîte du manifeste C2PA (un emplacement `self#jumbf`), la revendication doit être rejetée. Si ce champ n'est pas présent ou si l'URI ne peut être résolu, la revendication doit être rejetée avec un code d'échec `claimSignature.missing`.

Si la signature et la revendication ne sont pas contenues dans le même manifeste C2PA, ce manifeste C2PA ne sera pas considéré comme valide.

Pour tous les types de manifestes C2PA, la validation du certificat utilisé dans la signature doit être effectuée conformément au chapitre 14, *Modèle de confiance*.

Si le certificat n'est pas acceptable selon les exigences du type de certificat, la demande doit être rejetée avec un code d'échec `signingCredential.invalid`. Si l'algorithme de signature ne figure pas dans la liste des algorithmes autorisés ou obsolètes de la section 13.2, « Signatures numériques », la demande doit être rejetée avec un code d'échec `algorithm.unsupported`.

Il est alors nécessaire de vérifier la chaîne de confiance entre le certificat et une entrée dans l'une des listes d'ancrages de confiance applicables. Si cette chaîne de confiance ne peut être vérifiée, la revendication doit être rejetée avec un code d'échec `signingCredential.untrusted` ; sinon, la signature de la revendication doit se voir attribuer un code de réussite `signingCredential.trusted`.

Si la revendication n'a pas encore été rejetée, la validation se poursuit conformément à la procédure spécifiée à la section 13.2, « Signatures numériques ». Si la validation de la signature échoue, la revendication est rejetée avec un code d'échec `claimSignature.mismatch`. Sinon, la signature de la revendication se voit attribuer un code de réussite `claimSignature.validated`.

Dans la suite de ce chapitre, les en-têtes font référence à l'union de l'ensemble des paramètres d'en-tête protégés et non protégés dans la signature COSE. Sauf indication contraire dans la section 13.2, « Signatures numériques » ou la section 14.5, « Certificats X.509 », un en-tête peut apparaître dans l'un ou l'autre des groupes. Les en-têtes COSE sont décrits dans la RFC 8152, section 3.

15.8. Valider l'horodatage

15.8.1. Obtention du TimeStampToken

15.8.1.1. Intégré dans la signature de la revendication

Si l'en-tête `sigTst` ou `sigTst2` est présent, le tableau `tstTokens` doit contenir un seul `tstToken`. Si l'en-tête contient plusieurs `tstToken`, le validateur doit émettre un code d'information `timestamp.malformed` et ignorer les horodatages.

Un validateur prenant en charge `sigTst` doit effectuer les procédures suivantes pour valider la réponse d'horodatage :

- Récupérer la propriété `val` du `tstToken`, qui doit être une réponse d'horodatage (`TimeStampResp`) conforme à la norme RFC3161.

- Vérifier la valeur du champ `d'état PKIStatusInfo`, qui est la valeur du champ `d'état` de `TimeStampResp`.
    - Si elle contient une valeur autre que `0` (`accordé`) ou `1` (`accordé avec modifications`), le validateur doit émettre un code d'information `timeStamp.malformed` et ignorez cet horodatage.
    - S'il est égal à `0` (`accordé`) ou `1` (`accordé avec modifications`), poursuivez le reste du processus de validation de l'horodatage comme décrit ci-dessous.
- Récupérez la valeur du champ `timeStampToken` du `TimeStampResp` pour l'utiliser dans le reste du processus de validation.

Un validateur pour `sigTst2` doit récupérer la propriété `val` du `tstToken`, qui doit être un `timeStampToken` (TimeStampToken, TST) conforme à la norme RFC3161.

15.8.1.2. Référencé par une assertion d'horodatage

Si un validateur a déjà localisé un TimeStampToken dans un en-tête `sigTst` ou `sigTst2`, qui passe la validation (conformément à la section 15.8.2, « Validation du TimeStampToken »), il doit alors ignorer cette étape. Lorsqu'aucun en-tête de ce type n'existe ou que le TimeStampToken qui s'y trouve n'a pas passé la validation, cette étape doit être suivie.

Si un validateur a précédemment localisé des assertions d'horodatage, qui ont ensuite été conservées dans un mappage des identifiants du manifeste C2PA vers les TimeStampTokens, le validateur doit vérifier si l'identifiant du manifeste C2PA actuel est présent dans le mappage. Si c'est le cas, le validateur doit utiliser le TimeStampToken associé à l'identifiant dans le mappage pour le TimeStampToken dans le processus de validation décrit à la section 15.8.2, « Validation du TimeStampToken ». Si plusieurs TimeStampToken pour cet identifiant sont trouvés dans le mappage, le validateur doit les essayer tous jusqu'à ce que l'un d'entre eux passe la validation avec succès (et doit ensuite ignorer les autres). Si l'identifiant n'apparaît pas dans le mappage, aucune erreur n'est générée, car cela signifie simplement qu'il n'y a pas de TimeStampToken associé à ce manifeste C2PA dans le contexte actuel.

15.8.2. Validation du TimeStampToken

Tous les validateurs doivent poursuivre le processus comme suit :

- Si l'algorithme de signature dans le `timeStampToken` ne figure pas dans la liste des algorithmes autorisés ou obsolètes de la section 13.2, « Signatures numériques », le validateur doit émettre un code d'information `timestamp.untrusted` et ignorer l'horodatage.
- Valider la signature dans le `timeStampToken`, comme décrit dans la RFC 2630, section 5.6. Si la signature n'est pas valide, le validateur doit émettre un code d'information `timestamp.mismatch` et ignorer l'horodatage.
- Si le `timeStampToken` ne contient pas un champ `messageImprint`, le validateur doit émettre un `timestamp.malformed` et ignorer l'horodatage.
- Si l'algorithme de hachage de l'empreinte du message ne figure pas dans la liste des algorithmes autorisés ou obsolètes de la section 13.1, « Hachage », le validateur doit émettre un code d'information `timestamp.untrusted` et ignorer l'horodatage.
- Vérifier que la valeur du champ `messageImprint` (dans le `timeStampToken`) correspond à la revendication (v1,

`sigTst`) ou le champ `de signature` de la structure `COSE_Sign1_Tagged` (v2, `sigTst2`) du manifeste C2PA en cours de validation, comme décrit à la section 10.3.2.5.2, « Choix de la charge utile ». Si les valeurs ne correspondent pas, le validateur doit émettre un code d'information `timestamp.mismatch` et ignorer l'horodatage.

- Vérifier que le champ `des certificats` du `timeStampToken` est présent, que le certificat du TSA se trouve dans l'ensemble de certificats fourni dans ce champ et qu'il est possible de construire une chaîne de confiance entre le certificat du TSA et une entrée dans la liste de confiance TSA C2PA (ou toute autre liste d'ancres de confiance présente dans le validateur à cette fin). Si le certificat ne peut être localisé ou si une chaîne de confiance ne peut être établie, le validateur doit émettre un code d'information `timestamp.untrusted` et ignorer l'horodatage.
- Vérifiez que l'heure attestée, telle qu'elle figure dans le champ `genTime` (dans le `timeStampToken`), se situe dans la période de validité du certificat de signature de la TSA et de tous les certificats CA jusqu'à l'ancrage de confiance. Si ce n'est pas le cas, le validateur doit émettre un code d'information `timestamp.outsideValidity` et ignorer l'horodatage.
- Si la validation de l'horodatage ne s'arrête pas ou n'échoue pas en raison de l'une des conditions ci-dessus, le validateur doit émettre les codes de réussite `timeStamp.trusted` et `timeStamp.validated`.
- Si le validateur a émis les deux codes de réussite `timeStamp.trusted` et `timeStamp.validated`, il doit alors vérifier que l'heure attestée par l'autorité d'horodatage (TSA), telle qu'elle figure dans le champ `genTime` (dans le `timeStampToken`), se situe dans la période de validité du certificat de signature de la revendication et de tous les certificats CA jusqu'à l'ancrage de confiance. Si ce n'est pas le cas, le validateur doit rejeter la revendication avec un code d'échec `claimSignature.outsideValidity`.

REMARQUE Les horodatages restent valides même après l'expiration des informations d'identification de signature de l'autorité d'horodatage, donc tant que l'heure attestée se situe dans la période de validité du certificat de l'autorité de timbre horodateur. Il s'agit d'un type particulier de confiance accordé uniquement aux autorités de timbre horodateur.

Au moment de la validation, lorsqu'un horodatage est présent, fiable et validé, les validateurs doivent utiliser l'heure attestée, et non l'heure actuelle, pour déterminer la validité temporelle du certificat de signature et du certificat de l'autorité d'horodatage.

REMARQUE Ce document n'exige pas que le statut de révocation du certificat d'une autorité de horodatage soit enregistré au moment de la signature ni validé au moment de la validation.

Si les en-têtes `sigTst` et `sigTst2` ne sont pas présents, ou si au moins l'un d'entre eux est présent mais que leur jeton d'horodatage ne satisfait pas aux exigences ci-dessus, le manifeste C2PA est valide si l'heure actuelle au moment de la validation se situe dans la période de validité du certificat du signataire et de tous les certificats CA jusqu'à l'ancrage de confiance. Si c'est le cas, le validateur renvoie un code de réussite `claimSignature.insideValidity`. Si ce n'est pas le cas, le manifeste C2PA est rejeté avec un code d'échec `claimSignature.outsideValidity`.

15.8.3. Validation de « l'heure de signature déclarée »

Un validateur peut choisir de valider « l'heure de signature déclarée » telle qu'attestée par la valeur présente dans l'en-tête protégé `iat`. Si l'en-tête `iat` est présent, le validateur peut vérifier que l'heure attestée se situe dans la période de validité du certificat du signataire et de tous les certificats CA jusqu'à l'ancrage de confiance, et qu'elle n'est pas postérieure à l'heure attestée par tout horodatage de confiance associé. Si le validateur procède à la validation de cette valeur et que celle-ci se situe dans la période de validité,

le validateur renvoie le code d'information `timeOfSigning.insideValidity`, mais si elle se situe en dehors de la période de validité, le validateur renvoie le code d'information `timeOfSigning.outsideValidity`.

15.9. Valider les informations de révocation des informations d'identification

Le validateur doit tenter de déterminer le statut de révocation du certificat du signataire et de tous les certificats CA qui font partie de la chaîne de confiance.

Pour les certificats CA, le validateur doit déterminer le statut de révocation tel qu'indiqué dans l'extension Authority Information Access (AIA) décrite dans la section 4.2.2.1 de la RFC 5280. Le validateur doit utiliser les réponses OCSP pertinentes incluses dans le manifeste C2PA si l'extension AIA indique que l'OCSP est disponible.

Si le validateur détermine qu'un certificat CA a été révoqué à l'heure indiquée dans un horodatage fiable, ou à l'heure actuelle si aucun horodatage fiable n'est présent, la signature de la revendication doit être rejetée avec un statut d'échec `signingCredential.untrusted`.

Pour le certificat du signataire, le validateur doit utiliser le processus suivant.

- Si un certificat ne prend pas en charge le statut de révocation ou si l'émetteur du certificat n'a pas fourni de méthode pour interroger son statut de révocation, le validateur doit traiter le certificat comme non révoqué.
- Si le générateur de revendications a « agrafé » les réponses OCSP dans l'en-tête `rVals` de la structure `COSE_Sign1`, le validateur doit décoder et valider les réponses OCSP agrafées comme décrit dans la section 15.9.1, « Détermination de la révocation à l'aide des réponses OCSP dans le magasin de manifestes C2PA ».
- Si des générateurs de revendications ultérieurs ont ajouté des assertions d'état de certificat dans d'autres manifestes C2PA dans le magasin de manifestes C2PA, le validateur doit utiliser ces réponses OCSP dans le processus de validation décrit à la section 15.9.1, « Détermination de la révocation à l'aide des réponses OCSP dans le magasin de manifestes C2PA ». Si plusieurs réponses OCSP sont trouvées pour le certificat, le validateur doit les essayer toutes jusqu'à ce que l'une d'entre elles passe la validation avec succès (et ignorer ensuite les autres).

Si aucune information de révocation n'a été trouvée dans le magasin de manifestes C2PA, que le validateur est en ligne et qu'il souhaite vérifier le statut de révocation du certificat, il doit alors tenter de déterminer le statut de révocation du certificat en interrogeant le répondeur OCSP comme décrit dans la section 15.9.2, « Détermination de la révocation à partir de la réponse OCSP en ligne ».

15.9.1. Détermination de la révocation à l'aide des réponses OCSP dans le magasin de manifestes C2PA

Un validateur doit décoder les réponses OCSP conformément aux exigences de la RFC 6960, en particulier les exigences 1 à 4 de la section 3.2. Si une réponse OCSP est acceptée et si toutes les exigences suivantes sont remplies, cela établit que le certificat concerné n'était pas révoqué au moment de la signature.

- La signature de la revendication comporte une heure certifiée fournie par un horodatage signé valide.
- Il existe une `réponse unique` dans le tableau `des réponses` du champ `tbsResponseData` de la réponse OCSP, de sorte que toutes les conditions suivantes sont remplies :

- L'heure actuelle n'est pas antérieure à `thisUpdate`.
- L'heure attestée à partir de l'horodatage :
  - est antérieure à `thisUpdate`, ou
  - se situe dans l'intervalle `(thisUpdate,nextUpdate)`, si `nextUpdate` est présent, ou
  - se situe dans l'intervalle `(thisUpdate, producedAt + 24 heures)` où `producedAt` est le champ dans le `ResponseData` contenant, si `nextUpdate` n'est pas présent.
- Le champ `certStatus` de `SingleResponse` est `valide`, ou `révoqué` mais avec un `revocationReason` de `removeFromCRL`.

NOTE La valeur `removeFromCRL` est unique parmi les valeurs de `revocationReason,` car elle équivaut à une réponse `positive`. Bien qu'il s'agisse d'un type de réponse `révoquée`, cette réponse indique que le certificat avait été temporairement « suspendu » (motif `certificateHold`) en raison d'un problème d'intégrité, mais que ce problème a été résolu et que l'émetteur déclare que le certificat reste fiable (voir RFC 5280).

- Le signataire OCSP de la réponse est un « répondant autorisé » tel que défini par la RFC 6960, section 4.2.2.2.

Les validateurs doivent vérifier le `motif de révocation (revocationReason)` de toute réponse `révoquée` afin de distinguer le cas « `removedFromCRL` »
d'une révocation effective.

Si les conditions ci-dessus sont remplies pour toute réponse OCSP dans le magasin de manifestes C2PA, le certificat doit être considéré comme non révoqué au moment de la signature, et le validateur doit émettre un code de réussite `signingCredential.ocsp.notRevoked`.

Sinon, si une réponse OCSP dans le magasin de manifestes C2PA remplit toutes les conditions ci-dessus, sauf que le champ `certStatus` est `révoqué`, le certificat doit être considéré comme révoqué au moment de la signature et la revendication doit être rejetée avec un code d'échec `signingCredential.ocsp.revoked`.

15.9.2. Détermination de la révocation à partir d'une réponse OCSP en ligne

Si, pour un certificat donné, aucune réponse OCSP dans le magasin de manifestes C2PA ne satisfait aux conditions de la section 15.9.1, « Détermination de la révocation à partir des réponses OCSP dans le magasin de manifestes C2PA », ou si la signature de la revendication ne comporte pas d'horodatage, le validateur peut choisir d'interroger le répondeur OCSP, conformément à la RFC 6960, avec `l'emplacement d'accès` du répondeur trouvé via la RFC 6960, section 3.1.

REMARQUE L'interrogation de la méthode de statut des informations d'identification peut révéler à un observateur l'identité de l'actif en cours de validation. Cette interrogation est donc facultative.

Si le validateur choisit de ne pas effectuer une OCSP en ligne, il doit émettre un
`signingCredential.ocsp.skipped`.

Si le validateur tente d'interroger le répondeur OCSP mais ne parvient pas à recevoir de réponse, il doit émettre un `signingCredential.ocsp.inaccessible`.

Si une réponse est reçue et acceptée conformément aux exigences 1 à 4 de la RFC 6960, section 3.2, elle doit établir que le certificat du signataire n'a pas été révoqué au moment de la signature si l'une des conditions suivantes est remplie :

- Le claim signature a une horodatage horodatage, et le attesté horodatage tombe dans l'
  intervalle `(thisUpdate,nextUpdate)` de la réponse, ou
- La signature de la revendication ne comporte pas d'horodatage valide, mais l'heure réelle actuelle se situe dans l'
  intervalle `(cetteMiseÀjour,prochaineMiseÀjour)` de la réponse,

Et les deux conditions suivantes sont remplies :

- Le champ `certStatus` de la réponse est `correct`, ou `révoqué` mais avec un `revocationReason` de `removeFromCRL`, et
- Le signataire OCSP de la réponse est un « répondant autorisé » tel que défini par la RFC 6960, section 4.2.2.2.

Si le champ `certStatus` de la réponse est `révoqué` mais avec un `revocationReason` qui n'est pas `removeFromCRL`, il doit établir que le certificat du signataire n'était pas révoqué au moment de la signature si les deux conditions suivantes sont remplies :

- Le manifeste comporte un horodatage valide et l'heure attestée se situe dans l'intervalle `(thisUpdate,nextUpdate)` de la réponse, et
- Le `revocationTime` dans la réponse est postérieur à l'horodatage attesté.

Si les conditions ci-dessus sont remplies, le certificat est considéré comme non révoqué au moment de la signature, et le validateur émet un code de réussite `signingCredential.ocsp.notRevoked`.

Sinon :

- Si le champ `certStatus` de l' de la réponse est `inconnu`, la revendication doit être rejetée avec un `signingCredential.ocsp.unknown`.
- Sinon, le certificat doit être considéré comme révoqué au moment de la signature et la revendication doit être rejetée avec un `signingCredential.ocsp.revoked`.

15.10. Valider les assertions

15.10.1. Valider les assertions correctes pour le type de manifeste

15.10.1.1. Général

Selon le type de manifeste, certaines assertions sont obligatoires ou interdites. Un validateur doit vérifier les assertions obligatoires et non autorisées.

15.10.1.2. Assertions standard du manifeste

S'il s'agit d'un manifeste standard :

1. Vérifiez qu'il existe exactement une assertion de liaison forte au contenu : soit une `c2pa.hash.data`, une `c2pa.hash.boxes`, une `c2pa.hash.collection.data`, une `c2pa.hash.bmff.v2` (obsolète) ou une `c2pa.hash.bmff.v3`. Si aucune assertion de ce type n'est présente, le manifeste doit être rejeté avec un code d'échec `claim.hardBindings.missing`. S'il existe plusieurs assertions de ce type, le manifeste doit être rejeté avec u n code d'échec `assertion.multipleHardBindings`.
2. Vérifiez qu'il n'y a aucune ou une seule assertion `c2pa.ingredient` dont `la relation` est `parentOf`. S'il y en a plusieurs, le manifeste sera rejeté avec un code d'échec `manifest.multipleParents`.
3. Vérifiez qu'une action `c2pa.created` ou `c2pa.opened` est contenue dans exactement une assertion actions.

15.10.1.3. Mettre à jour les assertions du manifeste

S'il s'agit d'un manifeste de mise à jour :

1. Vérifiez qu'une seule assertion d'ingrédient est présente et que sa `relation` est `parentOf`. Si ce n'est pas le cas (c'est-à-dire s'il en manque une, s'il y en a plusieurs ou si la valeur de `la relation` n'est pas `parentOf`), le manifeste doit être rejeté avec un code d'échec `manifest.update.wrongParents`.
2. Vérifiez qu'il n'y a pas d'assertions `c2pa.hash.data`, `c2pa.hash.boxes`, `c2pa.hash.collection.data`, `c2pa.hash.bmff.v2` (obsolète), `c2pa.hash.bmff.v3` ou thumbnail. Si c'est le cas, le manifeste doit être rejeté avec un code d'échec `manifest.update.invalid`.
3. Vérifiez qu'il n'y a pas d'assertions `c2pa.hash.multi-asset`. Si c'est le cas, le manifeste doit être rejeté avec un code d'échec `manifest.update.invalid`.
4. S'il existe une ou plusieurs assertions `c2pa.actions` ou `c2pa.actions.v2`, vérifiez que le champ `action` de chaque action trouvée dans le tableau `actions` de toute assertion de ce type est l'une des valeurs prises en charge spécifiées dans Mettre à jour les manifestes. Si ce n'est pas le cas, le manifeste doit être rejeté avec un code d'échec `manifest.update.invalid`.

15.10.2. Préparation de la liste des assertions expurgées

Lors du traitement d'une déclaration, un validateur doit rassembler l'ensemble des assertions expurgées pour le manifeste de chaque ingrédient (le cas échéant) en fonction de chaque URI JUMBF répertorié dans son champ `redacted_assertions`. Le champ `redacted_assertions` d'une déclaration ne doit jamais inclure d'URI JUMBF renvoyant à l'une de ses propres assertions.

REMARQUE	Les affirmations peuvent être supprimées des ressources ingrédients à tout moment dans la provenance de la ressource finale. et pas nécessairement par le générateur de revendications qui utilise en premier lieu une ressource d'ingrédient comme ingrédient.

Pour plus de détails, reportez-vous à la section 15.11.3.2, « Effectuer une validation explicite ».

15.10.3. Validation des assertions

15.10.3.1. Général

Chaque assertion dans les champs `created_assertions` et `gathered_assertions` de la revendication (et dans le champ `assertions` d'une revendication v1) est une structure `hashed_uri`. Pour chaque assertion, le validateur doit d'abord déterminer si la référence URI dans le champ `url` figure dans la liste des assertions expurgées.

NOTE Même si les assertions répertoriées dans le champ `gathered_assertions` n'ont pas été créées par le générateur de revendications, elles font toujours partie de la revendication et sont donc également validées selon cet algorithme de validation.

Si elle figure dans la liste des assertions expurgées, alors si le libellé de l'assertion est `c2pa.actions` ou `c2pa.actions.v2`, la revendication doit être rejetée avec un code d'échec `assertion.action.redacted`, car les assertions `c2pa.actions` et `c2pa.actions.v2` ne doivent pas être expurgées. Si elle figure dans la liste des assertions expurgées, alors si le libellé de l'assertion est un lien contraignant vers une assertion de contenu - soit `c2pa.hash.data`, `c2pa.hash.boxes`, `c2pa.hash.collection.data`, `c2pa.hash.bmff.v2` (obsolète) ou `c2pa.hash.bmff.v3`), la revendication doit être rejetée avec un code d'échec `assertion.dataHash.redacted`, car ces types d'assertions ne doivent pas être expurgés. Sinon, l'assertion expurgée est considérée comme valide et la validation se poursuit en fonction du type d'assertion.

Pour toutes les autres assertions (qui ne figurent pas dans la liste des assertions expurgées), résolvez la référence URI dans le champ `url` pour obtenir ses données. Si l'URI ne renvoie pas à un emplacement dans le même manifeste C2PA (un emplacement `self#jumbf`), la revendication doit être rejetée avec un code d'échec `assertion.outsideManifest`. Si l'URI ne peut être résolue et les données récupérées, la revendication doit être rejetée avec un code d'échec `assertion.missing.`

Suivez la procédure décrite à la section 15.4, « Détermination de l'algorithme de hachage », pour déterminer l'algorithme de hachage et les codes d'échec possibles. Calculez le hachage de l'assertion à l'aide de cet algorithme et de la procédure décrite à la section 8.4.2.3, « Hachage des boîtes JUMBF », et comparez la valeur de hachage calculée avec la valeur dans le champ `de hachage`. Si elles ne correspondent pas, la revendication doit être rejetée avec un code d'échec `assertion.hashedURI.mismatch.` Sinon, un code de réussite `assertion.hashedURI.match` doit être enregistré.

Si le contenu d'une assertion standard n'est pas bien formé en CBOR ou n'est pas conforme à JSON, la revendication sera rejetée avec un code d'échec `assertion.cbor.invalid` ou `assertion.json.invalid`.

REMA Le format CBOR bien formé est défini dans la RFC 8949, annexe C.

RQUE La clause 2 de la RFC 8259 définit la grammaire à laquelle les données JSON doivent se conformer.

REMA

RQUE

Si une assertion présente dans le magasin d'assertions n'est pas référencée par un élément des tableaux `created_assertions` ou `gathered_assertions` dans la revendication (ou le tableau `assertions` dans la revendication v1), la revendication doit être rejetée avec un code d'échec `assertion.undeclared`.

Pour chaque URI du tableau `redacted_assertions` de la revendication, si l'URI pointe vers le manifeste de la revendication elle-même, la revendication doit être rejetée avec un code d'échec `assertion.selfRedacted`. Une revendication n'est pas autorisée à expurger ses propres

assertions.

15.10.3.2. Validation spécifique des assertions

Pour chaque assertion, le validateur doit vérifier l'étiquette de l'assertion et, si elle figure dans la liste ci-dessous, il doit effectuer les étapes de validation spécifiques à ce type d'assertion. Si l'étiquette de l'assertion ne figure pas dans la liste ci-dessous, ce type d'assertion ne nécessite aucune étape de validation supplémentaire au-delà de celles déjà décrites.

- `c2pa.cloud-data`, section 15.10.3.2.1, « Validation `c2pa.cloud-data` »
- `c2pa.actions` ou `c2pa.actions.v2`, section 15.10.3.2.2, « Validation `c2pa.actions` »
- `c2pa.metadata`, section 15.10.3.2.3, « Validation `c2pa.metadata` »

REMARQUE	Ingrédient assertions (`c2pa.ingredient` ou `c2pa.ingredient.v2` ou `c2pa.ingredient.v3`) sont soumis à une validation supplémentaire à un autre stade du processus de validation (voir section 15.11, « Validation des ingrédients »).

Si la valeur d'un champ d'une assertion standard est un `hashed_uri` ou `un hashed_ext_uri`, le validateur doit suivre les étapes décrites à la section 15.10.3.3, « Validation des références », à l'exception du champ `activeManifest` dans `c2pa.ingredient.v3`, pour lequel un comportement de validation spécial est spécifié à la section 15.11.3, « Validation des assertions d'ingrédients ».

15.10.3.2.1. Validation `c2pa.cloud-data`

Si le label de l'assertion est `c2pa.cloud-data` :

1. Vérifiez que l'assertion contient les champs suivants : `label`, `size`, `location` et `content_type`. Si l'un de ces champs est manquant, la revendication doit être rejetée avec un code d'échec `assertion.cloud-data.malformed`.
2. Si le champ `label` de l'assertion externe est `c2pa.hash.data`, `c2pa.hash.boxes`, `c2pa.hash.collection.data`, `c2pa.hash.bmff.v2` (obsolète), `c2pa.hash.bmff.v3`, la revendication doit être rejetée avec un code d'échec `assertion.cloud-data.hardBinding`.
3. Si le manifeste est un manifeste de mise à jour et que le champ `label` de l'assertion externe est `c2pa.actions` ou `c2pa.actions.v2`, la revendication doit être rejetée avec un code d'échec `assertion.cloud-data.actions`.
4. Le champ « `location` » doit être validé conformément à la section 15.10.4.2, « Validation des références externes ».

15.10.3.2.2. Validation `c2pa.actions`

Si le libellé de l'assertion est `c2pa.actions` ou `c2pa.actions.v2` :

1. S'assurer que il a un champ `actions`. Si ce n'est pas le cas, la revendication doit être rejetée avec un code d'échec de `assertion.action.malformed`.
2. Pour chaque action de la liste `des actions` :
   a. Si le champ `d'action` est `c2pa.created` ou `c2pa.opened`, la demande doit être rejetée avec un

code d'échec de `assertion.action.malformed` sauf si toutes les conditions suivantes sont vraies :

i. l'assertion est la première assertion d'actions dans le tableau `created_assertions` ou `gathered_assertions` (d'une revendication v2), ou la première assertion d'actions dans le tableau `assertions` d'une revendication v1, et

ii. l'action est le premier élément du tableau `actions` dans cette assertion.

b. Si le champ `action` est `c2pa.opened`, `c2pa.placed` ou `c2pa.removed` :

i. Si l'action n'a pas de champ `parameters` ou si la valeur de ce champ est vide, la revendication doit être rejetée avec un code d'échec `assertion.action.ingredientMismatch`.

ii. Si le champ `des paramètres` de l'action ne contient pas de champ `« ingrédients »` (ou de champ `« ingrédients »` pour `c2pa.actions`), la demande sera rejetée avec un code d'échec `assertion.action.ingredientMismatch`.

iii. Si la valeur du champ `ingrédients` n'est pas un tableau comportant au moins un élément, la demande sera rejetée avec un code d'échec `assertion.action.ingredientMismatch`.

iv. Vérifier les références aux assertions d'ingrédients :

A. Pour `c2pa.opened` : vérifiez que le champ `« ingredients »` (ou le champ `« ingredient »` pour `c2pa.actions`) contient exactement un URI haché valide pouvant être résolu en une assertion d'ingrédient dans le manifeste actuel dont le champ `« relationship »` est `« parentOf »`. Si ce n'est pas le cas, la revendication sera rejetée avec un code d'échec `« assertion.action.ingredientMismatch »`.

B. Pour `c2pa.placed` : vérifiez que le champ `ingredients` (ou le champ `ingredient` pour `c2pa.actions`) contient un ou plusieurs URI hachés valides, chacun pouvant être résolu en une assertion d'ingrédient dans le manifeste actuel dont le champ `relationship` est `componentOf`. Si ce n'est pas le cas, la revendication sera rejetée avec un code d'échec `assertion.action.ingredientMismatch`.

C. Pour `c2pa.removed` : vérifiez que le champ `ingredients` (ou le champ `ingredient` pour `c2pa.actions`) contient un ou plusieurs URI hachés valides, chacun pouvant être résolu en une assertion d'ingrédient dans un autre manifeste dont le champ `relationship` est `componentOf`. Si ce n'est pas le cas, la revendication sera rejetée avec un code d'échec `assertion.action.ingredientMismatch`.

c. Si le champ `action` est `c2pa.transcoded` ou `c2pa.repackaged` :

i. Si le champ `ingredients` (ou le champ `ingredient` pour `c2pa.actions`) est présent, vérifiez que chaque élément de ce champ est un URI haché valide pouvant être résolu en une assertion d'ingrédient dans le manifeste actuel avec la relation `parentOf`. Si ce n'est pas le cas, la revendication doit être rejetée avec un code d'échec `assertion.action.ingredientMismatch`.

d. Si le champ `d'action` est `c2pa.redacted` :

i. Vérifiez le champ `redacted` qui fait partie de l'objet `parameters` pour voir s'il contient une URI JUMBF. Si l'URI JUMBF n'est pas présente ou ne peut pas être résolue en une assertion, la revendication doit être rejetée avec un code d'échec `assertion.action.redactionMismatch`.

e. S'il existe un champ `softwareAgent` dans `action-common-map-v2` ou un ou plusieurs `softwareAgents` répertoriés dans le champ `softwareAgents` de `actions-map-v2` :

i. S'il existe un champ `icon` dans `generator-info-map`, il doit être validé comme décrit dans la section 15.10.3.3, « Validation des références ».

f. Pour chaque modèle de la liste `templates` :

i. S'il existe un champ `icon` dans `action-template-map-v2`, il doit être validé comme décrit dans la section 15.10.3.3, « Validation des références ».

15.10.3.2.3. Validation des métadonnées c2pa

Si le libellé de l'assertion est `c2pa.metadata`, le validateur doit s'assurer que l'assertion ne contient pas de champs ne figurant pas dans la liste autorisée. Si un champ contenu dans l'assertion ne figure pas dans la liste autorisée, la revendication doit être rejetée avec un code d'échec `assertion.metadata.disallowed`.

REMARQUE Cette exigence de validation nécessitera qu'un validateur analyse les données JSON-LD contenues dans l'assertion.

15.10.3.2.4. Validation c2pa.time-stamp

Si le libellé de l'assertion est `c2pa.time-stamp`, le validateur doit s'assurer que l'assertion est un CBOR bien formé composé d'une seule carte (type majeur `5`) avec au moins une paire clé/valeur. Si ce n'est pas le cas, la revendication doit être rejetée avec un code d'échec `assertion.timestamp.malformed`.

Étant donné que la validation du jeton d'horodatage est effectuée comme décrit dans la section 15.8.2, « Validation du jeton TimeStampToken », le validateur doit stocker le jeton d'horodatage (et son identifiant C2PA Manifest associé) pour une utilisation ultérieure.

15.10.3.3. Validation des références

Certaines assertions standard C2PA prennent en charge la référence à d'autres boîtes dans le manifeste C2PA via l'utilisation d'un `hashed_uri` et d'un `hashed_ext_uri`. Par exemple, il peut y avoir diverses références dans les assertions actions, ingredient et thumbnail.

Pour tous les champs `hashed_uri` et `hashed_ext_uri` dans les assertions standard, à l'exception du champ `activeManifest` dans `c2pa.ingredient.v3` (pour lequel un comportement de validation spécial est spécifié dans la section 15.11.3, « Validation des assertions d'ingrédients »), le validateur doit effectuer la validation suivante : . Pour un champ `hashed_ext_uri` dont le validateur choisit de récupérer la ressource, le validateur doit effectuer les étapes décrites dans la section 15.10.4.2, « Validation des références externes ». . Pour un champ `hashed_uri,` le validateur doit effectuer les étapes décrites ci-dessous.

La destination d'un `hashed_uri` se trouve dans son champ `url`. Si le champ n'est pas présent ou si la destination ne peut être localisée (c'est-à-dire que les données ne se trouvent pas là où elles sont censées être), cela doit être traité comme un échec de validation avec le code `hashedURI.missing`.

Si la destination peut être localisée, procédez comme suit : . Suivez la procédure décrite à la section 15.4, « Détermination de l'algorithme de hachage », pour déterminer l'algorithme de hachage et les codes d'échec possibles. . Assurez-vous que le champ `de hachage` est présent dans la structure `hashed_uri`. Si ce n'est pas le cas, la revendication doit être rejetée avec un code d'échec `hashedURI.mismatch`. . Calculez le hachage de l'assertion à l'aide de l'algorithme de hachage déterminé et de la

procédure décrite à la section 8.4.2.3, « Hachage des boîtes JUMBF ». . Comparez la valeur de hachage calculée avec la valeur du champ `de hachage`. Si elles ne correspondent pas, la revendication doit être rejetée avec un code d'échec `hashedURI.mismatch`.

15.10.4. Validation des données externes

15.10.4.1. Généralités

Le contenu d'une assertion de données dans le cloud contient les références URI et les hachages des données externes. Il est validé comme toute autre assertion, mais ces références ne sont pas récupérées et validées dans le cadre de la validation standard. Un validateur doit d'abord valider avec succès une revendication avant de tenter de récupérer les données externes référencées. Un validateur ne doit pas tenter de récupérer des données externes à partir d'une revendication rejetée. La récupération des données externes étant facultative, l'impossibilité de récupérer ou de valider des données externes ne doit pas entraîner le rejet d'une revendication.

Si un validateur choisit de récupérer l'une des données externes d'une assertion de données cloud, il doit suivre les étapes décrites à la section 15.10.4.2, « Validation des références externes ».

15.10.4.2. Validation des références externes

La procédure suivante doit être utilisée pour valider les données externes référencées dans une assertion de données cloud :

1. Résoudre la référence URI dans le champ `url` pour obtenir ses données. Si le champ `url` n'est pas présent ou si l'URI ne peut être résolu et les données récupérées, le validateur doit abandonner la tentative de récupération des données externes.
2. Si la taille des données récupérées n'est pas égale à la valeur du champ `size`, le validateur doit renvoyer un code d'échec `assertion.hashedURI.mismatch` à l'application et ne pas fournir les données récupérées.
3. Vérifiez que le type de contenu renvoyé dans l'en-tête `Content-Type` de la réponse HTTP est égal au type de contenu déclaré. S'ils ne correspondent pas, le validateur renvoie un code d'échec `assertion.hashedURI.mismatch` à l'application et ne fournit pas les données récupérées. Le type de contenu déclaré est déterminé par :
    a. Pour les données externes, le type de contenu est déterminé par le champ `dc:format` de la structure `hashed_ext_uri`. Si le champ `dc:format` est absent, la validation du type de contenu est toujours réussie.
    b. Pour une assertion de données cloud, si le champ `dc:format` est présent dans son champ `location`, celui-ci détermine le type de contenu et la valeur du champ `content_type` de l'assertion de données cloud est ignorée. Si `location` ne comporte pas de champ `dc:format`, alors le champ `content_type` de l'assertion détermine le type de contenu.
4. Déterminez l'algorithme de hachage à utiliser comme spécifié dans la section 15.4.2, « Pour les URI Ext hachées » ou les codes d'échec possibles.
5. Calculez le hachage des données à l'aide de l'algorithme de hachage déterminé et de la procédure décrite dans la section 8.4.2.3, « Hachage des boîtes JUMBF » sur le contenu récupéré. Pour les données externes, utilisez l'algorithme de hachage et le contenu exact récupéré comme entrée de la fonction de hachage.
    a. Comparez la valeur de hachage calculée avec la valeur du champ `de hachage`. Si le champ `de hachage` n'est pas présent ou s'ils ne correspondent pas, le validateur doit renvoyer un code d'échec `assertion.hashedURI.mismatch` à l'application et ne pas fournir les données récupérées.

b. Sinon, le validateur doit enregistrer un code de réussite `assertion.hashedURI.match` et fournir les données récupérées à l'application.

15.11. Valider les ingrédients

15.11.1. Explication

Un validateur doit effectuer les étapes de validation pour l'actif présenté et son manifeste actif. Si l'une des étapes conclut que le manifeste actif n'est pas valide, ce manifeste doit être rejeté avec le code d'échec indiqué.

Le manifeste actif d'un actif peut répertorier un ou plusieurs ingrédients, grâce à l'utilisation d'assertions d'ingrédients. Certains de ces ingrédients peuvent avoir leurs propres manifestes associés, et certains de ces manifestes peuvent eux-mêmes avoir des ingrédients et des manifestes d'ingrédients.

15.11.2. Traitement des manifestes d'ingrédients

15.11.2.1. Manifestes standard dans un ingrédient

Lors du traitement d'un manifeste standard, un validateur doit valider chaque ingrédient (quelle que soit la valeur de son champ `de relation`), comme décrit ci-dessous.

15.11.2.2. Mise à jour des manifestes dans un ingrédient

Pour les manifestes de mise à jour, l'ingrédient `parentOf` du manifeste de mise à jour doit être validé comme décrit ci-dessous.

15.11.2.3. Manifestes horodatés dans un élément

IMPORTANT	Cette fonctionnalité a été abandonnée au profit de l'assertion d'horodatage. Les informations ci-dessous sont conservées à des fins historiques.

Tout manifeste horodaté trouvé dans un élément doit être ignoré.

15.11.3. Validation de l'assertion d'ingrédient

15.11.3.1. Présentation de la validation

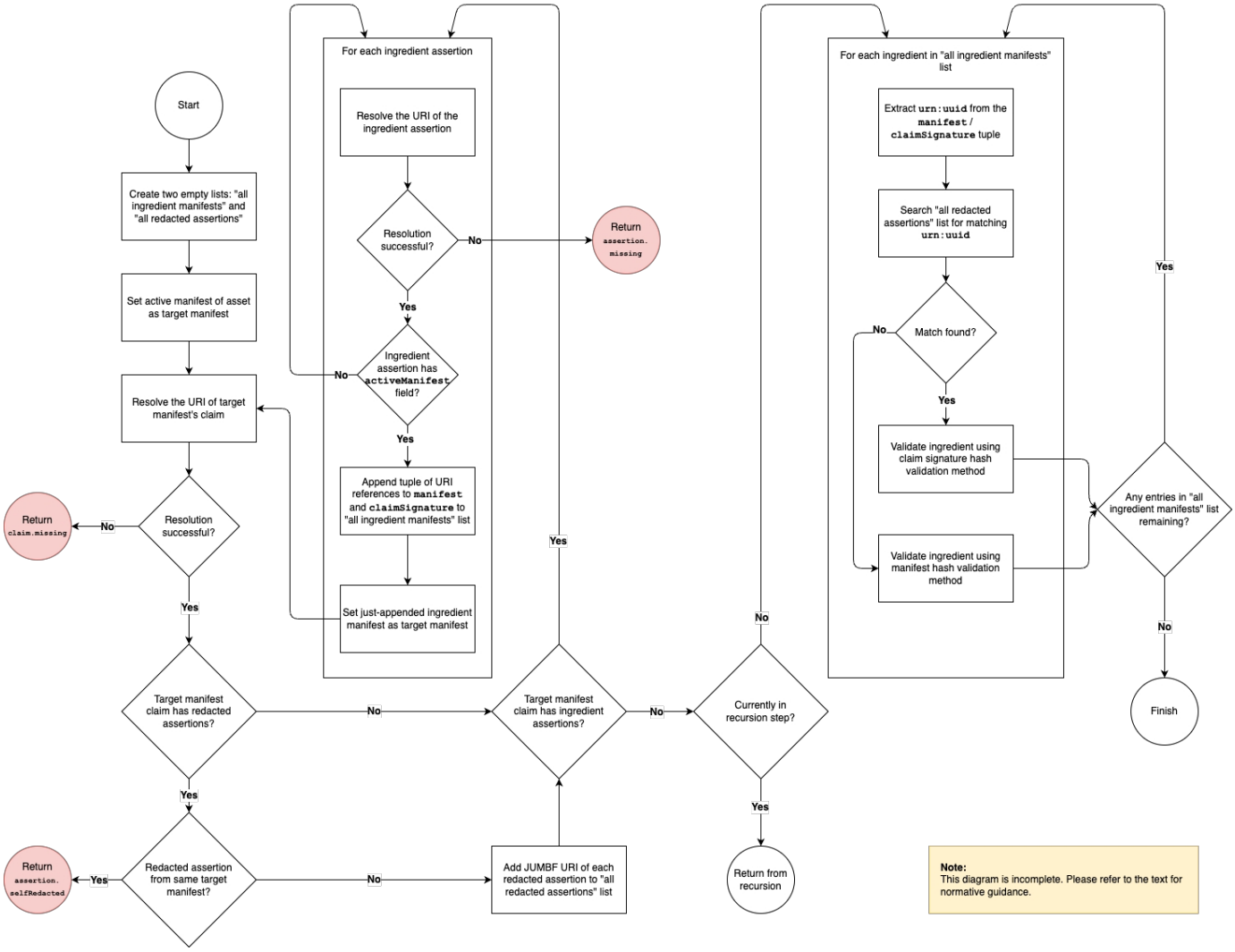

Figure 14. Validation des ingrédients

L'organigramme de la figure 14, « Validation des ingrédients », décrit le processus de validation des assertions relatives aux ingrédients contenues dans un manifeste C2PA donné.

REMARQUE En cas de divergence entre la représentation visuelle et le texte, le texte fait foi.

15.11.3.2. Effectuer une validation explicite

Si le champ « `relation` » n'est pas présent dans une assertion d'ingrédient, l'assertion doit être rejetée avec un code d'échec « `assertion.ingredient.malformed` ».

La valeur du champ `de relation` doit être l'une des suivantes : `parentOf`, `inputTo` ou `componentOf`. Si la valeur du champ `de relation` n'est pas l'une de celles-ci, l'allégation doit être rejetée avec un code d'échec `assertion.ingredient.malformed`.

15.11.3.3. Effectuer une validation récursive

Le validateur doit valider de manière récursive tous les manifestes d'ingrédients dans l'actif, par exemple en utilisant une recherche en profondeur comme

décrit ci-dessous. Un validateur n'est pas tenu de mettre en œuvre l'algorithme exactement tel qu'il est décrit, mais les résultats de la validation doivent être équivalents aux résultats de cet algorithme.

1. Créer deux listes vides :
    a. Une liste contenant les valeurs `hashed_uri` de tous les manifestes d'ingrédients utilisés dans l'actif, où qu'ils se trouvent dans sa lignée.
    b. Une liste pour contenir les URI JUMBF de toutes les assertions expurgées dans l'actif, où qu'elles se trouvent dans sa lignée.
2. Définissez le manifeste actif de l'actif en cours de validation comme manifeste cible.
3. Commencez la récursivité.
4. Localisez la revendication, comme décrit dans la section 15.6, « Localisation et validation de la revendication ». Si cela s'avère impossible, rejetez la revendication avec un code d'échec `claim.missing`.
5. Si la revendication du manifeste cible contient un champ `redacted_assertions`, vérifiez l'URI JUMBF de chaque assertion expurgée.
    a. Si l'assertion expurgée provient du manifeste cible, rejetez la revendication avec un code `d'échec assertion.selfRedacted` .
    b. Sinon, ajoutez l'URI JUMBF de l'assertion expurgée à la liste de toutes les assertions expurgées.
6. Si la déclaration du manifeste cible comprend des assertions relatives aux ingrédients :
    a. Pour chaque assertion relative aux ingrédients :
        i. Essayez de résoudre l'URI haché de l'assertion relative à l'ingrédient. Si l'URI ne se résout pas, si le hachage ne correspond pas ou si les cases de contenu JUMBF de l'assertion ne contiennent que des zéros, passez à l'assertion relative à l'ingrédient suivant.
        ii. Si l'assertion d'ingrédient comporte un champ `activeManifest` (ou un champ `c2pa_manifest` dans une assertion d'ingrédient v1 ou v2) :
            A. Ajoutez un tuple comprenant les valeurs suivantes à la liste de tous les manifestes d'ingrédients :
                - La valeur `hashed_uri` du champ `activeManifest` (ou `c2pa_manifest`) dans l'assertion ingredient
                - La valeur `hashed_uri` du champ `claimSignature` dans l'assertion ingredient
            B. Définissez le manifeste d'ingrédient qui vient d'être ajouté comme manifeste cible, puis répétez le processus ci-dessus à partir de l'étape « Commencer la récursivité ».
        iii. Si l'assertion d'ingrédient ne comporte pas de champ `activeManifest` (ou `c2pa_manifest`), enregistrez un code d'information `ingredient.unknownProvenance,` sauf si la valeur du champ `relationship` est `inputTo`, puis passez à l'assertion d'ingrédient suivante jusqu'à ce qu'elles soient toutes épuisées. À ce stade, revenez du niveau de récursivité actuel.
7. Si la revendication du manifeste cible ne comprend pas d'assertions d'ingrédients, revenez du niveau de récursivité actuel.
8. Fin de la récursivité.

Après avoir compilé une liste de tous les manifestes d'ingrédients et une liste de toutes les assertions expurgées, le validateur doit exécuter l'algorithme de validation suivant

algorithme de validation suivant :

1. Pour chaque manifeste d'ingrédients figurant dans la liste de tous les manifestes d'ingrédients :
    a. Extraire l'étiquette de la liste des ingrédients à partir de l'URI JUMBF de chaque tuple
    b. Rechercher dans la liste de toutes les assertions expurgées les assertions dont l'étiquette de manifeste correspond
    c. Si une ou plusieurs assertions expurgées correspondantes sont trouvées :
        i. Validez l'ingrédient à l'aide de la méthode de validation du hachage de signature de revendication, décrite à la section 15.11.3.3.1, « Méthode de validation du hachage de signature de revendication ».
    d. Si aucune assertion expurgée correspondante n'est trouvée :
        i. Validez l'ingrédient à l'aide de la méthode de validation du hachage du manifeste, décrite à la section 15.11.3.3.2, « Méthode de validation du hachage du manifeste », ou de la méthode de validation du hachage de la signature de la déclaration, décrite à la section 15.11.3.3.1, « Méthode de validation du hachage de la signature de la déclaration ».
    e. Si l'assertion d'ingrédient contient un champ `validationResults` :
        i. Pour chaque entrée dans la valeur du champ `validationResults`, si une entrée équivalente n'a pas été renvoyée dans le cadre du processus de validation, renvoyez-la dans le cadre des résultats de validation.
        ii. Si des entrées renvoyées dans le cadre du processus de validation ne sont pas présentes dans le champ champ `validationResults`, renvoyez-la dans le cadre des résultats de validation.
    f. Si aucun champ `validationResults` n'est présent et que l'assertion d'ingrédient est une assertion d'ingrédient v3 avec le champ champ `activeManifest` présent, renvoyer le code d'échec `assertion.ingredient.malformed`.

Les validateurs doivent ignorer tout manifeste C2PA supplémentaire qui apparaît dans le magasin de manifestes C2PA mais qui ne figure pas dans la liste des manifestes d'ingrédients.

REMARQUE Le fait d'ignorer les manifestes C2PA supplémentaires permet d'assurer la compatibilité avec les assertions personnalisées et les constructions futures qui peuvent faire référence aux manifestes C2PA d'une manière que le validateur ne reconnaît pas.

15.11.3.3.1. Méthode de validation du hachage de signature de revendication

Cette méthode comprend une validation complète de la revendication de l'ingrédient, comme celle effectuée pour le manifeste actif, sauf que les liaisons de contenu ne sont pas évaluées :

1. Résolvez la référence URI dans la valeur `url` du champ `claimSignature` pour obtenir la boîte de signature de la revendication de l'ingrédient. Si la référence URI ne peut pas être résolue ou si le champ `claimSignature` n'est pas présent, la revendication de l'ingrédient est rejetée avec un code d'échec `ingredient.claimSignature.missing`.
2. Déterminez l'identifiant de l'algorithme de hachage (ou le code d'échec éventuel) en suivant la procédure décrite à la section 15.4, « Détermination de l'algorithme de hachage ».
3. Calculez le hachage de la case de signature de la déclaration des ingrédients à l'aide de cet algorithme et de la procédure décrite à la section 8.4.2.3, « Hachage des cases JUMBF ».
4. Comparez le hachage calculé avec la valeur dans le champ `de hachage`.

a. Si les hachages ne sont pas identiques ou si le champ `de hachage` n'est pas présent :

   i. Rejetez la déclaration avec un code d'échec `ingredient.claimSignature.mismatch`.

b. Si les hachages sont égaux, émettez un code `ingredient.claimSignature.validated`.

   i. Valider la signature de la revendication, l'horodatage et les informations de révocation des informations d'identification conformément à la section 15.7, « Valider la signature », à la section 15.8, « Valider l'horodatage », et à la section 15.9, « Valider les informations de révocation des informations d'identification ».

   ii. Pour chaque URI de la liste des assertions expurgées avec une étiquette manifeste correspondante, si l'assertion référencée est présente et qu'une boîte de contenu JUMBF ou une boîte de remplissage qu'elle contient contient autre chose que zéro ou plusieurs octets `0x00`, la revendication doit être rejetée avec un code d'échec `assertion.notRedacted`.

   iii. Valider chaque assertion non expurgée conformément à la section 15.10, « Valider les assertions », à l'exception des assertions contraignantes, qui ne peuvent pas être validées pour les ingrédients.

Lorsqu'il utilise la méthode de validation du hachage de la signature de la revendication, le validateur ne doit pas enregistrer de codes d'échec de non-correspondance de hachage pour le champ `activeManifest`.

REMARQUE La raison en est que si les expurgations affectent le manifeste référencé, il est possible que le hachage de ce champ ne corresponde pas.

15.11.3.3.2. Méthode de validation du hachage du manifeste

Un manifeste d'ingrédients qui n'a pas été modifié en raison d'une rédaction peut être validé plus rapidement si le validateur actuel fait confiance aux résultats de validation du générateur de revendications précédent :

1. Résolvez la référence URI dans la valeur `url` du champ `activeManifest` pour obtenir la boîte du manifeste de l'ingrédient. Si le champ `url` n'est pas présent ou si la référence URI ne peut être résolue, la déclaration d'ingrédient est rejetée avec un code d'échec `ingredient.manifest.missing`.
2. Déterminez l'identifiant de l'algorithme de hachage (ou le code d'échec éventuel) en suivant la procédure décrite à la section 15.4, « Détermination de l'algorithme de hachage ».
3. Calculez le hachage de la boîte manifeste de l'ingrédient à l'aide de cet algorithme et de la procédure décrite à la section 8.4.2.3, « Hachage des boîtes JUMBF ».
4. Comparez le hachage calculé avec la valeur dans le champ `hachage`.

   a. Si les hachages ne sont pas identiques ou si le champ `hash` n'est pas présent :

      i. Rejetez la demande avec un code d'échec `ingredient.manifest.mismatch`.

   b. Si les hachages sont égaux, l'ingrédient est entièrement validé et un code de réussite `ingredient.manifest.validated` est émis.

15.12. Valider le contenu de l'actif

Le contenu de l'actif doit être validé à l'aide de la liaison forte dans le manifeste actif si celui-ci est un manifeste standard. Si le manifeste actif est un manifeste de mise à jour, la liaison forte doit être trouvée dans le manifeste `parentOf` de l'ingrédient

ou, si ce manifeste est également un manifeste de mise à jour, en suivant la chaîne des ingrédients `parentOf` jusqu'au premier manifeste standard. Si aucun manifeste standard n'est trouvé ou si le manifeste standard ne comporte aucune liaison forte, la revendication du manifeste actif doit être rejetée avec un code d'échec `claim.hardBindings.missing`.

Un actif peut également être composé de plusieurs parties, chacune ayant son propre hachage associé (voir section 18.9, « Hachage multi-actifs ») qui peut être validé séparément. Par exemple, un actif peut être composé de parties distinctes d'images statiques et de vidéos, chacune pouvant être validée séparément.

15.12.1. Validation d'un hachage de données

15.12.1.1. Généralités

Une fois qu'un manifeste standard (et ses liaisons) a été localisé, la ou les plages d'exclusion doivent être extraites de l'assertion
assertion `c2pa.hash.data`.

Si le décalage d'octet de fin d'une plage d'exclusion (`début` + `longueur`) est supérieur au décalage d'octet de début de la plage d'exclusion suivante dans le tableau, ou si une valeur `de début` ou `de longueur` est négative, le manifeste doit être rejeté avec un code d'échec `assertion.dataHash.malformed`.

Si des manifestes de mise à jour ont été rencontrés, la valeur `de longueur` de la plage d'exclusion dont la valeur `de début` est le décalage du début de l'ensemble du magasin de manifestes C2PA doit être traitée comme la longueur actuelle de l'ensemble du magasin de manifestes C2PA, plus les éventuels ajouts spécifiques au format de fichier.

L'algorithme de hachage (`alg`) spécifié dans `c2pa.hash.data` doit être calculé sur les octets de l'actif, à l'exception de ceux spécifiés dans la ou les plages d'exclusion. Si la fin d'une plage d'exclusion dépasse la fin de l'actif, le manifeste doit être rejeté avec un code d'échec `assertion.dataHash.mismatch`.

Si l'algorithme de hachage spécifié dans le champ `alg` n'apparaît pas dans la liste des algorithmes autorisés ou obsolètes de la section 13.1, « Hachage », le manifeste doit être rejeté avec un code d'échec `algorithm.unsupported`. Si le champ `hash` n'est pas présent, le manifeste doit être rejeté avec un code d'échec `assertion.dataHash.mismatch`.

La combinaison des plages d'exclusion et des valeurs de remplissage, en particulier le remplissage nécessaire pour prendre en charge les flux de travail à passages multiples, peut permettre à un attaquant de remplacer certaines parties de ce remplissage par des données arbitraires susceptibles d'avoir un impact sur la consommation de l'actif sans invalider le hachage. Pour cette raison, un validateur doit s'assurer que les données contenues dans la plage d'exclusion, y compris un magasin de manifestes C2PA, se composent uniquement du magasin de manifestes C2PA et d'un remplissage approprié (par exemple, des données à zéro) dans des champs de remplissage clairement marqués ou des cases libres/à ignorer. Dans les autres plages d'exclusion que celles du magasin de manifestes C2PA ci-dessus, tout ou partie des métadonnées de l'actif peuvent également être incluses, comme décrit à la section 9.2.5, « Liaisons de métadonnées d'actifs ». Si un validateur rencontre des données autres que celles autorisées ci-dessus, le manifeste doit être rejeté avec un code d'échec `assertion.dataHash.mismatch.` Si un validateur rencontre des plages d'exclusion autres que celles du magasin de manifestes C2PA et un remplissage approprié (par exemple, des données à zéro) dans des champs de remplissage clairement marqués ou des cases libres/à ignorer, un code d'information `assertion.dataHash.additionalExclusionsPresent` doit être défini.

Si aucune condition d'erreur n'a été rencontrée, le validateur doit ajouter le code de réussite `assertion.dataHash.match.`
à la liste qu'il renvoie finalement.

Si le hachage calculé sur toutes les données de l'actif (à l'exception des plages d'exclusion) ne correspond pas à la valeur du champ `hachage` dans `c2pa.hash.data`, le validateur doit alors rechercher la présence d'une assertion de hachage multi-actifs. Si tel est le cas, elle doit être validée comme décrit à la section 15.12.4, « Validation d'un hachage multi-actifs », mais si ce n'est pas le cas, le manifeste doit être rejeté avec un code d'échec `assertion.dataHash.mismatch`.

15.12.1.2. Hachage des fichiers JPEG 1

Dans les fichiers JPEG 1, les éléments supplémentaires du format de fichier décrits ci-dessus comprennent tous les marqueurs `APP11` et leurs octets de longueur de segment respectifs pour les segments `APP11`. Étant donné que les longueurs de segment se trouvent dans la plage d'exclusion, un validateur doit faire correspondre la longueur totale de la plage d'exclusion avec celle de la longueur totale de tous les segments `APP11` représentant le manifeste C2PA afin de s'assurer que la longueur n'a pas été altérée.

REMARQUE Un fichier JPEG 1 peut contenir des segments `APP11` pour des raisons autres que C2PA (par exemple, JPEG 360 ou JPEG Privacy and Security) et ceux-ci ne sont pas inclus dans ces calculs.

15.12.2. Validation d'un hachage BMFF

Pour toute partie d'un actif rendue pour présentation à un utilisateur, y compris, mais sans s'y limiter, l'audio, la vidéo ou le texte, la liaison forte correspondante au contenu rendu doit être validée conformément à la section 9.2, « Liaisons fortes ». Si la liaison forte standard n'est pas validée et qu'une assertion de hachage multi-actifs est présente, elle doit être validée comme décrit dans [validating_a_multi_asset_hash]. Si, à un moment quelconque, le contenu ne parvient pas à être validé, le validateur doit clairement signaler à l'utilisateur qu'une partie du contenu ne correspond pas à la revendication et, si possible, indiquer quelle partie du contenu n'a pas été validée. Si un contenu pour lequel il existe des liaisons de contenu est absent, la découverte de cette absence constitue également un échec de validation. Le validateur doit continuer à signaler que la validation a échoué, même si les parties ultérieures du contenu sont validées correctement.

Pour les contenus qui ne sont pas entièrement disponibles avant le début du rendu, comme lors d'un streaming à débit adaptatif (ABR) ou d'un téléchargement progressif, l'absence de parties du contenu qui ne sont pas encore disponibles n'est pas considérée comme un échec de validation. À mesure que le contenu devient disponible, le validateur doit valider chaque partie du contenu avant qu'elle ne soit rendue, comme décrit précédemment. En outre, le validateur doit vérifier que la séquence dudit contenu est la même que celle qui existait lors de la création du manifeste. À moins que le lecteur n'ait explicitement signalé au validateur qu'une discontinuité est attendue (par exemple, lorsque le consommateur effectue une recherche manuelle via l'interface utilisateur), le validateur doit clairement signaler à l'utilisateur qu'une discontinuité inattendue s'est produite chaque fois que la séquence ne correspond pas. Cela inclut la validation du fait que les valeurs `d'emplacement` pour un arbre Merkle donné commencent à zéro et augmentent d'une unité pour chaque bloc suivant ; de manière équivalente, la valeur `d'emplacement` indique toujours quel bloc est rendu.

Pour le contenu qui doit être validé pendant la lecture via un téléchargement progressif, les nœuds feuilles de l'arbre Merkle peuvent s'aligner sur les points de synchronisation de la piste vidéo dans le « `mdat` » (par exemple, les points d'accès aléatoire RAP). Lorsque `les « variableBlockSizes »` sont configurés pour obtenir un tel alignement, la validation pendant la lecture linéaire ou la recherche du temps de lecture souhaité peuvent toutes deux être réalisées via la même séquence. Les blocs souhaités doivent être récupérés, validés et les pistes qu'ils contiennent sélectionnées pour le rendu.

Pour les contenus qui ne sont pas rendus intentionnellement comme le générateur de revendications l'avait initialement prévu, par exemple lors d'un avance rapide, d'un retour en arrière ou d'une lecture à une vitesse différente, le validateur peut ne pas être en mesure de valider le contenu. Dans ce cas,

le validateur doit clairement signaler à l'utilisateur que le contenu ne peut pas être validé pendant l'opération correspondante.

Pour le contenu avec C2PA ContentProvenanceBox et box_purpose défini sur update presence, le manifeste actif est d'abord recherché dans C2PA ContentProvenanceBox avec box_purpose défini sur update, puis dans C2PA ContentProvenanceBox avec box_purpose défini sur original. Si le manifeste actif se trouve dans le C2PA ContentProvenanceBox avec box_purpose défini sur update, tracez la chaîne parentale des ingrédients (en recherchant soit dans le C2PA ContentProvenanceBox avec box_purpose défini sur update, soit dans original, selon les besoins) jusqu'à ce que le premier manifeste non mis à jour soit trouvé. Le hachage BMFF du contenu de ce manifeste doit être validé conformément à la section 9.2, « Liaisons rigides ». L'ajout d'une C2PA ContentProvenanceBox avec box_purpose défini sur update ne doit pas affecter le calcul du hachage, car elle a été ajoutée à la fin du fichier sans modifier aucun décalage.

Si le bmff-hash-map ne contient pas de champ exclusions ou si la valeur de ce champ n'est pas de type tableau avec au moins une entrée, le manifeste doit être rejeté avec un code d'échec assertion.bmffHash.malformed.

Déterminez l'identifiant de l'algorithme de hachage (ou le code d'échec éventuel) en suivant la procédure décrite à la section 15.4, « Détermination de l'algorithme de hachage ».

Si le décalage d'octet final d'une plage de sous-ensemble (décalage + longueur) est supérieur à la valeur de décalage de la plage suivante dans le tableau, ou si une valeur de décalage ou de longueur est négative, le manifeste doit être rejeté avec un code d'échec assertion.bmffHash.malformed. Le code d'échec assertion.bmffHash.mismatch est utilisé pour tous les autres échecs décrits dans cette section. Sinon, le validateur doit ajouter le code de réussite assertion.bmffHash.match à la liste qu'il renvoie finalement.

Si le processus de hachage BMFF produit un code d'échec assertion.bmffHash.mismatch, le validateur doit alors rechercher la présence d'une assertion de hachage multi-actifs. Si tel est le cas, le code d'échec assertion.bmffHash.mismatch ne doit pas être émis, et l'assertion de hachage multi-actifs doit être validée comme décrit à la section 15.12.4, « Validation d'un hachage multi-actifs » ; sinon, le manifeste doit être rejeté avec un code d'échec assertion.bmffHash.mismatch.

15.12.2.1. Actif non fragmenté utilisant un arbre de Merkle

Si le champ merkle dans le bmff-hash-map est présent, le validateur doit valider l'arbre Merkle. Si les champs fixedBlockSize et variableBlockSizes dans bmff-merkle-map ne sont pas présents, l'ensemble de la charge utile du mdat est traité comme un nœud feuille unique pour le calcul du hachage. Si le champ fixedBlockSize est présent et si le champ variableBlockSizes n'est pas présent, la charge utile du mdat est divisée en blocs de longueur fixe, chaque bloc étant traité comme un nœud feuille. Si le dernier bloc dépasse la fin de la charge utile du mdat, la taille du dernier bloc doit être définie de manière à ne s'étendre que jusqu'à la fin de la charge utile du mdat. Si variableBlockSize est présent et si fixedBlockSizes n'est pas présent, la charge utile du mdat est divisée en tailles définies par le tableau variableBlockSizes. Si le nombre d'éléments n'est pas égal à count ou si la somme des valeurs n'est pas égale à la taille de la charge utile du mdat, le manifeste doit être rejeté avec un code d'échec assertion.bmffHash.malformed. Si les variables fixedBlockSize et variableBlockSizes sont présentes dans bmff-merkle-map, le manifeste doit être rejeté avec un code d'échec assertion.bmffHash.malformed.

Si le nombre dans bmff-merkle-map est égal au nombre d'éléments de hachage dans bmff-merkle-map
et si le hachage du nœud feuille ne correspond pas à l'élément des hachages dans la bmff-merkle-map, alors le manifeste

est rejeté avec un code d'échec assertion.bmffHash.mismatch Si le nombre dans la bmff-merkle-map est inférieur au nombre d'éléments de hachage dans la bmff-merkle-map et si la boîte uuid C2PA auxiliaire n'existe pas comme décrit dans la section A.5.4, « Boîtes auxiliaires « c2pa » pour les fichiers volumineux et fragmentés », le manifeste doit être rejeté avec un code d'échec assertion.bmffHash.malformed. Si le hachage calculé à partir de la boîte uuid C2PA auxiliaire et du nœud feuille ne correspond pas à l'élément des hachages dans le bmff-merkle-map, le manifeste doit être rejeté avec un code d'échec assertion.bmffHash.mismatch. Si le nombre dans le bmff-merkle-map est supérieur au nombre d'éléments de hachage dans le bmff-merkle-map, le manifeste doit être rejeté avec un code d'échec assertion.bmffHash.malformed.

15.12.2.2. Actif fragmenté utilisant l'arbre Merkle

Si le champ merkle dans la bmff-hash-map est présent, le validateur doit valider l'arbre Merkle. Si la boîte uuid C2PA auxiliaire n'existe pas comme décrit dans la section A.5.4, « Boîtes auxiliaires « c2pa » pour les fichiers volumineux et fragmentés », le manifeste doit être rejeté avec un code d'échec assertion.bmffHash.malformed. Si le hachage calculé à partir de la boîte uuid C2PA auxiliaire et du nœud feuille ne correspond pas à l'élément des hachages dans le bmff-merkle-map, n'est pas égal, alors le manifeste doit être rejeté avec un code d'échec de assertion.bmffHash.mismatch

15.12.3. Validation d'un hachage de boîte général

Une fois qu'un manifeste standard (et ses liaisons) a été localisé, la liste des boîtes à valider doit être extraite du champ boxes de la structure box-map stockée dans l'assertion c2pa.hash.boxes. Si ce champ n'est pas présent, le manifeste doit être rejeté avec un code d'échec assertion.boxesHash.malformed.

Les boîtes doivent apparaître dans l'actif dans le même ordre que celui dans lequel elles apparaissent dans le tableau des boîtes, y compris la boîte contenant le manifeste C2PA. Si d'autres boîtes sont présentes dans l'actif, le manifeste doit être rejeté avec un code d'échec assertion.boxesHash.unknownBox. Si les boîtes apparaissent dans le désordre, le manifeste doit être rejeté avec un code d'échec assertion.boxesHash.mismatch.

Si la valeur de hachage d'une boîte ne correspond pas et que cette boîte ne comporte pas de champ exclu avec une valeur true, le manifeste doit être rejeté avec un code d'échec assertion.boxesHash.mismatch. Sinon, le validateur doit ajouter le code de réussite assertion.boxesHash.match à la liste qu'il renvoie finalement.

Si l'algorithme de hachage spécifié dans un champ alg n'apparaît pas dans la liste des algorithmes autorisés ou obsolètes de la section 13.1, « Hachage », ou si un champ alg n'apparaît ni dans la carte de boîtes ni dans une carte de hachage de boîtes spécifique, le manifeste doit être rejeté avec un code d'échec algorithm.unsupported.

Si une boîte-hash-map dans le tableau boxes ne contient pas de champ names, alors le manifeste sera rejeté avec un code d'échec assertion.boxesHash.malformed.

Pour chaque boîte répertoriée dans les tableaux names et boxes, l'algorithme de hachage spécifié doit être calculé sur les octets de la boîte (ainsi que sur tout en-tête associé). S'il existe plusieurs entrées dans un tableau names, la valeur de hachage pour cette plage de boîtes doit être calculée depuis le début de la première boîte (dans la plage) jusqu'à la fin de la dernière boîte (dans la plage). Cela inclut tous les octets arbitraires pouvant être présents entre les boîtes.

Si le champ de hachage n'est pas présent ou si le hachage résultant ne correspond pas à la valeur du champ de hachage pour ces boîtes, le manifeste doit être rejeté avec un code d'échec assertion.boxesHash.mismatch. Si le processus de hachage des boîtes produit un code d'échec assertion.boxesHash.mismatch, le validateur doit rechercher la présence d'une assertion de hachage multi-actifs. Si tel est le cas, il doit être validé comme décrit à la section 15.12.4, « Validation d'un hachage multi-actifs », mais s'il n'est pas présent, le manifeste doit être rejeté avec un code d'échec assertion.boxesHash.mismatch.

15.12.3.1. Traitement spécial JPEG

Lors de la validation d'un fichier JPEG, le validateur doit vérifier que chaque boîte identifiée par l'identifiant spécial C2PA est bien une APP11 contenant tout ou partie du magasin de manifestes C2PA. Le magasin de manifestes C2PA est identifié comme étant une super-boîte JUMBF avec une étiquette c2pa et un UUID de type JUMBF de 63327061-0011-0010-8000-00AA00389B71, comme décrit dans la section 11.1.4.2, « Magasin de manifestes ».

Si un APP11 qui ne fait pas partie du magasin de manifestes C2PA est présent et n'est pas inclus dans la liste des boîtes hachées, le manifeste doit être rejeté avec un code d'échec assertion.boxesHash.unknownBox.

15.12.3.2. Traitement spécial des polices

Lors de la validation d'une police, le validateur doit vérifier que la boîte correspondant à la table C2PA de la police est présente et déterminer si elle contient un manifeste intégré, une URI de manifeste distant ou les deux.

Si des tables de polices non couvertes par une case sont présentes, le manifeste doit être rejeté avec un code d'échec assertion.boxesHash.unknownBox.

15.12.4. Validation d'un hachage multi-actifs

Si la validation standard de la liaison forte de l'actif échoue et que l'actif contient une assertion de hachage multi-actifs, le validateur doit alors procéder à la validation de l'assertion de hachage multi-actifs. Si plusieurs assertions de hachage multi-actifs sont présentes, le manifeste doit être rejeté avec un code d'échec assertion.multiAssetHash.malformed.

La validation de l'assertion de hachage multi-actifs (c2pa.hash.multi-asset) doit être effectuée en itérant sur le tableau des parties dans le multi-asset-hash-map. Si le champ parts n'est pas présent, ou s'il est présent avec une valeur qui est un tableau vide, alors le manifeste doit être rejeté avec un code d'échec assertion.multiAssetHash.malformed.

Pour chaque partie, le validateur doit s'assurer qu'elle contient à la fois un localisateur valide et un champ hashAssertion valide. Si l'un ou l'autre de ces éléments est manquant, le manifeste doit être rejeté avec un code d'échec assertion.multiAssetHash.malformed.

Si le localisateur est un localisateur d'offset d'octets, le validateur doit s'assurer que les champs byteOffset et length sont présents, non négatifs et ne dépassent pas la longueur totale de la ressource. Si l'un de ces champs est manquant, négatif ou trop grand, le manifeste doit être rejeté avec un code d'échec assertion.multiAssetHash.malformed.

Si le localisateur est représenté par un bmffBox, le validateur doit s'assurer que la boîte spécifiée est présente dans l'

Si la boîte est non présente, alors le manifeste doit être rejeté avec un code d'échec de `assertion.multiAssetHash.malformed`.

Si le localisateur et le hachage sont valides, le validateur doit essayer de trouver la partie en utilisant les infos du localisateur. Si elle n'est pas là, et que le champ `optionnel` n'est pas là ou est là avec une valeur `fausse`, alors le manifeste doit être rejeté avec un code d'échec `assertion.multiAssetHash.missingPart`. Si le champ `optionnel` est là avec une valeur `vraie`, alors le validateur doit passer cette partie et continuer avec la suivante.

NOTE	Le rejet de certaines parties peut empêcher un validateur d'identifier sans ambiguïté le parties restantes. Dans la plupart des cas, seules une ou plusieurs parties à la fin du fichier, plutôt que des parties au milieu, peuvent être supprimées efficacement.

Si les parties localisées se chevauchent ou ne couvrent pas, dans leur ensemble, chaque octet de l'actif, le manifeste doit être rejeté avec un code d'échec `assertion.multiAssetHash.malformed`.

Pour chaque partie localisée, le validateur doit calculer le hachage de la partie à l'aide de l'algorithme et de la méthodologie spécifiés (c'est-à-dire le hachage des données, le hachage général de la boîte ou le hachage BMFF) sur les octets de la partie. Si le hachage résultant ne correspond pas à la valeur présente dans l'assertion de liaison forte référencée dans le champ `hashAssertion`, le manifeste doit être rejeté avec un code d'échec `assertion.multiAssetHash.mismatch`.

Si l'assertion de hachage pour chaque partie localisée est validée avec succès, le validateur doit enregistrer le code de réussite `assertion.multiAssetHash.match` et ne doit enregistrer aucun code d'échec associé à la liaison forte de l'actif.

15.12.5. Validation d'un hachage de données de collection

15.12.5.1. Généralités

La validation d'une assertion de hachage de données de collection (`c2pa.hash.collection.data`) qui a été localisée dans un manifeste standard doit être effectuée en itérant sur le tableau `d'URI` dans le `collection-data-hash-map`. Si aucun champ `URI` n'est présent, le manifeste doit être rejeté avec un code d'échec `assertion.collectionHash.malformed`.

L'algorithme de hachage spécifique à utiliser doit être déterminé à partir de la valeur du champ `alg` et traité comme spécifié dans la section 15.4.3, « Validation de l'algorithme ». Si le champ `alg` n'est pas présent, le manifeste doit être rejeté avec un code d'échec `assertion.collectionHash.malformed`.

Pour chaque `uri-hashed-data-map` dans le tableau `uris`, le validateur doit s'assurer qu'il contient à la fois un champ `uri` et un champ `hash`. Si l'un de ces champs est manquant, le manifeste doit être rejeté avec un code d'échec `assertion.collectionHash.malformed`.

Afin d'éviter tout problème de sécurité potentiel, un validateur doit valider les URI (c'est-à-dire la valeur du champ `uri`) avant leur utilisation, en s'assurant que ni `.` ni `..` n'apparaissent dans l'URI. Si l'un ou l'autre de ces caractères est trouvé dans une URI, le manifeste doit être rejeté avec un code d'échec `assertion.collectionHash.invalidURI`.

Pour l'actif récupéré à partir de l'URI, son hachage doit être calculé à l'aide de l'algorithme spécifié sur tous les octets de ses données. Si le hachage résultant ne correspond pas à la valeur du champ `de hachage`, le manifeste doit être rejeté avec un code d'échec `assertion.collectionHash.mismatch`. Sinon, le validateur doit ajouter le code de réussite `assertion.collectionHash.match` à la liste qu'il renvoie finalement.

Si le validateur ne trouve pas certains fichiers répertoriés dans l'assertion de hachage des données de la collection, le manifeste doit être rejeté avec un code d'échec `assertion.collectionHash.incorrectFileCount`.

15.12.5.2. Extras pour ZIP

Dans un fichier ZIP associé à un manifeste C2PA, le hachage des données de la collection contient le champ supplémentaire `zip_central_directory_hash`. Comme décrit précédemment, ce champ contient un hachage de chaque « en-tête du répertoire central » dans le répertoire central ZIP ainsi que la « fin de l'enregistrement du répertoire central » (qui est la dernière partie d'un fichier ZIP). L'algorithme de hachage utilisé pour ce champ est le même que celui utilisé pour le champ `de hachage` dans l'assertion `c2pa.hash.collection.data`.

Lors de la validation d'un fichier ZIP, le validateur doit vérifier que le champ `zip_central_directory_hash` est présent et que le hachage du répertoire central ZIP et de la « fin de l'enregistrement du répertoire central » correspond à sa valeur. Si le hachage ne correspond pas, le manifeste doit être rejeté avec un code d'échec `assertion.collectionHash.mismatch`.

Chapitre 16. Expérience utilisateur

16.1. Approche

La C2PA a pour objectif de fournir des recommandations et des conseils clairs aux développeurs d'expériences utilisateur (UX) basées sur la provenance. L'élaboration de ces recommandations est un processus continu qui implique divers acteurs, et dont les résultats doivent trouver un équilibre entre uniformité et familiarité d'une part, et utilité et flexibilité pour les utilisateurs d'autre part, quels que soient le contexte, la plateforme et l'appareil utilisés. Ces recommandations sont disponibles dans le document intitulé « User experience guidance » (Conseils sur l'expérience utilisateur).

16.2. Principes

Les recommandations en matière d'expérience utilisateur visent à définir les meilleures pratiques pour présenter la provenance C2PA aux consommateurs. Elles s'efforcent de décrire des expériences standard et facilement reconnaissables qui :

- fournissent aux créateurs d'actifs un moyen de capturer des informations et l'historique du contenu qu'ils créent, et
- fournir aux consommateurs d'actifs des informations et l'historique du contenu qu'ils consultent, leur permettant ainsi de comprendre d'où il provient et de décider dans quelle mesure ils peuvent s'y fier.

Les interfaces utilisateur conçues pour la consommation de la provenance C2PA doivent être informées par le contexte de l'actif. Nous avons étudié quatre groupes d'utilisateurs principaux et un ensemble de contextes dans lesquels les ressources C2PA sont rencontrées. Ces groupes d'utilisateurs ont été définis dans les principes directeurs C2PA comme étant les consommateurs, les créateurs, les éditeurs et les vérificateurs (ou enquêteurs). Afin de répondre aux besoins de chacun de ces groupes dans des contextes courants, des interfaces utilisateur exemplaires sont présentées pour de nombreux cas courants. Il s'agit de recommandations, et non d'obligations, et nous nous attendons à ce que les meilleures pratiques évoluent.

16.3. Niveaux de divulgation

Étant donné que l'ensemble complet des données C2PA pour un actif donné peut être trop volumineux pour un utilisateur, nous décrivons 4 niveaux de divulgation progressive qui guident les conceptions :

- Niveau 1 : indication de la présence de données C2PA et de leur statut de validation cryptographique.
- Niveau 2 : résumé des données C2PA disponibles pour un actif donné. Ce niveau doit fournir suffisamment d'informations sur le contenu, l'utilisateur et le contexte particuliers pour permettre au consommateur de comprendre dans une mesure suffisante comment l'actif est arrivé à son état actuel.
- Niveau 3 : affichage détaillé de toutes les données de provenance pertinentes. Il convient de noter que la pertinence de certains éléments par rapport à d'autres est contextuelle et déterminée par le responsable de la mise en œuvre de l'expérience utilisateur.
- Niveau 4 : pour une utilisation sophistiquée à des fins d'enquête judiciaire, il est recommandé d'utiliser un outil capable de révéler tous les détails granulaires des signatures et des signaux de confiance.

16.4. Examen public, commentaires et évolution

L'équipe qui rédige les recommandations UX est consciente de ses limites et de ses biais potentiels, et reconnaît que les commentaires, les examens, les tests utilisateurs et l'évolution continue sont des éléments essentiels à la réussite. Les recommandations seront donc un document évolutif, alimenté par des expériences concrètes de déploiement de l'UX C2PA dans une grande variété d'applications et de scénarios.

Chapitre 17. Sécurité de l'information

17.1. Menaces et considérations en matière de sécurité

Cette section fournit un résumé des considérations et des processus en matière de sécurité de l'information pour la technologie décrite dans la spécification de base C2PA. Des informations plus détaillées seront fournies dans les prochaines versions du matériel C2PA, y compris le document d'orientation.

17.1.1. Contexte

La sécurité de l'information est une préoccupation majeure de la C2PA. La C2PA maintient un modèle de menace et des considérations relatives à la sécurité pour la spécification C2PA. Cet effort complète d'autres travaux liés à la sécurité au sein de la C2PA. La documentation associée est actuellement en cours d'élaboration et peut être consultée à la rubrique Considérations relatives à la sécurité.

La C2PA élabore actuellement une documentation sur les considérations relatives à la sécurité qui comprend :

- Un résumé des fonctionnalités de sécurité pertinentes de la technologie C2PA
- Des considérations de sécurité pour l'utilisation pratique de la technologie C2PA
- Menaces pesant sur la technologie C2PA et traitement respectif de ces menaces, y compris les contre-mesures

17.1.2. Aperçu du processus de modélisation des menaces

La C2PA intègre la sécurité dans ses conceptions dès leur développement, mais prévoit également que la conception de la sécurité et la modélisation des menaces se poursuivront à mesure que le système, l'écosystème et le paysage des menaces évolueront.

À cette fin, la C2PA utilise un processus de modélisation des menaces ciblé pour soutenir le développement d'une conception solide en matière de sécurité et de confidentialité. Les résultats de ces efforts soutiennent directement l'élaboration d'une documentation explicite sur les menaces et les considérations de sécurité, mais facilitent également la réflexion sur la sécurité tout au long du processus de conception.

Le processus de modélisation des menaces combine des sessions synchrones (en direct) de modélisation des menaces, composées de groupes ciblés d'experts en la matière (SME), avec un développement asynchrone du contenu. Le nombre de participants à chaque session synchrone est limité afin de favoriser des discussions efficaces, mais tous les membres de la C2PA ont la possibilité de participer via l'une ou l'autre de ces modalités.

Comme pour les autres activités liées à la sécurité, nous nous attendons à ce que notre processus de modélisation des menaces évolue avec l'écosystème C2PA. La documentation du processus est considérée comme un guide plutôt que comme une directive stricte sur le fonctionnement de la modélisation des menaces au sein de la C2PA.

17.1.2.1. Références

Diverses références et expériences sont utilisées pour éclairer la modélisation des menaces et les activités de sécurité connexes pour la C2PA. Cette section fournit un sous-ensemble de documents publics à titre de référence.

- IETF sur les considérations de sécurité

- IETF sur les considérations relatives à la confidentialité (lignes directrices)
- Questionnaire d'auto-évaluation de la sécurité et de la confidentialité du W3C
- Modèle de menace OAuth2 (exemple)
- Modélisation des menaces : conception axée sur la sécurité
- Modélisation des menaces OWASP
- Modélisation des menaces Microsoft

17.2. Préjudices, utilisation abusive et détours

17.2.1. Introduction

Les principes directeurs de la C2PA stipulent que les spécifications de la C2PA doivent être examinées d'un œil critique afin de détecter tout abus ou utilisation abusive potentiel du cadre susceptible de causer des préjudices involontaires, de menacer les droits humains ou d'exposer des groupes vulnérables à des risques disproportionnés à l'échelle mondiale.

Afin de garantir que la C2PA respecte cet aspect de ses principes, l'évaluation des préjudices, des utilisations abusives et des détournements vise à identifier et à traiter les préoccupations potentielles lors de l'élaboration des spécifications et lors des mises en œuvre ultérieures.

De plus, les spécifications sont en cours de révision afin de :

- Anticiper et atténuer les abus et les utilisations abusives potentiels ;
- Répondre aux préoccupations courantes des utilisateurs en matière de confidentialité ; et
- Prendre en compte les besoins des utilisateurs et des parties prenantes à travers le monde.

17.2.2. Considérations

L'évaluation des préjudices, des utilisations abusives et des détournements est un processus continu. Les informations présentées dans la documentation sur la modélisation des préjudices ne doivent pas être considérées comme le résultat final d'une évaluation exhaustive, mais comme une base pour des discussions continues centrées sur les communautés touchées et visant à atténuer les abus et détournements potentiels et à protéger les droits humains.

Cette approche comporte deux aspects essentiels :

En cours

L'évaluation des dommages, des utilisations abusives et des détournements accompagne nécessairement les phases de conception et de développement, ainsi que celles de mise en œuvre et d'utilisation de la C2PA, en alimentant en permanence le processus d'élaboration des spécifications, les guides de mise en œuvre et d'expérience utilisateur, les efforts de sensibilisation, la gouvernance de la Coalition et, éventuellement, la coopération multilatérale pour la promotion d'un écosystème C2PA diversifié qui serve un large éventail de contextes mondiaux.

Multidisciplinaire et diversifié

L'évaluation des dommages, des utilisations abusives et des détournements doit être le fruit d'une collaboration entre des experts multidisciplinaires et un large éventail de parties prenantes ayant une expérience pratique et technique des questions abordées, issues de divers horizons géographiques, culturels et identitaires.

17.2.3. Évaluation

La modélisation des préjudices consiste à analyser comment un système socio-technique peut avoir un impact négatif sur les utilisateurs, les autres parties prenantes ou la société dans son ensemble, ou créer ou renforcer des structures d'injustice, des menaces pour les droits humains ou des risques disproportionnés pour les groupes vulnérables à l'échelle mondiale. Le processus de modélisation des dommages nécessite de combiner systématiquement les connaissances sur l'architecture d'un système et ses possibilités d'utilisation avec des données historiques et contextuelles sur l'impact de systèmes similaires existants sur différents groupes sociaux, ainsi qu'une consultation participative avec un éventail de communautés susceptibles d'être concernées par le système. Ces informations combinées permettent d'anticiper les dommages et d'identifier de manière proactive les réponses à apporter.

La documentation relative à la modélisation des dommages décrit le cadre et le processus mis en œuvre à ce jour, suivis de la méthodologie, d'un aperçu de l'évaluation, d'un aperçu de l'examen public et des commentaires, ainsi que des mesures de diligence raisonnable en cours d'élaboration pour accompagner la version 1.0 de ces spécifications, leur mise en œuvre et leur évolution.

17.2.4. Mesures de diligence raisonnable

L'évaluation des dommages, des utilisations abusives et des détournements a éclairé et devrait continuer à éclairer l'élaboration des spécifications techniques de la C2PA ainsi que la documentation qui l'accompagne :

- Guide à l'intention des responsables de la mise en œuvre
- Conseils sur l'expérience utilisateur
- Considérations en matière de sécurité
- Explications

En outre, l'évaluation des préjudices, des utilisations abusives et des détournements devrait éclairer la gouvernance de la Coalition et guider la coopération multilatérale potentielle en vue de promouvoir un écosystème C2PA diversifié qui favorise l'optimisation des avantages en termes de confiance dans les médias, de contrôle par les utilisateurs et de transparence qui ont motivé l'élaboration des spécifications C2PA.

Chapitre 18. Assertions standard C2PA

18.1. Introduction

Cette section du document répertorie l'ensemble standard d'assertions à utiliser pour les implémentations C2PA, en décrivant leur syntaxe, leur utilisation, etc. Pour simplifier, tous les exemples d'URI JUMBF ont été raccourcis à des fins d'illustration. Les URI complets sont nécessaires dans les données réelles.

Toutes les assertions doivent avoir une étiquette telle que décrite dans la section 6.2, « Étiquettes », et doivent être versionnées comme décrit dans le chapitre 5,
Gestion des versions.

Toutes les assertions normalisées C2PA utilisent le type de contenu JSON JUMBF, le type de contenu CBOR JUMBF ou le type de contenu Embedded File de la norme ISO 19566-5:2023. Les assertions spécifiques à une entité peuvent être n'importe laquelle de celles-ci, n'importe lequel des autres types de contenu JUMBF de la norme ISO 19566-5:2023, annexe B (tel que XML) ou peuvent créer leur propre type (conformément aux instructions de la norme ISO 19566-5:2023, tableau B.1). Le type de contenu Codestream ne doit pas être utilisé pour une assertion C2PA.

Sauf mention contraire, toutes les assertions documentées dans cet ensemble standard d'assertions doivent être sérialisées au format CBOR. Toutes les assertions sérialisées au format CBOR doivent être conformes aux exigences d'encodage déterministe de base du CBOR (voir RFC 8949, clause 4.2.1) et leurs schémas doivent être définis à l'aide d'une définition CDDL.

REMARQUE Toutes les CDDL sont considérées comme non normatives.

Pour ceux définis à l'aide de JSON, leurs schémas doivent être définis à l'aide de la dernière version de JSON Schema.

18.2. Régions d'intérêt

18.2.1. Description

Dans certains cas d'utilisation, une assertion donnée, telle qu'une assertion d'action, peut ne concerner qu'une partie spécifique d'un actif plutôt que l'ensemble de celui-ci. Dans ces cas, il est nécessaire de disposer d'un moyen de décrire cette région, qu'elle soit temporelle, spatiale, textuelle ou une combinaison de ces éléments. Une définition `de région` sert cet objectif.

18.2.2. Commun

La partie la plus importante de la définition `de la région` est le champ `« range » (plage)` qui sert à décrire une plage temporelle, une plage spatiale, une plage d'images, une plage textuelle ou une combinaison de celles-ci pour la région.

NOTE Bien que la spécification permette de spécifier une combinaison de plages, elle ne définit pas comment un manifeste Les consommateurs les utiliseront. Le groupe de travail sur l'expérience utilisateur de la C2PA devrait se pencher sur cette question à l'avenir.

Une `région` peut également contenir un ou plusieurs champs communs :

nom

chaîne de texte libre représentant un nom lisible par l'utilisateur pour la région, qui pourrait être utilisé dans une interface utilisateur.

identifiant

chaîne de texte libre représentant un identifiant lisible par machine, unique à cette assertion, pour la région.

type

une valeur issue d'un vocabulaire contrôlé tel que https://cv.iptc.org/newscodes/imageregiontype/ ou une valeur spécifique à une entité (par exemple, `com.litware.newType`) qui représente le type d'élément(s) représenté(s) par une région.

description

chaîne de texte libre.

Les anciennes versions de cette spécification comprenaient un champ « `rôle` ». Ce champ est désormais obsolète et ne doit plus être inclus lors de la génération d'une région d'intérêt.

18.2.2.1. Plages

Toutes les plages se composent d'un champ `type` dont la valeur est « spatial », « temporel », « cadre », « textuel » ou « identifié ». En outre, elles doivent contenir l'un des champs suivants dont les données sont un objet composé des données spécifiques à cette plage :

- `shape` (pour spatial) ;
- `time` (pour temporal) ;
- `frame` (pour temporel ou textuel) ;
- `text` (pour textuel) ;
- `item` (pour les éléments spécifiquement identifiés).

18.2.2.2. Spatial

Les plages spatiales sont décrites à l'aide d'un objet `de forme`. Une `forme` peut être utilisée pour représenter un rectangle, un cercle ou un polygone. Elle est modélisée à partir de la structure des limites régionales de l'IPTC.

18.2.2.3. Temporel

Les plages temporelles sont décrites à l'aide d'un objet `temporel`, qui représente une plage allant d'une heure de début à une heure de fin. Les heures sont décrites soit à l'aide du temps de lecture normal (`npt`) tel que décrit dans la RFC 2326 (comme recommandé dans la spécification W3C Media Fragments), soit à l'aide d'une « heure d'horloge murale » utilisant le profil Internet de la norme ISO 8601 tel que décrit dans la RFC 3339.

REMARQUE	Le « Wall Clock Time » est utile dans les cas où le média représente une activité qui s'est déroulée à une date et une heure spécifiques, comme un journal télévisé ou un événement en direct.

Si aucun champ `de type` n'est fourni, la plage est supposée être au format `npt`. Si aucun champ `de début` n'est fourni, la plage commence au début de la ressource. Si aucun champ `de fin` n'est fourni, la plage se termine à la fin de la ressource. Si aucun des deux n'est

fournis, la plage représente l'intégralité de l'actif.

18.2.2.4. Trames

Un objet `frame` définit une plage à l'aide des images ou des pages de début et de fin (incluses). Si aucun `début` n'est fourni, la plage commence au début de la ressource. Si aucune `fin` n'est fournie, la plage se termine à la fin de la ressource. Si aucun des deux n'est fourni, la plage représente l'intégralité de la ressource.

Les images sont représentées par des nombres ordinaux uniques, où `0` correspond à la première image.

Bien que les cadres soient généralement utilisés pour représenter les numéros de page d'un document, tel qu'un PDF, ils peuvent également être utilisés dans d'autres types de médias, tels que les animations, les vidéos et les fichiers audio. Il est recommandé, dans la mesure du possible, d'utiliser plutôt des plages `temporelles` pour les types de médias traitant de régions d'intérêt au fil du temps.

18.2.2.5. Textuel

Un objet `texte` définit une plage à l'aide d'un ou plusieurs identifiants de fragment d'URL, tels que définis par le sélecteur de fragment d'annotation Web du W3C. Il peut également affiner la plage à l'aide de décalages par rapport aux caractères de début et de fin (inclus). Si aucun `début` n'est fourni, la plage commence au début du fragment. Si aucune `fin` n'est fournie, la plage se termine à la fin du fragment. Si aucun des deux n'est fourni, la plage représente l'intégralité du fragment.

Lorsqu'il est utilisé seul, l'entrée `fragment` du `text-selector-map` représente l'intégralité de la plage textuelle spécifiée. Cependant, le `text-selector-range-map` prend en charge une paire d'objets `text-selector-map`. La valeur de `selector` correspond au début de la plage (ou à l'intégralité de celle-ci, si aucune entrée `end` n'est présente) et la valeur de `end` (si présente) représente la fin d'une plage contiguë. En outre, plusieurs paires peuvent être utilisées pour représenter une plage qui n'est pas contiguë.

18.2.2.6. Identifié

Un objet item définit une piste multimédia, un élément multimédia ou tout autre élément de contenu discret dans la ressource, permettant au générateur de revendications d'indiquer les assertions qui s'appliquent uniquement à un sous-ensemble du contenu transporté dans le conteneur de fichiers de la ressource. Par exemple, il pourrait être utilisé pour indiquer que seule la piste audio d'un fichier vidéo est pertinente.

Le média ou l'élément de contenu est identifié par une chaîne `d'identifiant` dont la valeur doit correspondre au schéma de nommage d'identification d'élément typique dans ce format de conteneur spécifique. Par exemple, la valeur de `l'identifiant` doit être `track_id` pour les fichiers MP4 et `item_ID` pour les fichiers HEIF. La valeur de `l'identifiant` est ensuite fournie dans le champ `value`. Par exemple, une `valeur` de 2 avec un `identifiant` `track_id` dans un conteneur de fichier vidéo MP4 indiquerait une assertion liée à la deuxième piste multimédia du fichier (qui pourrait être la piste audio).

Une autre utilisation des plages identifiées consiste à indiquer une région spécifique par une valeur sémantique connue. Par exemple, le modèle fondamental d'anatomie pourrait être utilisé pour identifier une région spécifique du corps humain. Dans ce cas, `l'identifiant` doit être l'URL ou l'URI où se trouve le schéma (mais pas nécessairement directement lisible par une machine).

18.2.3. Schéma

Le schéma de ce type est défini par la règle `region-map` dans la définition CDDL suivante :

```
region-map = {
     « region » :        [1* $range-map],             ; définition de la plage, une ou plusieurs plages
     ? « name » : tstr .size (1..max-tstr-length), ; chaîne de texte libre représentant un nom lisible par
l'utilisateur pour la région, qui pourrait être utilisé dans une interface utilisateur.
     ? « identifier » : tstr .size (1..max-tstr-length), ; chaîne de texte libre représentant un identifiant
lisible par machine, unique à cette assertion, pour la région.
     ? « type » :   tstr .size (1..max-tstr-length),           ; provenant d'une liste contrôlée
     ? « role » :   $role-choice,       ; OBSOLÈTE
     ? « description » : tstr .size (1..max-tstr-length), ; description lisible par l'utilisateur de la région
     ? « metadata » : $assertion-metadata-map, ; informations supplémentaires sur l'assertion
}

$range-choice /= « spatial »           ; une plage identifiée par une zone physique
$range-choice /= « temporal »          ; une plage identifiée par une période
$range-choice /= « frame »             ; une plage identifiée par une série d'images ou de pages
$range-choice /= « textuel »           ; une plage identifiée par une plage de texte
$range-choice /= « identifié »         ; une plage identifiée par un identifiant et une valeur spécifiques

range-map = {
     « type » :     $range-choice,          ; soit « spatial », « temporel », « cadre », « textuel » ou «
identifié »
     ? « shape » :  $shape-map,              ; description de la forme d'une plage spatiale
     ? « time » :   $time-map,               ; description des limites temporelles d'une plage temporelle
     ? « frame » :  $frame-map,              ; description des limites de la trame d'une plage temporelle

     ? « texte » :  $text-map,              ; description des limites d'une plage textuelle
     ? « item » :   $item-map,              ; description des limites d'une plage identifiée
}

coordinate-map = {
     « x » : float,      ; coordonnée le long de l'axe x « y
     » : float,          ; coordonnée le long de l'axe y
}

$choix-forme /= « rectangle »          ; une forme rectangulaire
$shape-choice /= « circle »            ; une forme circulaire
$shape-choice /= « polygone »          ; une forme polygonale

$unit-choice /= « pixel »         ; les unités sont en pixels
$unit-choice /= « pourcentage »   ; les unités sont exprimées en pourcentage de la taille totale


shape-map = {
     « type » :     $choix de forme,             ; soit « rectangle », « cercle » ou « polygone » «
     unité » :      $unit-choice,                ; soit « pixel », soit « pourcentage »
     « origine » :  $coordinate-map,             ; coordonnée de départ/origine de la forme.
     ? « width » : nombre à virgule flottante,   ; largeur pour les rectangles, diamètre pour les
cercles (ignoré pour les polygones)
     ? « hauteur » : nombre à virgule flottante  ; hauteur pour les rectangles
     ? « inside » : booléen,                     ; à l'intérieur ou à l'extérieur de la forme, la valeur par défaut
     est « true »
     ? « vertices » :    [1* $coordinate-map]          ; points/sommets restants du polygone
}

; les heures de début et de fin npt et utc ont des formats d'expression
régulière différents time-map = npt-time-map / wall-clock-time-map
```

```
npt-time-map = {
     ? « type » :   « npt » ; si absent, supposer une carte temporelle « npt »
     ? « start » : tstr .regexp « ^(?:(?:([01]?\d|2[0-3]):)?([0-5]?\d):)?([0-
5]?\d)(\.(\d{1,9}))?$ »,         ; heure de début (ou début de l'actif si non présent).
     ? « end » :                 ; heure de fin (ou fin de l'actif si elle n'est pas
présente).
}

wall-clock-time-map = {
     « type » :     "wallClock" ; doit être présent pour la carte temporelle "wall-clock"
     ? « start » : tstr .regexp « ^(\d{4})-(\d{2})-(\d{2})T(\d{2}):(\d{2}):(\d{2})(\.\d+)?(([+-
]\d{2}:\d{2})|Z)$", ; heure de début (ou début de l'actif si non présent).
     ? « end » :
]\d{2}:\d{2})|Z) », ; heure de fin (ou fin de l'actif/bord actif s'il n'est pas présent).
}

; ceci peut être utilisé soit pour les images d'une vidéo, soit pour les pages d'un
document frame-map = {
     ? « start » : int,       ; image de départ (ou début de l'actif s'il n'est pas présent).
     ? « fin » :    int       ; image de fin (ou fin de la ressource si elle n'est pas présente).
}

; ceci est calqué sur le modèle de sélecteur d'annotation Web du W3C
text-selector-map = {
     « fragment » :      tstr,     ; identifiant de fragment, conformément à la norme RFC3023 ou ISO 32000-2, annexe
     O
     ? « début » :       int,      ; décalage du caractère de début (ou début du fragment s'il n'est pas
présent).
     ? « fin » :         int       ; décalage du caractère de fin (ou fin du fragment s'il n'est pas présent).
}

; une ou deux cartes de sélection de texte utilisées pour identifier
la plage text-selector-range-map = {
     « sélecteur » :     $text-selector-map,           ; sélecteur de texte de début (ou unique)
     ? « end » :         $text-selector-map            ; s'il est présent, représente la fin de la plage de
texte
}

text-map = {
     « sélecteurs » : [1* $text-selector-range-map] ; tableau de plages de texte (éventuellement discontinues)
}

item-map = {
     « identifier » :    tstr .size (1..max-tstr-length),           ; terme spécifique au conteneur utilisé
pour identifier les éléments, tel que « track_id » pour MP4 ou « item_ID » pour HEIF
     « value » :         tstr .size (1..max-tstr-length),           ; la valeur de l'identifiant, par exemple
une valeur de « 2 » pour un identifiant « track_id » impliquerait la piste 2 de la ressource
}

; Ces valeurs sont obsolètes
$role-choice /= « c2pa.areaOfInterest » ; zone arbitraire méritant d'être identifiée
$role-choice /= « c2pa.cropped »               ; cette zone est tout ce qui reste après une action de recadrage
$role-choice /= « c2pa.edited »                ; cette zone a fait l'objet de modifications
$role-choice /= « c2pa.placed »                ; la zone où un ingrédient a été placé/ajouté
$role-choice /= « c2pa.redacted »              ; quelque chose dans cette zone a été expurgé
$role-choice /= « c2pa.subjectArea »           ; zone spécifique à un sujet (humain ou non)
$role-choice /= « c2pa.deleted »               ; une série d'informations a été supprimée/effacée
$role-choice /= « c2pa.styled »                ; un style a été appliqué à cette zone
$role-choice /= « c2pa.watermarked »           ; un filigrane a été appliqué à cette zone à des fins de
reliure souple
```

18.2.4. Exemples

Une série d'exemples en notation diagnostique CBOR (RFC 8949, clause 8) est présentée ci-dessous :

```
// exemple d'une plage temporelle et spatiale combinée dans une vidéo //
{
  « region » : [
    {
      « type » : « temporel », «
      temps » : {
        « type » : « npt »,
        « début » : « 0 »,
        « fin » : « 5.2 »
      }
    },
    {
      « type » : « spatial », «
      shape » : {
        « type » : « rectangle »,
        « unité » : « pixel »,
        « origine » : {
          « x » : 10,0,
          « y » : 10,0
        },
        « width » : 200,0,
        « hauteur » : 112,0
      }
    }
  ],
  « nom » : « Logo animé », « identifiant » :
  « logo-clip »,
  « description » : « 5,2 secondes du logo animé d'ouverture, dans un rectangle situé à 10 pixels en dessous du
haut et à gauche, 200 px par 112 px »
}

// exemple d'une plage textuelle dans un fichier Word/DOCX //
{
  « region » : [
    {
      « type » : « textuel », «
      texte » : {
        « selectors » : [
          [
            {
              « fragment » : « xpointer(/w:document/w:body/w:p/) »
            }
          ]
        ]
      }
    },
  ],
  « description » : « Contenu édité par l'assistant IA »
}

// exemple d'une plage textuelle dans un fichier PDF balisé //
{
  « region » : [
    {
      « type » : « textuel », «
      texte » : {
        « selectors » : [
```

```
            [
              {
                « sélecteur » : {
                  « fragment » : « path=/Document/Sect[0]/P[3] », « start
                  » : 10,
                  « fin » : 20
                }
              }
            ]
          ]
        }
      },
  ],
  « description » : « Rédaction effectuée conformément à la demande FOIA »
}

// exemple d'une plage textuelle dans un fichier PDF non balisé //
// dans ce cas, nous ne pouvons spécifier qu'une page et une zone rectangulaire //
{
  « region » : [
    {
      « type » : « textuel », «
      texte » : {
        « selectors » : [
          [
            {
              « sélecteur » : {
                « fragment » : « page=1,rect=10,10,450,500 », « start »
                : 10,
                « end » : 20
              }
            }
          ]
        ]
      }
    },
  ],
  « description » : « Rédaction effectuée conformément à la demande FOIA »
}

// exemple de suppression de certaines pages d'un PDF //
{
  « region » : [
    {
      « type » : « frame », «
      frame » : {
        « start » : 27,
        « fin » : 30
      }
    },
  ],
  « description » : « pages inutiles supprimées avant la distribution »
}

// exemple d'une plage de cellules dans Excel //
{
  « region » : [
    {
      « type » : « textuel », «
      texte » : {
        « selectors » : [
          [
            {
```

```
                « sélecteur » : {
                  « fragment » : « xpointer(Sheet1!A5:A10) »,
                }
              }
            ],
            [
              {
                « sélecteur » : {
                  « fragment » : « xpointer(Sheet1!B5:B10) »,
                }
              }
            ]
          ]
        }
      },
  ],
  « description » : « application d'un style à une plage de cellules dans Excel »
}

// exemple d'une plage contiguë de cellules de tableau //
{
  « region » : [
    {
      « type » : « textuel », «
      texte » : {
        « sélecteurs » : [
          [
            {
              « sélecteur » : {
                « fragment » : « xpointer(//table[1]/tr[1]/td[2]) »,
              },
              « end » : {
                « fragment » : « xpointer(//table[1]/tr[1]/td[4]) »,
              }
            }
          ]
            ]
      }
    },
  ],
  « description » : « a effacé certaines cellules du tableau »
}

// exemple d'une plage d'une piste spécifique d'une vidéo //
{
  « region » : [
    {
      « type » : « temporel », «
      temps » : {
        "type": "npt",
        « début » : « 0 »,
        « fin » : « 5.2 »
      }
    },
    {
      « type » : « identifié »,
      « élément » : {
        « identifiant » : « track_id », «
        valeur » : « 2 »
      }
    }
  ],
  « description » : « amélioration de certaines pistes audio »
```

```
}


// exemple indiquant que les yeux ont été modifiés dans l'ensemble de la ressource //
{
  « region » : [
    {
      « type » : « temporel »,
      « time » : {},
    },
    {
      « type » : « identified », « item
      » : {
        « identifiant » : « https://bioportal.bioontology.org/ontologies/FMA », « valeur » :
        « ensemble d'yeux »
      },
    }
  ]
  « description » : « s'est assuré qu'il avait l'air de dormir pendant toute la réunion »
}
```

18.3. Métadonnées sur les assertions

18.3.1. Description

Dans de nombreux cas, il est utile, voire nécessaire, de fournir des informations supplémentaires sur une assertion, telles que la date et l'heure auxquelles elle a été générée ou d'autres données susceptibles d'aider les consommateurs de manifestes à prendre des décisions éclairées sur la provenance ou la véracité des données d'assertion.

REMARQUE

Un consommateur de manifeste n'est pas tenu de lire une partie quelconque des métadonnées d'assertion. Il peut choisir lesquelles, le cas échéant, les champs qu'il souhaite utiliser, en fonction du type d'assertion auquel il est appliqué.

Vous trouverez ci-dessous les schémas de base utilisés dans d'autres assertions.

La définition CDDL pour la règle `assertion-metadata-map` se trouve dans CDDL pour les métadonnées d'assertion :

CDDL pour les métadonnées d'assertion

```
;Décrit des informations supplémentaires sur une assertion, y compris une référence hashed-uri vers celle-ci.
Nous utilisons ici un socket/plug pour permettre à hashed-uri-map d'être utilisé dans des fichiers individuels
sans que la carte soit définie dans le même fichier
$assertion-metadata-map /= {
  ? « dateTime » : tdate, ; La chaîne date-heure RFC 3339 lorsque l'assertion a été créée/générée
  ? « reviewRatings » : [1* rating-map], ; Notes attribuées à l'assertion (peut être vide)
  ? « reference » : $hashed-uri-map, ; référence hashed_uri à une autre assertion concernée par cette
révision
  ? « dataSource » : source-map, ; Description de la source des données d'assertion, sélectionnée dans une liste
prédéfinie
  ? « localizations » : [1* localization-data-entry] ; localisations pour les chaînes dans l'assertion
  ? « regionOfInterest » : $region-map ; décrit une région de l'actif où cette assertion est pertinente
```

```
}

$source-type /= « signer »
$source-type /= « claimGenerator.REE »
$source-type /= « claimGenerator.TEE »
$source-type /= « localProvider.REE »
$source-type /= « localProvider.TEE »
$source-type /= « remoteProvider.1stParty »
$source-type /= « remoteProvider.3rdParty »
$source-type /= « humanEntry »
; les deux valeurs suivantes de source-type sont obsolètes depuis la version 2.0
$source-type /= « humanEntry.anonymous »
$source-type /= « humanEntry.identified »


; REMARQUE : une version antérieure de cette spécification comprenait également un champ « actors », mais
celui-ci a été supprimé dans la version 2.0.
source-map = {
  « type » : $source-type, ; Une valeur parmi la liste énumérée indiquant si la source de l'assertion est un
générateur de revendications fonctionnant dans un environnement d'exécution riche (REE), un générateur de
revendications fonctionnant dans un environnement d'exécution fiable (TEE), un fournisseur de données local
dans REE (par exemple, l'API de localisation d'un système d'exploitation mobile), une donnée locale
s'exécutant dans un TEE (par exemple, une application de localisation fiable d'un fournisseur de puces), un
fournisseur de données distant tel qu'un serveur (par exemple, le service API de géolocalisation de Google) ou
une entrée effectuée par un humain.
  ? « details » : tstr .size (1..max-tstr-length), ; Chaîne lisible par l'utilisateur fournissant des détails
sur la source des données d'assertion, par exemple l'URL du serveur distant qui a fourni les données
}

int-range = 1..5

$review-code /= « actions.unknownActionsPerformed »
$review-code /= « actions.missing »
$review-code /= « actions.possiblyMissing »
$review-code /= « depthMap.sceneMismatch »
$review-code /= « ingredient.modified »
$review-code /= « ingredient.possiblyModified »
$review-code /= « thumbnail.primaryMismatch »

; les trois valeurs suivantes de review-code sont obsolètes depuis la version 2.0
$review-code /= « stds.iptc.location.inaccurate »
$review-code /= « stds.schema-org.CreativeWork.misattributed »
$review-code /= « stds.schema-org.CreativeWork.missingAttribution »

rating-map = {
  « valeur » : plage d'entiers, ; « Une valeur comprise entre 1 (la pire) et 5 (la meilleure) pour la note
  attribuée à l'élément »
  ? « code » : $review-code, ; Chaîne au format étiquette décrivant la raison de la note
  ? « explication » : tstr .size (1..max-tstr-length), ; Chaîne lisible par l'utilisateur expliquant pourquoi la
note est telle qu'elle est
}

; Structures de données utilisées pour stocker les dictionnaires de localisation
$localization-data-entry /= {
  * $$chaîne-langue
}

chaîne-de-langue /= tstr .size (1..max-tstr-length)
```

Exemple en notation diagnostique CBOR (RFC 8949, clause 8) :

```
{
  « reference » : {
       "url": "self#jumbf=c2pa.assertions/c2pa.metadata", "alg":
       "sha256",

    },
  « source_donnée » : {
    « type » : « localProvider.REE »,
    « details » : « Données GPS EXIF fournies par l'API de géolocalisation du système d'exploitation »
  }
}
```

Dans la plupart des cas, ces métadonnées spécifiques à l'assertion apparaîtront directement dans d'autres assertions (par exemple, les ingrédients) en tant que valeur de leur champ `de métadonnées`. Cependant, il est parfois nécessaire ou souhaitable de stocker les métadonnées de l'assertion dans une assertion de métadonnées distincte et indépendante, par exemple lorsqu'une assertion n'est pas au format JSON ou CBOR, comme les vignettes.

Le libellé de l'assertion de métadonnées d'assertion est `c2pa.assertion.metadata`.

18.3.2. Source des données

Ce champ `dataSource` est un champ facultatif qui permet au générateur de revendications d'informer les consommateurs de manifestes en aval de la source d'où provient le contenu de l'assertion. Si aucun `dataSource` n'est fourni pour une assertion donnée, le `dataSource` est considéré comme étant le `signataire`.

REMARQUE Par défaut, toutes `les assertions créées` sont attribuées au signataire, car le modèle de confiance repose sur la confiance accordée au signataire, qui est généralement aussi le générateur de revendications.

La valeur du champ est un objet `dataSource` composé de deux champs : `type` et `details`.

Le champ dataSource `type` définit le type de la source de données. Il est assemblé avec des étiquettes au format décrit dans la section 6.2, « Étiquettes ». La valeur peut être l'une des valeurs définies par la spécification dans le tableau 5, « Types de sources de données », ou des espaces de noms spécifiques à l'entité peuvent être utilisés comme mécanisme d'extension.

Tableau 5. Types de sources de données

Valeur du `type`	**Signification**
`signataire`	Le contenu de l'assertion provient du signataire
`claimGenerator.REE`	Le contenu de l'assertion provient d'un générateur de revendications fonctionnant dans un environnement d'exécution riche (REE), tel qu'un système d'exploitation de bureau ou mobile
`claimGenerator.TEE`	Le contenu de l'assertion provient d'un générateur de revendications fonctionnant dans un environnement d'exécution fiable (TEE), tel qu'un système d'exploitation fiable
`localProvider.REE`	Le contenu de l'assertion provenait d'une source de données s'exécutant dans un REE sur le même dispositif informatique physique que le générateur de revendications.
Valeur de `type`	**Signification**

localProvider.TEE	Le contenu de l'assertion provient d'une source de données s'exécutant dans un TEE sur le même dispositif informatique physique que le générateur de revendications.
remoteProvider	Le contenu de l'assertion provient d'une source de données distante contrôlée par le signataire ou le fournisseur du générateur de revendications.
remoteProvider.external	Le contenu de l'assertion provient d'une source de données externe et distante qui n'est pas le signataire ou le fournisseur du générateur de revendications
humanEntry	Le contenu de l'assertion a été saisi par un être humain

Le champ « details » est une chaîne lisible par l'homme qui fournit des informations supplémentaires sur la source de données, par exemple le nom de l'API utilisée pour fournir le contenu de l'assertion ou l'URL du serveur à partir duquel le contenu a été fourni. Par exemple, une source d'assertion de localisation générale peut avoir une valeur de type « remoteProvider.3rdParty », avec la valeur « details » définie sur «www.googleapis.com/geolocation/v1/geolocate ».

18.3.3. Notes d'évaluation

Lorsqu'il est présent, le tableau reviewRatings permet au générateur de revendications de fournir un ou plusieurs objets d'évaluation sur la qualité (ou l'absence de qualité) d'une assertion. Un reviewRatings ne doit pas être présent si un objet dataSource est présent avec un champ type dont la valeur est soit humanEntry.anonymous, soit humanEntry.credentialed.

Le champ « valeur » de l'objet « notation » doit contenir une valeur entière comprise entre *1* (la plus mauvaise) et *5* (la meilleure). S'il est présent, le champ « explication » doit contenir une description sous forme de chaîne de caractères lisible par l'utilisateur du type de notation. En outre, un champ « code » lisible par machine, qui définit les codes de résultat d'évaluation spécifiques à l'assertion, peut être fourni en option. La valeur du champ code est définie en utilisant le même format que celui décrit à la section 6 .2, « Étiquettes ». La valeur peut être l'une des valeurs définies par la spécification dans le tableau 6, « Valeurs du champ code », ou des espaces de noms spécifiques à l'entité peuvent être utilisés comme mécanisme d'extension.

Tableau 6. Valeurs du champ de code

Valeur du code	**Assertion applicable**	**Signification**
actions.unknownActionsPerformed	c2pa.actions	L'assertion actions ne contient pas la liste complète de toutes les actions effectuées dans l'outil de création (par exemple, en raison de l'utilisation d'un filtre tiers dont l'effet est inconnu de l'outil de création).
actions.placedIngredientNotFound	c2pa.actions	L'assertion d'actions examinée comporte une action placée sans URI d'ingrédient résolvable. La valeur doit être 1.
ingredient.actionManquant	c2pa.ingredient	L'assertion d'ingrédient examinée ne comporte pas au moins une action qui y fait référence dans sa revendication. La valeur doit être 1.
ingredient.notVisible	c2pa.ingredient	L'affirmation relative à l'ingrédient en cours d'examen n'est pas visible dans le contenu numérique lié à ce manifeste. La valeur doit être 1.
Valeur du code	**Affirmation applicable**	**Signification**
depthMap.sceneMismatch	c2pa.depthmap.GDe pth	Le contenu de l'assertion de la carte de profondeur ne correspond pas à la scène représentée dans la présentation principale de la ressource (par exemple, en raison d'une attaque par image dans l'image).

thumbnail.primary Mismatch	c2pa.thumbnail.cl aim	Le contenu de la vignette ne correspond pas au contenu de la présentation principale dans la ressource.

18.3.4. Références

Étant donné que le champ de référence de l'assertion de métadonnées d'assertion est un hashed_uri standard, il est également possible qu'une assertion de métadonnées d'assertion fasse référence à des assertions dans d'autres manifestes que celui qui est actif. Par exemple, le manifeste actif peut inclure une assertion de métadonnées d'assertion qui valide l'assertion c2pa.metadata présente dans le manifeste d'un ingrédient.

REMARQUE Étant donné que la revendication est un type particulier d'assertion, cette même méthode peut être utilisée pour faire référence à des revendications dans d'autres manifestes.

18.3.5. DateTime

Si un champ dateTime est présent, sa valeur doit être une chaîne de caractères représentant une date et une heure conforme aux dates/heures CBOR (RFC 8949, 3.4.1).

18.3.6. Région d'intérêt

L'affirmation peut être spécifique à une partie seulement d'un actif, telle qu'une série d'images dans une vidéo ou une zone spécifique d'une image. Une telle partie peut être identifiée à l'aide d'un champ regionOfInterest, dont la valeur est un objet region-map (tel que défini dans la section 18.2, « Régions d'intérêt »).

18.3.7. Localisation

18.3.7.1. Général

Il est important que les consommateurs de manifestes C2PA puissent comprendre les informations dans leur langue maternelle, dans la mesure du possible. À cette fin, il est possible d'ajouter des informations de localisation pour une assertion à l'aide d'un dictionnaire inclus dans les métadonnées de l'assertion.

18.3.7.2. Dictionnaire de localisation

Un dictionnaire de localisation doit être constitué d'un seul objet, dont chacune des clés représente les traductions à l'aide de la technique d'indexation linguistique de JSON-LD. Si la valeur à traduire n'est pas associée à une clé de niveau supérieur, la « notation par points » (.) doit être utilisée pour référencer les clés imbriquées dans les objets. La notation d'indexation de tableau ([n], n>=0) doit être utilisée lorsqu'un élément spécifique d'un tableau doit être parcouru. Lorsque la valeur à traduire est elle-même un tableau, un élément spécifique peut être référencé. Quelques exemples sont présentés dans l'exemple 4, « Exemples de dictionnaires de localisation

Dictionnaires » :

Exemple 4. Exemples de dictionnaires de localisation

```
{
  « dc:title » : {
    « en-US » : « Les cinq chats de Kevin »,
    « en-GB » : « Les cinq chats de Lord Kevin »,
    « es-MX » : « Los Cinco Gatos de Kevin », «
    es-ES » : « Los Thinco Gatos de Kevin », « fr
    » : « Les Cinq Chats de Kevin »,
    « jp » : « ケヴィンの５匹の猫 »
  }
}
```

```
{
  « actions[0].softwareAgent » : { « en-
    US » : « Joe's Photo Editor », « en-
    GB » : « Joe's Photo Editor »,
    « es » : « Editor de fotos de Joe », « fr
    » : « L'éditeur de photos de Joe »,
    « jp » : « ジョーの写真編集者 »
  }
}
```

Toutes ces clés ou valeurs tierces doivent être nommées de la même manière que dans la section 6.2.1, « Espaces de noms », par exemple `com.litware`. Afin qu'un consommateur de manifeste puisse afficher des informations lisibles par l'utilisateur sur ces clés et valeurs, le générateur de revendications doit fournir les chaînes via cette approche de localisation.

Les actions localisées montrent son utilisation dans la localisation d'actions personnalisées, en l'utilisant dans les métadonnées d'assertion d'un Assertion `c2pa.actions`.

Actions localisées

```
{
  « com.litware.blur » : {
    « en-US » : « Blur »,
    "fr-FR": "Brouiller",
  },
  « com.litware.filter » : { «
    en-US » : « Filter »,
    "es-ES": "Filtrar",
    « jp-JP » : « フィルター »
  }
}
```

18.4. Résumé des assertions C2PA standard

Les assertions C2PA standard sont répertoriées dans le tableau 7, « Assertions C2PA standard » :

Tableau 7. Affirmations C2PA standard

Type	Assertion	Schéma	Sérialisation
Actions	c2pa.actions, c2pa.actions.v2	C2PA	CBOR
Assertion Métadonnées	c2pa.assertion.metadata	C2PA	CBOR
Référence d'actif	c2pa.asset-ref	C2PA	CBOR
Type d'actif	c2pa.asset-type (obsolète), c2pa.asset-type.v2	C2PA	CBOR
Hachage basé sur BMFF	c2pa.hash.bmff (supprimé), c2pa.hash.bmff.v2 (obsolète), c2pa.hash.bmff.v3	C2PA	CBOR
Statut du certificat	c2pa.certificate-status	C2PA	CBOR
Données cloud	c2pa.cloud-data	C2PA	CBOR
Hachage des données de collecte	c2pa.hash.collection.data	C2PA	CBOR
Hachage des données	c2pa.hash.data	C2PA	CBOR
Carte de profondeur	c2pa.depthmap.GDepth	https://developers.google.com/depthmap-metadata/reference	CBOR
Données intégrées	c2pa.embedded-data	C2PA	JUMBF Fichier intégré
Informations sur les polices	font.info	C2PA	CBOR
Hachage général de la boîte	c2pa.hash.boxes	C2PA	CBOR
Ingrédient	c2pa.ingredient, c2pa.ingredient.v2, c2pa.ingredient.v3	C2PA	Fichier intégré JUMBF
Métadonnées	c2pa.metadata	C2PA	JSON-LD
Hachage multi-actifs	c2pa.hash.multi-asset	C2PA	CBOR
Liaison souple	c2pa.soft-binding	C2PA	CBOR
Vignette	c2pa.thumbnail.claim (heure de création de la revendication), c2pa.thumbnail.ingredient (importation d'un ingrédient)	C2PA	Fichier intégré
Horodatage	c2pa.time-stamp	C2PA	CBOR

18.5. Hachage de données

18.5.1. Description

La manière la plus courante de vérifier de manière unique l'intégrité de parties d'un actif non basé sur BMFF consiste à utiliser les liaisons fixes (c'est-à-dire le hachage cryptographique) présentes dans les assertions de hachage de données. Cependant, pour les formats de type « boîte » mais non compatibles avec BMFF, l'assertion de hachage de boîte générale est recommandée.

L'assertion de hachage des données prend en charge la création et le stockage des hachages comme décrit dans la section 13.1, « Hachage », et la valeur doit être présente dans le champ `de hachage`.

Chaque assertion de hachage de données définit une plage spécifiée d'octets sur laquelle le hachage a été calculé. Si seule une partie de l'actif doit être hachée, la ou les plages à exclure doivent être présentes dans la valeur du tableau du champ `des exclusions`. Ces plages exclues doivent être classées par ordre croissant de position de départ et ne doivent pas se chevaucher.

Pour les plages d'exclusion de hachage de données, la plage doit commencer et se terminer dans la même unité logique (par exemple, boîte, segment, objet) et ne doit pas chevaucher les champs d'en-tête ou de longueur associés à cette unité, à l'exception des données freebox ou pad. Il incombe au générateur de revendications de définir les plages d'exclusion de manière à garantir que les données qu'un attaquant pourrait placer dans ces plages ne puissent pas affecter de manière significative l'interprétation de l'actif. En outre, le générateur de revendications doit s'assurer que la plage d'exclusion ne contient que du contenu provenant du magasin de manifestes C2PA ou des métadonnées d'actifs (par exemple, métadonnées EXIF, IPTC). Les métadonnées qui pourraient être ignorées sont, par exemple, les noms d'utilisateur non vérifiés ou les informations de rotation des images.

Une version précédente de cette spécification fournissait un champ `url` pour indiquer l'emplacement des données hachées, mais celui-ci n'a jamais été utilisé. Ce champ est désormais obsolète au profit de l'assertion de référence d'actif. Les générateurs de revendications ne doivent pas ajouter ce champ à une assertion de hachage de données, et les consommateurs doivent ignorer ce champ lorsqu'il est présent, sauf si cela n'affecte pas l'inclusion du champ dans le contenu validé, comme décrit dans la section 15.10.3, « Validation des assertions ».

Une assertion de hachage de données doit avoir pour étiquette `c2pa.hash.data`. Une assertion de hachage de données ne doit pas apparaître dans une assertion de données cloud.

Une assertion de hachage de données ne doit pas être utilisée avec un manifeste compressé.

> **REMARQUE** Cette restriction existe pour pallier une incompatibilité technique entre les deux.

18.5.2. Schéma et exemple

Le schéma de ce type est défini par la règle `data-hash-map` dans la définition CDDL suivante :

```
; Vérifiez également l'optionalité dans le hash-map
; Structure de données utilisée pour stocker le hachage cryptographique de tout ou partie des données de l'actif

; et les informations supplémentaires nécessaires au calcul du
hachage. data-hash-map = {
   ? « exclusions » : [1* EXCLUSION_RANGE-map], ; Les plages ont des valeurs « start » croissantes de manière
monotone, et deux plages ne peuvent pas se chevaucher.
   ? « alg » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme de hachage cryptographique
utilisé pour calculer le hachage dans cette assertion, tirée de la liste des identifiants d'algorithmes de
hachage C2PA. Si ce champ est absent, l'algorithme de hachage est pris comme valeur « alg » du
```

```
structure englobante. Si les deux sont présents, le champ de cette structure est utilisé. Si aucune valeur
n'est présente à aucun de ces emplacements, cette structure n'est pas valide ; il n'y a pas de valeur par
défaut.
  « hash » : bstr, ; chaîne d'octets de la valeur de hachage
  « pad » : bstr, ; chaîne d'octets remplie de zéros utilisée pour remplir l'espace
  ? « pad2 » : bstr, ; chaîne d'octets remplie de zéros facultative utilisée pour remplir l'espace
  ? « name » : tstr .size (1..max-tstr-length), ; (facultatif) description lisible par l'utilisateur de ce que
couvre ce hachage
  ? « url » : uri, ; Inutilisé et obsolète.
}

EXCLUSION_RANGE-map = {
  « start » : uint, ; Octet de départ de la plage « length » :
  uint, ; Nombre d'octets de données à exclure
}
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous :

```
{
  « alg » : « sha256 »,
  « pad » : « 0000 »,
  "hash": 'Auxjtmax46cC2N3Y9aFmBO9Jfay8LEwJWzBUtZ0sUM8gA=', "name": "JUMBF
  manifest"
  "exclusions" : [
    {
      « start » : 9960,
      « length » : 4213
    }
  ],
}
```

Normalement, les valeurs `de début` et de `longueur` d'une `exclusion` doivent être écrites dans leur sérialisation préférée (c'est-à-dire « aussi courtes que possible »). Cependant, lorsqu'une assertion de hachage de données doit être créée mais que les valeurs `de début` et `de longueur` ne sont pas encore connues, elles doivent être créées « aussi grandes que possible », c'est-à-dire sous la forme d'un entier de 32 bits.

La valeur `de remplissage` doit toujours être présente, mais doit être une chaîne d'octets remplie de zéros de longueur 0, sauf si elle est utilisée pour remplacer (c'est-à-dire « remplir ») des octets lors d'un traitement en plusieurs passes. `pad2` est une chaîne d'octets remplie de zéros facultative qui est utilisée si le remplissage souhaité ne peut pas être obtenu avec `pad`.

REMARQUE La section 10.4, « Traitement en plusieurs étapes », décrit comment remplir les valeurs correctes et ajuster le remplissage.

18.5.3. Considérations particulières pour JPEG 1

Lors du hachage d'un fichier JPEG 1 (.jpg) dans lequel le manifeste C2PA sera intégré, le marqueur APP11 (`FFEB`) et la longueur (`Lp`) de tous les segments APP11 contenant les données JUMBF doivent être inclus dans la plage d'exclusion.

REMARQUE Tous les segments APP11 contenant le manifeste C2PA JUMBF sont contigus, de sorte qu'une seule plage est nécessaire.

18.5.4. Considération particulière pour les fichiers PNG

Lors du hachage d'un fichier PNG (.png) dans lequel le manifeste C2PA sera intégré, il est important que la `longueur` et le « `caBX` » (représentant le type de chunk) du chunk contenant les données JUMBF soient inclus dans la plage d'exclusion.

18.6. Hachage basé sur BMFF

18.6.1. Description

Les parties d'un actif basé sur BMFF qu'un générateur de revendication souhaite identifier de manière unique à l'aide d'un lien contraignant (c'est-à-dire un hachage cryptographique) doivent être décrites à l'aide d'assertions de hachage basées sur BMFF.

Une assertion de hachage basée sur BMFF doit avoir une étiquette `c2pa.hash.bmff.v3`.

REMARQUE	Les versions antérieures de cette norme documentaient également les assertions `c2pa.hash.bmff` et `c2pa.hash.bmff.v2`.
IMPORTANT	Les validateurs doivent ignorer toute assertion `c2pa.hash.bmff` et traiter le manifeste comme si l'assertion n'était pas présente.

Une assertion de hachage basée sur BMFF ne doit pas apparaître dans une assertion de données cloud.

Une version précédente de cette spécification fournissait un champ `url` pour indiquer l'emplacement des données hachées, mais celui-ci n'a jamais été utilisé. Ce champ est désormais obsolète au profit de l'assertion de référence d'actif. Les générateurs de revendications ne doivent pas ajouter ce champ à une assertion de hachage BMFF, et les consommateurs doivent ignorer ce champ lorsqu'il est présent, sauf si cela n'affecte pas l'inclusion du champ dans le contenu validé, comme décrit dans la section 15.10.3, « Validation des assertions ».

18.6.2. Calcul du hachage

Pour calculer le hachage spécifié dans le champ de valeur d'un hachage BMFF, tous les octets du fichier sont ajoutés au hachage, à l'exception des boîtes BMFF ou sous-ensembles de celles-ci qui correspondent à une entrée d'exclusion dans le tableau `des exclusions`.

Les boîtes qui sont incluses dans leur intégralité comprennent également leurs en-têtes dans les données d'entrée fournies au hachage. De même, les boîtes qui sont exclues dans leur intégralité excluent également leurs en-têtes des données d'entrée fournies au hachage. Lorsqu'une boîte est partiellement exclue des données d'entrée contribuant au hachage grâce à l'utilisation d'un champ `de sous-ensemble` dans la spécification d'exclusion, la ou les parties de la boîte à exclure définies par les décalages d'octets relatifs dans le champ de `sous-ensemble` sont des décalages par rapport au début de la boîte, y compris les en-têtes de boîte, et non des décalages par rapport au début du contenu de la boîte. Ces plages `de sous-ensembles` doivent être classées par ordre croissant de valeur `de décalage` et ne doivent pas se chevaucher.

Dans une assertion `c2pa.hash.bmff.v2` (obsolète) et `c2pa.hash.bmff.`v2 (obsolète) et c2pa`.`hash`.bmff.`v3, pour toute boîte racine non exclue dans son intégralité, les données d'entrée contribuant au hachage de cette boîte sont constituées de la concaténation des chaînes binaires `offset || data`, où `offset` est défini comme le décalage absolu du fichier de la boîte sous la forme d'un entier de 8 octets au format big-endian, et `data` est défini comme le contenu de la boîte, y compris les en-têtes, moins les exclusions éventuelles. Dans cette définition, « || »

représente la concaténation binaire des deux. Le décalage ne doit pas être inclus pour les hachages d'arborescence Merkle lorsque le bmff-hash-map comprend à la fois les champs `hash` et `merkle`.

De plus, les assertions `c2pa.hash.bmff.v2` (obsolète) et `c2pa.hash.bmff.v3` incluent les fonctionnalités suivantes :

- Le décalage absolu en octets du fichier est inclus au début des données d'entrée contribuant au hachage pour toute boîte racine. Cela garantit qu'une boîte racine incluse dans le hachage ne peut pas changer de position dans le fichier.
- La boîte `mdat` n'est plus exclue dans son intégralité lorsque la bmff-hash-map inclut à la fois les champs `hash` et `merkle` . Au lieu de cela, une entrée obligatoire dans la liste d'exclusion exclut la majeure partie de la boîte.

REMARQUE

Ces deux caractéristiques garantissent que le `mdat` ne peut pas changer de position dans le fichier tout en éliminant le besoin de décalage pour chaque hachage d'arbre Merkle individuel lorsque la bmff-hash-map inclut à la fois les champs `de hachage` et `Merkle`.

Une boîte correspond à une entrée d'exclusion dans le tableau `des exclusions` si et seulement si toutes les conditions suivantes sont remplies :

- L'emplacement de la boîte dans le fichier correspond au champ `xpath` de l'entrée `exclusions-map`. Par exemple, l'exclusion xpath `/foo/bar[2]` correspondrait à `/foo[2]/bar[2]`, `/foo[3]/bar[1]` ou `/foo[3]/bar[2]/baz[1]`. correspondre à emplacements mais `/foo[3]/bar[2]` et pas
- Si `la longueur` est spécifiée dans l'entrée `exclusions-map`, la longueur de la boîte correspond exactement au champ de longueur de l'entrée `exclusions-map` . Remarque : la longueur inclut les en-têtes de la boîte.
- Si `la version` est spécifiée dans l'entrée `exclusions-map`, la boîte est une FullBox et la version de la boîte correspond exactement au champ `version` de l'entrée `exclusions-map`.
- Si `flags` (tableau d'octets de 3 octets exactement) est spécifié dans l'entrée `exclusions-map` et que la boîte est une FullBox. Si `exact` est défini sur true ou n'est pas spécifié, les indicateurs de la boîte (bit(24), c'est-à-dire 3 octets) correspondent également exactement au champ `flags` de l'entrée `exclusions-map`. Si `exact` est défini sur false, le bitwise-and des indicateurs de la boîte (bit(24), c'est-à-dire 3 octets) avec le champ `flags` de l'entrée `exclusions-map` correspond exactement au champ `flags` de l'entrée `exclusions-map` (c'est-à-dire que la boîte a au moins ces bits définis, mais peut également avoir des bits supplémentaires définis).
- Si `des données` (tableau d'objets) sont spécifiées dans l'entrée `exclusions-map`, alors pour chaque élément du tableau, les données binaires de la boîte au niveau du champ `d'offset` d'octet relatif de cet élément correspondent exactement au champ `d'octets` de cet élément.

La syntaxe de la chaîne du champ `xpath` doit être limitée au sous-ensemble strict suivant.

- Seule la syntaxe abrégée doit être utilisée.
- Seuls les chemins complets doivent être utilisés.
- Seule la sélection de nœuds via `node` ou `node[entier]` doit être utilisée.
- La syntaxe descendante, c'est-à-dire `//`, ne doit PAS être utilisée.
- Tous les nœuds doivent être des codes BMFF `4cc`.

Exemple 5. Syntaxe complète pour le champ `xpath`

```
   xpath = '/' nodes nodes =
   node
          | node '/' nodes node =
   box4cc
         | box4cc '[' integer ']'
Où :
   box4cc est n'importe quel 4cc autorisé par la norme ISO/IEC 14496-12 pour une
   boîte BMFF. integer est n'importe quel entier positif non nul sans zéros en tête.
```

Toute entrée d'exclusion donnée peut correspondre à zéro ou plusieurs cases. Il n'est pas nécessaire qu'une entrée d'exclusion corresponde exactement à une case.

Un nœud xpath non feuille ne doit pointer que vers une boîte conteneur qui ne possède pas de champs propres (c'est-à-dire qui ne contient pas de données, seulement des boîtes enfants) et qui n'hérite pas de FullBox. Cela garantit qu'un validateur C2PA n'a pas besoin de connaître la syntaxe et la sémantique des boîtes inhabituelles qui contiennent d'autres boîtes. Si une boîte enfant d'une telle boîte inhabituelle doit être exclue en tout ou en partie, le champ `xpath` de l'entrée `exclusions-map` doit pointer vers la boîte inhabituelle elle-même et le champ `subset-map` doit exclure la ou les plages d'octets contenant les données de la boîte enfant exclue. Par exemple, la boîte « `sgpd` » contient d'autres boîtes, mais elle est inhabituelle en ce sens qu'elle hérite de FullBox ; ainsi, s'il est nécessaire d'exclure tout ou partie des boîtes enfants de « `sgpd` », l'assertion doit utiliser un champ `xpath` pointant vers « `sgpd` » elle-même (par exemple, `/moof/traf/sgpd`) et doit utiliser le champ `subset-map` pour exclure les octets souhaités.

Si le manifeste C2PA est intégré au fichier, la boîte qui le contient doit être l'une des entrées du tableau `exclusions` . Pour plus d'informations, reportez-vous à la section A.5, « Intégration de manifestes dans des ressources basées sur BMFF ».

Si une boîte exclue non racine est supprimée après la création du manifeste C2PA, elle doit être remplacée par une boîte « `libre` » de même taille afin de garantir que les données d'entrée contribuant au hachage des autres boîtes ne soient pas invalidées. Si la taille du magasin du manifeste C2PA est réduite à l'aide d'un manifeste compressé après la création du manifeste C2PA, une boîte « `libre` » doit être insérée à sa place afin de garantir que les décalages restent les mêmes. S'il est prévu qu'une boîte exclue non racine puisse être ajoutée après la création du manifeste C2PA, alors, au moment de la création du manifeste, une boîte « `libre` » doit être insérée avec un espace suffisant pour la boîte exclue et cette boîte « `libre` » doit également être exclue par une entrée d'exclusion utilisant son chemin d'accès complet. Lorsque la boîte exclue est ajoutée ou que la taille du magasin du manifeste C2PA est augmentée, la boîte « `libre` » doit être réduite (ou supprimée) pour compenser les données ajoutées. Toutefois, si l'espace dans la boîte « `libre` » est insuffisant, un manifeste standard doit être utilisé.

L'intégration des données C2PA dans une ressource BMFF via des boîtes MP4 modifie les décalages de fichiers dans d'autres boîtes MP4 ainsi que les décalages absolus d'octets de fichiers inclus dans les données d'entrée contribuant au hachage pour toute boîte racine. Ces boîtes et décalages doivent être inclus dans les données d'entrée contribuant au hachage avec leurs valeurs post-intégration, et non leurs valeurs pré-intégration, sinon l'assertion de hachage BMFF ne sera pas validée.

Il existe trois façons possibles pour une implémentation de garantir que les valeurs post-intégration de tous les décalages d'octets de fichier sont hachées :

1. Utiliser des boîtes « `libres` ».

a. Déterminez la ou les tailles maximales raisonnables pour la ou les boîtes C2PA qui seront intégrées. Toutes les boîtes MP4 pour C2PA prennent en charge les octets de remplissage inutilisés à la fin, il est donc possible de surestimer la taille des boîtes « `libres` » car les octets supplémentaires seront ignorés.

b. Insérez des boîtes « `libres` » de la ou des tailles indiquées dans le ou les fichiers de ressources et mettez à jour tous les décalages de manière appropriée.

c. Effectuez le hachage de la ressource avec « /free » dans la liste d'exclusion.

d. Créez et signez le manifeste. Créez la ou les boîtes C2PA.

e. Remplacez la ou les boîtes « `libres` » par la ou les boîtes C2PA.

2. Utilisez une approche en deux étapes.

a. Calculez les tailles exactes de l'assertion de hachage basée sur BMFF et des boîtes `Merkle`, le cas échéant. Cette dernière opération nécessitera l'analyse des fichiers d'actifs afin de déterminer la taille de l'arbre Merkle.

b. Calculez la taille exacte du manifeste final.

c. Effectuez le hachage du ou des fichiers de ressources. Mettez à jour toute boîte contenant des décalages de fichiers afin de corriger les valeurs avant d'inclure cette boîte dans les données d'entrée contribuant au hachage. Calculez les données d'entrée contribuant au hachage à l'aide de (`décalage || données`) en utilisant le décalage absolu mis à jour comme décrit ci-dessus. Comme indiqué ci-dessus, le décalage n'est pas inclus dans les données contribuant aux hachages de l'arbre Merkle lorsque la bmff-hash-map comprend à la fois les champs `hachage` et `merkle`.

d. Créez et signez le manifeste. Créez la ou les cases C2PA.

e. Insérez la ou les cases C2PA.

3. Placez le magasin de manifeste mis à jour à la fin du fichier BMFF.

a. Définissez le magasin de manifeste d'origine `box_purpose` de `manifeste` à `d'origine`.

b. Créez et signez le manifeste.

c. Créez une C2PA ContentProvenanceBox avec `box_purpose` défini sur `update`.

d. Insérez le manifeste mis à jour dans C2PA ContentProvenanceBox.

e. Insérez le C2PA ContentProvenanceBox à la fin du fichier BMFF.

f. Si un manifeste standard est ajouté alors qu'un magasin de manifestes de mise à jour est présent, le contenu du magasin de manifestes de mise à jour est déplacé vers le manifeste « original ».

g. Le magasin de manifests mis à jour est ensuite supprimé de la fin du fichier, ce qui permet une compatibilité ascendante avec un seul manifeste pour les cas d'utilisation courants.

h. Le magasin de manifestes « original » `box_purpose` est rétabli en tant que `manifeste` et le manifeste standard est ajouté comme d'habitude.

REMARQUE

Le champ `box_purpose` n'est pas inclus dans le hachage et peut être modifié sans invalider quoi que ce soit. hachage existant. De même, l'ajout du nouveau C2PA ContentProvenanceBox n'invalide pas les hachages existants.

Bien que la méthode en deux étapes soit nettement plus complexe, elle permet un hachage correct sans aucune

connaissance préalable de la taille maximale du manifeste. Elle minimise également la taille finale de la ressource. Les boîtes courantes (**liste non** exhaustive) avec des décalages de fichiers comprennent « iloc », « stco », « co64 », « tfhd », « sidx » et « saio ».

La possibilité de placer les manifestes mis à jour à la fin du fichier BMFF permet d'effectuer des mises à jour lorsqu'il n'y a pas de boîte « libre » suffisamment grande ou lorsque la complexité de l'approche en deux passes n'est pas souhaitée. Cette option prend également en charge les décalages de blocs dans les boîtes atom « stco » avec des informations de décalage de données partielles.

18.6.3. Schéma et exemple

Le schéma des assertions `c2pa.hash.bmff.v2` (obsolète) et `c2pa.hash.bmff.v3` est défini par la règle règle `bmff-hash-map` dans la définition CDDL suivante :

```
bmff-hash-map = {
  « exclusions » : [1* exclusions-map],
  ? « alg » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme de hachage cryptographique utilisé pour calculer ce hachage, tirée de la liste des identifiants d'algorithmes de hachage C2PA. Si ce champ est absent, l'algorithme de hachage est tiré d'une structure englobante telle que définie par cette structure. Si les deux sont présents, le champ de cette structure est utilisé. Si aucune valeur n'est présente à l'un de ces emplacements, cette structure n'est pas valide ; il n'y a pas de valeur par défaut.
  ? « hash » : bstr, ; Pour les MP4 non fragmentés, il s'agit du hachage de l'ensemble du fichier BMFF, à l'exclusion des boîtes répertoriées dans le tableau des exclusions. Pour les MP4 fragmentés, ce champ doit être absent.
  ? « merkle » : [1* merkle-map], ; Ensemble de lignes d'arbre Merkle et de données associées nécessaires pour permettre la vérification d'une seule boîte « mdat », de plusieurs boîtes « mdat » et/ou de fichiers fragmentés individuels dans l'actif.
  ? « name » : tstr .size (1..max-tstr-length), ; facultatif) une description lisible par l'utilisateur de ce que couvre ce hachage.
  ? « url » : uri, ; inutilisé et obsolète.
}

;(facultatif) Chaîne d'octets CBOR de 3 octets exactement.
flag-type = bytes

flag-t = flag-type .eq 3

exclusions-map = {
  « xpath » : tstr, ; Emplacement des boîtes à exclure du hachage à partir du nœud racine sous la forme d'une chaîne au format xpath de version https://www.w3.org/TR/xpath-10/ avec une syntaxe très contraignante.
  ? « length » : uint, ; (facultatif) Longueur que doit avoir une boîte la plus en feuille pour être exclue du hachage.
  ? « data » : [1* data-map], ; (facultatif) Les données contenues dans la boîte la plus à gauche à l'offset relatif spécifié doivent être identiques aux données spécifiées pour que la boîte soit exclue du hachage.
  ? « subset » : [1* subset-map], ; (facultatif) Seule cette partie de la boîte exclue est exclue du hachage. Chaque entrée du tableau doit avoir un décalage relatif en octets monotone croissant. Aucun sous-ensemble du tableau ne peut se chevaucher. La dernière entrée peut avoir une longueur nulle, ce qui indique que le reste de la boîte à partir de ce décalage relatif en octets est exclu. U n  décalage relatif en octets ou un décalage relatif en octets plus une longueur supérieure à la longueur de la boîte est autorisé ; les octets au-delà de la fin de la boîte ne sont jamais hachés.
  ? « version » : int, ; (facultatif) Version qui doit être définie dans une boîte la plus à la feuille pour que la boîte soit exclue du hachage. Spécifié uniquement pour une boîte qui hérite de FullBox.
  ? « flags » : flag-t, ; (facultatif) chaîne d'octets de 3 octets exactement. Les indicateurs 24 bits qui doivent être définis dans une boîte la plus basse pour que la boîte soit exclue du hachage. Spécifié uniquement pour une boîte qui hérite de FullBox.
  ? « exact » : bool, ; (facultatif) indique que les indicateurs doivent correspondre exactement. Si non spécifié, la valeur par défaut est true. Uniquement spécifié pour une boîte qui hérite de FullBox et lorsque
```

```
flags est également spécifié.
}

data-map = { « offset »
   : uint, « value » :
   bstr,
}
subset-map = { « offset
   » : uint, « length »
   : uint,
}

; Chaque entrée d'une carte correspond à des lignes d'arbre Merkle et aux données associées nécessaires pour
permettre la validation d'un seul
; Boîte « mdat » ou plusieurs boîtes « mdat » dans l'actif. », merkle-
map = {
   « uniqueId » : int, ; identifiant unique basé sur 1 utilisé pour différencier les fichiers afin de déterminer
quel arbre Merkle doit être utilisé pour valider une boîte « mdat » donnée.
   « localId » : int, ; Identifiant local indiquant l'arbre Merkle.
   « count » : int, ; Nombre de nœuds feuilles dans l'arbre Merkle. Les nœuds nuls ne sont pas inclus dans ce
comptage.
   ? « alg » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme de hachage cryptographique utilisé
pour calculer les hachages dans cet arbre Merkle, tirée de la liste des identifiants d'algorithmes de hachage C2PA.
Si ce champ est absent, l'algorithme de hachage est pris à partir de la valeur « alg » de la structure englobante
telle que définie par cette structure. Si les deux sont présents, le champ de cette structure est utilisé. Si
aucune valeur n'est présente à l'un de ces emplacements, cette structure n'est pas valide ; il n'y a pas de valeur
par défaut.
   ? « initHash » : bstr, ; Pour les ressources MP4 fragmentées qui sont réparties sur plusieurs fichiers, ce
champ doit être présent et correspond au hachage de l'ensemble du fichier du segment d'initialisation pour les
morceaux hachés par cet arbre Merkle, à l'exclusion des boîtes répertoriées dans le tableau des exclusions.
Pour les ressources MP4 fragmentées qui sont stockées sous la forme d'un seul fichier MP4 plat, ce champ doit
être présent et correspond au hachage de tous les octets précédant la première boîte « moof », à l'exclusion des
boîtes répertoriées dans le tableau des exclusions. Pour les MP4 non fragmentés, ce champ doit être absent.
   « hashes » : [1* bstr], ; Tableau ordonné représentant une seule ligne de l'arbre Merkle qui peut être la
ligne la plus à droite, la ligne racine ou toute ligne intermédiaire. La profondeur de la ligne est implicite
(calculée à partir) du nombre d'éléments dans ce tableau.
   ? « fixedBlockSize » : uint, ; Pour les ressources MP4 non fragmentées où la boîte mdat est validée par
morceaux, ce champ peut être présent. Ce champ correspond à la taille non négative en octets d'un nœud
feuille donné dans l'arbre Merkle. Pour les MP4 fragmentés, ce champ n'est pas présent.
   ? « variableBlockSizes » : [1* int], ; Pour les ressources MP4 non fragmentées où la boîte mdat est
validée par morceaux, ce champ peut être présent. Chaque entrée du tableau correspond à la taille non
négative en octets d'un nœud feuille donné dans l'arbre Merkle. Le nombre d'éléments est égal à « count » et
la somme des valeurs est égale à la taille de la charge utile de mdat. Pour les MP4 fragmentés, ce champ
n'est pas présent.
}
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) pour un fichier MP4 monolithique dont la boîte `mdat` est validée en tant qu'unité est présenté ci-dessous :

```
{
   « hash » : b64'EiAuxjtmax46cC2N3Y9aFmBO9Jfay8LEwJWzBUtZ0sUM8gA=', « name » : «
   Exemple d'assertion `c2pa.hash.bmff.v2` »,
   "exclusions": [
      {
         « data » : [
            {
               « value » : b64'2P7D1hsOSDySl1goh37EgQ==', « offset » : 8
            }
```

```
      ],
      « xpath » : « /uuid »
    },
    {
      « xpath » : « /ftyp »
    },
    {
      « xpath » : « /mfra »
    },
    {
      « xpath » : « /moov[1]/pssh »
    },
    {
      « xpath » : « /emsg »,
      « data » : [
        {
          « value » : b64'r3avWCpXHkmKHATFsV0Q5g==', « offset » :
          20
        }
      ]
    }
  ]
}
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) pour un actif composé de fichiers MP4 fragmentés est présenté ci-dessous :

```
{
  « alg » : « sha256 »,
  "name": "Exemple d'assertion `c2pa.hash.bmff.v3` pour fMP4", "merkle": [
    {
      "count": 23,
      « hachages » : [ b64'HvWZOxKMfkSatRAygs8DJfnEEcN/G1BNi359NdIDxbQ=',
b64'HvWZOxKMfkSatRAygs8DJfnEEcN/G1BNi359NdIDxbQ=' ],
      « localId » : 19,
      « initHash » : b64'Hf0IgeqbL0m+FTTLpUWwsDGR8pvhUR1AlwvaXjQ0qGY=', « uniqueId
      » : 17
    },
    {
      « count » : 69,
      "hashes": [ b64'9Zk7Eox+RJq1EDKCzwMl+cQRw38bUE2Lfn010gPFtB0=',
b64'9Zk7Eox+RJq1EDKCzwMl+cQRw38bUE2Lfn010gPFtB0=', b64'mTsSjH5EmrUQMoLPAyX5xBHDfxtQTYt+fTXSA8W0Hf0=',
b64'mTsSjH5EmrUQMoLPAyX5xBHDfxtQTYt+fTXSA8W0Hf0=', b64'OxKMfkSatRAygs8DJfnEEcN/G1BNi359NdIDxbQd/Qg=' ],
      « localId » : 38,
      « initHash » : b64'Hf0IgeqbL0m+FTTLpUWwsDGR8pvhUR1AlwvaXjQ0qGY=', « uniqueId
      » : 34
    },
    {
      « count » : 46,
      « hashes » : [ b64'OxKMfkSatRAygs8DJfnEEcN/G1BNi359NdIDxbQd/Qg=' ], « localId » : 57,
      « initHash » : b64'Hf0IgeqbL0m+FTTLpUWwsDGR8pvhUR1AlwvaXjQ0qGY=', « uniqueId
      » : 51
    }
  ],
  « exclusions » : [
    {
```

```
      « données » : [
        {
          « valeur » : b64'2P7D1hsOSDySl1goh37EgQ==', «
          décalage » : 8
        }
      ],
      « xpath » : « /uuid »
    },
    {
      « xpath » : « /ftyp »
    },
    {
      « xpath » : « /mfra »
    },
    {
      « xpath » : « /moov[1]/pssh »
    },
    {
      « data » : [
        {
          « value » : b64'9Q==', « offset »
          : 5
        },
        {
          « valeur » : b64'UAJXD79SlkG9rfnmcsqTUA==', «
          décalage » : 20
        },
        {
          « valeur » : b64'OxKM', «
          décalage » : 70
        }
      ],
      « flags » : b64'ZDNx',
      « xpath » : « /emsg », «
      length » : 200,
      « sous-ensemble » : [
        {
          « length » : 7,
          « offset » : 5
        },
        {
          « length » : 28,
          « décalage » : 20
        },
        {
          « length » : 63,
          « décalage » : 45
        },
        {
          « length » : 112,
          « décalage » : 80
        }
      ],
      « version » : 1
    }
  ]
}


{
  « alg » : « sha256 »,
  « name » : « Exemple d'assertion `c2pa.hash.bmff.v3` pour MP4 non fragmenté »,
 "merkle" : [
```

```
    {
      « count » : 3,
      « hachages » : [ b64'HvWZOxKMfkSatRAygs8DJfnEEcN/G1BNi359NdIDxbQ=',
b64'HvWZOxKMfkSatRAygs8DJfnEEcN/G1BNi359NdIDxbQ=' ], "variableBlockSizes" : [ 100,
      30, 20 ],
      « localId » : 19,
      « initHash » : b64'Hf0IgeqbL0m+FTTLpUWwsDGR8pvhUR1AlwvaXjQ0qGY=', « uniqueId
      » : 17
    }
  ],
  « exclusions » : [
    {
      « données » : [
        {
          « valeur » : b64'2P7D1hsOSDySl1goh37EgQ==', «
          décalage » : 8
        }
      ],
      « xpath » : « /uuid »
    },
    {
      « xpath » : « /ftyp »
    },
    {
      « xpath » : « /mfra »
    },
    {
      « xpath » : « /moov[1]/pssh »
    },
    {
      « data » : [
        {
          « value » : b64'9Q==', « offset »
          : 5
        },
        {
          « valeur » : b64'UAJXD79SlkG9rfnmcsqTUA==', «
          décalage » : 20
        },
        {
          « valeur » : b64'OxKM', «
          décalage » : 70
        }
      ],
      « flags » : b64'ZDNx',
      « xpath » : « /emsg », «
      length » : 200,
      « sous-ensemble » : [
        {
          « length » : 7,
          « offset » : 5
        },
        {
          « length » : 28,
          « décalage » : 20
        },
        {
          « length » : 63,
          « offset » : 45
        },
        {
          « length » : 112,
          « décalage » : 80
```

```
          }
        ],
        « version » : 1
      }
    ]
}
```

Une implémentation en pseudo-code de cet algorithme est présentée dans l'exemple 6, « Pseudo-code pour l'assertion de hachage basée sur BMFF ».

Exemple 6. Pseudo-code pour l'assertion de hachage basée sur BMFF

```
décalage = 0
Tant que (décalage < longueur du fichier)
     À partir du décalage, localisez le premier octet de la première boîte qui correspond à une entrée
du tableau des exclusions, appelez-le first_excluded_byte
          Si aucune boîte de ce type n'est trouvée, définissez first_excluded_byte = longueur
     du fichier Déterminez la longueur de cette boîte, appelez-la excluded_byte_count
          Si aucune boîte de ce type n'a été trouvée, définissez excluded_byte_count = 0
     Ajoutez au hachage tous les octets compris entre l'offset et first_excluded_byte moins un (inclus)
     Si first_excluded_byte < longueur du fichier et qu'il existe un sous-ensemble de tableau dans l'exclusion
qui a déterminé la valeur de first_excluded_byte
          définissez next_included_begin = premier_octet_exclu
          Pour chaque entrée du sous-ensemble de tableaux dans l'exclusion qui a déterminé la valeur
de first_excluded_byte
               Définir next_excluded_begin = champ de décalage de cette entrée du tableau de sous-ensembles plus
first_excluded_byte
               Si next_excluded_begin > next_included_begin
                    Ajouter au hachage tous les octets compris entre next_included_begin et
next_excluded_begin moins un (inclus)
               Définir next_included_begin = champ de longueur de cette entrée du tableau de sous-ensembles plus
next_excluded_begin
          Si next_included_begin < first_excluded_byte + excluded_byte_count Ajouter au
               hachage tous les octets entre next_included_begin et
first_excluded_byte + excluded_byte_count moins un (inclus) Définir offset =
     first_excluded_byte + excluded_byte_count
```

Un exemple de génération d'un hachage pour la carte Merkle est présenté dans l'exemple 7, « Exemple suggéré d'une carte Merkle ».

Exemple 7. Exemple suggéré d'une carte Merkle

```
Si les champs « fixedBlockSize » et « variableBlockSizes » ne sont pas présents
     Au hachage, ajoutez tous les octets entre l'adresse de début et la dernière adresse de la charge
utile mdat Si le champ « fixedBlockSize » est présent et que le champ « variableBlockSizes » n'est pas
présent
     Tant que (1)
          next_address = begin_address + fixedBlockSize
          Si next_address > dernière adresse de la charge utile mdat next_address =
               dernière adresse de la charge utile mdat plus un hash_complete = true
          Ajoutez au hachage tous les octets compris entre begin_address et next_address moins un (inclus)
          Si hash_complete est vrai, break
          adresse_début = adresse_suivante
```

```
Si le champ `variableBlockSizes` est présent et que le champ `fixedBlockSize` n'est pas présent
     Pour (blockSize dans variableBlockSizes) next_address
          = begin_address + blockSize
          Si next_address > dernière adresse de la charge utile mdat next_address =
               dernière adresse de la charge utile mdat plus un hash_complete = true
          Ajouter au hachage tous les octets entre begin_address et next_address moins un (inclus)
          Si hash_complete est vrai,
               terminer
          adresse_début = adresse_suivante
```

18.6.4. Profils de liste d'exclusion

18.6.4.1. Général

Cette section décrit un ensemble de profils prédéfinis et nommés de listes d'extensions.

18.6.4.2. Profil de base

Les médias non chronométrés classiques (par exemple, les photos) et les médias chronométrés (par exemple, les vidéos avec ou sans piste audio, fragmentées ou non) doivent uniquement inclure les exclusions obligatoires répertoriées dans les exigences relatives à la liste d'exclusion.

18.6.5. Diagramme d'entité BMFF fragmenté

La figure 15, « Diagramme d'entités BMFF fragmenté », montre la relation entre les objets C2PA composant un manifeste BMFF fragmenté.

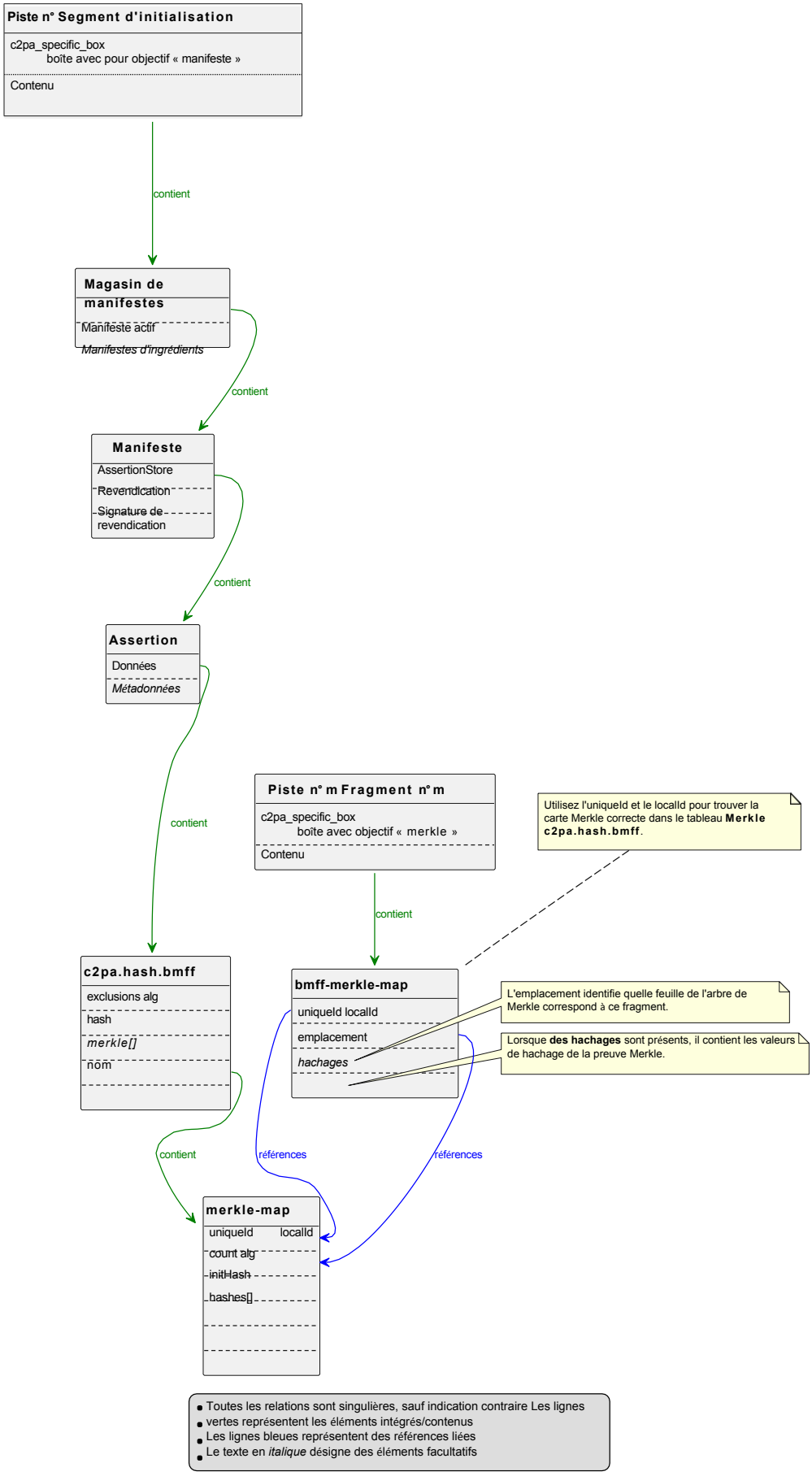

Figure 15. Diagramme d'entités BMFF fragmenté

18.6.6. Validation

Pour valider un chunk donné, il faut d'abord valider le `champ initHash` de `merkle-map` sur le segment d'initialisation correspondant, puis localiser l'entrée correcte dans le tableau `hashes` du champ `merkle-map` et la valider par rapport au hachage des données du chunk. Si nécessaire, il faut dériver ce hachage à l'aide de la preuve Merkle à partir des `hachages` spécifiés dans le `bmff-merkle-map` du chunk.

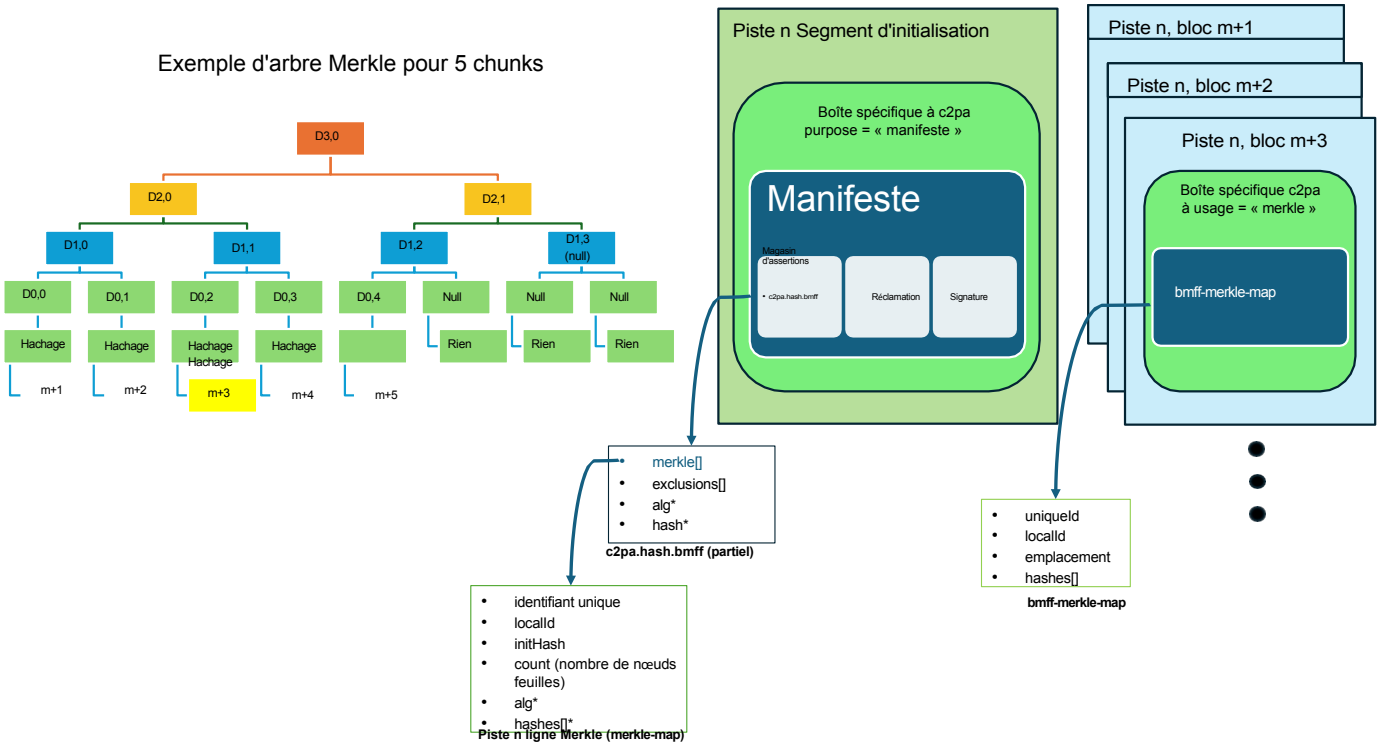

Figure 16. Validation du segment d'initialisation et exemple de données d'un bloc

Pour vérifier le segment **m+3,** vous devez d'abord vérifier le segment d'initialisation correspondant. La boîte manifeste spécifique à c2pa dans le segment d'initialisation de chaque piste contiendra le magasin de manifestes. Si l'actif contient plusieurs segments d'initialisation, le magasin de manifestes doit être identique dans chacun d'eux. Cela permet aux validateurs de vérifier qu'une piste appartient à l'ensemble plus large. L'assertion `c2pa.bmff.hash` du manifeste actif contiendra un champ `merkle` avec un tableau d'objets `merkle-map`, un par piste.

18.6.6.1. Étapes

1. À partir du `bmff-merkle-map` dans la boîte merkle spécifique à c2pa du chunk, obtenez `l'uniqueId` et `le localId`. Utilisez `l'uniqueId` et `le localId` pour trouver une `merkle-map` correspondante dans le tableau `merkle` de l'assertion `c2pa.bmff.hash` dans le segment init.
2. Si le hachage du segment init utilisant les `exclusions c2pa.bmff.hash` et `l'algorithme merkle-map` est égal à l' `initHash` à l'intérieur de la `carte Merkle` que vous venez de localiser, le segment d'initialisation est vérifié.

REMARQUE	Les paramètres `alg` & `hash` au niveau supérieur de la `bmff-hash-map` sont utilisés pour les MP4 monolithiques, tandis que `les paramètres alg` & `hashes` dans la `merkle-map` sont utilisés pour les MP4 fragmentés.

Pour terminer la vérification du chunk **m+3** : nous examinons `la merkle-map` de la piste n trouvée à l'étape 1, qui contient dans cet exemple la ligne 2 de l'arbre Merkle - **D2,0** et **D2,1**.

3. Hacher le bloc **m+3** à l'aide du tableau `d'exclusions c2pa.hash.bmff` et de `l'algorithme` issu de la `carte Merkle`, ce qui donne **D0,2(dérivé)**.
4. Le tableau `de hachages bmff-merkle-map` du bloc m+3 (preuve Merkle) contiendra le hachage du bloc m+4 **(D0,3)** et la valeur de hachage de la ligne 1 **D1,0**.
5. Hachage **de D0,2(dérivé)** et **D0,3** pour obtenir **D1,1(dérivé)**. Hachage **de D1,0** avec **D1,1(dérivé)** pour obtenir **D2,0(dérivé)**.
6. Si **D2,0(dérivé)** = **D2,0** tel qu'enregistré dans le paramètre `de hachage de la carte Merkle` d'assertion, et que le segment d'initialisation correspondant a été vérifié à l'étape 2, alors le bloc m+3 a été vérifié.

18.7. Hachage général

18.7.1. Description

Un générateur de revendications doit utiliser une assertion de hachage de boîte générale pour vérifier l'intégrité, avec une liaison forte (c'est-à-dire un hachage cryptographique), des actifs dont les formats utilisent un format de boîte non basé sur BMFF, tel que JPEG, PNG ou GIF.

Une assertion générale de hachage de boîte doit avoir une étiquette `c2pa.hash.boxes`. Une telle assertion se compose d'un tableau de structures, chacune répertoriant une ou plusieurs boîtes (par leur nom/identifiant) et un hachage qui couvre les données de ces boîtes (et toutes les données éventuelles présentes dans le fichier entre elles), ainsi que l'algorithme utilisé pour le hachage. Les boîtes doivent apparaître dans l'assertion dans le même ordre que celui dans lequel elles apparaissent dans l'actif, y compris la boîte contenant le manifeste C2PA. Si d'autres boîtes présentes dans l'actif ne sont pas explicitement incluses dans cette assertion, ou si les boîtes apparaissent dans un ordre incorrect, le manifeste sera rejeté lors de la validation, comme décrit à la section 15.12.3, « Validation d'un hachage de boîte général ».

Une boîte peut également comporter un champ `exclu`, qui est une valeur booléenne indiquant si un validateur peut ignorer cette boîte (et le hachage associé) lors de la validation. Si ce champ est absent, ou s'il est présent et que sa valeur est `fausse`, la boîte doit être hachée et les valeurs comparées. Pour les boîtes qui ont un champ `exclu` avec une valeur `vraie,` le générateur de revendications doit inclure un hachage précis pour assurer la compatibilité avec les validateurs plus anciens qui ne reconnaissent pas le champ `exclu`. Si le générateur de revendications ne se soucie pas de la compatibilité ascendante, il doit écrire la chaîne binaire `00` (un seul octet avec une valeur de 0) pour le hachage.

Dans le cas où il existe plusieurs instances du même type de boîte, telles que plusieurs segments `APP1` dans un fichier JPEG 1, chaque instance doit être répertoriée séparément dans l'assertion. Les segments JPEG qui sont des fragments partageant le même identifiant de segment sont également répertoriés comme des boîtes distinctes, à l'exception des segments composant le magasin de manifestes C2PA (tel que décrit ci-dessous).

La création des hachages est décrite dans la section 13.1, « Hachage », et la valeur doit être présente dans le champ `de hachage`. La valeur de hachage pour une plage de boîtes doit être calculée à partir du début de la première boîte (dans la plage) jusqu'à la fin de la dernière boîte (dans la plage). Cela inclut tous les octets arbitraires qui peuvent être présents entre les boîtes.

REMARQUE Lorsque vous utilisez une série de boîtes, toutes les données comprises entre le début de la première boîte et la fin de la dernière boîte sont

incluses dans le hachage. Cependant, lorsque chaque boîte est répertoriée séparément, les données supplémentaires ne sont pas incluses, seules les données contenues dans la boîte répertoriée le sont.

La boîte contenant le magasin de manifeste C2PA (par exemple `caBX` pour PNG ou `21FF` pour GIF) doit également être répertoriée dans son propre tableau. Afin de l'identifier clairement comme la boîte de manifeste C2PA, elle doit porter le nom `C2PA` et la valeur du `hachage` doit être la chaîne binaire `00` (un seul octet avec une valeur de 0). Le magasin de manifestes C2PA doit être représenté sous la forme d'une seule boîte, même dans le cas d'un fichier JPEG où la boîte est fragmentée en plusieurs segments de marqueurs `APP11`.

REMARQUE

Comme les validateurs sont souvent utilisés en combinaison avec la sortie des analyseurs de fichiers, il est recommandé, pour des raisons de sécurité, de

Hachage de tout le contenu du fichier en dehors du magasin de manifestes C2PA. Cela garantira l'intégrité du média et du manifeste lié.

La valeur `de remplissage` doit toujours être présente et doit être une chaîne d'octets remplie de zéros, sauf si elle a été remplacée par autre chose lors d'un traitement en plusieurs passes, auquel cas aucun `remplissage` ne doit être présent.

REMARQUE

La section 10.4, « Traitement en plusieurs étapes », décrit comment remplir les valeurs correctes et ajuster le remplissage.

Une assertion General Box Hash ne doit pas apparaître dans une assertion Cloud Data.

18.7.2. Traitement spécial des actifs en plusieurs parties

Pour prendre en charge les formats de fichiers composés de plusieurs parties (comme décrit dans la section 18.9, « Hachage multi-ressources »), une boîte logique supplémentaire est définie pour les cas où les données d'une ou plusieurs parties viennent après les données basées sur des boîtes de la partie principale. Cette boîte doit être étiquetée `c2pa.after` (pour les données arbitraires au-delà de la fin de la structure de la boîte). La boîte `c2pa.after`, si elle est présente, doit être la dernière boîte répertoriée, et son hachage doit être calculé à partir de l'octet suivant la dernière boîte jusqu'à la fin du fichier physique.

L'assertion de liaison forte, qui couvre l'ensemble de l'actif, doit être la seule assertion pouvant inclure une boîte `c2pa.after`
. Une assertion de hachage pour une partie individuelle ne doit couvrir que le contenu de cette partie elle-même, et aucune autre partie.

18.7.3. Gestion des formats spécifiques

18.7.3.1. Traitement spécifique au format JPEG

Lorsque vous travaillez avec JPEG, la boîte `APP11` est utilisée pour les normes autres que C2PA (c'est-à-dire JPEG 360). Dans ces situations, toutes les boîtes `APP11` non C2PA doivent être incluses dans la liste des boîtes hachées. Les boîtes `APP11` contenant le magasin de manifestes C2PA doivent être identifiées par `C2PA.` Toutes les autres boîtes doivent être identifiées par le symbole figurant dans la norme ISO 10918-1:1994, tableau B.1.

Le magasin de manifestes C2PA peut être identifié comme étant une superboîte JUMBF avec une étiquette `c2pa` et un UUID de type JUMBF de `63327061-0011-0010-8000-00AA00389B71,` comme décrit à la section 11.1.4.2, « Magasin de manifestes ».

REMARQUE

Les boîtes Start of Scan et Restart, étiquetées `SOS` et `RST[n]`, comprendront les segments codés par entropie suivant le marqueur respectif.

L'extension MPF (Multi-Picture Format) du format JPEG peut également être prise en charge à l'aide de cette méthode en répertoriant toutes les boîtes contenues dans le fichier telles qu'elles apparaissent, en supposant qu'il n'y ait pas de données entre `l'EOI` d'une image individuelle et le `SOI` de l'image suivante. La liste `des boîtes` énumérerait les segments de chaque image individuelle dans le MPF dans l'ordre (`SOI`, …, `EOI`, `SOI`, …, `EOI`, …). Cependant, si le générateur de revendications prévoit de traiter le fichier MPF comme un actif en plusieurs parties, la case `c2pa.after` doit être utilisée pour hacher les parties supplémentaires qui suivent `l'EOI` de la première image individuelle (la partie principale).

18.7.3.2. Traitement spécifique au format PNG

Un fichier PNG commence toujours par un en-tête de 8 octets (`89 50 4E 47 0D 0A 1A 0A`). Pour l'inclure, utilisez la valeur spéciale `PNGh` comme première boîte dans la liste des boîtes et commencez le hachage à partir du premier octet de l'image.

18.7.3.3. Traitement spécifique au format TIFF

Un fichier TIFF commence toujours par un en-tête de 8 octets. Pour l'inclure, utilisez la valeur spéciale `TIFh` comme première boîte dans la liste des boîtes.

Un fichier TIFF se compose d'un ou plusieurs IFD (répertoires de fichiers image) qui sont équivalents à des « super boîtes ». Chaque IFD contient un tableau d'entrées appelées « entrées IFD » ou « champs TIFF » qui représentent les « boîtes ». Le `nom de boîte` pour chaque entrée IFD doit être la valeur du champ `Tag` convertie en une chaîne de sa valeur décimale.

Contrairement à d'autres formats de type boîte, les données d'une entrée IFD peuvent ne pas être contenues dans l'entrée (à moins qu'elles ne soient d'une longueur inférieure ou égale à 4 octets), mais exister ailleurs dans le fichier.

REMARQUE La longueur des données d'une entrée IFD est déterminée en multipliant le nombre de valeurs de données (comme déterminé dans le champ `Count` de l'entrée IFD) par la taille de chaque valeur de données (déterminée par le Champ `de type` dans l'entrée IFD).

Le hachage d'une entrée IFD doit être calculé sur les 12 octets de l'entrée IFD. Si la longueur de l'entrée IFD est supérieure à 4 octets, le hachage doit être calculé à partir de la concaténation de ces 12 octets avec les octets du fichier référencé par l'entrée, en commençant par le décalage d'octet spécifié dans le champ `Décalage de valeur` de l'entrée IFD et en continuant sur toute la longueur des données.

Pour certaines entrées IFD bien connues (`StripOffsets` (273), `TileOffsets` (324) et `FreeOffsets` (288)), les données référencées par l'entrée IFD sont elles-mêmes une liste de décalages par rapport aux données réelles. Dans ces cas, les données sur lesquelles le hachage est calculé doivent être la concaténation des éléments suivants dans l'ordre indiqué :

1. Les 12 octets de l'IFD,
2. Les octets commençant à `la valeur Offset` d'une longueur `Count` multipliée par la taille de `Type` contenant les décalages, et
3. Pour chaque décalage dans l'ordre où il apparaît, les octets à ce décalage, avec la longueur donnée par l'entrée de comptage d'octets associée au type : `StripByteCounts` (279), `TileByteCounts` (325) et `FreeByteCounts` (289), respectivement.

REMARQUE Les données d'image dans un fichier TIFF seraient donc hachées à l'aide de cette combinaison de « décalages » et de « nombres d'octets ».

Le format TIFF prend également en charge les SubIFD, un type IFD qui pointe vers un ou plusieurs IFD et les intègre donc par référence. Ceux-ci comprennent non seulement le type appelé `SubIFD` (330), mais aussi `EXIF` (34665), `GPS` (34853) et `Interopérabilité` (40965). Pour tous ces types d'IFD, et tout autre type d'IFD qui fait référence à d'autres IFD de cette manière, les données sur lesquelles le hachage est calculé doivent être la concaténation des éléments suivants dans l'ordre indiqué :

1. Les 12 octets de l'IFD,
2. Soit :
    a. Si `N = 1`, les octets commençant à `la valeur Offset` de longueur égale à la taille du `Type` contenant le décalage de l'IFD référencé, ou
    b. Si `N > 1`, les octets commençant à `la valeur Offset` de longueur égale à la taille `du type` contenant le décalage vers le tableau des décalages IFD, concaténés avec les octets commençant à ce décalage de longueur égale à `Count` fois la taille du `type` qui contiennent les décalages vers chaque IFD « arborescent ».
3. Pour chaque IFD référencé, calculez de manière récursive les données pour le hachage de cet IFD à ce décalage, comme spécifié dans cette section.

18.7.3.4. Traitement spécifique au format GIF

Le hachage d'une boîte contenant un attribut « Packed Fields » (Champs compressés) hachera également les données facultatives indiquées par cet attribut. Par exemple, le descripteur d'image inclura le bloc Local Color Table (Table de couleurs locale) et le descripteur d'écran logique inclura le bloc Global Color Table (Table de couleurs globale), s'ils existent.

Pour toutes les boîtes contenant une étiquette de bloc, la convention de nommage sera la suivante : « <Block Label> ».

Pour tous les blocs d'extension, la convention de nommage est la suivante : « <Extension Introducer><Extension Label> ». Les seuls

autres blocs qui ne sont pas décrits par la convention de nommage ci-dessus sont :

- L'en-tête sera marqué « GIF89a ».
- Les données d'image basées sur un tableau seront marquées « TBID ».
- Le descripteur d'écran logique sera marqué « LSD ».

Par exemple :

- En-tête : « GIF89a ».
- Fin : « 3B ».
- Descripteur d'image : « 2C ».
- Extension de commentaire : « 21FE ».

18.7.3.5. Traitement spécifique au format RIFF

Les segments de fichiers RIFF peuvent être imbriqués dans une structure arborescente d'une profondeur arbitraire. La racine de cette structure se compose d'un ou plusieurs

morceaux **L0**, chacun avec l'identifiant de morceau `RIFF`. Ces morceaux `RIFF` sont définis avec la structure suivante :

- Octets 0-3 : identifiant du bloc, toujours `RIFF`.
- Octets 4-7 : longueur du bloc (moins 8 octets pour les champs d'identifiant et de longueur du bloc).
- Octets 8-11 : identifiant du type de média.
- Octets 12-n : données de chunk (tous les chunks **L1**).

Après l'identifiant du type de média, le bloc `RIFF` peut contenir un ou plusieurs sous-blocs **L1**, chacun ayant la structure suivante :

- Octets 0-3 : identifiant du bloc.
- Octets 4-7 : longueur du bloc (moins 8 octets pour les champs d'identifiant et de longueur du bloc).
- Octets 8-n : données du bloc.
- Octet n+1 : octet de remplissage (si nécessaire).

Un identifiant de bloc spécial `LIST` peut être utilisé pour imbriquer des blocs dans un bloc **L1**. Ces blocs `LIST` imitent la structure des blocs `RIFF` **L0** :

- Octets 0-3 : identifiant de chunk, toujours `LIST`.
- Octets 4 à 7 : longueur du bloc (moins 8 octets pour les champs d'identifiant et de longueur du bloc).
- Octets 8 à 11 : identifiant du type de liste.
- Octets 12 à n : données de chunk (tous les chunks **L2**).
- Octet n+1 : octet de remplissage (si nécessaire).

Aux fins du calcul d'un hachage général de boîte, chaque morceau **L0** doit être traité comme une boîte unique d'une taille exacte de 12 octets, dont le nom est égal à l'identifiant du type de média (octets 8 à 11). Chaque bloc **L1** `non LIST` doit être traité comme une boîte dont le nom est égal à l'identifiant du bloc (octets 0 à 3) et dont le contenu s'étend du début de l'identifiant du bloc (octet 0) jusqu'à l'octet de remplissage, le cas échéant, inclus. Chaque bloc `LIST` **L1** doit être traité comme une boîte dont le nom est égal à l'identifiant de type de liste (octets 8 à 11) et dont le contenu s'étend du début de l'identifiant de bloc (octet 0) à l'octet de remplissage, le cas échéant, inclus. Tous les chunks imbriqués dans un chunk `LIST` **L1** (**L2** et supérieur) doivent être traités comme faisant partie du contenu du chunk `LIST` **L1** et hachés comme une seule boîte.

Dans tous les cas, les octets de remplissage doivent être traités comme faisant partie du contenu du bloc précédent et doivent être inclus dans le hachage de cette case.

18.7.3.6. Traitement spécifique aux polices

Les tables d'une police correspondent directement aux boîtes de hachage, y compris la table `C2PA`. Les tables

sont toujours énumérées dans l'ordre dans lequel elles apparaissent dans le répertoire des tables de la police.

Notez que le répertoire des tables lui-même ne fait pas partie du contenu haché et n'est donc couvert par aucune boîte.

La valeur `checkSumAdjustment` doit être traitée comme zéro (`0`) lors du calcul du hachage de la boîte contenant la

table d'en-tête.

Le regroupement, ou l'absence de regroupement, des tables de polices dans l'assertion générale de hachage de boîte dépend du générateur de revendications.

Remarque : Les polices créées pour une large distribution peuvent tirer profit de l'attribution de chaque table à une boîte individuelle ; ainsi, si la police est reconditionnée dans un autre format, son hachage continuera à être validé correctement. En revanche, les systèmes qui génèrent automatiquement un grand nombre de polices, tels que les sous-ensembles, peuvent choisir de combiner les tables dans un nombre réduit de boîtes afin de rationaliser le traitement. Dans ce cas, le ou les hachages de boîte peuvent ne pas être validés après une transformation de format, en raison de l'inclusion d'un remplissage inter-table.

Étant donné que les consommateurs de polices ne doivent pas réagir aux tables qu'ils ne reconnaissent pas, l'infrastructure existante de gestion des polices s'attendra à ce que la valeur `checkSumAdjustment` de la table `principale` intègre le contenu final définitif de la table `C2PA` elle-même, y compris tout manifeste local dans son intégralité.

18.7.4. Schéma et exemple

Le schéma de ce type est défini par la règle `box-map` dans la définition CDDL dans CDDL pour Box Hash :

CDDL pour Box Hash

```
box-map = {
   « boxes » : [1* box-hash-map],
   ? « alg » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme de hachage cryptographique
utilisé pour calculer le hachage dans cette assertion, tirée de la liste des identifiants d'algorithmes de
hachage C2PA. Si ce champ est absent, l'algorithme de hachage est pris à partir de la valeur « alg » de la
structure englobante. Si les deux sont présents, le champ de cette structure est utilisé. Si aucune valeur
n'est présente à l'un de ces emplacements, cette structure n'est pas valide ; il n'y a pas de valeur par
défaut.
}

box-hash-map = {
   « names » : [1* box-name], ; Tableau de chaînes représentant les identifiants des boîtes par ordre d'apparition
(par exemple, « APP0 », « IHDR »)
   ? « alg » : tstr .size (1..max-tstr-length), ; Chaîne identifiant l'algorithme de hachage cryptographique
utilisé pour calculer le hachage dans cette assertion, tirée de la liste des identifiants d'algorithmes de
hachage C2PA. Si ce champ est absent, l'algorithme de hachage est pris à partir de la valeur `alg` de la
structure englobante. Si les deux sont présents, le champ de cette structure est utilisé. Si aucune valeur
n'est présente à l'un de ces emplacements, cette structure est invalide ; il n'y a pas de valeur par défaut.
   « hash » : bstr, ; chaîne d'octets de la valeur de hachage
   ? « excluded » : booléen ; valeur booléenne indiquant si un validateur peut ignorer cette case (et le
hachage associé) lors de la validation. Si ce champ est absent, la case est hachée et les valeurs sont
comparées.
   « pad » : bstr, ; chaîne d'octets remplie de zéros utilisée pour remplir l'espace
}

box-name /= tstr .size (1..10)
```

Cinq exemples en notation diagnostique CBOR (RFC 8949, clause 8) sont présentés dans l'exemple Box Hash :

1. JPEG ;
2. PNG ;
3. GIF ;

4. DNG (TIFF), avec un SubIFD ;
5. TTF.

Exemple de hash de boîte

```
// Exemple JPEG //
{
     « alg » : « sha256 », «
     boxes » : [
          {
               « noms » : [« SOI », « APP0 », « APP2 »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« C2PA »],
               « hash » : b64'AA==',
               « pad » : b64'',
          },
          {
               « noms » : [« DQT », « SOF0 », « DHT », « SOS », « RST0 », « RST1 », « EOI »],
               « hash » : b64'', « pad
               » : b64'',
          }
     ]
}

// Exemple PNG //
// avec la boîte XMP exclue //
{
     « alg » : « sha256 », «
     boxes » : [
          {
               « names » : [« PNGh », « IHDR »],
               "hash" : b64'...',
               "pad" : b64'',
          },
          {
               « noms » : [« C2PA »],
               « hash » : b64'AA==',
               « pad » : b64'',
          },
          {
               « noms » : [« sBIT »],
               « hash » : b64'', « pad
               » : b64'',
          },
          {
               « noms » : [« iTXt »],
               « hash » : b64'...', «
               excluded » : true, « pad
               » : b64'',
          },
          {
               « noms » : [« IDAT », « IEND »],
               « hash » : b64'...', «
               pad » : b64'',
          }
     ]
}
```

```
// Exemple GIF //
{
     « alg » : « sha256 », «
     boxes » : [
          {
               « names » : [« GIF89a », « LSD »]
               "hash" : b64'...',
               "pad" : b64'',
          },
          {
               « noms » : [« 2C », « TBID », « 2C », « TBID »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« 21FE »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« 21F9 »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« 3B »],
               « hash » : b64'...', «
               pad » : b64'',
          },
     ]
}

// Exemple TIFF/DNG //
{
     « alg » : « sha256 », «
     boxes » : [
          {
               « noms » : [« TIFh », « 254 », « 256 », « 257 », « 258 », « 259 », « 262 »],
               « hash » : b64'...', « pad
               » : b64'',
          },
          {
               « noms » : [« 273 », « 277 », « 278 », « 279 », « 284 »],
               « hash » : b64'...', « pad
               » : b64'',
          },
          {
               // il s'agit d'un SubIFD contenant une image secondaire //
               « names » : [« 330 », « 254 », « 256 », « 257 », « 258 », « 259 », « 262 », « 277 », « 278 », « 279 »,
« 284 »],
               « hash » : b64'...', « pad
               » : b64'',
          },
          {
               « names » : [« 700 », « 34665 »],
               « hash » : b64'...', « pad
               » : b64'',
          },
          {
               « noms » : [« C2PA »],
               « hash » : b64'AA==',
               « pad » : b64'',
```

```
          }
     ]
}

// Exemple TTF //
{
     « alg » : « sha256 », «
     boxes » : [
          {
               « names » : [« C2PA »],
               « hash » : b64'AA==',
               « pad » : b64'',
          },
          {
               « noms » : [« PCLT »],
               « hash » : b64'', « pad
               » : b64'',
          },
          {
               « noms » : [« cmap »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« cvt »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« fpgm »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« gasp »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« glyf »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« tête »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« hhea »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« hmtx »],
               « hash » : b64'...', «
               pad » : b64'',
          },
          {
               « noms » : [« loca »],
               « hash » : b64'...', «
               pad » : b64'',
          },
```

```
          {
               « noms » : [« maxp »],
               « hash » : b64'...', « pad
               » : b64'',
          },
          {
               « noms » : [« nom »],
               « hash » : b64'...', « pad
               » : b64'',
          },
          {
               « noms » : [« publication »],
               « hash » : b64'...', « pad
               » : b64'',
          },
          {
               « noms » : [« prep »],
               « hash » : b64'...', « pad
               » : b64'',
          }
     ]
}
```

18.8. Hachage des données de collection

18.8.1. Description

Dans les flux de travail où l'on sait à l'avance que le manifeste C2PA fera référence à un ensemble d'actifs plutôt qu'à un seul actif, l'assertion de hachage des données de l'ensemble sera utilisée comme méthode pour spécifier les liaisons fixes (c'est-à-dire les hachages cryptographiques) pour les actifs de l'ensemble.

REMARQUE	Il est possible de décrire chaque dossier de l'ensemble de données d'entraînement d'un modèle d'IA/ML en faisant de chaque dossier un élément distinct du manifeste complet de l'ensemble de données d'entraînement.

Une assertion de hachage de données de collection doit avoir une étiquette

`c2pa.hash.collection.data`. Une assertion de hachage de données de collection ne doit pas apparaître

dans une assertion de données cloud.

18.8.2. Schéma et exemple

Le schéma de ce type est défini par la règle `collection-data-hash-map` dans la définition CDDL suivante :

```
; Tableau d'URI et de leurs hachages associés
$collection-data-hash-map /= { « uris » :
   [1* uri-hashed-data-map],
   « alg » : tstr .size (1..max-tstr-length), ; Une chaîne identifiant l'algorithme de hachage cryptographique
utilisé pour calculer le hachage de chaque entrée du tableau « uris », tiré de la liste des identifiants
d'algorithmes de hachage C2PA.
     ? « zip_central_directory_hash » : bstr,
}

; Structure de données utilisée pour stocker une référence à un URI et son hachage.
$uri-hashed-data-map /= {
```

```
  « uri » : type-url-relative, ; référence URI relative « hash » :
  bstr, ; chaîne d'octets contenant la valeur de hachage
  ? « size » : type-taille, ; nombre d'octets de données
  ? « dc:format » : chaîne de format, ; type de média IANA des données
  ? « data_types » : [1* $asset-type-map], ; informations supplémentaires sur le type des données
}

; avec CBOR Head (#) et tail ($) sont introduits dans regexp, donc pas besoin de les mentionner explicitement
relative-url-type /= tstr .regexp « [-a-zA-Z0-9@:%._\\+~#=]{2,256}\\.[a-z]{2,6}\\b[-a-zA-Z0-
9@:%_\\+.~#?&//=]*"
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous :

```
// exemple d'une liste d'URL distantes //
{
     « alg » : « sha256 », « uris
     » : [
          {
               "uri": "photos/id/870.jpg"
               "hash": b64'+ddHMTUUEpuSF6dNaHFa9uFc1sSnY+O3l3MMPFvX5Ws=', "dc:format":
               "image/jpeg"
          },
          {
               « url » : « deepmind/bigbigan-resnet50/1 », «
               hash » : b64'...,
               « dc:format » : « application/octet-stream », « data_types »
               : [
                    {
                         « type » : « c2pa.types.generator »,
                    },
                    {
                         « type » : « c2pa.types.model.tensorflow », «
                         version » : « 1.0.0 »,
                    },
                    {
                         « type » : « c2pa.types.tensorflow.hubmodule », « version »
                         : « 1.0.0 »,
                    }
               ]
          }
     ]
}

// exemple d'une liste d'URI de fichiers (relatifs) //
{
     « alg » : « sha256 », « uris
     » : [
          {
               "uri" : "image1.png",
               "hash": b64'U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8='
          },
          {
               « uri » : « document.pdf »,
               « hash » : b64'G5hfJwYeWTlflxOhmfCO9xDAK52aKQ+YbKNhRZeq92c='
          },
     ]
}

// exemple d'une liste de chemins relatifs à l'intérieur d'un EPUB (qui est un fichier ZIP) //
{
```

```
    « alg » : « sha256 », «
    uris » : [
        {
            « uri » : « mimetype »
            "hash": b64'+ZXhhbXBsZSBvZiBhIGxpc3Qgb2YgcmVsYXRpdmUgc8=', "dc:format":
            "text/text"
        },
        {
            « uri » : « META-INF/container.xml »
            « hash » : b64'+ddHMTUUEpuSF6dNaHFa9uFc1sSnY+O3l3MMPFvX5Ws=', « dc:format » : «
            text/xml »
        },
        {
            « uri » : « cover_page.svg »,
            « hash » : b64'U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8='
        },
        {
            « uri » : « chapter1.html »,
            « hash » : b64'G5hfJwYeWTlflxOhmfCO9xDAK52aKQ+YbKNhRZeq92c='
        },
    ]
}
```

18.8.3. Champs

Le champ `uris` est constitué d'un tableau de valeurs `uri-hashed-data-map` qui représente une collection d'actifs. Le champ `alg`, tel que décrit dans la section 13.1, « Hachage », garantit que tous les éléments de contenu de la liste sont hachés à l'aide du même algorithme.

Pour chaque `uri-hashed-data-map`, le champ `uri` doit être présent et doit être un URI relatif valide. Tous les URI doivent être considérés comme relatifs à l'emplacement du manifeste, qu'il soit local, dans un conteneur (par exemple, ZIP) ou dans le cloud. Comme un URI relatif peut contenir des éléments de navigation (par exemple, `../`), il est possible de faire référence à des éléments de contenu qui ne se trouvent pas dans le même dossier que le manifeste, ce qui constituerait un problème de sécurité. Un générateur de revendications doit valider ou nettoyer les URI avant leur utilisation, en s'assurant que ni . ni `..` n'apparaissent dans l'URI.

Le champ `hash` est une chaîne d'octets représentant la valeur de hachage valide pour l'élément de contenu, telle que déterminée par le champ `alg` . Le hachage doit porter sur tous les octets (de 0 à n) de l'élément de contenu, sans exception. Les autres

champs sont identiques à ceux d'une assertion d'ingrédient.

18.8.4. Hachage des membres de la collection

Chaque fichier de la collection doit être haché individuellement à l'aide de l'algorithme de hachage spécifique défini dans le champ `alg`. La valeur de hachage obtenue doit être stockée dans le champ `hash` de la `table uri-hashed-data-map` associée à `l'URI` du fichier.

Tous les fichiers d'une hiérarchie donnée ne doivent pas nécessairement être inclus dans une collection hachée.

REMARQUE	Bien que cela soit utile dans les cas où certains fichiers ne nécessitent pas d'être hachés, cela permet également à un adversaire d'ajouter des fichiers sans invalider la liaison.

18.9. Hachage multi-actifs

18.9.1. Description

Il existe un certain nombre de formats de fichiers composés de plusieurs parties, chacune d'entre elles étant un format de fichier valide, comme lorsque plusieurs images individuelles sont regroupées dans un seul fichier. Voici quelques exemples :

- Format multi-images CIPA (MPF)
- Format Android Ultra HDR (qui utilise le MPF)
- ISO 21496 HDR (qui utilise le format MPF)
- Format Android Motion Photo (qui n'utilise pas le MPF, mais peut coexister avec le MPF dans le même fichier)

Dans certains cas, il peut être souhaitable, voire nécessaire, de vérifier l'intégrité de chaque partie individuelle du fichier, plutôt que celle du fichier dans son ensemble. Par conséquent, l'ensemble actuel d'assertions contraignantes n'est pas suffisant pour vérifier séparément l'intégrité de chaque partie. De plus, les parties individuelles peuvent avoir leurs propres manifestes C2PA qui doivent être enregistrés. L'assertion de hachage multi-actifs est utilisée pour fournir cette fonctionnalité.

Un autre cas particulier est celui où une partie individuelle est facultative, ce qui signifie qu'elle peut être supprimée dans le cadre d'un flux de travail qui n'implique pas de signataire de confiance, mais où la possibilité de vérifier l'intégrité du reste du fichier reste souhaitable.

18.9.2. Détails

Une assertion de hachage multi-actifs doit porter l'étiquette `c2pa.hash.multi-asset`. Bien qu'elle contienne des hachages et modifie le traitement de la liaison forte, elle n'est pas considérée comme une liaison forte.

Une assertion de hachage multi-actifs ne doit pas apparaître dans une assertion de données cloud.

Une assertion de hachage multi-actifs ne doit pas être utilisée avec un manifeste compressé.

REMARQUE	Il n'est pas certain qu'il existe une incompatibilité technique entre les deux, il est donc recommandé d'éviter de les utiliser ensemble jusqu'à ce qu'une évaluation plus approfondie soit effectuée.

Chaque partie, y compris la partie principale, doit être représentée sous la forme d'un objet `part-hash-map` dans le tableau `parts`. Le champ `location` doit contenir un objet `locator` qui décrit l'emplacement de la partie dans le fichier. L'objet `locator` doit contenir soit un champ `bmffBox`, soit des champs `byteOffset` et `length`. Le champ `byteOffset` doit contenir le décalage en octets (à partir du début physique du fichier) de la partie dans le fichier, et le `champ length` doit contenir la longueur de la partie en octets. Le champ `bmffBox` doit contenir la boîte BMFF de la partie, lorsque celle-ci est contenue dans la partie principale mais sous la forme d'une boîte BMFF spécifique (par exemple, `mpvd` tel qu'utilisé par Motion Photo). Pour une partie décrite par un champ `bmffBox`, le contenu de la partie doit être considéré comme la charge utile de cette boîte uniquement, à l'exclusion de l'en-tête de la boîte.

Les parties contenues dans le tableau `des parties` doivent être répertoriées dans l'ordre dans lequel elles apparaissent dans le fichier, et les parties doivent être contiguës, ne pas se chevaucher et couvrir chaque octet de la ressource.

REMARQUE	L'apparition dans le fichier est définie comme leur ordre séquentiel tel qu'ils seraient situés si l'on commençait à partir de l'octet 0 et que l'on parcourait jusqu'au dernier octet du fichier.

Le champ `hashAssertion` doit contenir un URI haché vers l'assertion de hachage de la partie. L'assertion de hachage d'une partie doit être une assertion standard à liaison forte (par exemple, `c2pa.hash.data`), mais l'étiquette doit comporter la chaîne `.part` et tout identifiant d'instance multiple ajouté. Par exemple, `c2pa.hash.data.part 2`.

REMARQUE	L'ajout de ces suffixes d'étiquette permet de préciser que les assertions à liaison forte pour les parties ne sont pas prises en compte. des assertions standard à liaison forte et qu'il peut donc en exister plusieurs instances dans un manifeste C2PA.

Le champ `facultatif` doit être une valeur booléenne indiquant si la présence de la partie est facultative. La valeur par défaut est « `false` » si elle n'est pas présente.

Si une partie possède son propre manifeste C2PA, qui n'est pas autonome au sein de cette partie (par exemple, des images individuelles dans une ressource multi-images), il est recommandé de stocker ce manifeste C2PA dans le magasin de manifestes de la ressource et de créer un composant « `componentOf` » pour y faire référence.

18.9.3. Schéma et exemple

Le schéma de ce type est défini par la règle `multi-asset-hash-map` dans la définition CDDL suivante :

```
multi-asset-hash-map = {
  « parts » : [* part-hash-map] ; Tableau contenant un ou plusieurs hachages pour les différentes parties
du fichier multipart.
}

byte-range-locator = (
  « byteOffset » : uint      ; Décalage en octets de la partie dans le fichier «
  length » : uint            ; La longueur de la partie
)

; il s'agit d'une carte CDDL spéciale de choix (ce qui signifie que seul l'un des éléments suivants peut être
présent)
locator-map = {
  byte-range-locator //          ; Décalage en octets et longueur de la partie dans le fichier «
  bmffBox » : tstr           ; Un XPath vers la boîte BMFF de la partie
}

part-hash-map = {
  « location » : locator-map, ; Emplacement de la partie dans le fichier « hashAssertion » : $hashed-
  uri-map, ; hashed_uri vers l'assertion de hachage de la partie
  ? « optional » : bool, ; Si la partie est facultative et peut être supprimée
}
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous :

```
// assertion multi-asset-hash //
// L'actif (de 33 333 octets) comprend une partie JPEG en octets [0,11111) et une autre
// partie en octets [11111,33333).
{
  « parties » : [
```

```
    {
      « location » : { «
        byteOffset » : 0,
        « length » : 11111
      },
      « hashAssertion » : « self#jumbf=c2pa.assertions/c2pa.hash.boxes.part »
    },
    {
      « location » : { « byteOffset »
        : 11111,
        « length » : 22222
      },
      « hashAssertion » : « self#jumbf=c2pa.assertions/c2pa.hash.data.part »
    }
  ]
}

// c2pa.hash.boxes.part - hachage de la boîte pour la première partie de l'actif //
{
  « alg » : « sha256 »,
  « boxes » : [
    {
      "names" : ["SOI", "APP0", "APP2"],
      "hash" : b64'...', "pad"
      : b64'',
    },
    {
      « noms » : [« C2PA »],
      « hash » : b64'AA==',
      « pad » : b64'',
    },
    {
      « noms » : [« DQT », « SOF0 », « DHT », « SOS », « RST0 », « RST1 », « EOI »],
      « hash » : b64'', « pad »
      : b64'',
    }
  ]
}

// c2pa.hash.data.part - hachage des données pour la deuxième partie de l'actif //
{
  « alg » : « sha256 »,
  « pad » : « 0000 »,
  « hash » : b64'...',
}

// c2pa.hash.boxes - hachage global de l'actif, couvrant l'ensemble de l'actif en deux parties //
{
  « alg » : « sha256 »,
  « boxes » : [
    {
      "names" : ["SOI", "APP0", "APP2"],
      "hash" : b64'...', "pad"
      : b64''
    },
    {
      « noms » : [« C2PA »],
      « hash » : b64'AA==',
      « pad » : b64''
    },
    {
      « noms » : [« DQT », « SOF0 », « DHT », « SOS », « RST0 », « RST1 », « EOI »],
      « hash » : b64'...,
```

```
        « pad » : b64''
      },
      {
        « noms » : [« c2pa.after »],
        « hash » : b64'...', «
        pad » : b64''
      }
    ]
}
```

Un exemple d'assertion de hachage multi-actifs de ce type peut être inclus dans une image, comme illustré dans [_multi_asset_hdr_image].

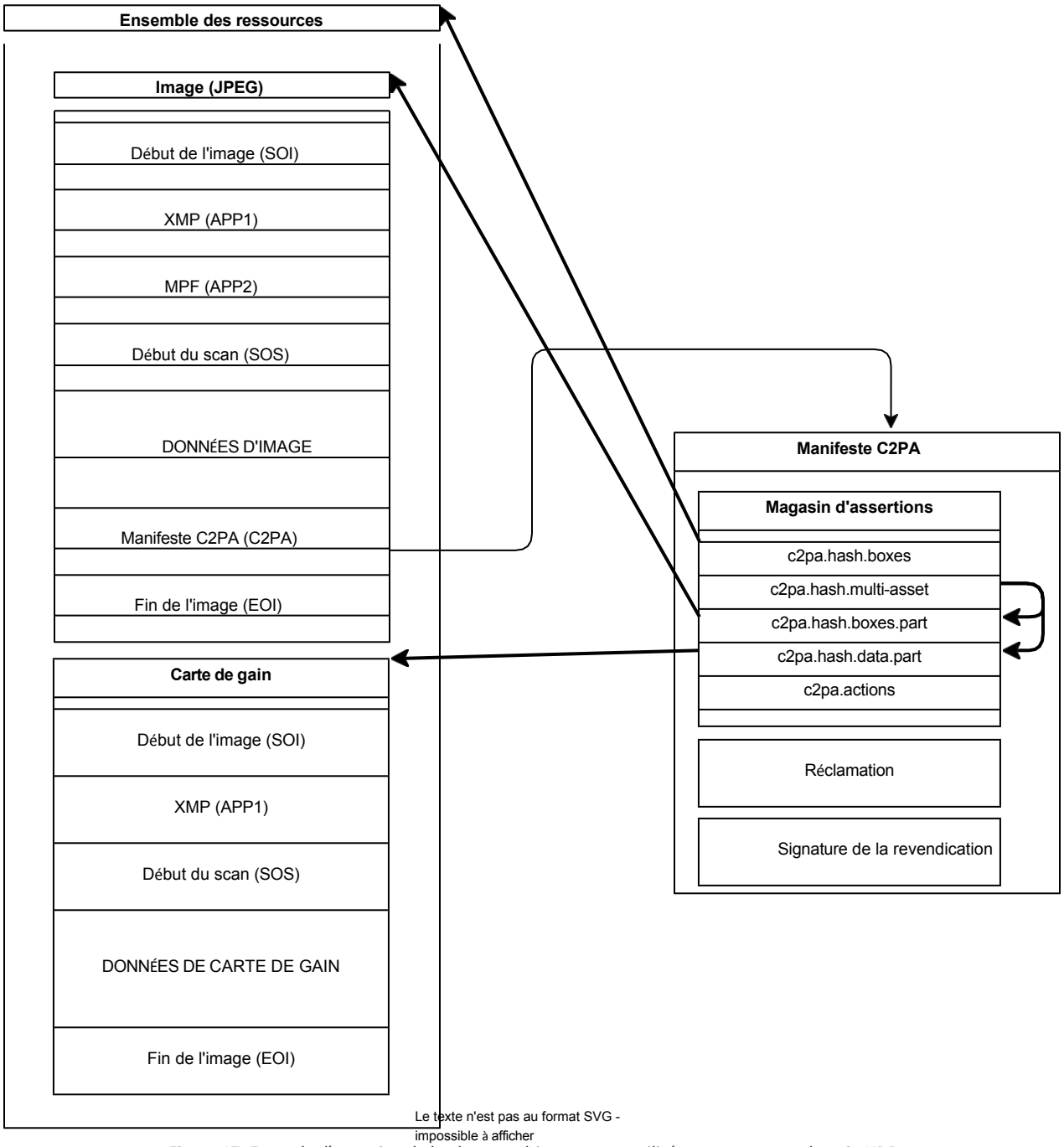

Figure 17. Exemple d'assertion de hachage multi-ressources utilisée pour une carte de gain HDR

18.10. Liaison souple

18.10.1. Description

Si un générateur de revendications fournit une liaison souple pour le contenu de l'actif, celle-ci doit être décrite à l'aide d'une assertion de liaison souple. Les types de liaisons souples qui peuvent être créés et stockés dans une telle assertion sont décrits à la section 18.10, « Liaison souple ».

Une version précédente de cette spécification fournissait un champ `url` pour indiquer l'emplacement des données hachées, mais celui-ci n'a jamais été utilisé. Ce champ est désormais obsolète au profit de l'assertion de référence d'actif. Les générateurs de revendications ne doivent pas ajouter ce champ à une assertion à liaison souple, et les consommateurs doivent ignorer ce champ lorsqu'il est présent, sauf si cela n'affecte pas l'inclusion du champ dans le contenu validé, comme décrit dans la section 15.10.3, « Validation des assertions ».

Une version précédente de cette spécification fournissait un champ « `extent` » dans le champ « `scope` » pour décrire une partie du contenu numérique couvert par l'assertion de liaison souple, dans un format spécifique à l'algorithme. Ce champ est désormais obsolète au profit du champ « `region` ». Les générateurs de revendications ne doivent pas ajouter ce champ à une assertion de liaison souple, et les consommateurs doivent ignorer ce champ lorsqu'il est présent. Cela n'affecte pas l'inclusion du champ dans le contenu validé, comme décrit dans la section 15.10.3, « Validation des assertions ».

Une assertion à liaison souple doit avoir une étiquette `c2pa.soft-binding`.

18.10.2. Schéma et exemple

Le schéma de ce type est défini par la règle `soft-binding-map` dans la définition CDDL suivante :

```
;Aligner la structure des objets des régions d'intérêt dans les assertions de liaison souple avec celle
utilisée à d'autres fins
;# inclure les régions d'intérêt

;Structure de données utilisée pour stocker une ou plusieurs liaisons souples sur tout ou partie du contenu
de l'actif
soft-binding-map = {
  « alg » : tstr, ; Chaîne identifiant l'algorithme de liaison souple et la version de cet algorithme utilisée
pour calculer la valeur, tirée de la liste des algorithmes de liaison souple C2PA. Si ce champ est absent,
l'algorithme est tiré de la valeur « alg_soft » de la structure englobante. Si les deux sont présents, le champ
de cette structure est utilisé. Si aucune valeur n'est présente à l'un de ces emplacements, cette structure
n'est pas valide ; il n'y a pas de valeur par défaut.
  « blocks » : [1* soft-binding-block-map],
  « pad » : octets, ; chaîne d'octets remplie de zéros utilisée pour remplir l'espace
  ? « pad2 » : octets, ; chaîne d'octets facultative remplie de zéros utilisée pour remplir l'espace
  ? « name » : tstr .size (1..max-tstr-length), ; (facultatif) description lisible par l'utilisateur de ce que
couvre ce hachage
  ? « alg-params » : bstr, ; (facultatif) chaîne d'octets CBOR décrivant les paramètres de l'algorithme de
liaison souple.
  ? « url » : uri, ; inutilisé et obsolète.
}

soft-binding-block-map = { « scope » :
  soft-binding-scope-map,
  « value » : bstr, ; Chaîne d'octets CBOR décrivant, dans un format spécifique à l'algorithme, la valeur de
```

```
la liaison souple calculée sur ce bloc de contenu numérique »
}

soft-binding-scope-map = {
  ? « extent » : bstr, ; obsolète, chaîne d'octets CBOR décrivant, dans un format spécifique à l'algorithme, la
partie du contenu numérique sur laquelle la valeur de liaison souple a été calculée »
  ? « timespan » : soft-binding-timespan-map,
  ? « region » : region-map, ; objet CBOR défini dans regions-of-interest.cddl
}

soft-binding-timespan-map = {
  « start » : uint, ; Début de la plage de temps (en millisecondes à partir du début du média) sur laquelle la
valeur de liaison souple a été calculée.
  « end » : uint,   ; Fin de la plage de temps (en millisecondes à partir du début du média) sur laquelle la
valeur de liaison souple a été calculée.
}
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous :

```
{
  « alg » : « phash »,
  « pad » : h'00',
  « url » : 32(« http://example.c2pa.org/media.mp4 »), « blocks » :
  [
    {
      « scope » : {
        « timespan » : { «
          end » : 133016
          « start » : 0,
        }
      },
      « value » : b64'dmFsdWUxCg=='
    },
    {
      « scope » : {
        « timespan » : { «
          end » : 245009
          « début » : 133017,
        }
      },

    }
  ]
}
```

18.10.3. Exigences

L'algorithme de liaison souple utilisé doit être présent en tant que valeur du champ `alg`, et les blocs sur lesquels il a été appliqué doivent être répertoriés dans le champ `blocks`. Si l'algorithme utilisé nécessite des paramètres supplémentaires, ceux-ci doivent être présents en tant que valeur de `alg-params`.

Le champ `scope` peut contenir soit un champ `region`, soit un champ `timespan` pour décrire la partie du contenu numérique sur laquelle la liaison souple a été calculée. Le champ `region`, lorsqu'il est présent, contient un objet `region-map` (tel que défini dans la section 18.2, « Régions d'intérêt »). Le champ `timespan`, lorsqu'il est présent, décrit l'intervalle de temps sur lequel la liaison souple a été calculée en millisecondes à partir du début du contenu.

18.10.4. Liste des algorithmes de liaison souple

La liste des algorithmes de liaison souple est une liste lisible par machine des valeurs autorisées pour le champ `alg`. Le champ `alg` doit correspondre au champ `alg` d'un algorithme présent dans cette liste. Le format des champs `alg-params` et `value` est spécifique à l'algorithme et décrit via une page d'informations lisible par l'utilisateur référencée par `informationalUrl` dans l'entrée `alg` de la liste.

La liste est conservée sous forme de document JSON par la C2PA à l'adresse suivante : https://github.com/c2pa-org/ softbinding-algorithm-list

Les entrées de la liste des algorithmes de liaison souple dont le champ `deprecated` est défini sur `true` doivent être considérées comme obsolètes et ne doivent pas être utilisées pour créer des assertions de liaison souple dans les manifestes. Les algorithmes de liaison souple marqués comme obsolètes peuvent être utilisés pour résoudre les liaisons souples, mais cette pratique est déconseillée.

Le schéma JSON pour les entrées de la liste des algorithmes de liaison souple est présenté ci-dessous :

```
{
     « $schema » : « https://json-schema.org/draft/2020-12/schema », « type »
     : « object »,
     "properties": {
          "identifier": {
               "type": "integer",
               "minimum": 0,
               « maximum » : 65535,
               « description » : « Cet identifiant sera attribué lorsque l'algorithme de liaison souple sera
ajouté à la liste. »
          },
          « obsolète » : { « type » :
               « booléen », « valeur
               par défaut » : faux,
               « description » : « Indique si cet algorithme de liaison souple est obsolète.
Les algorithmes obsolètes ne doivent pas être utilisés pour créer des liaisons souples. Les algorithmes
obsolètes peuvent être utilisés pour résoudre des liaisons souples, mais cette pratique est déconseillée. »
          },
          « alg » : {
               « type » : « chaîne »,

               « description » : « Espace de noms spécifique à l'entité, tel que spécifié pour les
étiquettes d'assertions C2PA, qui doit commencer par le nom de domaine Internet de l'entité, de la même
manière que les paquets Java sont définis (par exemple, « com.example.algo1 », « net.example.algos.algo2 »)
».
          },
          « type » : {
               « type » : « chaîne », «
               énumération » : [
                    « watermark », «
                    fingerprint »
               ],
               « description » : « Type de liaison souple implémenté par cet algorithme. »
          },
          « decodedMediaTypes » : { «
               type » : « array », «
               minItems » : 1,
               « items » : {
                    « type » : « chaîne »,
                    « énumération » : [
                         « application »,
```

```
                        « audio »,
                        « image »,
                        « modèle »,
                        « texte
                        », «
                        vidéo »
                    ],
                    « description » : « Type de média de niveau supérieur IANA (rendu) auquel s'applique cet
algorithme de liaison souple. »
               }
          },
          « encodedMediaTypes » : { «
               type » : « array », «
               minItems » : 1,
               « items » : {
                    « type » : « chaîne »,
                    « description » : « Type de média IANA auquel s'applique cet algorithme de liaison souple,
par exemple application/pdf »,

\\+]+)*)$"
               }
          },
          « entryMetadata » : { « type
               » : « object », «
               properties » : {
                    « description » : { «
                         type » : « chaîne »,
                         « description » : « Description lisible par l'utilisateur de l'algorithme. »
                    },
                    « dateEntered » : { «
                         type » : « string »,
                         « format » : « date-time »,
                         « description » : « Date d'entrée en vigueur de cet algorithme. »
                    },
                    « contact » : {
                         « type » : « chaîne »,
                         « format » : « email »
                    },
                    « informationalUrl » : {
                         « type » : « string
                         »,
                         « format » : « uri »,
                         « description » : « Une page Web contenant plus de détails sur l'algorithme. »
                    }
               },
               « required » : [
                    « description », «
                    dateEntered », « contact
                    », « informationalUrl »
               ]
          },
          « softBindingResolutionApis » : { « type » :
               « array »,
               « items » : {
                    « type » : « chaîne »,
                    « format » : « uri »
               },
               « description » : « Liste des API de résolution de liaison souple prenant en charge cet
algorithme."
          }
     },
     « requis » : [
          « identifier », «
          alg »,
```

```
          « type », «
          entryMetadata »
     ],
     « oneOf » : [
          {
               « requis » : [
                    « decodedMediaTypes »
               ]
          },
          {
               « requis » : [
                    « encodedMediaTypes »
               ]
          }
     ]
}
```

Un exemple JSON d'une entrée dans la liste des algorithmes de liaison souple est présenté ci-dessous :

```
{
     « identifier » : 1, «
     deprecated » : false,
     « alg » : « com.example.product », «
     type » : « watermark », «
     decodedMediaTypes » : [
          "audio",
          "video",
          « texte
          », «
          image »
     ],
     « entryMetadata » : {
          « description » : « Algorithme de tatouage numérique version 1.2 de Foo Inc. », «
          dateEntered » : « 2024-04-23T18:25:43.511Z »,
          « contact » : «foo.bar@example.com », « informationalUrl » : «
          https://example.com/wmdetails »
     },

     ]
}
```

Le nom unique de l'algorithme est indiqué dans le champ `alg` et correspond à la chaîne qui doit être utilisée dans le champ `alg` d'une assertion de liaison souple qui utilise cet algorithme. Le nom doit respecter les exigences en matière d'espaces de noms et représenter le propriétaire de l'algorithme. Un identifiant numérique unique est également attribué à chaque algorithme. Si différentes versions d'un algorithme sont fournies, chacune doit avoir une entrée distincte dans la liste des algorithmes de liaison souple.

Le type de l'algorithme doit être soit « watermark » (filigrane), soit « fingerprint » (empreinte digitale) pour indiquer que l'algorithme est un filigrane invisible ou une empreinte digitale.

Le statut de dépréciation de l'algorithme est indiqué dans le champ « `deprecated` ». Un validateur ne doit pas résoudre les liaisons souples qui utilisent des algorithmes dépréciés. Les manifestes C2PA ne doivent pas être rédigés à l'aide de liaisons souples dépréciées.

L'entrée de la liste des algorithmes de liaison souple doit contenir une liste des types de médias pris en charge, soit sous la forme `encodedMediaTypes`, soit sous la forme `decodedMediaTypes`. Les types de médias pris en charge pour `decodedMediaTypes` doivent correspondre à un ou plusieurs des types suivants

les types de médias IANA de niveau supérieur comprenant : « application », « audio », « image », « modèle », « texte », « vidéo ». Les types de médias pris en charge pour `encodedMediaTypes` doivent correspondre à un ou plusieurs sous-types IANA enregistrés d'un `decodedMediaType` répertorié dans la phrase précédente. Ces types de niveau supérieur et sous-types IANA sont répertoriés à l'adresse https://www.iana.org/assignments/media-types/media-types.xhtml

Des informations supplémentaires doivent accompagner chaque entrée de la liste des algorithmes de liaison souple, dans le champ `entryMetadata`. Il s'agit d'une description lisible par l'homme de l'algorithme (`description`) et de la date à laquelle il a été proposé pour être inscrit dans la liste des algorithmes de liaison souple (`dateEntered`).

Les coordonnées du propriétaire de l'entrée doivent être fournies sous forme d'adresse électronique (`contact`, obligatoire). Une URL informative (`informationalUrl`, obligatoire) doit être fournie, renvoyant à une page lisible par l'utilisateur décrivant les caractéristiques de l'algorithme de liaison souple. Les informations contenues dans cette page ne sont pas soumises à des contraintes, mais peuvent inclure des détails tels que la manière d'interpréter le champ `de valeur` dans le registre de liaison souple, qui est codé sous une forme spécifique à l'algorithme.

18.10.5. API de résolution de liaison souple

L'API de résolution de liaison souple est une API Web qui fournit un moyen standard de récupérer les magasins C2PA Manifest à partir d'un point de terminaison API de résolution de liaison souple à partir d'une valeur de liaison souple, d'un identifiant de manifeste ou d'un actif. L'entrée de la liste des algorithmes de liaison souple peut contenir une liste d'URI d'API de résolution de liaison souple dans le champ `softBindingResolutionApis`. Si plusieurs URI sont fournis, n'importe lequel peut être utilisé pour une résolution de liaison souple.

La spécification et la documentation de l'API sont disponibles ici.

18.10.5.1. Validation des correspondances de liaison souple

Une utilisation courante des liaisons souples consiste à découvrir le manifeste actif, à partir d'un référentiel de manifestes, pour un actif dont le manifeste C2PA est absent ou invalide.

La découverte du manifeste C2PA doit être effectuée à l'aide d'un ou plusieurs algorithmes identifiés par le champ `alg` dans la liste des algorithmes de liaison souple C2PA. La liste est conservée sous forme de document JSON par la C2PA à l'emplacement suivant : https://github.com/c2pa-org/softbinding-algorithm-list

Si un manifeste C2PA est trouvé dans un référentiel de manifestes et que ce manifeste contient une ou plusieurs assertions de liaison souple, le comparateur doit s'assurer que toutes les assertions de liaison souple du manifeste localisé correspondent aux liaisons souples utilisées pour effectuer la découverte.

Une assertion de liaison souple est considérée comme une correspondance si l'identifiant de l'algorithme (`alg`) et la valeur (`value`) décrits dans l'assertion correspondent à l'identifiant de l'algorithme (`alg`) et à la valeur (`value`) utilisés pour effectuer la correspondance. La correspondance est effectuée de la manière prescrite par l'algorithme spécifié.

18.11. Données cloud

18.11.1. Description

Dans certains cas, il est préférable de stocker les données pour l'assertion à distance, par exemple dans le cloud, plutôt que de les intégrer dans l'actif, en particulier lorsque les données sont volumineuses. Dans ce cas, il est possible d'utiliser un type spécial d'assertion qui sert de référence à ces informations. Pour des raisons de confidentialité et de fiabilité, les données référencées par une assertion de données dans le cloud doivent être considérées comme facultatives : leur contenu ne doit pas être récupéré dans le cadre de la validation du manifeste. Un validateur peut récupérer le contenu ultérieurement pour répondre à un besoin lié à l'application, tel que l'exploration plus approfondie de l'historique de provenance.

Si les métadonnées d'assertion sont incluses dans une autre assertion, elles feront également partie des informations référencées à partir d'une assertion de données cloud. Il est également possible de stocker à distance des assertions de métadonnées d'assertion individuelles, comme pour les autres types d'assertions.

Une assertion de données cloud doit porter le label `c2pa.cloud-data`.

Une assertion de données cloud ne doit pas faire référence à une assertion portant l'étiquette `c2pa.hash.data`, `c2pa.hash.boxes`, `c2pa.hash.collection.data`, `c2pa.hash.bmff.v2` (obsolète) ou `c2pa.hash.bmff.v3`.

18.11.2. Schéma et exemple

Le schéma de ce type est défini par la règle `cloud-data-map` dans la définition CDDL suivante :

```
; Assertion qui fait référence à l'assertion réelle stockée dans le cloud cloud-
data-map = {
  « label » : tstr, ; étiquette pour l'assertion basée sur le cloud (par exemple
  c2pa.actions) « size » : size-type, ; nombre d'octets de données
  « location » : $hashed-ext-uri-map, ; URL http(s) où se trouve l'assertion hébergée dans le cloud
  « content_type » : tstr .regexp « ^[-\\w.]+/[-+\\w.]+$ », ; type de média/MIME pour les données
  ? « metadata » : $assertion-metadata-map, ; informations supplémentaires sur l'assertion
}

; taille minimale 1, par multiples de 1,0
size-type = int .ge 1
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous :

```
{
  « size » : 98765,
  « label » : « c2pa.thumbnail.claim », « location »
  : {
    "url": "https://some.storage.us/foo",
    "hash": b64'zP84FPSremIrAQHlhw+hRYQdZp/+KggnD0W8opXlIQQ='
  },
  « type_de_contenu » : « application/jpeg »
}
```

18.12. Données intégrées

18.12.1. Description

Dans les versions précédentes de cette spécification, le concept de boîte de données en tant que type spécial de boîte JUMBF était utilisé pour permettre l'intégration arbitraire de données dans un manifeste C2PA, telles que des vignettes, des icônes et des ingrédients `inputTo`. Il a été déterminé que cette approche via un nouveau type de boîte introduisait des complexités inutiles et des fonctionnalités manquantes, telles que l'impossibilité de modifier les boîtes de données. En conséquence, ce concept a été abandonné au profit d'une assertion standard qui utilise une boîte de type de contenu JUMBF Embedded File standard pour contenir les données.

Une assertion de données intégrées doit avoir une étiquette commençant par `c2pa.embedded-data` et respecter les règles relatives aux étiquettes d'assertion en ce qui concerne les instances multiples. De plus, certains autres types d'assertions seront techniquement équivalents à une assertion de données intégrées, mais auront leurs propres étiquettes uniques (par exemple, `c2pa.thumbnail.claim`).

18.12.2. Détails techniques

Étant donné que l'assertion de données intégrées est basée sur une boîte de type de contenu JUMBF Embedded File, sa boîte Embedded File Description doit contenir un type de média IANA (par exemple, `image/png`) comme valeur du champ *MEDIA TYPE*, et peut contenir un nom de fichier comme valeur du champ *FILE NAME*. Le bit `External` toggle ne doit pas être activé.

REMARQUE Suffixes structurés de l'IANA (https://www.iana.org/assignments/media-type-structured-suffix/media-type-structured-suffix.xhtml), tels que `+json` et `+zip`, sont également pris en charge comme valeurs du champ *MEDIA TYPE*.

La zone Données binaires de l'assertion de données intégrées doit contenir les bits d'un fichier (tel qu'une image raster ou une invite de texte) dans le format souhaité par le générateur de revendications, mais correspondant au type de média spécifié dans la zone Description du fichier intégré.

18.13. Vignette

18.13.1. Description

Une assertion de vignette fournit une représentation visuelle approximative de l'actif à un moment précis du cycle de vie de celui-ci. Il existe actuellement deux événements spécifiques :

- importation d'ingrédients et création de revendications ;
- chacun utilisant une étiquette unique pour l'assertion.

18.13.1.1. Miniatures des revendications

Pour les vignettes d' , a créé à l'adresse claim creation time, la vignette assertion doit avoir le label `c2pa.thumbnail.claim`. Il ne doit pas y avoir plus d'une assertion de vignette avec cette étiquette dans un manifeste C2PA. Les versions précédentes de cette spécification exigeaient que le type de média du registre IANA de la vignette soit inclus dans le

nom du label (par exemple, `c2pa.thumbnail.claim.png`). Cette convention de nommage est désormais obsolète.

18.13.1.2. Miniatures des ingrédients

Lors de l'importation d'un ingrédient (voir section 10.3.2.2, « Ajout d'ingrédients »), il convient de faire référence à la vignette stockée dans le manifeste de cet ingrédient. Cependant, certains ingrédients peuvent ne pas inclure d'assertion de vignette, voire de manifeste. Dans ce cas, une nouvelle vignette de l'ingrédient doit être générée et une nouvelle assertion de vignette doit être créée dans le manifeste actif.

L'assertion de vignette d'un ingrédient doit avoir une étiquette commençant par `c2pa.thumbnail.ingredient` et respecter les règles relatives aux étiquettes d'assertion en ce qui concerne les instances multiples. Par exemple, une vignette d'ingrédient peut avoir l'étiquette `c2pa.thumbnail.ingredient 1`.

Les versions précédentes de cette spécification exigeaient que le type de média du registre IANA de la vignette soit inclus dans le nom de l'étiquette (par exemple, `c2pa.thumbnail.claim.png`). Cette convention de nommage est désormais obsolète.

Les versions précédentes de cette spécification exigeaient un suffixe `_1` pour la première instance et un seul trait de soulignement. La spécification actuelle, en adoptant une nomenclature cohérente avec toutes les assertions, utilise `c2pa.thumbnail.ingredient` pour la première instance, `c2pa.thumbnail.ingredient 1` pour la deuxième, etc. L'ancienne convention de nomenclature est désormais obsolète.

18.13.1.3. Détails techniques

Une assertion miniature est une assertion de données intégrée, mais avec une étiquette spéciale identifiant ce cas d'utilisation spécifique.

18.14. Actions

18.14.1. Description

Une assertion d'action fournit des informations sur les modifications et autres actions effectuées qui affectent le contenu de l'actif. Il y aura un ensemble d'actions, chacune déclarant *ce* qui s'est passé sur l'actif et (éventuellement) *quand* cela s'est produit, ainsi que d'autres informations possibles telles que le logiciel qui a effectué l'action. Sauf indication contraire dans la section 18.14.2, « Présence obligatoire d'au moins une assertion d'actions », l'ordre des actions dans cet ensemble n'est pas spécifié et n'implique pas l'ordre dans lequel les actions ont été effectuées.

Il existe deux versions de l'assertion d'actions : la version originale v1 (qui doit porter le label `c2pa.actions`) et la nouvelle version améliorée v2 (qui doit porter le label `c2pa.actions.v2`). Les actions sont modélisées d'après les ResourceEvents XMP, mais avec un certain nombre d'ajustements spécifiques à C2PA.

Les actions v1 sont entièrement spécifiées dans leur tableau `d'actions`. Cependant, dans la v2, une action peut être soit entièrement spécifiée dans un élément du tableau `d'actions`, soit dérivée d'un élément du tableau `de modèles` portant le même nom d'action.

Pour chaque action présente dans les tableaux `actions` ou `templates`, la valeur du champ `action` doit être soit une action prédéfinie action name (`c2pa.resized`, `c2pa.edited`, etc.) ou spécifique à l'entité action name

(`com.fabrikam.gaussianBlur`, etc.).

L'ensemble des noms prédéfinis, préfixés par `c2pa.`, est répertorié dans le tableau 8, « Liste des actions prédéfinies » :

Tableau 8. Liste des actions prédéfinies

Action	Signification
c2pa.addedText	(visible) Du contenu textuel a été inséré dans la ressource, par exemple sur un calque de texte ou sous forme de légende.
c2pa.adjustedColor	Modifications apportées à la tonalité, à la saturation, etc.
c2pa.changedSpeed	Réduction ou augmentation de la vitesse de lecture d'une piste vidéo ou audio
c2pa.color_adjustments	[OBSOLÈTE] Modifications apportées à la tonalité, à la saturation, etc.
c2pa.converted	Le format de la ressource a été modifié.
c2pa.created	La ressource a été créée pour la première fois.
c2pa.cropped	Certaines parties du contenu numérique de l'actif ont été supprimées.
c2pa.deleted	Certaines zones du contenu numérique de l'actif ont été supprimées.
c2pa.drawing	Modifications à l'aide d'outils de dessin, notamment des pinceaux ou une gomme.
c2pa.dubbed	Des modifications ont été apportées à l'audio, généralement à une ou plusieurs pistes d'un élément composite.
c2pa.edited	Actions généralisées qui seraient considérées comme des transformations éditoriales du contenu.
c2pa.edited.metadata	Modifications apportées aux métadonnées d'un élément ou à une assertion de métadonnées, mais pas au contenu numérique de l'élément.
c2pa.enhanced	Améliorations appliquées telles que la réduction du bruit, la compression multibande ou la netteté, qui représentent des transformations non éditoriales du contenu.
c2pa.filtered	Modifications de l'apparence à l'aide de filtres, de styles, etc.
c2pa.opened	Un élément existant a été ouvert et est défini comme `parentOf` ingrédient.
c2pa.orientation	Modifications apportées à la direction et à la position du contenu.
c2pa.placed	Ajout/placement d'un ou plusieurs `composants` d'ingrédients dans l'actif.
c2pa.published	L'actif est diffusé à un public plus large.
c2pa.redacted	Une ou plusieurs assertions ont été supprimées.
c2pa.removed	Un `composant` de l'ingrédient a été supprimé.

c2pa.repackaged	Conversion d'un format d'emballage ou de contenant en un autre. Le contenu est reconditionné sans transcodage. Cette action est considérée comme une transformation non éditoriale de l'ingrédient `parentOf`.
c2pa.resized	Modifications apportées aux dimensions du contenu, à la taille du fichier ou aux deux.
c2pa.transcoded	Conversion d'un encodage vers un autre, y compris le redimensionnement de la résolution, l'ajustement du débit binaire et le changement de format d'encodage. Cette action est considérée comme une transformation non éditoriale de l'ingrédient `parentOf`.
c2pa.translated	Modifications apportées à la langue du contenu.
c2pa.trimmed	Suppression d'une partie du contenu.
c2pa.unknown	Quelque chose s'est produit, mais le générateur de réclamations ne peut pas préciser quoi.
c2pa.watermarked	Un filigrane invisible a été inséré dans le contenu numérique afin de créer une reliure souple.

De plus, l'ensemble de noms prédéfinis suivant (dans le tableau 9, « Liste des actions sur les polices »), précédé du préfixe `font.` est utilisé spécifiquement pour les ressources de polices :

REMARQUE Une version antérieure de cette spécification les désignait sous le nom de `c2pa.font`, mais cela a été abandonné au profit du préfixe `font`, plus court.

Tableau 9. Liste des actions de police

Action	Signification
font.charactersAdded	Caractères ou jeux de caractères ajoutés.
font.charactersDeleted	Caractères ou jeux de caractères supprimés.
font.charactersModified	Caractères ou jeux de caractères ajoutés et supprimés.
font.createdFromVariableFont	La police a été instanciée, en tout ou en partie, à partir d'une police variable.
font.edited	La police a subi une action d'édition qui n'est décrite par aucune action plus spécifique.
font.hinted	Hinting appliqué.
font.merged	La police est une combinaison de polices antérieures.
font.openTypeFeatureAdded	Fonctionnalité OpenType ajoutée à la police.
font.openTypeFeatureModified	Fonctionnalité OpenType modifiée.
font.openTypeFeatureRemoved	Fonctionnalité OpenType supprimée de la police.
font.subset	La police a été réduite pour prendre en charge un sous-groupe arbitraire (sui generis) de caractères.

18.14.2. Présence obligatoire d'au moins une assertion d'action

Il doit être au au une actions affirmation présent dans soit la `les assertions créées` ou Tableau `gathered_assertions` de la revendication d'un manifeste C2PA standard. De plus :

- Si l'actif a été créé *de novo* (par exemple, à la suite d'une opération `Fichier → Nouveau` dans un outil créatif, de la capture d'une photo ou d'une vidéo, ou de la génération du média par un modèle d'IA générative), alors le tableau `actions` dans la première assertion `c2pa.actions` du tableau `created_assertions` ou `gathered_assertions` de la revendication doit avoir une action `c2pa.created` comme premier élément.
    - Pour tous les actifs, un champ `digitalSourceType` correspondant, avec une valeur appropriée, doit être enregistré avec l'action `c2pa.created`, afin d'indiquer la nature de l'actif à sa création. Si l'actif est créé sans contenu numérique, le champ `digitalSourceType` doit avoir la valeur `http://c2pa.org/ digitalsourcetype/empty`.
- Si l'actif a été créé en ouvrant un actif existant en tant qu'ingrédient `parentOf` pour modification, le tableau `d'actions` dans la première assertion `c2pa.actions` du tableau `created_assertions` ou `gathered_assertions` de la revendication doit avoir une action `c2pa.opened` comme premier élément. Aucun champ `digitalSourceType` n'est requis en conjonction avec une action `c2pa.opened`.

REMA Cette exigence ne s'applique pas aux manifestes de mise à jour.

RQUE Lorsque vous enregistrez des actions dans `gathered_assertions`, gardez à l'esprit que ces assertions ne sont pas attribuées au signataire (voir chapitre 10, *Revendications*).

REMA

RQUE

L'ensemble complet des assertions d'actions dans un manifeste C2PA ne doit contenir qu'une seule action de type `c2pa.created` ou `c2pa.opened`. Si l'une de ces actions apparaît dans `created_assertions`, aucune ne doit apparaître dans `gathered_assertions`, et si l'une apparaît dans `gathered_assertions`, aucune ne doit apparaître dans `created_assertions`.

EXEMPLE : un modèle d'IA générative génère une vidéo en réponse à une invite textuelle. Le manifeste actif de la ressource vidéo résultante comporterait une assertion `c2pa.actions` commençant par une action `c2pa.created`, elle-même ayant une valeur de `http://cv.iptc.org/newscodes/digitalsourcetype/trainedAlgorithmicMedia` dans le champ `digitalSourceType` correspondant.

EXEMPLE : un utilisateur ouvre Emily's Mobile Poster Maker pour créer une image destinée à être publiée sur les réseaux sociaux. L'utilisateur sélectionne un modèle, puis commence à le personnaliser, en important certaines photos existantes au cours du processus. Le manifeste actif de la ressource image résultante comporterait une assertion `c2pa.actions` commençant par une action `c2pa.created` et aucun champ `digitalSourceType`, indiquant qu'il s'agit d'un nouveau fichier. Il comporterait également une action `c2pa.placed` pour chaque photo importée par l'utilisateur, chacune pointant vers une assertion d'ingrédient correspondante où une relation `componentOf` est indiquée. Enfin, il comporterait des actions supplémentaires enregistrées pour les autres opérations effectuées par l'utilisateur.

EXEMPLE : Le service média d'un journal souhaite modifier une photo prise par un photojournaliste à l'aide d'un appareil photo compatible C2PA. Le rédacteur en chef ouvre la photo et applique des opérations de recadrage et de vignettage. Le manifeste actif de la photo modifiée contient une assertion `c2pa.actions` avec une action `c2pa.opened` pointant vers une assertion d'ingrédient pour la photo originale, où une relation `parentOf` est indiquée. Il contiendrait également des actions pour le

recadrage et à la vignette.

18.14.3. Toutes les actions incluses

La `carte d'actions v2` peut inclure un champ, `allActionsIncluded`, qui est une valeur booléenne. Si `allActionsIncluded` est présent et a une valeur true, alors le générateur de revendications indique que seules les actions répertoriées dans l'assertion d'actions ont été effectuées sur l'actif. Si `allActionsIncluded` n'est pas présent ou a une valeur false, alors un consommateur de manifeste peut supposer que d'autres actions ont été effectuées mais n'ont pas été répertoriées.

18.14.4. Champs dans l'assertion actions

18.14.4.1. Description

Une action peut inclure une description en texte libre, dans le champ `description`, de ce que fait l'action. Cela est particulièrement utile pour les actions non standard, mais peut également être utilisé pour fournir des informations supplémentaires sur une action standard. Par exemple, une action `c2pa.edited` pourrait avoir une `description` indiquant « Outil pinceau ».

18.14.4.2. Raison

Si présent, le champ « `raison` » doit contenir l'une des valeurs standard suivantes, ou une valeur personnalisée conforme à la même syntaxe que l'espace de noms spécifique à l'entité, pour justifier l'action :

- `c2pa.PII.present` ;
- `c2pa.invalid.data` ;
- `c2pa.trade-secret.present` ;
- `c2pa.government.confidential`.

REMARQUE Bien que le champ « `raison` » puisse être utilisé pour toutes les actions, seules les valeurs `c2pa` axées sur la rédaction sont définies pour le moment.

Lors de l'utilisation d'une action `c2pa.redacted`, le champ « `raison` » doit contenir la justification de la rédaction. Les exigences supplémentaires relatives à l'action `c2pa.redacted` sont disponibles à la section 18.14.4.7, « Paramètres ».

18.14.4.3. Lorsque

La date et l'heure auxquelles l'action a eu lieu peuvent également être indiquées dans le champ « `when` ». Si elles sont incluses, les valeurs du champ « `when` »
doit être conforme aux dates/heures CBOR (RFC 8949, 3.4.1).

REMARQUE Le champ « `when` » sert de simple horodatage non fiable. Il est recommandé d'utiliser des heures basées sur le temps universel coordonné (UTC).

18.14.4.4. SoftwareAgent

Le logiciel ou le matériel utilisé pour effectuer l'action peut être identifié via le champ `softwareAgent`. Dans une action v1, il s'agit d'une simple chaîne de texte. Cependant, pour la v2, `softwareAgent` utilise la structure plus riche `generator-info-map` comme

décrit à la section 10.2.3.2, « Carte d'informations du générateur ». Lorsque plusieurs agents logiciels sont utilisés, comme décrit à la section 18.14.6.2, « Agents logiciels », le champ `softwareAgentIndex` doit être utilisé pour référencer l'agent logiciel par son index basé sur 0 dans le tableau `softwareAgents`. Une action donnée ne doit avoir qu'un seul champ `softwareAgent` ou `softwareAgentIndex`.

REMARQUE

Ces champs sont utiles lorsque le `softwareAgent` n'est pas le même programme que le générateur de revendications.

Une version antérieure de cette spécification comprenait également un champ `actors`, mais celui-ci a été supprimé dans la version 2.0.

REMARQUE

18.14.4.5. Type de source numérique

Une action peut inclure une clé `digitalSourceType`, dont la valeur doit être l'un des termes définis par l'IPTC ou une valeur spécifique à la C2PA figurant dans la liste ci-dessous :

`http://c2pa.org/digitalsourcetype/empty`

Médias dont le contenu numérique est effectivement vide, comme une toile vierge ou une vidéo de longueur nulle.

`http://c2pa.org/digitalsourcetype/trainedAlgorithmicData`

Données résultant de l'utilisation algorithmique d'un modèle dérivé de contenus et de données échantillonnés. Diffère de `l'http://cv.iptc.org/newscodes/digitalsourcetype/trainedAlgorithmicMedia`, car le résultat n'est pas un type de média (par exemple, une image ou une vidéo), mais un format de données (par exemple, CSV, pickle).

REMARQUE

One common use case for the `digitalSourceType` key is in conjunction with the `c2pa.created` action permettant de préciser comment l'élément multimédia a été créé, par exemple « capture numérique », « numérisé à partir d'un négatif » ou « média algorithmique entraîné ».

Pour les ressources et les données « algorithmiques entraînées », telles que celles créées par l'IA générative, un ou plusieurs ingrédients peuvent être ajoutés au manifeste C2PA afin de fournir des informations sur les entrées qui ont conduit à la production de la ressource. Ils peuvent être référencés à partir d'une action `c2pa.placed` ou `c2pa.created`, comme indiqué dans l'exemple 8, « Exemple d'action pour l'IA générative ».

18.14.4.6. Modifications

L'action peut être spécifique à une partie seulement d'un élément, comme une série d'images dans une vidéo ou une zone spécifique d'une image. Dans la version 1, la valeur était une simple chaîne de texte. Dans la version 2, elles sont identifiées à l'aide d'un champ « `changes` », dont la valeur est un tableau d'objets « `region-map` » (comme défini dans la section 18.2, « Régions d'intérêt »).

18.14.4.7. Paramètres

Une action peut inclure une clé `de paramètres` qui permet de spécifier certaines informations spécifiques à l'action via des champs prédéfinis ainsi que l'inclusion ouverte de tout champ personnalisé (et de leurs valeurs associées). Les champs personnalisés doivent respecter la même syntaxe que l'espace de noms spécifique à l'entité, par exemple `com.litware.someFieldName`.

REMARQUE Ceci est utile pour fournir des informations supplémentaires qui seraient utiles à un workflow spécifique ou à un consommateur de manifeste C2PA.

Un générateur de revendications qui effectue la même action à plusieurs reprises, avec les mêmes paramètres et réglages, peut utiliser le champ `multipleInstances` pour indiquer si l'action a été effectuée plusieurs fois ou non. Si le champ `multipleInstances` n'est pas présent, il est impossible de savoir si l'action a été effectuée plusieurs fois.

Lors de l'utilisation d'une action `c2pa.opened` ou `c2pa.placed`, le champ `ingredient` (pour la v1) ou le champ `ingredients` (pour la v2) dans l'objet `parameters` doit contenir les URI JUMBF hachés vers une ou plusieurs assertions d'ingrédients associées. Dans une action `c2pa.removed`, ce champ doit contenir l'URI JUMBF haché vers une assertion d'ingrédient `componentOf` dans un manifeste différent. Dans certains cas, seule une partie d'un ingrédient est pertinente pour l'action. Dans ce cas, l'assertion d'ingrédient doit contenir des métadonnées d'assertion comprenant un champ `regionOfInterest` qui sera utilisé pour spécifier les régions pertinentes de l'ingrédient (comme décrit dans la section 18.15.13, « Métadonnées d'ingrédient »).

REMARQUE Dans les versions précédentes de cette spécification, les actions `c2pa.transcoded` et `c2pa.repackaged` devaient faire référence à l'assertion `parentOf` ingredient référencée par l'action précédente action `c2pa.opened` `précédente` ; les générateurs de revendications peuvent le faire pour assurer la compatibilité avec les anciens validateurs.

Lors de l'utilisation d'une action `c2pa.translated`, les champs `sourceLanguage` et `targetLanguage` de l'objet paramètres doivent contenir les codes de langue RFC 5646, BCP 47.

Exemple 8. Exemple d'action pour l'IA générative

L'action `c2pa.created` pour une image créée par un modèle d'IA générative pourrait ressembler à ceci, en notation diagnostique CBOR (RFC 8949, clause 8) :

```
// une assertion d'action utilisée pour décrire la sortie de l'IA générative //
{
  « actions » : [
    {
      « action » : « c2pa.created »,

      « softwareAgent » : {
          « nom » : « Joe's Photo Editor », « version » :
          « 2.0 »,
          « operating_system » : « Windows 10 »
      },
      « digitalSourceType » : « http://cv.iptc.org/newscodes/digitalsourcetype/trainedAlgorithmicMedia »,
      « parameters » : { « ingredients »
        : [
          {
            « url » : « self#jumbf=c2pa.assertions/c2pa.ingredient.v3 », « alg » :
            « sha256 »,
            « hash » : b64'...',
          },
          {
            « url » : « self#jumbf=c2pa.assertions/c2pa.ingredient.v3 1 », « alg »
            : « sha256 »,
            « hash » : b64'...',
          }
        ]
      }
    }
  ]
}
```

Lorsqu'une action `c2pa.redacted` est utilisée, le champ `redacted` dans l'objet `parameters` doit contenir l'URI JUMBF vers l'assertion qui a été expurgée.

18.14.5. Filigrane

Lorsque vous utilisez une action `c2pa.watermarked`, une assertion de liaison souple doit également être incluse dans le manifeste C2PA pour décrire le filigrane inséré.

18.14.6. Modèles d'action

18.14.6.1. Modèles

Les éléments du tableau `des modèles`, dans une action v2, sont décrits à l'aide d'une combinaison d'éléments communs aux actions, ainsi que de certaines valeurs spécifiques aux modèles. Ces valeurs sont combinées par un consommateur de manifeste C2PA avec actions du même nom, ou avec toutes les actions (si la valeur du champ `action` est la `valeur` spéciale `), pour obtenir une vue d'ensemble d'une action. S'il existe plusieurs modèles qui s'appliquent à la même action, les valeurs sont fusionnées en commençant par le` modèle (s'il existe), puis appliquées dans l'ordre dans lequel elles apparaissent dans le tableau `des modèles`.

Exemple 9. Exemple de modèle d'action

Une action et un modèle, en notation diagnostique CBOR (RFC 8949, clause 8) :

```
// exemple d'un modèle unique appliqué à plusieurs actions //
{
     « actions » : [
          {
               « action » : « com.joesphoto.filter », «
               when » : 0 (« 2020-02-11T09:00:00Z »)
          },
          {
               « action » : « c2pa.edited »,

          },
          {
               « action » : « com.joesphoto.filter », «
               when » : 0 (« 2020-02-11T09:20:00Z »)
          },
          {
               « action » : « c2pa.cropped »,

          }
     ],
     « templates » : [{
          « action » : « com.joesphoto.filter », «
          description » : « Filtre magique »,
```

```
          « digitalSourceType » : « http://cv.iptc.org/newscodes/digitalsourcetype/compositeSynthetic »,
          « softwareAgent » : {
               « nom » : « Joe's Photo Editor », « version
               » : « 2.0 »,
               « système_d'exploitation » : « Windows 10 »
          }
     }]
}

// exemple d'utilisation du modèle spécial toutes les actions/`*`, pour toutes les actions //
{
     « actions » : [
          {
               « action » : « c2pa.created », « when
               » : 0 (« 2024-03-09T20:04Z »)
          },
          {
               « action » : « c2pa.edited »,

          },
          {
               « action » : « c2pa.cropped »,

          }
     ],
     « templates » : [{
          « action » : « * », «
          digitalSourceType » :
« http://cv.iptc.org/newscodes/digitalsourcetype/humanEdits », «
          softwareAgent » : {
               « nom » : « Outil de création humaine de Jane », «
               version » : « 1.0 »
          }
     }]
}
```

Un consommateur de manifeste C2PA doit prendre les valeurs du modèle et les superposer aux valeurs de l'action elle-même, ce qui entraînera le remplacement de toutes les valeurs portant le même nom.

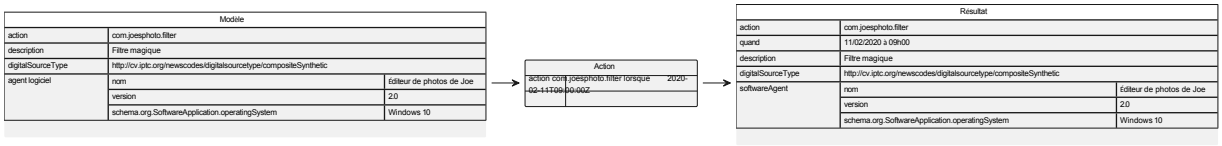

Figure 18. Flux du modèle d'actions

Un modèle peut inclure une clé `templateParameters` qui permet d'inclure d'autres clés (et leurs valeurs associées). Cela est utile pour fournir des informations supplémentaires qui seraient utiles à un workflow spécifique ou à un consommateur de manifeste C2PA.

18.14.6.2. SoftwareAgents

Si plusieurs agents logiciels ont été utilisés, ils peuvent être répertoriés dans le champ `softwareAgents`. Ce champ est un tableau d'objets `generator-info-map`, chacun décrivant un logiciel ou un matériel différent qui peut ensuite être référencé par son index via le champ `softwareAgentIndex` d'une action ou d'un modèle donné.

Exemple 10. Exemple d'agents logiciels

Exemple de spécification de plusieurs agents pour plusieurs actions, en notation diagnostique CBOR (RFC 8949, clause 8) :

```
{
    « actions » : [
        {
            « action » : « com.joesphoto.magic-avatar », « when »
            : 0 (« 2020-02-11T09:00:00Z »),
            « softwareAgentIndex » : 0
        },
        {
            « action » : « c2pa.edited »,

            « softwareAgentIndex » : 1
        },
        {
            « action » : « com.joesphoto.beauty-filter », « when
            » : 0 (« 2020-02-11T09:20:00Z »),
            « softwareAgentIndex » : 0
        },
        {
            « action » : « com.joesphoto.all-smiles », « when » :
            0 (« 2020-02-11T09:40:00Z »),
            « softwareAgentIndex » : 0
        },
        {
            « action » : « c2pa.cropped »,

            « softwareAgentIndex » : 1
        },
        {
            « action » : « com.joesphoto.green-screen », « when »
            : 0 (« 2020-02-11T09:50:00Z »),
            « softwareAgentIndex » : 0
        }
    ],
    « agents logiciels » : [
        {
            « nom » : « Filtre IA de Joe »,
            « version » : « 1.0 »,
            « operating_system » : « Windows 10 »
        }
        {
            « nom » : « Éditeur de photos de
            Joe », « version » : « 2.0 »,
            « système_d'exploitation » : « Windows 10 »
        }
    ]
}
```

18.14.6.3. Icônes

Un modèle peut également inclure une icône, c'est-à-dire une image (raster ou vectorielle) qui peut être utilisée dans l'expérience utilisateur du consommateur de manifeste C2PA afin de fournir une représentation graphique de l'action. Étant donné qu'un consommateur de manifeste connaîtra toutes les

les actions définies, ces icônes ne doivent être présentes que dans les modèles d'actions spécifiques à une entité.

La valeur du champ `icône`, s'il est présent, doit être un URI haché. Cet URI haché doit renvoyer soit à une assertion de données intégrée, soit à une assertion de données cloud. Si une assertion de données intégrée est utilisée, son libellé doit être `c2pa.icon` et doit respecter les règles relatives aux libellés d'assertion en ce qui concerne les instances multiples.

> **REMARQUE** Ce champ `d'icône` est identique dans sa structure au champ `d'icône` dans la carte d'informations du générateur de la revendication.

Les consommateurs Manifest doivent également prendre en charge l'approche « data box » recommandée par les versions antérieures de cette spécification.

18.14.7. Localisations

Si les `métadonnées` d'une assertion d'action contiennent un dictionnaire de localisation pour un modèle, les localisations s'appliquent également à toute action basée sur ce modèle.

18.14.8. Actions connexes

Lorsqu'une série d'actions sont liées entre elles et se déroulent généralement au même moment, il peut être utile de les associer en conséquence. Le champ « `related` » (`lié`) de l'action v2 permet de répertorier les actions supplémentaires qui sont liées. Chaque action liée doit être un sous-ensemble de l'action principale et ne doit inclure que les champs qui diffèrent. Tout comme pour un modèle d'action, les valeurs sont fusionnées avec celles de l'action principale par un consommateur de manifeste C2PA afin d'obtenir une vue d'ensemble de chaque action associée.

18.14.9. Rendu des ressources

Les rendus d'actifs sont monnaie courante lors de la distribution de médias sur Internet. Ces rendus sont souvent créés dans le but de fournir des médias à des consommateurs disposant de connectivités, de résolutions d'écran et d'autres environnements différents. Nous pouvons utiliser l'assertion `d'actions` pour aider les acteurs consommateurs à comprendre l'intention de certains créateurs de revendications de créer des rendus d'actifs.

La présence des seules actions `c2pa.published`, `c2pa.transcoded` et `c2pa.repackaged` dans une assertion `c2pa.actions` indique au consommateur du manifeste que le signataire affirme qu'aucune modification éditoriale n'a été apportée au contenu numérique entre les ressources composants et celle-ci.

La présence supplémentaire d'un seul élément « parentOf » fournit un signal supplémentaire au consommateur de manifeste indiquant que le signataire affirme que l'élément a été dérivé directement de ce parent.

18.14.10. Recherche de liaison souple

Lorsqu'il effectue une action `c2pa.opened` ou `c2pa.placed` avec un actif qui ne contient pas de manifeste C2PA, le générateur de revendications peut utiliser une recherche de liaison souple pour trouver le manifeste C2PA de cet actif. En cas de succès, le générateur de revendications doit ajouter le manifeste C2PA localisé comme valeur du champ `activeManifest` dans l'assertion d'ingrédient. Dans ce cas, l'assertion d'ingrédient doit également contenir un champ `softBindingsMatched` avec une valeur true et un champ `softBindingAlgorithmsMatched` dont la valeur contient au moins une entrée dans le tableau.

REMARQUE	L'ajout de ces champs indique au consommateur du manifeste C2PA qu'une recherche de liaison souple a été utilisée.
REMARQUE	Étant donné que la plupart des manifestes récupérés par liaison souple contiendront une assertion de liaison forte qui ne correspond pas à l'actif recherché, il faut s'attendre à ce que des échecs de validation soient signalés dans l'assertion d'ingrédient.

Exemple d'action d'un ingrédient montrant que son manifeste a été récupéré via une liaison souple, en notation diagnostique CBOR (RFC 8949, clause 8) :

```
// une assertion d'ingrédient dont le manifeste a été récupéré via une liaison souple //
{
  « dc:title » : « image 1.jpg », «
  dc:format » : « image/jpeg », «
  relationship » : « parentOf », «
  softBindingsMatched » : true, «
  softBindingAlgorithmsMatched » : [
    "com.foo.watermark.1"
  ]
  « activeManifest » : {
      "url": "self#jumbf=/c2pa/urn:c2pa:5E7B01FC-4932-4BAB-AB32-D4F12A8AA322", "hash":
      b64'1kjJTO108b71cL95UxgfHD3eDgk9VrCedW8n3fYTRMk='
  },
}



// une assertion d'actions pointant vers l'ingrédient //
{
  « actions » : [
    {
      « action » : « c2pa.opened »,

      « softwareAgent » : {
        « nom » : « Joe's Photo Editor », « version
        » : « 2.0 »,
        « operating_system » : « Windows 10 »
      },
      « paramètres » : { «
        ingrédients » : [
          {
            « url » : « self#jumbf=c2pa.assertions/c2pa.ingredient.v3 », « alg »
            : « sha256 »,
            « hash » : « b64'...' »
          }
        ]
      }
    }
  ]
}
```

18.14.11. Actions obsolètes

Les actions suivantes faisaient partie des versions précédentes de cette spécification et sont désormais obsolètes :

- `c2pa.copied` ;

- `c2pa.formatted` ;
- `c2pa.version_updated` ;
- `c2pa.printed` ;
- `c2pa.managed` ;
- `c2pa.produit` ;
- `c2pa.enregistré`.

Ils ne seront plus écrits dans les assertions `c2pa.actions` ou `c2pa.actions.v2`, mais pourront apparaître dans les manifestes C2PA préexistants.

18.14.12. Schéma et exemple

Le schéma pour `c2pa.actions` est défini par la règle `actions-map`, et le schéma pour `c2pa.actions.v2` est défini par la règle `actions-map-v2` dans la définition CDDL suivante :

CDDL pour les actions

```
actions-map = {
  « actions » : [1* action-items-map], ; liste d'actions
  ? « métadonnées » : $assertion-metadata-map, ; informations supplémentaires sur l'assertion
}

$action-choice /= « c2pa.addedText »
$action-choice /= « c2pa.adjustedColor »
$action-choice /= « c2pa.changedSpeed »
$action-choice /= « c2pa.color_adjustments »
$action-choice /= « c2pa.converted »
$action-choice /= « c2pa.copied »
$action-choice /= « c2pa.created »
$action-choice /= « c2pa.cropped »
$action-choice /= « c2pa.supprimé »
$action-choice /= « c2pa.dessin »
$action-choice /= « c2pa.doublé »
$action-choice /= « c2pa.edited »
$action-choice /= « c2pa.modifié.métadonnées »
$action-choice /= « c2pa.filtered »
$action-choice /= « c2pa.formaté »
$action-choice /= « c2pa.managed »
$action-choice /= « c2pa.opened »
$action-choice /= « c2pa.orientation »
$action-choice /= « c2pa.produced »
$action-choice /= « c2pa.placed »
$action-choice /= « c2pa.imprimé »
$action-choice /= « c2pa.published »
$action-choice /= « c2pa.redacted »
$action-choice /= « c2pa.removed »
$action-choice /= « c2pa.repackaged »
$action-choice /= « c2pa.resized »
$action-choice /= « c2pa.saved »
$action-choice /= « c2pa.transcoded »
$action-choice /= « c2pa.translated »
$action-choice /= « c2pa.trimmed »
$action-choice /= « c2pa.unknown »
```

```
$action-choice /= « c2pa.version_updated »
$action-choice /= « c2pa.watermarked »
$action-choice /= « font.edited »
$action-choice /= « font.subset »
$action-choice /= « font.createdFromVariableFont »
$action-choice /= « font.charactersAdded »
$action-choice /= « font.charactersDeleted »
$action-choice /= « font.charactersModified »
$action-choice /= « font.hinted »
$action-choice /= « font.openTypeFeatureAdded »
$action-choice /= « font.openTypeFeatureModified »
$action-choice /= « font.openTypeFeatureRemoved »
$action-choice /= « font.merged »
$action-choice /= tstr .regexp « ([\\da-zA-Z_-]+\\.)+[\\da-zA-Z_-]+ » buuid =

#6.37(bstr)

; REMARQUE : une version antérieure de cette spécification comprenait également un champ « acteurs », mais
celui-ci a été supprimé dans la version 2.0.
action-items-map = { « action »
  : $action-choice,
  ? « when » : tdate, ; horodatage du moment où l'action s'est produite.
  ? « softwareAgent » : tstr .size (1..max-tstr-length), ; L'agent logiciel qui a effectué l'action.
  ? « changed » : tstr .size (1..max-tstr-length), ; Liste délimitée par des points-virgules des parties de la
ressource qui ont été modifiées depuis l'historique des événements précédent. Si ce champ n'est pas présent, il
est considéré comme indéfini. Lors du suivi des modifications et lorsque la portée des composants modifiés est
inconnue, on peut supposer que tout a pu être modifié.
  ? « instanceID » : buuid, ; Valeur de la propriété xmpMM:InstanceID pour la ressource modifiée (sortie)
  ? « parameters » : parameters-map, ; Paramètres supplémentaires de l'action. Ceux-ci varient souvent en
fonction du type d'action
  ? « digitalSourceType » : tstr .size (1..max-tstr-length), ; L'un des types de source définis sur
https://cv.iptc.org/newscodes/digitalsourcetype/
}

parameters-map = {
  ? « ingrédient » : $hashed-uri-map, ; Une uri hachée vers l'assertion d'ingrédient sur laquelle cette action
agit
  ? « description » : tstr .size (1..max-tstr-length) ; Description supplémentaire de l'action
  * tstr => any
}


; Version 2 (v2) de l'assertion d'actions

$action-reason /= « c2pa.PII.present »
$action-reason /= « c2pa.invalid.data »
$action-reason /= « c2pa.tradesecret.present »
$action-reason /= « c2pa.government.confidential »
$action-reason /= tstr .regexp "([\\da-zA-Z_-]+\\.)+[\\da-zA-Z_-]+"

actions-map-v2 = {
  « actions » : [1* action-item-map-v2],            ; liste des actions
  ? « templates » : [1* $action-template-map-v2], ; liste des modèles pour les actions
  ? « softwareAgents » : [1* $generator-info-map], ; liste des logiciels/matériels ayant effectué l'action
  ? « metadata » : $assertion-metadata-map,          ; informations supplémentaires sur l'assertion
  ? « allActionsIncluded » : bool                    ; Si présent et vrai, indique qu'aucune action non
incluse dans la liste des actions n'a été effectuée.
}

action-common-map-v2 = {
```

```
  ? « softwareAgent » : $generator-info-map, ; Description du logiciel/matériel qui a effectué l'action
  ? « softwareAgentIndex » : int, ; Index basé sur 0 dans le tableau softwareAgents dans actions-map-2
  ? « description » : tstr .size (1..max-tstr-length), ; Description supplémentaire de l'action, importante pour
les actions personnalisées
  ? « digitalSourceType » : tstr .size (1..max-tstr-length), ; L'un des types de source définis sur
https://cv.iptc.org/newscodes/digitalsourcetype/ ou dans cette spécification
}

; REMARQUE : une version antérieure de cette spécification comprenait également un champ « acteurs », mais
celui-ci a été supprimé dans la version 2.0.
action-item-map-v2 = {
  « action » : $action-choice , ; le type d'action action-
  common-map-v2, ; désormais des éléments communs
  supplémentaires
  ? « when » : tdate, ; horodatage du moment où l'action s'est produite.
  ? « changes » : [1* region-map], ; liste des régions d'intérêt de la ressource qui ont été modifiées. Si ce
champ n'est pas présent, il est considéré comme indéfini.
  ? « related » : [1* action-item-map-v2], ; liste des actions connexes
  ? « reason » : $action-reason, ; la raison pour laquelle cette action a été effectuée, requise lorsque l'action
est « c2pa.redacted »
  ? « parameters » : parameters-map-v2 ; Paramètres supplémentaires de l'action. Ceux-ci varient souvent en
fonction du type d'action
}

action-template-map-v2 = {
  « action » : $action-choice / « * », ; les modèles prennent en charge l'option spéciale supplémentaire
  « * » action-common-map-v2,       ; éléments communs supplémentaires
  ? « icon » : $hashed-uri-map, ; référence hashed_uri à une assertion de données intégrée
  ? « templateParameters » : parameters-common-map-v2 ; Paramètres supplémentaires du modèle.
}

parameters-common-map-v2 = (
  * tstr => any
)

paramètres-map-v2 = {
  ? « redacted » : $jumbf-uri-type, ; Une URI JUMBF vers l'assertion redacted, requise lorsque l'action est «
c2pa.redacted ».
  ? « ingredients » : [1* $hashed-uri-map], ; Liste des URI JUMBF hachés vers les assertions d'ingrédients (v2
ou v3) sur lesquelles cette action agit
  ? « sourceLanguage » : tstr .size (1..max-tstr-length), ; Code BCP-47 de la langue source d'une action «
c2pa.translated »
  ? « targetLanguage » : tstr .size (1..max-tstr-length), ; Code BCP-47 de la langue cible d'une action «
c2pa.translated »
  ? « multipleInstances » : bool, ; cette action a-t-elle été effectuée plusieurs fois ?
  parameters-common-map-v2, ; tout élément provenant des paramètres communs
}
```

Les actions standard spécifiques aux ressources de polices sont décrites dans :

CDDL pour les actions de police

```
; Cartes, plages et paramètres pour les actions spécifiques aux polices.

; Les actions multiples sur les polices fonctionnent par rapport à des plages de
valeurs Unicode. font-unicode-range-map = {
  « start » : uint, ; Début inclusif « stop
  » : uint, ; Fin inclusive
}
```

```
; Paramètre de police utilisé par font.subset, font.charactersAdded,
; font.charactersDeleted et font.charactersModified. font-
parameter-unicode-ranges-map = {
  « ranges » : [1* font-unicode-range-map] ; Tableau de plages Unicode
}

; Plages pour les paramètres d'instanciation des polices
font-weight-range = 1..1000 ; Poids ou épaisseur valides pour la police. 400 est la valeur normale.
font-width-range = 0.0..1000.0 ; Pourcentage par rapport à la valeur normale, compris entre 0 % et 1000
%. 100 % correspond à la largeur normale.
font-slant-range = -90,0..90,0 ; Angle d'inclinaison, 0 degré correspondant à aucune inclinaison.

; Paramètres de police utilisés lors de la création d'une instance d'une police à partir d'une police variable.
; Les différents « axes de variation » pour les polices sont détaillés ici. Les noms des balises
; des différents axes sont indiqués entre parenthèses dans les commentaires de chaque
; paramètre.
font-parameter-created-from-variable-font-map = {
  ? « weight » : font-weight-range, ; Poids (wght) ou épaisseur de la police à instancier.
  ? « width » : font-width-range, ; Largeur (wdth) ou étroitesse des formes des lettres de la police à instancier.
  ? « italic » : bool, ; Obtenir la version italique (ital) de la police.
  ? « slant » : font-slant-range, ; Angle d'inclinaison (slnt) de la police.
  ? « optical-size » : int / float, ; La taille optique (opsz) de la police, généralement celle que vous souhaitez
faire correspondre à la taille de police demandée.
  * tstr => any ; Nom et type des axes personnalisés.
}
```

Exemple 11. Exemple d'une action v2

Un exemple d'action v2, en notation diagnostique CBOR (RFC 8949, clause 8), est présenté ci-dessous :

```
{
  « actions » : [
    {
      « action » : « c2pa.filtered »,

      « parameters » : {

      },
      « softwareAgent » : {
          « nom » : « Joe's Photo Editor », « version » :
          « 2.0 »,
          « operating_system » : « Windows 10 »
      }
    },
    {
      « action » : « c2pa.cropped »,

    }
  ],
  « metadata » : {

    « notes » : [
      {
        « value » : 1,
        « explication » : « Les liaisons de contenu n'ont pas été validées »
      }
    ]
  }



}
```

18.15. Ingrédient

18.15.1. Description

Lorsque des ressources sont assemblées, par exemple en plaçant une image dans un calque dans un outil d'édition d'images ou un clip audio dans une vidéo dans un outil d'édition vidéo, il est important que les informations relatives à toute revendication concernant la ressource placée soient enregistrées dans la nouvelle ressource afin de permettre de comprendre l'historique complet de la nouvelle ressource composée. Cela vaut également lorsqu'une ressource existante est utilisée pour créer une ressource dérivée ou une représentation de ressource.

Une autre utilisation courante d'un ingrédient consiste à décrire certains actifs ou données qui ont été utilisés comme entrées dans un processus, tels que les demandes d'entraînement ou d'inférence associées à un modèle d'IA/ML.

Il existe trois versions de l'assertion d'ingrédients : la version originale v1 (qui doit porter l'étiquette `c2pa.ingredient`), la version améliorée v2 (qui doit porter l'étiquette `c2pa.ingredient.v2`) et la version encore affinée v3 (qui doit porter l'étiquette `c2pa.ingredient.v3`), qui traite de la question de la validation des ingrédients après expurgation.

NOTE Comme il y aura très probablement plusieurs assertions d'ingrédients, l'utilisation de la fonction monotone augmentant index en le étiquette serait être utilisé (par exemple, `c2pa.ingredient.v3`, `c2pa.ingredient.v3 1`, `c2pa.ingredient.v3 2`).

18.15.2. Établissement d'identifiants uniques

Si l'ingrédient ajouté contient un manifeste C2PA, son identifiant unique doit être tiré de l'étiquette du manifeste de la superboîte JUMBF contenant le manifeste C2PA actif de l'ingrédient, et il n'est pas nécessaire de fournir le champ facultatif `instanceID` de l'assertion d'ingrédient. Lorsque le générateur de revendications fournit le champ facultatif `instanceID` de l'assertion d'ingrédient, la valeur de l'identifiant unique doit être déterminée comme spécifié à la section 8.3, « Identification des actifs non C2PA ».

REMARQUE Un générateur de revendications peut fournir un champ `instanceID` dans l'assertion d'ingrédient même si l'ingrédient dispose d'un manifeste C2PA.

18.15.3. Relation

Lors de l'ajout d'un ingrédient, sa relation avec l'actif actuel doit être décrite. Les valeurs possibles du champ `de relation` et leur signification sont indiquées dans le Tableau 10, « Relations entre les ingrédients ».

Tableau 10. Relations entre les ingrédients

Valeur	Signification
`parentOf`	L'actif actuel est un actif dérivé ou une représentation de cet ingrédient. Cette valeur de relation est également utilisée avec les manifestes de mise à jour.
`composantDe`	L'actif actuel est composé de plusieurs parties, dont cet ingrédient fait partie.
`inputTo`	Cet ingrédient a été utilisé comme entrée dans un processus de calcul, tel qu'un modèle d'IA/ML, qui a conduit à la création ou à la modification de cet actif.

Lors de l'ajout d'une assertion d'ingrédient, un générateur de revendications doit ajouter une assertion `c2pa.actions` (voir section 18.14, « Actions »), si celle-ci n'existe pas déjà dans le manifeste actif. En fonction du type d'ingrédient, l'une des nouvelles entrées suivantes doit être ajoutée au tableau `d'actions` d'une assertion `c2pa.actions`.

- Lors de l'ajout d'un ingrédient avec une relation `parentOf`, une action `c2pa.opened` doit être ajoutée au tableau `actions`.
- Lors de l'ajout d'un ingrédient avec une relation `componentOf`, une action `c2pa.placed` doit être ajoutée au tableau `actions`.

Cette exigence s'applique uniquement aux manifestes standard, car l'enregistrement des actions n'est pris en charge que dans ce type de manifeste.

18.15.4. Titre

Si présent, la valeur de `dc:title` doit être un nom lisible par l'utilisateur pour l'ingrédient, qui peut être tiré soit du XMP de la ressource, soit du nom de la ressource dans un système de fichiers local ou distant (par exemple, basé sur le cloud). Si l'ingrédient n'a pas de nom spécifique, une description de l'ingrédient peut être utilisée à la place.

18.15.5. Format

Si elle est présente, la valeur de `dc:format` doit être le type de média IANA pour l'ingrédient. Il est recommandé qu'un générateur de revendications fournisse ce champ et qu'il contienne une valeur valide. Lors de la description d'un ingrédient composé de plusieurs fichiers, tel que l'ensemble de données d'un modèle d'IA/ML, le champ `dc:format` doit être défini sur `multipart/mixed`.

18.15.6. Schéma et exemple

La définition CDDL pour ce type est la suivante :

```
; Affirmation décrivant un ingrédient utilisé dans l'actif ingredient-
map = {
  « dc:title » : tstr, ; nom de l'ingrédient
  « dc:format » : chaîne de format, ; type de média de l'ingrédient
  ? « documentID » : tstr, ; valeur de l'ingrédient « xmpMM:DocumentID » « instanceID » : tstr, ;
  identifiant unique, tel que la valeur de l'ingrédient
`xmpMM:InstanceID`
```

```
   "relationship": $relation-choice, ; relation entre cet ingrédient et l'actif dont il est un ingrédient.
   ? « c2pa_manifest » : $hashed-uri-map, ; référence hashed_uri au manifeste C2PA de l'ingrédient
   ? « thumbnail » : $hashed-uri-map, ; référence hashed_uri à une vignette de l'ingrédient
   ? « validationStatus » : [1* $status-map] ; statut de validation de l'ingrédient
   ? « metadata » : $assertion-metadata-map ; informations supplémentaires sur l'assertion
}

; Version 2 (v2) de l'assertion relative à l'ingrédient
; Assertion décrivant un ingrédient utilisé dans l'actif ingredient-
map-v2 = {
   « dc:title » : tstr, ; nom de l'ingrédient
   « dc:format » : format-string, ; Type de média de l'ingrédient
   « relationship » : $relation-choice, ; relation entre cet ingrédient et l'actif dont il est un ingrédient.
   ? « documentID » : tstr, ; valeur de l'ingrédient « xmpMM:DocumentID »
   ? « instanceID » : tstr, ; identifiant unique, tel que la valeur de l'ingrédient
`xmpMM:InstanceID`
   ? « data » : $hashed-uri-map / $hashed-ext-uri-map, ; référence hashed_uri à une assertion de données
intégrée ou hashed_ext_uri à des données externes
   ? « data_types » : [1* $asset-type-map], ; informations supplémentaires sur le type de données pour la
structure V2 de l'ingrédient.
   ? « c2pa_manifest » : $hashed-uri-map, ; référence hashed_uri au manifeste C2PA de l'ingrédient
   ? « thumbnail » : $hashed-uri-map, ; référence hashed_uri à une vignette dans une assertion de données intégrée
   ? « validationStatus » : [1* $status-map] ; statut de validation de l'ingrédient
   ? « description » : tstr .size (1..max-tstr-length) ; Description supplémentaire de l'ingrédient
   ? « informational_URI » : tstr .size (1..max-tstr-length) ; URI vers une page d'information sur
l'ingrédient ou ses données
   ? « metadata » : $assertion-metadata-map ; informations supplémentaires sur l'assertion
}

; Version 3 (v3) de l'assertion d'ingrédient
; Déclaration décrivant un ingrédient utilisé dans l'actif ingredient-
map-v3 = {
   ? « dc:title » : tstr, ; nom de l'ingrédient
   ? « dc:format » : format-string, ; Type de média de l'ingrédient
   « relationship » : $relation-choice, ; relation entre cet ingrédient et l'actif dont il est un ingrédient.
   ? « validationResults » : $validation-results-map, ; Résultats du générateur de revendications effectuant une
validation complète de l'actif ingrédient
   ? « instanceID » : tstr, ; identifiant unique tel que la valeur de l'ingrédient
`xmpMM:InstanceID`
   ? « data » : $hashed-uri-map / $hashed-ext-uri-map, ; référence hashed_uri à une assertion de données
intégrée ou hashed_ext_uri à des données externes
   ? « dataTypes » : [1* $asset-type-map], ; informations supplémentaires sur le type de données pour la
structure V3 des ingrédients
   ? « activeManifest » : $hashed-uri-map, ; hashed_uri vers la case correspondant au manifeste actif de
l'ingrédient
   ? « claimSignature » : $hashed-uri-map, ; hashed_uri vers la boîte Claim Signature dans le manifeste C2PA de
l'ingrédient
   ? « thumbnail » : $hashed-uri-map, ; référence hashed_uri à une vignette dans une assertion de données intégrée
   ? « description » : tstr .size (1..max-tstr-length), ; Description supplémentaire de l'ingrédient
   ? « informationalURI » : tstr .size (1..max-tstr-length), ; URI vers une page d'information sur
l'ingrédient ou ses données
   ? « softBindingsMatched » : bool, ; Si les liaisons souples ont été mises en correspondance
   ? « softBindingAlgorithmsMatched » : [1* tstr] ; Tableau des noms d'algorithmes utilisés pour
découvrir le manifeste actif
```

```
  ? « metadata » : $assertion-metadata-map ; informations supplémentaires sur l'assertion
}

format-string = tstr .regexp "^\\w+\/[-+.\\w]+$"

; Choix qui décrivent la raison pour laquelle l'ingrédient est lié à l'actif
$relation-choice /= « parentOf »
$relation-choice /= « componentOf »
$relation-choice /= « inputTo »
```

Exemple en notation diagnostique CBOR (RFC 8949, clause 8) :

```
{
  « dc:title » : « image 1.jpg », « metadata
  » : {

    « reviewRatings » : [
       {
         « value » : 5,
         « explication » : « Liens de contenu validés »
       }
    ]
  }
  « dc:format » : « image/jpeg », «
  thumbnail » : {
       « url » : « self#jumbf=c2pa.assertions/c2pa.thumbnail.ingredient », « hash »
       : b64'UjRAYWiAq4lfCRDmksWAlDJN/XtHHFFwMWymsZsm3j8='
  },
  « relation » : « parentOf », « manifeste
  actif » : {
       "url": "self#jumbf=/c2pa/urn:c2pa:5E7B01FC-4932-4BAB-AB32-D4F12A8AA322", "hash":
       b64'1kjJTO108b71cL95UxgfHD3eDgk9VrCedW8n3fYTRMk='
  },
  « signature de revendication » : {


  },
  « validationResults » : { « activeManifest
    » : {
       « success » : [
       {
         « code » : « claimSignature.validated »,

       },{
         « code » : « signingCredential.trusted »,

       },{
         « code » : « timeStamp.validated »,

       },{
         « code » : « timeStamp.trusted »,

       },{
         « code » : « assertion.hashedURI.match »,
```

```
D4F12A8AA322/c2pa.assertions/c2pa.ingredient.v3"
        }
        ],
        « informationnel » : [{
          « code » : « signingCredential.ocsp.skipped »,

        }],
        « échec » : []
      },
      « ingredientDeltas » : [{
        « ingredientAssertionURI » : « self#jumbf=/c2pa/urn:c2pa:5E7B01FC-4932-4BAB-AB32-
D4F12A8AA322/c2pa.assertions/c2pa.ingredient.v3 »,
        « validationDeltas » : { «
          success » : [],
          « informationnel » : [],
          « échec » : [{
            « code » : « assertion.hashedURI.mismatch »,

        }]
        }
      },{
        « ingredientAssertionURI » : « self#jumbf=/c2pa/urn:c2pa:F095F30E-6CD5-4BF7-8C44-
CE8420CA9FB7/c2pa.assertions/c2pa.ingredient.v3 »,
        « validationDeltas » : { «
          success » : [],
          « informationnel » : [],
          « échec » : [{
            « code » : « signingCredential.untrusted »,

        }]
        }
      }
      ]
    }
}
```

18.15.7. Champ Description

Un ingrédient peut inclure une description libre, dans le champ `Description`, indiquant ce qu'est l'ingrédient ou à quoi il sert. Cela est utile dans les cas où ni le titre ni le format ne suffisent.

18.15.8. Données sur les ingrédients

18.15.8.1. Utilisation standard

Dans certains cas d'utilisation, tels que l'IA générative, il peut être important de disposer d'ingrédients dont les données sont fournies, soit intégrées dans le manifeste C2PA, soit via une URL qui renvoie aux données. Cela est possible grâce au champ `de données` de l'ingrédient, qui utilise un `hachage URI` pour pointer vers une assertion de données intégrée ou un `hachage ext-uri` pour pointer vers une référence externe.

REMARQUE Les versions précédentes de cette spécification permettaient à `l'URI haché` de pointer vers une boîte de données.

L'utilisation d'une assertion de données intégrée implique que son contenu sera intégré dans ce manifeste C2PA et dans tout futur manifeste C2PA (sauf s'il est expurgé) qui contient cet élément en tant qu'ingrédient. Les générateurs de revendications doivent tenir compte de la taille de ce champ lorsqu'ils choisissent d'intégrer ou non des données.

Exemple 12. Exemple d'ingrédients avec données

Exemple d'ingrédients avec données, en notation diagnostique CBOR (RFC 8949, clause 8) :

```
// boîte de données de l'invite //
{
  « dc:format » : « text/plain »,
  "data" : 'pirate avec un oiseau sur l'épaule'
  "dataTypes": [{
    "type": "c2pa.types.generator.prompt",
  }]
}

// ingrédient (invite) //
{
  « dc:title » : « prompt », «
  dc:format » : « text/plain », «
  relationship » : « inputTo », «
  data » : {
    "url": "self#jumbf=c2pa.assertions/c2pa.embedded-data", "alg" :
    "sha256",
    « hash » : b64'...',
  }
}


// ingrédient (modèle) //
{
  « dc:title » : « modèle »,
  "dc:format" : "application/octet-stream", "dataTypes" :
  [
    {
      "type": "c2pa.types.generator",
    },
    {
      « type » : « c2pa.types.model.tensorflow », «
      version » : « 1.0.0 »,
    },
    {
      « type » : « c2pa.types.tensorflow.hubmodule », «
      version » : « 1.0.0 »,
    }
  ],
  « relation » : « inputTo », «
  données » : {
    « url » : « https://tfhub.dev/deepmind/bigbigan-resnet50/1?tf-hub-format=compressed », « alg » : «
    sha256 »,
    « hash » : b64'...',
  },
  « description » : « Modèle d'apprentissage de génération et de représentation d'images BigBiGAN
non supervisé, entraîné sur ImageNet avec une architecture d'encodeur plus petite (ResNet-50). »,
  « informationalURI » : « https://tfhub.dev/deepmind/bigbigan-resnet50/1 »,
}
```

Il existe également des cas d'utilisation où il est important d'identifier les informations relatives aux données des ingrédients, mais où il n'est pas possible de les intégrer ni de fournir une URL valide, par exemple lors de la description de l'utilisation d'un modèle d'IA privé/interne. Dans ces cas, un type d'actif, en tant que valeur du champ `data_types`, peut être fourni pour plus de clarté sur le format et la description de ces données.

Exemple 13. Exemple d'ingrédients avec data_types

Exemple d'ingrédient sans URI haché, en notation diagnostique CBOR (RFC 8949, clause 8) :

```
// ingrédient (modèle privé) //
{
     « dc:title » : « modèle »,
     "dc:format": "application/octet-stream", "relationship":
     "inputTo",
     "dataTypes": [
          {
               « type » : « c2pa.types.generator »,
          },
          {
               « type » : « c2pa.types.model.tensorflow », «
               version » : « 1.5.0 »,
          }
     ],
     « description » : « Modèle d'IA génératif privé de Joe », « informationalURI » : «
     https://www.example.com/joes-model-info.html »
}
```

18.15.8.2. Ingrédients multi-fichiers

Dans certains cas, un ingrédient peut être représenté comme un ensemble de plusieurs fichiers, tels que l'ensemble de données d'entraînement pour un modèle d'IA/ML. Dans ces cas, il est recommandé d'inclure le manifeste C2PA dans l'assertion d'ingrédient et que le manifeste C2PA pour l'ensemble complet de données comprenne une assertion de référence d'actif qui indique où trouver ces fichiers.

REMARQUE Cette méthode est particulièrement adaptée lorsque vous travaillez avec une collection d'actifs dont tous les fichiers ne sont pas contenus dans la même hiérarchie.

18.15.9. URI d'information

Lorsqu'il est nécessaire de fournir une URL vers une page web contenant des informations sur l'ingrédient, telles que des informations détaillées sur un modèle d'IA/ML, celle-ci doit être placée comme valeur du champ `informationalURI` de l'assertion d'ingrédient.

REMARQUE L'`URI informationnel` n'est pas un lien authentifié vers le contenu de l'ingrédient lui-même, mais quelque chose qui intéresse plus généralement un utilisateur humain.

REMARQUE Ancien (et obsolètes) versions de l' ingrédient affirmation nommée ce champ `URI_informative`.

18.15.10. Vignettes

Lors de l'ajout d'un ingrédient, il peut être utile d'inclure également une vignette de l'ingrédient afin de déterminer l'état de celui-ci au moment de l'importation. À cette fin, une vignette doit être ajoutée en tant qu'assertion de vignette et référencée ici via une référence hashed-uri.

Les consommateurs de manifestes doivent également prendre en charge l'approche par boîte de données recommandée dans les versions précédentes de cette spécification.

18.15.11. Manifestes existants

18.15.11.1. Général

Si l'ingrédient dispose d'un magasin de manifestes C2PA existant, tous les manifestes C2PA du magasin de manifestes C2PA de l'ingrédient qui ont été validés et qui n'existent pas déjà dans le magasin de manifestes C2PA de l'actif doivent être copiés par le générateur de revendications dans le magasin de manifestes C2PA de l'actif, sauf dans les cas décrits à la section 18.15.12, « Copie des manifestes existants » ou lorsqu'il est demandé de ne pas le faire (par exemple via une entrée utilisateur ou via la configuration).

Le générateur de revendications doit également copier dans le magasin de manifestes C2PA de l'actif tous les manifestes C2PA supplémentaires qui n'ont pas été validés, ainsi que toutes les boîtes JUMBF et superboîtes supplémentaires apparaissant dans le magasin de manifestes C2PA qui ne sont pas reconnues comme des manifestes C2PA.

REMARQUE	La copie de ces éléments supplémentaires permet d'assurer la compatibilité avec les assertions personnalisées et les futures qui pourraient faire référence à des éléments du magasin de manifestes C2PA d'une manière que le générateur de revendications ne reconnaît pas.

18.15.12. Copie des manifestes existants

18.15.12.1. Déterminer le besoin

Pour déterminer si un manifeste existant provenant du magasin de manifestes C2PA de l'ingrédient doit être copié dans le magasin de manifestes C2PA de l'actif, le générateur de revendications doit :

1. Valider l'ingrédient conformément au processus décrit à la section 18.15.12.4, « Validation des ingrédients ». En cas d'échec de la validation, le générateur de revendications peut ignorer le reste de ces étapes s'il en reçoit l'instruction (par exemple, via une saisie utilisateur ou via la configuration).
2. Pour chaque manifeste du magasin de manifestes C2PA de l'ingrédient, comparer son identifiant URN avec les identifiants URN de chaque manifeste C2PA déjà présent dans le magasin de manifestes C2PA de l'actif.
    a. Si une correspondance est trouvée, calculez et comparez le hachage de la boîte de manifeste du magasin de manifestes C2PA de l'ingrédient au hachage de la boîte de manifeste correspondante du magasin de manifestes C2PA de l'actif.
        i. Si les hachages correspondent, le générateur de revendications ne doit pas copier le manifeste du magasin de manifestes C2PA de l'ingrédient vers le magasin de manifestes C2PA de l'actif.
        ii. Si les hachages ne correspondent pas :

A. Le générateur de revendications doit vérifier si des assertions provenant de l'un ou l'autre manifeste ont été expurgées (en utilisant éventuellement la liste des expurgations compilée dans le processus de validation explicite).

I. Si le validateur est en mesure de déterminer que les hachages diffèrent uniquement en raison de la rédaction, alors :

1. Si toutes les rédactions ont été appliquées au manifeste déjà présent dans le magasin de manifestes C2PA de l'actif, le générateur de revendications ne doit pas copier le manifeste du magasin de manifestes C2PA de l'ingrédient dans le magasin de manifestes C2PA de l'actif.

2. Si toutes les suppressions ont été appliquées au manifeste du magasin de manifestes de l'ingrédient, le générateur de revendications doit remplacer le manifeste du magasin de manifestes C2PA de l'actif par le manifeste du magasin de manifestes C2PA de l'ingrédient.

3. Si différentes rédactions ont été appliquées à la fois au manifeste C2PA provenant du magasin de manifestes C2PA de l'ingrédient et au magasin de manifestes C2PA de l'actif, le générateur de revendications doit alors rédiger autant d'assertions que nécessaire à partir du manifeste existant dans le magasin de manifestes C2PA de l'actif afin d'obtenir une union des deux ensembles de rédactions.

II. Dans tous les autres cas, le générateur de revendications doit copier le manifeste du magasin de manifestes C2PA de l'ingrédient, le renommer avec un URN mis à jour conformément au processus décrit dans la section Identifiants uniques, et insérer la version renommée dans le magasin de manifestes C2PA de l'actif.

> Le processus permettant de déterminer si les hachages diffèrent uniquement en raison de la rédaction est laissé à la discrétion du validateur.

REMARQUE

18.15.12.2. Exemples

EXEMPLE : prenons le cas d'un générateur de revendications `D` qui importe des ingrédients. Il commence par importer l'ingrédient `B`, qui contient lui-même un manifeste `A`. Après avoir validé les deux manifestes, le générateur de revendications `D` copie les manifestes `B` et `A` dans le magasin de manifestes C2PA de l'actif `D`. Il importe ensuite l'ingrédient C, qui comprend également une version expurgée du manifeste A. Après avoir validé le manifeste C et le manifeste A expurgé, il compare les hachages des deux versions du manifeste A. Sachant que la version du manifeste A dans l'ingrédient C a été expurgée, le générateur de revendications D écrase la version du manifeste A déjà présente dans le magasin de manifestes C2PA de l'actif D avec la version expurgée du manifeste A provenant de l'ingrédient C.

EXEMPLE : considérons le même scénario que ci-dessus, sauf que la version du manifeste `A` dans l'ingrédient `C` a échoué à la validation car l'une de ses assertions a échoué à la comparaison de hachage. Dans cette situation, le générateur de revendications `D` copie le manifeste `A` de l'ingrédient `C`, le renomme avec un nouvel URN et place la copie renommée dans le magasin de manifestes C2PA de l'actif `D`.

REMARQUE

Un magasin de manifestes C2PA peut contenir des boîtes JUMBF ou des superboîtes qui ne sont pas des manifestes C2PA. Elles n'ont pas besoin d'être copiées dans le cadre de ce processus.

18.15.12.3. Ajout de références de manifeste à l'assertion d'ingrédient

Si le manifeste actif de l'ingrédient a été copié dans le magasin de manifestes C2PA de l'actif, une référence URI à la boîte de manifeste C2PA active de l'ingrédient doit être stockée comme valeur du champ `activeManifest` dans l'assertion d'ingrédient, et une référence URI supplémentaire à la boîte de signature de revendication C2PA du manifeste actif doit être stockée comme valeur de `claimSignature`.

Pour un manifeste C2PA présent dans le magasin de manifestes C2PA, `les hashed_uri doivent être utilisés comme valeurs pour` les champs `activeManifest` et `claimSignature de l'assertion d'ingrédient`.

REMARQUE La fourniture des deux valeurs permet une validation efficace des ingrédients et prend également en charge la validation si l'une des assertions de l'ingrédient a été expurgée.

18.15.12.4. Validation des ingrédients

18.15.12.4.1. Généralités

En outre, lorsque l'assertion relative à l'ingrédient fait référence à un manifeste C2PA, le générateur de revendications doit également agir en tant que validateur, en effectuant la validation de l'ingrédient comme décrit dans les étapes de validation. Le résultat de cette validation (tous les codes de réussite, codes d'information et codes d'échec) doit être utilisé pour remplir le champ `validationResults` ou `validationStatus` de l'assertion relative à l'ingrédient, comme décrit ci-dessous. Ce champ doit être présent afin de pouvoir être utilisé dans de futures validations.

NOTE

La présence d'un `validationStatus` (ingrédient v2) ou d'`un validationResults` (ingrédient v3) avec un statut d'échec est considérée comme une déclaration explicite du générateur de revendications selon laquelle un acteur a reconnu des erreurs de validation dans la revendication C2PA de l'ingrédient lui-même et a choisi de poursuivre avec incorporer l'ingrédient.

Comme décrit dans la section 15.3, « Affichage des informations du manifeste », il est souhaitable qu'un générateur de revendications affiche des avertissements bien visibles afin qu'un acteur utilisant un actif dont l'historique de provenance est erroné puisse prendre une décision éclairée sur la marche à suivre.

18.15.12.4.2. Assertions d'ingrédients V2 (OBSOLÈTES)

Dans une assertion d'ingrédient v2 sans champ `c2pa_manifest`, le champ `validationStatus` est facultatif, mais s'il est présent, il peut contenir un tableau vide.

Dans une assertion d'ingrédient v2 avec le champ `c2pa_manifest`, chaque objet du tableau `validationStatus` comprend un champ `code` dont la valeur décrit l'état de validation d'une partie spécifique du manifeste, ainsi qu'un champ `success` facultatif dont la valeur booléenne indique si le code reflète une réussite (vrai) ou un échec (faux). Une description facultative de l'état de validation peut être présente dans le champ `explanation` s'il est nécessaire d'ajouter une explication lisible par l'utilisateur. En outre, chaque objet `status-map` possède un champ `url` qui doit contenir, en cas d'échec, une référence URI JUMBF à l'élément spécifique du manifeste auquel le statut fait référence. En fonction du code, `l'url` renverra vers u n e revendication C2PA, u n e signature de revendication C2PA ou u n e assertion C2PA spécifique. Les codes de statut sont définis dans la section 15.2.2, « Codes de statut standard ».

Les codes d'état personnalisés sont autorisés lorsqu'un générateur de revendications a besoin d'enregistrer certaines informations d'état spécifiques au processus. Le code doit être conforme à la même syntaxe que les espaces de noms spécifiques à l'entité (par exemple `com.litware.malformedFrobber`) et l'objet `validationStatus` doit contenir une valeur booléenne `de réussite`.

18.15.12.4.3. Assertions d'ingrédients V3

Dans une assertion d'ingrédient v3 sans champ `activeManifest`, le champ `validationResults` ne doit pas être présent. Dans une assertion d'ingrédient v3 avec un champ `activeManifest`, le champ `validationResults` doit contenir un objet `validation-results-map` qui contient à son tour :

1. Dans `activeManifest`, les résultats complets de la validation pour le manifeste actif de l'ingrédient.
2. Dans `ingredientDeltas`, les résultats de validation delta pour chaque assertion d'ingrédient contenant un champ `activeManifest` dans chaque manifeste du magasin de manifestes C2PA de l'ingrédient. Les résultats de validation delta pour une assertion d'ingrédient doivent contenir les éléments suivants :
    a. Dans `ingredientAssertionURI`, l'URI de l'assertion d'ingrédient.
    b. Dans `validationDeltas`, les résultats de validation pour le manifeste référencé par l'assertion d'ingrédient, en omettant toutes les valeurs d'état présentes dans le champ `activeManifest` du champ `validationResults` dans l'assertion d'ingrédient (ou pour les assertions d'ingrédient v1 ou v2, le champ `validationStatus`). Cette comparaison des valeurs d'état doit tenir compte du type d'état (réussite, information ou échec), `du code` et `de l'URL`, en ignorant les autres champs.

EXEMPLE : Prenons un manifeste `E` à plusieurs ingrédients avec une lignée complexe. Le générateur de revendications `E` ajoute les manifestes `C` et `D` en tant qu'ingrédients via des assertions d'ingrédients. Le manifeste C lui-même comporte les manifestes A et B ajoutés via des assertions d'ingrédients. Le manifeste D comporte également le manifeste A ajouté via une assertion d'ingrédient. Lors de l'ajout du manifeste C, le générateur de revendications E crée une assertion d'ingrédient avec un objet `validationResults` qui stocke les résultats de validation pour le manifeste actif de C dans `activeManifest`, et les résultats de validation delta pour les manifestes A et B dans `ingredientDeltas`. Le tableau `ingredientDeltas` comportera deux éléments : un pour les résultats delta comparés à l'objet `activeManifest` dans l'objet `validationResults` dans l'assertion d'ingrédient de Manifest C de Manifest B (avec un lien `hashed-uri` vers ladite assertion d'ingrédient dans Manifest C), et un autre élément avec les mêmes attributs mais pour l'assertion d'ingrédient de Manifest C de Manifest A. De même, lors de l'ajout de Manifest D, le générateur de revendications E crée une assertion d'ingrédient qui stocke `les validationResults` pour `l'activeManifest` de D, ainsi que `les ingredientDeltas` avec un seul élément de tableau contenant les résultats de validation delta comparés à l'objet `activeManifest` dans l'objet `validationResults` dans l'assertion d'ingrédient du manifeste D du manifeste A.

REMARQUE Bien qu'il s'agisse d'un exemple volontairement artificiel, il est conçu pour illustrer la manière dont la structure de données `validationResults` doit être utilisée.

Chaque résultat de validation (tel que décrit à l'aide d'une `carte des codes d'état`) se compose d'un tableau de codes `de réussite`, `d'information` et `d'échec`. Chaque code est représenté par un objet `de carte d'état` qui doit contenir un champ `code` avec le code d'état. En outre, il peut contenir un champ `url` avec une référence URI JUMBF vers le spécifique

élément dans le manifeste auquel se réfère le statut, et un champ `explicatif` facultatif contenant une explication lisible par l'utilisateur du statut. Les codes de statut sont définis dans la section 15.2.2, « Codes de statut standard ».

Les codes d'état personnalisés sont autorisés lorsqu'un générateur de revendications a besoin d'enregistrer certaines informations d'état spécifiques au processus. Le code doit respecter la même syntaxe que les espaces de noms spécifiques à l'entité (par exemple, `com.litware.malformedFrobber`).

18.15.13. Métadonnées sur les ingrédients

Comme décrit dans les métadonnées d'assertion, le champ de `métadonnées` de l'assertion d'ingrédient peut contenir des métadonnées sur l'ingrédient, telles que la date et l'heure de sa génération ou d'autres données pouvant aider les consommateurs de manifestes à prendre des décisions éclairées sur la provenance ou la véracité des données d'assertion.

Le champ `de métadonnées` est couramment utilisé lorsqu'une partie seulement d'un ingrédient est utilisée dans la création ou la modification d'un actif. Dans ce cas, le champ `de métadonnées` doit contenir un champ `regionOfInterest` (tel que décrit à la section 18.3.6, « Région d'intérêt ») qui décrit les parties pertinentes de l'ingrédient qui ont été utilisées. Vous trouverez un exemple dans l'exemple 14, « Exemple d'ingrédient avec des métadonnées contenant des régions ».

NOTE	Bien que le champ ne contienne qu'une seule région d'intérêt, l'objet `region-map` peut spécifier plusieurs régions comme valeurs de son champ `region`. Cela peut être utile lorsque plusieurs parties d'un même ingrédient sont concernées.

Exemple 14. Exemple d'ingrédient avec métadonnées contenant des régions

Exemple d'ingrédient contenant une région d'intérêt dans ses métadonnées, en notation diagnostique CBOR (RFC 8949, clause 8) :

```
{
  « dc:title » : « someVideo.mp4 », «
  metadata » : {
     "regionOfInterest" : {
          « description » : « 10 secondes d'audio », «
          région » : [
               {
                    « type » : « temporel », «
                    temps » : {
                         « type » : « npt »,
                         « début » : « 10 »,
                         « fin » : « 20 »
                    }
               },
               {
                    « type » : « identifié », «
                    élément » : {
                         « identifiant » : « track_id », «
                         valeur » : « 3 »
                    }
               }
          ]
     }
  }
  « dc:format » : « video/mp4 »,
```

```
   « relation » : « composantDe », «
   manifesteActif » : {
        "url": "self#jumbf=/c2pa/urn:c2pa:98782815-5116-4d78-93de-3f5d8b4f4615", "hash":
        b64'TEWww2UCIR/e8mmR0XvzkFVZYTJ59Q8Ip4nkYxrS/Ys='
   },
   « claimSignature » : {

        « hash » : b64'ICJkYzpmb3JtYXQiOiAiaW1hZ2UvanBlZyIsCiAgImR='
   },
   « validationResults » : { ... }
}
```

18.15.14. Liens souples

Un manifeste actif peut inclure un manifeste C2PA en tant qu'ingrédient (via une relation `parentOf`) qui a été découvert à l'aide d'une recherche de liaison souple. Si le générateur de revendications inclut un tel manifeste C2PA, il doit alors inclure un champ `softBindingsMatched` indiquant « true » et un champ `softBindingAlgorithmsMatched` contenant un tableau de chaînes (de noms d'algorithmes de liaison souple qui ont été utilisés pour découvrir le manifeste C2PA ingrédient). Les noms des algorithmes doivent être répertoriés dans la liste des algorithmes de liaison souple C2PA, tels qu'identifiés dans le champ `alg` des entrées de cette liste.

18.16. Métadonnées

18.16.1. Description

Dans les versions précédentes de cette spécification, il existait des assertions individuelles pour chaque norme de métadonnées (par exemple, IPTC, EXIF). Dans cette version, il existe désormais une catégorie d'assertions qui doit être utilisée pour représenter les métadonnées, dans une sérialisation normalisée. Le fait d'avoir les métadonnées dans une assertion établit que les métadonnées dans cette assertion sont significatives, car elles ont été explicitement incluses dans le manifeste C2PA et signées par un signataire spécifique, ce qui permet la validation cryptographique et l'attribution des données. De plus, l'utilisation d'une sérialisation commune permet aux consommateurs du manifeste de le traiter de manière cohérente.

REMARQUE Ces assertions peuvent représenter des normes existantes ou des spécifications privées.

18.16.2. Exigences communes

Une assertion de métadonnées doit avoir une étiquette qui se termine par la chaîne `.metadata` et qui est précédée soit de l'identifiant standard `c2pa`, soit de tout autre identifiant conforme à la même syntaxe que les espaces de noms spécifiques à l'entité. Par exemple, une assertion `com.litware.metadata` serait valide.

Chaque assertion de métadonnées doit contenir une seule boîte de type de contenu JSON contenant la sérialisation JSON-LD d'une ou plusieurs valeurs de métadonnées. La propriété `@context` dans l'objet JSON-LD doit être incluse et utilisée pour fournir le contexte/les espaces de noms pour les normes de métadonnées spécifiées. La procédure recommandée pour créer cet objet JSON-LD consiste à créer d'abord une représentation du modèle de données XMP des métadonnées, puis à la sérialiser en JSON-LD selon la sérialisation JSON-LD de XMP. Le JSON-LD serait alors stocké sous forme de boîte de type de contenu JSON.

18.16.3. L'assertion `c2pa.metadata`

Cette spécification définit une assertion de métadonnées, dont l'étiquette est `c2pa.metadata`, qui est utilisée pour représenter un sous-ensemble de schémas de métadonnées courants pouvant être utilisés dans tout manifeste C2PA. Les champs de métadonnées pouvant être inclus dans cette assertion sont documentés dans l'annexe B, *Détails de mise en œuvre pour `c2pa.metadata`*.

REMARQUE Les assertions de métadonnées avec étiquette personnalisée peuvent contenir n'importe quelle valeur provenant de n'importe quel schéma.

Exemple 15. Assertion `c2pa.metadata` pour une image

Exemple d'assertion `c2pa.metadata` pour une image :

```
{
     "@context" : {
          « exif » : « http://ns.adobe.com/exif/1.0/ », « exifEX
          » : « http://cipa.jp/exif/2.32/ », « tiff » : «
          http://ns.adobe.com/tiff/1.0/ »,
          « Iptc4xmpExt » : « http://iptc.org/std/Iptc4xmpExt/2008-02-29/ », « photoshop » :
          « http://ns.adobe.com/photoshop/1.0/ »
     },
     « photoshop:DateCreated » : « 31 août 2022 », «
     Iptc4xmpExt:DigitalSourceType » :
« http://cv.iptc.org/newscodes/digitalsourcetype/digitalCapture », «
     exif:GPSVersionID » : « 2.2.0.0 »,
     « exif:GPSLatitude » : « 39,21.102N », «
     exif:GPSLongitude » : « 74,26.5737W », «
     exif:GPSAltitudeRef » : 0, « exif:GPSAltitude
     » : « 100963/29890 », "exif:GPSTimeStamp":
     "18:22:57",
     « exif:GPSDateStamp » : « 2019:09:22 », «
     exif:GPSSpeedRef » : « K », « exif:GPSSpeed »
     : « 4009/161323 », « exif:GPSImgDirectionRef
     » : « T », « exif:GPSImgDirection » : «
     296140/911 », « exif:GPSDestBearingRef » : «
     T », « exif:GPSDestBearing » : « 296140/911 »,
     « exif:GPSHPositioningError » : « 13244/2207 », «
     exif:ExposureTime » : « 1/100 », « exif:FNumber » :
     4,0,
     « exif:ColorSpace » : 1,
     « exif:DigitalZoomRatio » : 2,0, « tiff:Make
     » : « CameraCompany », « tiff:Model » : «
     Shooter S1 », « exifEX:LensMake » : «
     CameraCompany », « exifEX:LensModel » : «
     17,0-35,0 mm »,
     « exifEX:LensSpecification » : { « @list » : [ 1,55, 4,2, 1,6, 2,4 ] }

}
```

Exemple 16. Assertion `c2pa.metadata` pour un PDF

Exemple d'assertion `c2pa.metadata` pour un fichier PDF :

```
{
```

```
    « @context » : {
       "dc" : "http://purl.org/dc/elements/1.1/", "xmp" :
       "http://ns.adobe.com/xap/1.0/", "pdf" :
       "http://ns.adobe.com/pdf/1.3/", "pdfx":
       "http://ns.adobe.com/pdfx/1.3/"
    },
    « dc:created » : « 3 février 2015 », « dc:title
    » : [
        « Ceci est un fichier test »
    ],
    « xmp:CreatorTool » : « TeX », « pdf:Producer »
    : « pdfTeX-1.40.14 », « pdf:Trapped » : «
    Inconnu »,
    « pdfx:PTEX.Fullbanner » : « Ceci est pdfTeX, version 3.1415926-2.5-1.40.14 (TeX Live 2013)
kpathsea version 6.1.1 »
}
```

18.16.4. Rédaction de `c2pa.metadata`

Bien que le processus de rédaction fonctionne de telle manière que seule une assertion entière puisse être rédigée (voir section 6.8, « Rédaction des assertions »), l'utilisation d'un manifeste de mise à jour permet une rédaction partielle en supprimant l'original, puis en plaçant les nouvelles versions réduites dans le manifeste de mise à jour. Cette nouvelle assertion serait présentée dans une expérience utilisateur en association avec le signataire du manifeste de mise à jour et non avec le signataire du manifeste C2PA qui a été expurgé.

Par exemple, une assertion de métadonnées contenant à la fois des données de localisation et des informations sur la caméra dont les données de localisation doivent être expurgées pourrait être effectuée via un manifeste de mise à jour avec une nouvelle assertion de métadonnées contenant uniquement les informations sur la caméra.

18.17. Horodatages

18.17.1. Description

Dans certains flux de travail liés à la provenance, un manifeste standard ou de mise à jour est créé hors ligne, où il n'est pas possible d'obtenir un horodatage fiable (conformément à la norme RFC 3161) auprès d'une TSA au moment de la signature. Cependant, dans de tels cas, ces certificats de signature expireront après un certain temps, ce qui rendra le manifeste C2PA invalide.

Pour éviter cette expiration, un horodatage fiable peut être ajouté ultérieurement (à condition que le certificat n'ait pas encore expiré), fournissant ainsi une « preuve d'existence » pour ce manifeste C2PA et (dans le cas du manifeste actif) l'actif qui lui est associé. Cette assertion d'horodatage est utilisée pour fournir un horodatage fiable pour ces manifestes C2PA.

18.17.2. Schéma et exemple

Le schéma de ce type est défini par la règle `time-stamp-map` dans la définition CDDL suivante :

```
; Les structures de données utilisées pour stocker un tableau de
; URN manifestes pour horodater des « blobs »
$time-stamp-map /= {
```

```
  * $$time-stamp-entry => bstr
}

time-stamp-entry /= tstr .regexp "^urn:c2pa:[\\da-zA-Z_-]+$"
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous :

```
{
  « urn:c2pa:d61c74e0-ce26-4439-b92d-690dcce6b58e » : h'...', «
  urn:c2pa:ab8c2751-8711-455a-9a8b-37143bfc92c2 » : h'...'
}
```

18.17.3. Exigences

Une assertion d'horodatage doit avoir une étiquette `c2pa.time-stamp`, et il ne doit y avoir qu'une seule assertion d'horodatage par manifeste C2PA.

L'assertion d'horodatage consiste en une carte CBOR (définie comme une `carte d'horodatage`) qui doit contenir au moins une paire clé-valeur (définie comme une `entrée d'horodatage`). La clé doit être l'URN du manifeste C2PA, tel que défini ici, pour le manifeste C2PA qui est horodaté, et la valeur doit être une chaîne d'octets CBOR dont le contenu est décrit dans le paragraphe suivant.

La valeur de chaque `entrée d'horodatage` doit être la même que les données binaires figurant dans le champ `timeStampToken` de la structure `TimeStampResp` reçue en réponse d'une autorité d'horodatage (TSA) conforme à la norme RFC 3161 (RFC 3161) utilisant le mode de contenu détaché. La structure `TimeStampResp` elle-même doit être obtenue en utilisant le même processus que celui décrit à la section 10.3.2.5.3, « Obtention de l'horodatage », à l'exception que la valeur de `la charge utile` doit être la valeur du champ `de signature` de la structure `COSE_Sign1_Tagged` contenue dans la boîte C2PA Claim Signature du manifeste C2PA qui est horodaté.

18.18. Statut du certificat

18.18.1. Description

Dans certains flux de travail liés à la provenance, un manifeste standard ou de mise à jour est créé hors ligne, où il n'est pas possible d'obtenir les informations de révocation (via OCSP) au moment de la signature. Sans ces informations disponibles pendant le processus de validation, un validateur peut avoir besoin de se connecter à Internet pour déterminer le statut de révocation du certificat. Cette assertion est utilisée pour fournir le statut de certificat de confiance pour ces manifestes C2PA, en ajoutant les informations après coup.

18.18.2. Schéma et exemple

Le schéma de ce type est défini par la règle `cert-status-map` dans la définition CDDL suivante :

```
certificate-status-map = { « ocspVals »
  : [1* bstr]
}
```

Un exemple au format CBOR Diagnostic Format (`.cbordiag`) est présenté ci-dessous :

```
{
  « ocspVals » : [ h'...',
    h'...'
  ]
}
```

18.18.3. Exigences

Une assertion d'état de certificat doit avoir une étiquette `c2pa.certificate-status`, et un manifeste C2PA doit contenir au maximum une assertion d'état de certificat.

L'assertion d'état du certificat se compose d'une carte CBOR (définie comme une `carte d'état du certificat`) et doit contenir au moins une entrée dans le tableau `ocspVals`. Comme décrit dans la section 14.5.2, « Révocation de certificat », le générateur de revendications interroge le service OCSP indiqué par le certificat de signature, capture la réponse et doit la stocker dans le même format binaire que celui utilisé lorsqu'elle est stockée en tant qu'élément du tableau `ocspVals` de l'en-tête `rVals` (voir l'exemple 3, « CDDL pour rVals »).

18.19. Référence d'actif

18.19.1. Description

Cette assertion est utilisée pour indiquer un ou plusieurs emplacements où une copie de la ressource peut être obtenue. Ces emplacements doivent être décrits à l'aide d'une assertion de référence de ressource. L'emplacement doit être exprimé via un URI. L'URI peut renvoyer à une seule ressource ou faire référence à un répertoire. Dans ce dernier cas, il sert à fournir l'emplacement d'une collection de ressources, qui serait hachée via un hachage de données de collection.

NOTE

L'expression d'un `URI` offre une certaine flexibilité pour obtenir la ressource à partir d'emplacements Web ou distribués. de systèmes de fichiers tels que comme IPFS (voir https://docs.ipfs.tech/how-to/address-ipfs-on-web/#subdomain-gateway pour ce dernier).

Une assertion de référence d'actif doit avoir une étiquette `c2pa.asset-ref`.

L'horodatage dans les métadonnées de l'assertion sert de base pour déterminer la fraîcheur du lien décrit comme référence.

18.19.2. Schéma et exemple

Le schéma de ce type est défini par la règle `asset-ref-map` dans la définition CDDL suivante :

```
;L'assertion de référence d'actif (ARA) décrit où une copie de l'actif peut être obtenue. asset-ref-map = {
  « references » : [1* ara-reference-block-map]
}
```

```
ara-reference-block-map = { « référence » :
  ara-reference-uri-map,
  ? « description » : tstr, ; Description lisible par l'utilisateur de l'emplacement.
}

ara-reference-uri-map = {
  « uri » : tstr, ; URI référençant un emplacement où une copie de la ressource peut être obtenue
}
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous :

```
{
  « references » : [
    {
      « description » : « Une copie de l'actif sur le Web », « reference » :
      {
           "uri": "https://some.storage.us/foo"
        }
    },
    {
      « description » : « Une copie de la ressource sur IPFS », «
      référence » : {
           « uri » : « ipfs://cid »
        }
    }
  ]
}
```

18.20. Type d'actif

18.20.1. Description

L'assertion de type d'actif permet de décrire plus complètement un actif, en particulier en fournissant des informations supplémentaires sur la manière de l'analyser ou de le traiter. Cette assertion permet de spécifier une valeur de type de média IANA et/ou des informations supplémentaires sur le type, car de nombreux actifs ont des formats qui ne peuvent être décrits complètement par une seule valeur de type de média.

L'assertion de type d'actif doit avoir pour étiquette `c2pa.asset-type.v2`. Il ne doit y avoir qu'une seule assertion de type d'actif dans un manifeste C2PA.

REMARQUE Les versions antérieures de cette norme documentaient une assertion `c2pa.asset-type`, qui est désormais obsolète.

Si présent, la valeur du champ `dc:format` doit être le type de média IANA de la ressource.

Si elle est présente, la valeur du champ `types` doit être un tableau de zéro ou plusieurs mappages (`asset-type-map`) spécifiant les types associés à la ressource. La valeur du champ `type` dans cette carte doit provenir soit du tableau 11, « Valeurs des types de ressources », soit utiliser un espace de noms spécifique à l'entité (par exemple, `com.litware.types.abc`), conformément à la syntaxe définie pour les étiquettes d'assertion dans la section 6.2.2, « Nommage des étiquettes ». Le cas échéant, la version de l'actif (par exemple, la version d'un ensemble de données ou d'un modèle) peut être documentée dans le champ `version` de la `carte des types d'actifs.`

REMARQUE Étant donné que le C2PA est adopté pour fournir la provenance des actifs IA/ML (c'est-à-dire intelligence artificielle/apprentissage automatique) à l'avenir, le manifeste C2PA peut être intégré dans les actifs de modèles et d'ensembles de données, et l'assertion de type d'actif utilisée pour spécifier le type de ces actifs de modèles et d'ensembles de données.

Tableau 11. Valeurs du type d'actif

Type C2PA	Description du type C2PA de l'actif
c2pa.types.dataset	Ensemble de données IA/ML pouvant être traité par plusieurs cadres IA/ML ou non décrit par une autre valeur
c2pa.types.dataset.jax	Ensemble de données JAX
c2pa.types.dataset.keras	Ensemble de données Keras
c2pa.types.dataset.ml_net	Ensemble de données ML.NET
c2pa.types.dataset.mxnet	Ensemble de données MXNet
c2pa.types.dataset.onnx	Ensemble de données ONNX
c2pa.types.dataset.openvino	Ensemble de données OpenVINO
c2pa.types.dataset.pytorch	Ensemble de données PyTorch
c2pa.types.dataset.tensorflow	Ensemble de données TensorFlow
c2pa.types.model	Modèle IA/ML qui n'est décrit par aucun autre type de modèle
c2pa.types.model.jax	Modèle JAX
c2pa.types.model.keras	Modèle Keras
c2pa.types.model.ml_net	Modèle ML.NET
c2pa.types.model.mxnet	Modèle MXNet
c2pa.types.model.onnx	Modèle ONNX
c2pa.types.model.openvino.parameter	Paramètre du modèle OpenVINO
c2pa.types.model.openvino.topology	Topologie du modèle OpenVINO
c2pa.types.model.pytorch	Modèle PyTorch
c2pa.types.model.tensorflow	Modèle TensorFlow
c2pa.types.numpy	Stocké au format NumPy sérialisé
c2pa.types.protobuf	Stocké au format Protocol Buffer
c2pa.types.pickle	Stocké au format Python pickle
c2pa.types.savedmodel	Stocké au format TensorFlow SavedModel

18.20.2. Schéma et exemple

Le schéma de ce type est défini par la règle `asset-types` dans la définition CDDL suivante :

```
; L'assertion de type d'actif fournit un moyen de décrire le type ou le format d'un actif,
; plus précisément, elle fournit des informations supplémentaires sur la manière de l'analyser ou de le traiter.
; Elle peut également être utilisée pour décrire des actifs référencés en externe ou associés, tels que les
modèles d'IA/ML.

$type-choice /= « c2pa.types.classifier »
$type-choice /= « c2pa.types.cluster »
$type-choice /= « c2pa.types.dataset »
$type-choice /= « c2pa.types.dataset.jax »
$type-choice /= « c2pa.types.dataset.keras »
$type-choice /= « c2pa.types.dataset.ml_net »
$type-choice /= « c2pa.types.dataset.mxnet »
$type-choice /= « c2pa.types.dataset.onnx »
$type-choice /= « c2pa.types.dataset.openvino »
$type-choice /= « c2pa.types.dataset.pytorch »
$type-choice /= « c2pa.types.dataset.tensorflow »
$type-choice /= « c2pa.types.format.numpy »
$type-choice /= « c2pa.types.format.protobuf »
$type-choice /= « c2pa.types.format.pickle »
$type-choice /= « c2pa.types.generator »
$type-choice /= « c2pa.types.generator.prompt »
$type-choice /= « c2pa.types.generator.seed »
$type-choice /= « c2pa.types.model »
$type-choice /= « c2pa.types.model.jax »
$type-choice /= « c2pa.types.model.keras »
$type-choice /= « c2pa.types.model.ml_net »
$type-choice /= « c2pa.types.model.mxnet »
$type-choice /= « c2pa.types.model.onnx »
$type-choice /= « c2pa.types.model.openvino »
$type-choice /= « c2pa.types.model.openvino.parameter »
$type-choice /= « c2pa.types.model.openvino.topology »
$type-choice /= « c2pa.types.model.pytorch »
$type-choice /= « c2pa.types.model.tensorflow »
$type-choice /= « c2pa.types.regressor »
$type-choice /= « c2pa.types.tensorflow.hubmodule »
$type-choice /= « c2pa.types.tensorflow.savedmodel »
$type-choice /= tstr .regexp "([\\da-zA-Z_-]+\\.)+[\\da-zA-Z_-]+"

asset-type-map = {
  « type » : $type-choice, ; l'un des choix répertoriés ou une valeur personnalisée
  ? "version" : tstr .regexp "^(0|[1-9]\\d*)\\.(0|[1-9]\\d*)\\.(0|[1-9]\\d*)(?:-((?:0|[1-

]*))*))?(?:\\+([0-9a-zA-Z-]+(?:\\.[0-9a-zA-Z-]+)*))?$"
}

asset-types = {
  ? « dc:format » : format-string, ; type de média IANA de l'actif
  ? « types » : [* asset-type-map], ; ensemble de types liés à l'actif
  ? « metadata » : $assertion-metadata-map ; informations supplémentaires sur l'assertion
}
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous. Dans cet exemple, l'actif est un fichier de modèle TensorFlow de version 2.11.0 stocké au format SavedModel.

```
{
  « types » :
  [
    {
      "type": "c2pa.types.model.tensorflow",
```

```
        « version » : « 2.11.0 »
      },
      {
        « type » : « c2pa.types.savedmodel », «
        version » : « 2.11.0 »
      }
    ]
}
```

18.20.3. Détails sur la sélection d'une valeur pour `le type`

Si le type exact d'un actif est spécifié dans le type d'application du registre IANA ou le type de texte du registre IANA, y compris les types JSON, CSV et XML, cette information doit être incluse dans le champ `dc:format` de l'assertion du type d'actif.

> Par exemple, si l'actif est un fichier texte au format CSV, le champ `dc:format` sera `text/csv`.

Une assertion de type d'actif peut contenir à la fois un format Dublin Core et un type d'actif standard ou personnalisé C2PA afin de fournir des informations supplémentaires sur le type d'actif. Certains types Dublin Core existants couramment utilisés dans une assertion de type d'actif en combinaison avec d'autres types d'actifs sont spécifiés dans le tableau 12, « Formats DC courants ».

Tableau 12. Formats DC courants

dc:format Valeur	Description du type Dublin Core de l'actif
application/json	Stocké au format JSON
application/gzip	Stocké au format GZIP
application/vnd.rar	Stocké au format RAR
application/zip	Stocké au format ZIP
application/octet-stream	Stocké à l'aide d'un format binaire arbitraire
text/csv	Stocké au format CSV
text/plain	Stocké au format texte brut
text/tab-separated-values	Stocké au format texte TSV (valeurs séparées par des tabulations)
text/xml	Stocké au format XML

Les suffixes structurés IANA, tels que `+json` et `+zip`, sont également pris en charge dans le champ `dc:format` de la revendication C2PA pour spécifier des types supplémentaires.

Certains types dc:format sont couramment utilisés mais ne sont pas spécifiés dans le registre IANA. Les types `dc:format` suivants
Les valeurs sont valables pour les actifs C2PA, comme indiqué dans le tableau 13, « Formats supplémentaires ».

Tableau 13. Formats supplémentaires

dc:format Valeur	Description du type Dublin Core de l'actif
application/x-hdf5	Stocké au format HDF5
application/x-7z-compressed	Stocké au format 7Z

18.21. Carte de profondeur

18.21.1. Description

Une assertion de carte de profondeur fournit une description en 3D de la scène capturée par une caméra. Une assertion de carte de profondeur peut contenir une carte de profondeur précalculée ou des données qui peuvent être utilisées ultérieurement pour calculer une carte de profondeur par un logiciel d'ingestion ou de visualisation en aval (par exemple, des images stéréo gauche/droite).

Toutes les assertions de carte de profondeur doivent avoir une étiquette commençant par `c2pa.depthmap` et suivie d'une troisième section qui identifie le type de carte de profondeur.

Les assertions de carte de profondeur C2PA doivent être capturées optiquement, et non déduites à partir d'une seule image 2D via, par exemple, un modèle d'apprentissage automatique.

18.21.2. Carte de profondeur GDepth

Une assertion de carte de profondeur GDepth utilise le format GDepth bien établi pour encoder une carte de profondeur précalculée. Une

assertion de carte de profondeur GDepth doit avoir pour étiquette `c2pa.depthmap.GDepth`.

Le schéma des données stockées dans cette assertion doit toujours refléter le schéma disponible à l'adresse https://developers.google.com/depthmap-metadata/reference.

REMARQUE Il n'y a aucun problème à diviser les données GDepth lorsqu'elles dépassent 64 Ko, car cette limite existait dans XMP pour s'adapter aux limitations de taille du segment `APP1`.

18.21.3. Schéma et exemple

Le schéma de ce type est défini par la règle `depthmap-gdepth-map` dans la définition CDDL suivante :

```
; Assertion qui encode une carte de profondeur 3D au format GDepth de la scène capturée
depthmap-gdepth-map = {
  « GDepth:Format » : type de format, ; Format décrivant comment convertir les données de la carte de profondeur
en une carte de profondeur à virgule flottante valide. Les valeurs valides actuelles sont « RangeInverse » et «
RangeLinear ».
  « GDepth:Near » : float, ; La valeur proche de la carte de profondeur en unités de
  profondeur « GDepth:Far » : float, ; La valeur éloignée de la carte de profondeur en unités
  de profondeur
  « GDepth:Mime » : type MIME, ; Le type MIME pour la chaîne base64 décrivant le contenu de l'image de
profondeur, par exemple « image/jpeg » ou « image/png ».
  « GDepth:Data » : type de chaîne base64, ; L'image de profondeur encodée en base64. Voir la page d'encodage
GDepth sur developers.google.com. La carte de profondeur sera étirée pour s'adapter à l'image couleur
correspondante.
  ? « GDepth:Units » : type d'unité,    ; Les unités de la carte de profondeur, par exemple « m » pour les
mètres ou « mm » pour les millimètres
```

```
   ? « GDepth:MeasureType » : type de mesure de profondeur, ; Le type de mesure de profondeur. Les valeurs
valides actuelles sont « OpticalAxis » et « OpticRay
   ? « GDepth:ConfidenceMime » : type-mime-confiance, ; Le type MIME pour la chaîne base64 décrivant le contenu de
l'image de confiance, par exemple « image/png ». »,
   ? « GDepth:Confidence » : type chaîne base64,          ; L'image de confiance encodée en base64. Voir la
page d'encodage GDepth sur developers.google.com. La carte de confiance doit avoir la même taille que la
carte de profondeur
   ? « GDepth:Manufacturer » : tstr .size (1..max-tstr-length), ; Le fabricant de l'appareil qui a créé cette
carte de profondeur
   ? « GDepth:Model » : tstr .size (1..max-tstr-length), ; Le modèle de l'appareil qui a créé cette carte de
profondeur
   ? « GDepth:Software » : tstr .size (1..max-tstr-length), ; Le logiciel qui a créé cette carte de profondeur
   ? « GDepth:ImageWidth » : float, ; Largeur en pixels de l'image couleur originale associée à cette carte de
profondeur. Il ne s'agit PAS de la largeur de la carte de profondeur. Si elle est présente, les applications
doivent mettre à jour cette propriété lors du redimensionnement, du recadrage ou de la rotation de l'image
couleur. Les clients utilisent cette propriété pour vérifier l'intégrité de la carte de profondeur par rapport
à l'image couleur
   ? « GDepth:ImageHeight » : float, ; Hauteur en pixels de l'image couleur originale associée à cette carte de
profondeur. Il ne s'agit PAS de la hauteur de la carte de profondeur. Si elle est présente, les applications
doivent mettre à jour cette propriété lors du redimensionnement, du recadrage ou de la rotation de l'image
couleur. Les clients utilisent cette propriété pour vérifier l'intégrité de la carte de profondeur par rapport
à l'image couleur.
   ? « metadata » : $assertion-metadata-map, ; informations supplémentaires sur l'assertion
}

base64-string-type = tstr

$mime-choice /= « image/jpeg »
$mime-choice /= « image/png »

mime-type = $mime-choice .default « image/jpeg » confidence-mime-
type = $mime-choice .default « image/png »

$format-choice /= « RangeInverse »
$format-choice /= « RangeLinear »

format-type = $format-choice .default « RangeInverse »

; L'unité peut être le mètre, représenté par « m », ou le millimètre, représenté par « mm ».
$unit-choice /= « m »
$choix-unité /= « mm »
type-unité = $choix-unité .par défaut « m »

$depth-meas-choice /= « OpticalAxis »
$depth-meas-choice /= « OpticRay »
type-mesure-profondeur = $choix-mesure-profondeur .par défaut « OpticalAxis »
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous :

```
{
   « GDepth:Far » : 878,7,
   « GDepth:Data » : « hoOspQQ1lFTy/4Tp8Epx670E5QW5NwkNR+2b30KFXug= », « GDepth:Mime
   » : « image/jpeg »,
   « GDepth:Near » : 29,3,
   « GDepth:Model » : « CameraCompany Shooter S1 », « GDepth:Units »
   : « mm »,
   « GDepth:Format » : « RangeInverse »,
   «GDepth:Software » : « Firmware Truepic Foresight pour QC QRD8250 v0.01 », « GDepth:Confidence » :
   « acdbpQQ1lFTy/4Tp8Epx670E5QW5NwkNR+2b30KFXug= », « GDepth:ImageWidth » : 32,2,
   « GDepth:ImageHeight » : 43,6
```

```
  « GDepth:MeasureType » : « OpticalAxis », «
  GDepth:Manufacturer » : « CameraCompany », «
  GDepth:ConfidenceMime » : « image/png »,
}
```

Conformément à la spécification GDepth, les champs suivants doivent être présents dans toutes les assertions de carte de profondeur GDepth :

- GDepth:Format ;
- GDepth:Near ;
- GDepth:Far ;
- GDepth:Mime;
- GDepth:Data.

18.22. Informations sur la police

18.22.1. Description

Une assertion Font Information est utilisée pour garantir que les métadonnées de base relatives à la police, telles que le nom, le format, l'attribution au créateur et la licence, sont ajoutées à la ressource d'une manière pouvant être validée cryptographiquement.

Une assertion d'informations sur les polices doit porter le label `font.info`, et il ne doit y avoir qu'une seule assertion d'informations sur les polices par manifeste.

18.22.2. Schéma et exemple

Le schéma de ce type est défini par la règle `font-info-map` dans la définition CDDL suivante :

```
; Données d'assertion pour l'assertion font.info.
font-info-map = {
  « fullName » : tstr, ; Nom complet de la police.
  ; Une version au format de versionnement sémantique (semver).
  ? « version » : tstr .regexp « ^(0|[1-9]\\d*)\\.(0|[1-9]\\d*)\\.(0|[1-9]\\d*)(?:-((?:0|[1-

]*))*))?(?:\\+([0-9a-zA-Z-]+(?:\\.[0-9a-zA-Z-]+)*))?$",
  ? « versionUrl » : ext-url-type, ; URL vers les notes de mise à jour associées à cette version de la police.
  ? « releaseDate » : tdate, ; Date à laquelle cette version de la police a été publiée. « familyName » : tstr, ;
  Famille de polices.
  « style » : $font-style, ; Le style de la police, par exemple italique ou normal. « weight
  » : font-weight-map, ; Le poids de la police avec nom et valeur.
  ; Le nom PostScript, ID 6, provenant du tableau « name » de la police.
  « postScriptName » : tstr .regexp « ^(?!.*[\\[\\]\\(\\)\\{\\}<>\\/%])[!-~]{1,63}$ », ; Caractères ASCII 33-
126 à l'exception des suivants : [](){}<>/%
  « format » : $font-format-choice, ; Le format de cette police. « copyrightNotice
  » : tstr, ; Le copyright associé à cette police.
  ? « copyrightHolder » : font-entity-map, ; L'entité qui détient les droits d'auteur de la police.
  ? « copyrightYears » : [1* font-copyright-year-range], ; Les années pour lesquelles le détenteur revendique
le copyright.
  ? « designers » : [1* font-designer-map], ; Les personnes qui ont conçu la police.
 ? « designFoundry » : font-entity-map, ; La fonderie qui a conçu la police.
```

```
  ? « sourceFoundry » : font-entity-map, ; La fonderie qui distribue la police.
  ? « identifiant » : tstr, ; Identifiant interne de la police destiné à être utilisé par le fabricant ou le
  fournisseur.
}

; Formats de police
$font-format-choice /= « TrueType »
$font-format-choice /= « OpenType »

; Plage d'années de copyright
font-copyright-year-range = 1..9999

; Plage d'épaisseur de police
font-weight-range = 1..1000

; Descripteurs de classe de graisse de police
$font-weight-class /= « Microline »
$font-weight-class /= « Hairline »
$font-weight-class /= « UltraThin »
$font-weight-class /= « ExtraThin »
$font-weight-class /= « Thin »
$font-weight-class /= « UltraLight »
$font-weight-class /= « ExtraLight »
$font-weight-class /= « Light »
$font-weight-class /= « SemiLight »
$font-weight-class /= « Book »
$font-weight-class /= « Normal »
$font-weight-class /= « Regular »
$font-weight-class /= « Medium »
$font-weight-class /= « DemiBold »
$font-weight-class /= « SemiBold »
$font-weight-class /= « Gras »
$font-weight-class /= « Heavy »
$font-weight-class /= « ExtraBold »
$font-weight-class /= « UltraBold »
$font-weight-class /= « SemiBlack »
$font-weight-class /= « Noir »
$font-weight-class /= « ExtraBlack »
$font-weight-class /= « UltraBlack »
$font-weight-class /= « MegaBlack »

; Le style de police
$font-style /= « Normal »
$font-style /= « Italic »
$font-style /= « Oblique »
$font-style /= « Roman »
$font-style /= « Regular »

; Données relatives à
l'épaisseur d'une police
font-weight-map = {
  « class » : $font-weight-class, ; Nom descriptif de la classe de graisse, par exemple « bold » (gras) ou «
thin » (fin).
  « valeur » : font-weight-range, ; La valeur de l'épaisseur.
}

; Données pour une entité avec un nom et des informations
d'identification font-entity-map = {
  « nom » : tstr, ; Le nom de la personne ou de la fonderie.
  ? « url » : ext-url-type, ; Une URL pour obtenir des informations supplémentaires sur cette personne ou cette
  fonderie.
}

; Données relatives à un
concepteur de polices font-
designer-map = {
  « person » : font-entity-map, ; La personne qui a conçu la police.
```

```
  ? « foundry » : font-entity-map, ; Le nom de la fonderie à laquelle le concepteur était associé lorsqu'il a
contribué à la conception de la police.
  ? « contribution » : tstr, ; Description de la contribution du concepteur à la police. Par exemple, « Tous les
caractères latins et arabes ».
  ? « startDate » : tdate, ; « Date à laquelle le concepteur a commencé à contribuer à la conception de la
  police.
  ? « endDate » : tdate, ; Date à laquelle le concepteur a cessé de contribuer à la conception de la police.
}
```

Un exemple simple en notation diagnostique CBOR (RFC 8949, clause 8), ne contenant que les champs obligatoires, est présenté ci-dessous :

```
{
     « fullName » : « Example Two Italic », «
     familyName » : « ExampleTwo », « style » :
     « Italic »,
     « weight » : {
          "class": "Regular", "value":
          400
     },
     « postScriptName » : « Exemple-Deux-Italique »,
     « format » : « TrueType »,
     « copyrightNotice » : « Copyright 2011 Les auteurs du projet Exemple deux
(https://www.example.com/lifonts/Example-Two), avec le nom de police réservé « Exemple deux ». »,
     « copyrightHolder » : { « nom »
          : « Fabrikam »
     },
     « designers » : [
          {
               « personne » : {
                  « nom » : « John Doe »,
                  « url » : « https://fabrikam.example.com/jdoefonts »
               }
          }
     ]
}
```

Cet exemple détaillé montre également les champs facultatifs :

```
{
     « fullName » : « Example Font Bold Italic », «
     version » : « 7.0.4-beta »,
     « versionUrl » : « https://fabrikam.example.com/release/efbi/7.0 », «
     familyName » : « ExampleFont »,
     « style » : « Italic »,
     « weight » : {
          "class": "Bold",
          "value": 700
     },
     « postScriptName » : « ExampleFont-BoldItalic », « format » :
     « OpenType »,
     « copyrightNotice » : « © 2017 Fabrikam, Inc. Tous droits réservés. », «
     copyrightHolder » : {
          « nom » : « Fabrikam Inc. »
     },
     « copyrightYears » : [
          1982,
          2017
     ],
     « designers » : [
```

```
          {
               « personne » : {
                  « nom » : « John Doe »,
                  « url » : « https://fabrikam.example.com/browse/designers/john-doe »
               },
               « foundry » : {
                   « nom » : « Fabrikam Fonts »
               },
               « contribution » : « Ligatures. »
          },
          {
               « personne » : {
                  « nom » : « Jane Doe »
               },
               « foundry » : {
                    « nom » : « Fabrikam Fonts »
               },
               « contribution » : « Tous les caractères. »
          }
     ],
     « designFoundry » : {
          « nom » : « Fabrikam Fonts »,
          « url » : « https://fabrikam.example.com »
     },
     « sourceFoundry » : {
          « nom » : « Fonts Direct 2 U », « url »
          : « https://fd2u.example.com »
     },
     « identifier » : « ExampleFont Bold Italic (Fabrikam) »
}
```

Chapitre 19. Politique en matière de brevets

La C2PA a adopté une politique de brevets ouverte via le mode brevet du W3C (2004) :

Engagement en matière de licence. Pour les éléments autres que le code source ou les ensembles de données développés par le groupe de travail, chaque participant au groupe de travail s'engage à mettre à disposition ses revendications essentielles, telles que définies dans la politique en matière de brevets du W3C (disponible à l'adresse http://www.w3.org/Consortium/Patent-Policy-20040205), conformément aux exigences de licence RF du W3C, section 5 (http://www.w3.org/Consortium/Patent-Policy-20040205), dans les livrables approuvés adoptés par ce groupe de travail, comme si ces livrables approuvés étaient une recommandation du W3C. Le code source développé par le groupe de travail est soumis à la licence énoncée dans la charte du groupe de travail.

Pour exclusion. Avant l'adoption d'un projet de livrable en tant que livrable approuvé, un participant au groupe de travail peut exclure des revendications essentielles de ses engagements de licence en vertu du présent accord en adressant une notification écrite de cette intention au président du groupe de travail (« notification d'exclusion »). L'avis d'exclusion pour les brevets délivrés et les demandes publiées doit inclure le ou les numéros de brevet ou le titre et le ou les numéros de demande, selon le cas, pour chacun des brevets délivrés ou des demandes de brevet en instance que le participant au groupe de travail souhaite exclure de l'engagement de licence énoncé à la section 1 de la présente politique en matière de brevets. Si un brevet délivré ou une demande de brevet en instance susceptible de contenir des revendications essentielles n'est pas mentionné dans l'avis d'exclusion, ces revendications essentielles continuent d'être soumises aux engagements de licence prévus par le présent accord. L'avis d'exclusion pour les demandes de brevet non publiées doit fournir soit : (i) le texte de la demande déposée ; soit (ii) l'identification de la ou des parties spécifiques du projet de livrable dont la mise en œuvre fait de la revendication exclue une revendication essentielle. Si l'option (ii) est choisie, l'effet de l'exclusion sera limité à la ou aux parties identifiées du projet de livrable. Le président du groupe de travail publiera les avis d'exclusion.

Annexe A : Manifestes d'intégration

A.1. Formats pris en charge

Un manifeste C2PA est intégré à une ressource dans le cadre du magasin de manifestes C2PA pour cette ressource.

Lors de l'intégration du magasin de manifestes C2PA dans un actif, l'emplacement varie en fonction du type ou du format de l'actif. Voici quelques formats de fichiers courants et l'emplacement du magasin de manifestes C2PA dans chacun d'eux :

JPEG

Pour plus d'informations, reportez-vous à la section A.3.1, « Intégration de manifestes dans JPEG ».

JPEG-XL

Pour plus d'informations, reportez-vous à la section A.3.8, « Intégration de manifestes dans JPEG XL ».

PNG

Pour plus d'informations, reportez-vous à la section A.3.2, « Intégration de manifestes dans des fichiers PNG ».

SVG

Pour plus d'informations, reportez-vous à la section A.3.3, « Intégration de manifestes dans SVG ».

FLAC

Pour plus d'informations, reportez-vous à la section A.3.4, « Intégration de manifestes dans ID3 ».

MP3

Pour plus d'informations, reportez-vous à la section A.3.4, « Intégration de manifestes dans ID3 ».

GIF

Pour plus d'informations, reportez-vous à la section A.3.7, « Intégration de manifestes dans les GIF ».

DNG

Pour plus d'informations, reportez-vous à la section A.3.5, « Intégration de manifestes dans des ressources au format TIFF ».

Formats basés sur TIFF

Pour plus d'informations, reportez-vous à la section A.3.5, « Intégration de manifestes dans des ressources au format TIFF ».

WAV et BWF

Pour plus d'informations, reportez-vous à la section A.3.6, « Intégration de manifestes dans des ressources basées sur RIFF ».

AVI

Pour plus d'informations, reportez-vous à la section A.3.6, « Intégration de manifestes dans des ressources basées sur RIFF ».

WebP

Pour plus d'informations, reportez-vous à la section A.3.6, « Intégration de manifestes dans des ressources basées sur RIFF ».

Autres formats basés sur RIFF

Pour plus d'informations, reportez-vous à la section A.3.6, « Intégration de manifestes dans des ressources basées sur RIFF ».

Polices

Pour plus d'informations, reportez-vous à la section A.3.9, « Intégration de manifestes dans les polices ».

PDF

Pour plus d'informations, consultez la section A.4, « Intégration de manifestes dans des fichiers PDF ».

EPUB

Pour plus d'informations, reportez-vous à la section A.6, « Intégration de manifestes dans des formats ZIP ».

OOXML

Pour plus d'informations, reportez-vous à la section A.6, « Intégration de manifestes dans des formats ZIP ».

Open Document

Pour plus d'informations, reportez-vous à la section A.6, « Intégration de manifestes dans des formats basés sur ZIP ».

OpenXPS

Pour plus d'informations, reportez-vous à la section A.6, « Intégration de manifestes dans des formats basés sur ZIP ».

Autres formats ZIP

Pour plus d'informations, reportez-vous à la section A.6, « Intégration de manifestes dans des formats ZIP ».

MP4

Pour plus d'informations, reportez-vous à la section A.5, « Intégration de manifestes dans des ressources BMFF ».

MOV

Pour plus d'informations, reportez-vous à la section A.5, « Intégration de manifestes dans des ressources BMFF ».

AAC

Pour plus d'informations, reportez-vous à la section A.5, « Intégration de manifestes dans des ressources basées sur BMFF ».

ALAC

Pour plus d'informations, reportez-vous à la section A.5, « Intégration de manifestes dans des ressources basées sur BMFF ».

HEIF

Pour plus d'informations, reportez-vous à la section A.5, « Intégration de manifestes dans des ressources basées sur BMFF ».

Autres formats basés sur BMFF

La case spécifiée dans la section A.5, « Intégration de manifestes dans des ressources basées sur BMFF ».

REMARQUE Les formats audio non basés sur BMFF dont l'ajout à cette spécification est envisagé comprennent Ogg Vorbis et la version native du conteneur Free Lossless Audio Codec (Native FLAC).

A.2. Intégration de manifestes dans des ressources en plusieurs parties

Lorsqu'un manifeste C2PA est intégré à un actif en plusieurs parties (« multi-actif »), un magasin de manifestes C2PA doit être intégré à la partie principale de l'actif (qui contient le manifeste actif), bien que des parties supplémentaires puissent également contenir leurs propres magasins de manifestes C2PA. Le manifeste actif de la partie principale doit contenir une assertion de hachage multi-actifs qui décrit l'emplacement et le hachage de chaque partie au sein de l'actif et doit décrire la provenance de l'ensemble de l'actif en plusieurs parties.

A.3. Intégration de manifestes dans des ressources non basées sur BMFF

A.3.1. Intégration de manifestes dans JPEG

Le magasin de manifestes C2PA doit être intégré en tant que données contenues dans un segment de marqueur **APP11** tel que défini dans JPEG XT, ISO/IEC 18477-3.

Étant donné qu'un segment de marqueur unique dans JPEG 1 ne peut pas dépasser 64 Ko, il est probable que plusieurs segments **APP11** soient nécessaires, et ceux-ci doivent être construits conformément à la norme JPEG 1 et à la norme ISO 19566-5:2023, D.2. Lors de l'écriture de plusieurs segments, ceux-ci doivent être écrits dans un ordre séquentiel et être contigus (c'est-à-dire qu'un segment doit suivre immédiatement le suivant).

A.3.2. Intégration de manifestes dans PNG

Le magasin de manifestes C2PA doit être intégré à l'aide d'un type de bloc auxiliaire, privé, non copiable, `de type « caBX »` (conformément à la norme PNG, 4.7.2). Il est recommandé que le bloc `« caBX »` précède les blocs `« IDAT »`.

REMARQUE Bien que le format PNG le prenne en charge, il est considéré comme incorrect d'avoir un bloc de données après le `« IDAT »` et avant le `« IEND »`. (À l'exception des blocs PNG animés)

A.3.3. Intégration de manifestes dans SVG

Le format SVG est un format basé sur XML qui peut exister de manière autonome ou être intégré à d'autres formats textuels tels que HTML. À ce titre, il est nécessaire d'encoder en Base64 le magasin de manifestes C2PA binaire pour effectuer l'intégration. Bien que cette section décrive comment procéder, l'utilisation d'un manifeste externe est préférable.

Le magasin de manifestes C2PA doit être intégré en tant que valeur codée en Base64 d'un élément `c2pa:manifest` dans l'élément `metadata` du SVG. Étant donné que le XML, et le SVG en particulier, recommandent fortement la déclaration d'un espace de noms avant son utilisation, une déclaration d'attribut `xmlns:c2pa = « http://c2pa.org/manifest »` doit être ajoutée à l'élément `svg`.

Exemple 17. Exemple d'un magasin de manifestes C2PA dans un SVG

Exemple de magasin de manifeste C2PA dans un SVG (les données réelles du manifeste C2PA ont été omises).

```
<?xml version="1.0" standalone="yes"?>
<svg width="4in" height="3in" version="1.1" xmlns =
     "http://www.w3.org/2000/svg" xmlns:c2pa =
     "http://c2pa.org/manifest">
     <metadata>
          <c2pa:manifest>...Les données Base64 vont ici...</c2pa:manifest>
     </metadata>
</svg>
```

A.3.4. Intégration des manifestes dans ID3

Le magasin de manifestes C2PA doit être intégré dans un fichier audio compressé compatible ID3v2 (par exemple, MP3 ou FLAC) en tant que données d'objet encapsulées d'un objet encapsulé général (GEOB), tel que défini dans https://id3.org/id3v2.3.0. Le champ `de type MIME` du GEOB doit être présent et utiliser la valeur du type de média pour JUMBF, comme décrit dans la section 11.4, « Manifestes externes ».

A.3.5. Intégration de manifestes dans des ressources au format TIFF

Le format Digital Negative ou DNG permet aux fabricants d'appareils photo de fournir leurs formats RAW de manière standardisée. Le format DNG est basé sur le format TIFF/EP (qui est lui-même basé sur le format TIFF).

Le magasin de manifestes C2PA doit être intégré dans un fichier compatible TIFF (c'est-à-dire TIFF/EP, DNG ou d'autres formats RAW basés sur TIFF) en tant que données d'une balise avec l'ID 52545 (décimal) ou 0xCD41 (hexadécimal), avec un type de balise de 7.

Bien que le format TIFF prenne en charge le concept de pages ou de couches multiples (via plusieurs IFD), il ne doit y avoir qu'un seul magasin de manifeste C2PA pour l'ensemble des ressources, et non un par IFD. Ainsi, le magasin de manifeste C2PA doit être la seule boîte présente dans le dernier IFD, c'est-à-dire l'IFD précédant immédiatement la fin du fichier.

REMARQUE Les versions précédentes de cette spécification exigeaient l'utilisation de l'IFD 0, mais il a été reconnu que cela limitait son utilisation dans les formats RAW basés sur TIFF.

A.3.6. Intégration de manifestes dans des ressources basées sur RIFF

Le format RIFF (Resource Interchange File Format) fournit un format conteneur générique pour stocker des données dans des blocs balisés. Il est principalement utilisé pour stocker des fichiers multimédias tels que des images, des sons et des vidéos. Il sert de format conteneur pour les formats WAV, BWF, Broadcast Wave, AVI et WebP.

REMARQUE Le format RIFF est basé sur un format plus ancien appelé IFF.

Le magasin de manifestes C2PA doit être intégré dans un fichier compatible RIFF (c'est-à-dire WAV, AVI ou WebP) en tant que données d'un bloc avec un identifiant `C2PA`. Pour des raisons de compatibilité, ce bloc C2PA doit apparaître à la fin du bloc RIFF.

A.3.7. Intégration des manifestes dans les fichiers GIF

Le magasin de manifestes C2PA doit être divisé en blocs d'une taille maximale de 255 octets et intégré dans des sous-blocs de données contigus (conformément à GIF, 15) au sein d'un bloc d'extension d'application spécialisé C2PA (conformément à GIF, 26), spécifié ci-dessous.

REMARQUE Dans ce bloc d'extension d'application C2PA, le code d'authentification de l'application n'est pas utilisé pour authentifier l'application produisant le bloc. Il est plutôt utilisé comme version de bloc, initialement défini comme version majeure 1, version mineure 0, et encodé comme spécifié ci-dessous.

```
Introducteur d'extension : 0x21
Étiquette d'extension d'application :
0xFF Taille du bloc : 0xB
Identifiant de l'application : 0x43, 0x32, 0x50, 0x41, 0x5F, 0x47, 0x49, 0x46 (« C2PA_GIF ») Code
d'authentification de l'application : 0x010000 (0x[Version majeure][Version mineure]00) Données
d'application : le magasin de manifestes C2PA, encodé sous forme d'une série de sous-blocs de données,
chacun contenant 1 octet suivi d'un maximum de 255 octets de données
Terminateur de bloc : 0x00 (ajouté après le dernier sous-bloc de données du magasin de manifestes C2PA) Quantité
: un
```

Ce bloc doit être intégré après l'en-tête et avant la première boîte de descripteur d'image.

A.3.8. Intégration des manifestes dans JPEG XL

Comme décrit dans la norme ISO/IEC 18181-2:2024, clause 4, JPEG XL prend en charge deux formats différents pour les données. Il peut utiliser une structure en boîte compatible avec JPEG 2000 et JPEG XS ou il peut s'agir d'un flux de code JPEG XL direct sans structure en boîte. Un fichier JPEG XL qui utilise la structure en boîtes doit contenir au maximum une superboîte JUMBF (`jumb`) (ISO/IEC 18181-2:2024, clause 9.3) contenant une boîte JUMBF C2PA Manifest, qui contient le manifeste C2PA tel que décrit dans la section 11.1.4.2, « Manifest Store ». Un fichier JPEG XL qui n'est qu'un flux de code ne peut pas inclure de manifeste C2PA intégré.

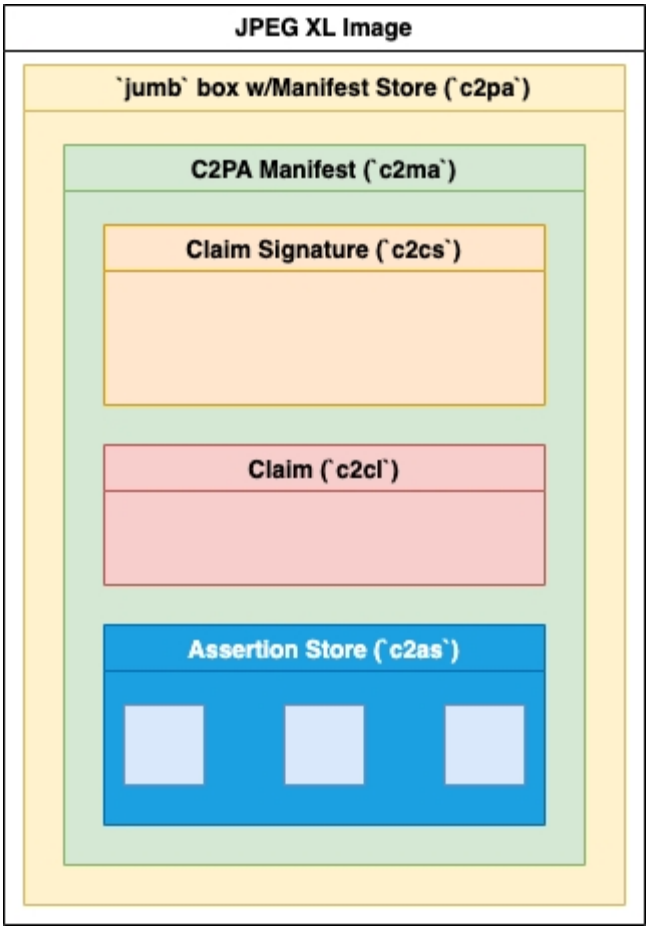

Figure 19. Manifeste C2PA intégré dans une image JPEG XL

A.3.9. Intégration de manifestes dans les polices

Les polices conformes au format Open Font Format ou à la spécification OpenType peuvent inclure une table `C2PA`. Lorsqu'elle est présente, cette table peut inclure un manifeste intégré, un URI de manifeste distant, ou les deux.

Le format de la table `C2PA` n'est pas encore défini dans le format Open Font Format ni dans la spécification OpenType ; la définition suivante est préliminaire :

A.3.9.1. Balise de table

L'enregistrement de la table C2PA sera identifié par la balise de table suivante : `C2PA`.

A.3.9.2. Enregistrement de table

La table C2PA prend entièrement en charge les magasins de manifestes intégrés, distants ou les deux. L'enregistrement de table est défini comme suit :

Tableau 14. Enregistrement de la table C2PA

Type	Nom	Description
`uint16`	`majorVersion`	Spécifie la version majeure de la table de polices C2PA.

uint16	minorVersion	Spécifie la version mineure de la table de polices C2PA.
Offset32	activeManifestUriOffset	Décalage entre le début de la table de polices C2PA et la section contenant une URI vers le manifeste actif. Si aucune URI n'est fournie, un décalage NULL = 0x0000 doit être utilisé.
uint16	activeManifestUriLength	Longueur de l'URI en octets.
uint16	réservé	Réservé pour une utilisation future.
Offset32	manifestStoreOffset	Décalage entre le début de la table de polices C2PA et la section contenant un magasin de manifestes C2PA. Si aucun magasin de manifestes n'est fourni, un décalage NULL = 0x0000 doit être utilisé.
uint32	manifestStoreLength	Longueur des données du magasin de manifestes C2PA en octets.

Le manifeste C2PA non intégré peut être distant ou local sur le même système de stockage. Si la référence est une URI JUMBF, elle doit être une référence valide dans le magasin de manifestes C2PA.

A.4. Intégration de manifestes dans des fichiers PDF

A.4.1. Général

Tous les magasins de manifestes C2PA doivent être intégrés à l'aide de flux de fichiers intégrés (ISO 32000, 7.11.4). Le dictionnaire de spécification des fichiers intégrés doit comporter une clé `Subtype` dont la valeur est `application/c2pa` et une clé `AFRelationship` (ISO 32000, 7.11.3) dont la valeur est `C2PA_Manifest.` Si un magasin de manifestes C2PA est intégré dans un PDF crypté, le flux de fichiers intégré doit utiliser un filtre cryptographique `Identity`.

A.4.2. Manifestes au niveau du document

A.4.2.1. Ajout du manifeste à un PDF

Lors de l'ajout d'un manifeste C2PA à l'ensemble du PDF, le dictionnaire du catalogue de documents doit contenir une entrée `AF` dont la valeur est une référence indirecte à la spécification de fichier intégré contenant le manifeste actif. Cette spécification de fichier intégré doit également être référencée, via un objet indirect, soit à partir de l'arborescence des noms des fichiers `intégrés` (`/Catalog/Names/EmbeddedFiles`), soit à partir d'une annotation `FileAttachment`. L'approche par annotation doit être utilisée lors de l'ajout d'un magasin de manifestes C2PA à un PDF qui possède déjà une signature de certification PDF existante afin d'éviter d'invalider ses restrictions `DocMDP`.

REMARQUE Les valeurs *1* ou *2* du champ `P` dans le dictionnaire DocMDP ne permettent pas ce type de modification. Seule la valeur *3* le permet.

Dans la plupart des autres formats, il n'existe qu'un seul magasin de manifestes C2PA qui contient tous les manifestes C2PA pour l'actif. Cependant, en raison de la fonctionnalité de « mise à jour incrémentielle » du format PDF, il est nécessaire de prendre en charge plusieurs manifestes dans un seul fichier PDF. Dans ce scénario, le magasin de manifestes C2PA trouvé dans le fichier PDF de base doit être considéré comme le manifeste initial et celui de la mise à jour la plus récente comme le manifeste actif. Un consommateur de manifeste C2PA doit traiter tous les manifestes C2PA dans tous les magasins de manifestes C2PA comme s'ils étaient contenus dans un seul magasin de manifestes C2PA.

REMARQUE

Étant donné qu'une URI JUMBF est toujours une URI complète, ce qui signifie qu'elle commence à un manifeste C2PA donné, et que toutes les manifestes C2PA sont considérés comme contenus dans un seul magasin de manifestes C2PA, l'utilisation d'un tel URI pour faire référence à un élément `parentOf` dans les magasins de manifestes C2PA dans un PDF est acceptable.

A.4.2.2. Compatibilité avec les signatures PDF

Lors de l'ajout d'un nouveau magasin de manifestes C2PA, il est nécessaire de savoir si une signature PDF (certification ou approbation) sera également appliquée. Étant donné que la signature PDF modifiera les données du PDF après la signature du manifeste C2PA, la taille et l'emplacement de la clé `Contents` du dictionnaire de signature PDF doivent être déterminés avant la signature C2PA. Cette plage d'octets doit être ajoutée à la liste des exclusions dans l'assertion `c2pa.hash.data`, afin que la signature C2PA ne soit pas invalidée par l'ajout de la signature PDF. La signature PDF doit couvrir l'ensemble du PDF, y compris le magasin de manifestes C2PA associé.

REMARQUE

L'ajout de la signature PDF en plus de la signature de revendication C2PA améliore la compatibilité avec l'écosystème PDF existant.

A.4.3. Manifestes au niveau des objets

En plus de pouvoir fournir la provenance du PDF lui-même, via des manifestes au niveau du document, les objets individuels d'un document peuvent également être associés à un magasin de manifestes C2PA. Pour ce faire, il suffit d'ajouter une entrée `AF` au flux ou au dictionnaire de l'objet. La valeur de l'entrée `AF` doit être une référence indirecte à la spécification du fichier intégré contenant le magasin de manifestes C2PA, intégré comme décrit ci-dessus.

Cette fonctionnalité est principalement utilisée pour fournir la provenance des images intégrées, qu'il s'agisse d'images, de formulaires XObjects ou de polices. Elle peut également être utilisée pour fournir la provenance de contenus spécifiques en ajoutant l'entrée `AF` à l'objet (via la liste des propriétés) ou à un élément de structure, comme décrit dans la clause « Fichiers associés » de la norme ISO 32000-2 (14.13.1).

Il est recommandé que tout manifeste au niveau de l'objet ajouté soit référencé à partir du manifeste actif en tant que composant « `componentOf` ». Cela permettra au consommateur de manifeste C2PA de parcourir facilement toute la chaîne de provenance de la ressource.

En général, tout flux PDF ou dictionnaire peut être associé à un manifeste C2PA, à condition que le flux ou le dictionnaire représente une ressource d'informations réelle. En cas d'ambiguïté quant au flux ou au dictionnaire pouvant contenir l'entrée `AF`, le manifeste doit être associé autant que possible à l'objet qui stocke réellement la ressource de données décrite.

REMARQUE

Le manifeste C2PA décrivant une image raster serait joint au flux Image XObject.

qui la décrit, et le manifeste des fichiers de polices intégrés serait joint aux flux de fichiers de polices plutôt

qu'aux dictionnaires de polices.

A.4.4. Exemple

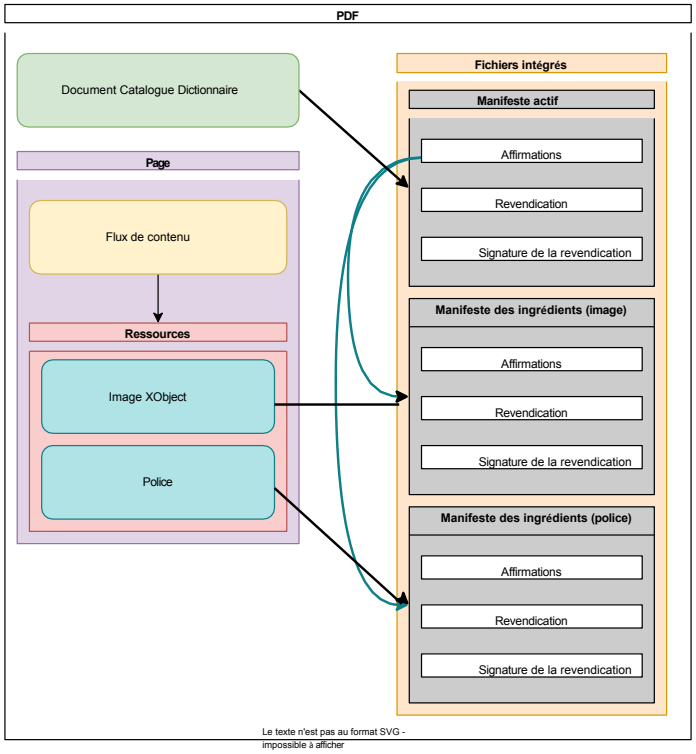

Figure 20. Exemple de fichier PDF contenant plusieurs manifestes d'ingrédients

A.5. Intégration de manifestes dans des ressources basées sur BMFF

A.5.1. La case « `uuid` » pour C2PA

Tous les actifs C2PA basés sur BMFF, qu'il s'agisse de médias audiovisuels chronométrés (par exemple, des vidéos avec ou sans piste audio), non chronométrés (par exemple, des photos fixes) ou mixtes (par exemple, des photos en direct ou animées), doivent utiliser une boîte « `uuid` » qui respecte la syntaxe et la sémantique définies ci-dessous.

REMARQUE La raison pour laquelle une boîte « `uuid` » est utilisée à la place d'une boîte « `c2pa` » est que les navigateurs basés sur Chromium échouent immédiatement la lecture lorsqu'ils rencontrent des boîtes de niveau supérieur inconnues.

Certains formats de fichiers basés sur BMFF et pris en charge par cette méthode comprennent :

- les points de code MPEG-4, complets (`.mp4`) ou fragmentés (`.m4s`) ; les fichiers audio téléchargeables (`.m4a`) ;
- HEIF (`.heif`, `.heic`) ;
- AVIF (`.avif`).

A.5.1.1. Définition

```
Type de boîte : « uuid »
```

```
Type de boîte étendue : 0xD8, 0xFE, 0xC3, 0xD6, 0x1B, 0x0E, 0x48, 0x3C, 0x92, 0x97, 0x58, 0x28, 0x87, 0x7E, 0xC4,
0x81
Conteneur : fichier
Obligatoire : non Quantité
: zéro ou plus
```

La boîte « uuid » de C2PA intègre la provenance dans BMFF. Une telle boîte contient un magasin de manifestes C2PA, et il peut y avoir une ou plusieurs boîtes auxiliaires contenant des informations supplémentaires nécessaires à la validation.

A.5.1.2. Syntaxe

```
aligné(8) classe ContentProvenanceBox étend FullBox('uuid', extended_type = 0xD8 0xFE 0xC3 0xD6 0x1B 0x0E 0x48
0x3C 0x92 0x97 0x58 0x28 0x87 0x7E 0xC4 0x81, version = 0, 0) {
     string box_purpose;
     bit(8) data[];
}
```

A.5.1.3. Concernant les identifiants uniques

Dans certains cas, tels que les MP4 fragmentés (fMP4), l'ID d'un sous-ensemble de la ressource, tel que le champ track_id de la boîte « tkhd », n'est unique que localement pour un sous-ensemble de la ressource globale, et non globalement pour la ressource.

Étant donné qu'un identifiant unique au niveau mondial est nécessaire pour déterminer ce qui doit être haché, un identifiant unique est inclus. Cet identifiant unique ne correspond à aucune valeur de la ressource d'origine ; chaque valeur est plutôt définie lors de la création du manifeste. L'identifiant unique est ensuite combiné à un identifiant local associé pour former un identifiant unique au niveau mondial pour l'ensemble de la ressource.

A.5.2. Sémantique

L'objectif de chaque boîte (box_purpose) et les champs qui en dépendent (data) sont décrits ci-dessous pour chaque boîte.

A.5.3. Boîte contenant le manifeste

La boîte contenant le magasin de manifestes C2PA doit apparaître avant la première boîte « mdat » du fichier et avant toute boîte « moov » du fichier. Afin de permettre la vérification des marques principales et des marques compatibles, elle doit être placée après la boîte « ftyp ». Lorsque le manifeste actif d'un actif est un manifeste de mise à jour, le magasin de manifestes C2PA standard précédent est situé comme indiqué ci-dessus, avec box_purpose changé en original. Le magasin de manifestes C2PA mis à jour doit être la dernière boîte du fichier, avec box_purpose défini sur update.

Les champs de la case correspondante décrite ci-dessus doivent être remplis comme suit.

box_purpose

Pour un magasin de manifestes C2PA, cette valeur doit être manifest, original ou update.

data

Lorsque box_purpose est manifest, les 8 premiers octets dans « data » doivent être le décalage absolu en octets du fichier vers la première boîte C2PA auxiliaire « uuid » avec box_purpose égal à merkle. Si ce fichier ne contient aucune boîte de ce type, ces 8 octets

doit être égal à zéro. Ces 8 octets doivent être suivis des octets bruts du magasin de manifestes C2PA, puis d'un ou plusieurs octets de remplissage inutilisés. Lorsque `box_purpose` est `original`, cela indique qu'une autre boîte C2PA dont la valeur `box_purpose` est définie sur `update` est présente. Les « `données` » contenues dans cette boîte `originale` restent inchangées. Lorsque `box_purpose` est `update`, le magasin de manifestes C2PA ne doit contenir que des manifestes de mise à jour.

NOTE

Le champ « `data` » à l'intérieur de la boîte « `uuid` » de type `manifeste` ou `original` comprend le fichier absolu , le manifeste et les octets de remplissage. Les boîtes original et manifeste sont identiques, à l'exception de la valeur de box_purpose, et les liaisons de hachage ne sont donc pas modifiées. Aucune donnée hachée n'est déplacée par l'ajout de la boîte « update ».

Les octets de remplissage ne sont pas autorisés en dehors de la boîte « `uuid` », sauf s'ils sont contenus dans leur propre boîte mp4, telle qu'une boîte
boîte « `free` ».

Pour les fichiers MP4 fragmentés (fMP4), une boîte « `uuid` » C2PA identique de type `manifeste` doit être présente dans chaque segment d'initialisation ; le magasin de manifestes C2PA doit être identique.

A.5.4. Boîtes « `c2pa` » auxiliaires pour les fichiers volumineux et fragmentés

A.5.4.1. Généralités

Certains fichiers comportent une ou plusieurs boîtes « `mdat` » très volumineuses (par exemple, des fichiers vidéo ou image volumineux qui peuvent être téléchargés et rendus progressivement) ou un grand nombre de boîtes « mdat » indépendantes (par exemple, fMP4 où chaque fragment peut être téléchargé indépendamment).

Dans ces cas, il n'est pas raisonnable d'exiger d'un client qu'il télécharge complètement toutes les boîtes « mdat » avant de valider une partie quelconque de la ressource. Il est possible d'éviter cette nécessité en utilisant plusieurs hachages.

Pour chaque grande boîte « `mdat` », les sous-ensembles de la boîte ont des hachages individuels qui peuvent être validés indépendamment ; la manière de déterminer ces sous-ensembles est précisée ci-dessous. Pour le contenu fMP4 où chaque boîte « `mdat` » peut être téléchargée indépendamment, chaque fragment a son propre hachage individuel.

Dans le cas le plus simple, tous ces hachages sont stockés dans le manifeste actif. Chaque sous-ensemble dispose d'une boîte C2PA « `uuid` » auxiliaire qui indique comment localiser son hachage dans le manifeste actif ; reportez-vous à la remarque concernant les identifiants uniques ci-dessus pour comprendre pourquoi il en est ainsi.

Cependant, pour les actifs suffisamment volumineux, l'inclusion du hachage de chaque sous-ensemble dans le manifeste lui-même augmenterait la taille du magasin de manifestes C2PA à un ou plusieurs mégaoctets.

Pour éviter un magasin de manifestes C2PA aussi volumineux pour un actif de grande taille, on utilise un ou plusieurs arbres de Merkle.

- Pour un actif non fragmenté de grande taille contenant une ou plusieurs boîtes « `mdat` » dans un seul fichier volumineux, un arbre Merkle est utilisé pour chaque boîte « `mdat` ».
- Pour un actif fragmenté volumineux contenant un ensemble de boîtes « `mdat` » pour une seule piste pouvant être répartie sur plusieurs fichiers, un arbre Merkle est utilisé pour chaque piste.

Dans les deux cas :

- Chaque nœud feuille d'un arbre Merkle donné est le hachage du sous-ensemble.
- Le manifeste stocke une ligne de chaque arbre Merkle.
- La case auxiliaire « uuid » C2PA qui existe pour chaque sous-ensemble indique quelle ligne de l'arbre Merkle dans le manifeste actif elle requiert et quel nœud feuille elle représente. Elle comprend également tout hachage supplémentaire provenant de l'arbre Merkle nécessaire pour dériver un hachage dans la ligne de l'arbre Merkle du manifeste actif.

Le choix de la ligne de l'arbre Merkle à stocker dans le manifeste crée un compromis en termes de taille au sein de l'actif. Plus précisément, le stockage d'un seul hachage par arbre Merkle dans le manifeste minimise la taille du manifeste, mais nécessite le stockage de log2(sous-ensembles) dans chaque boîte spécifique au sous-ensemble. Chaque fois que le nombre de hachages stockés dans le manifeste pour un arbre Merkle est doublé (en descendant d'une ligne de l'arbre Merkle), le nombre de hachages stockés dans chaque boîte spécifique à un sous-ensemble diminue d'un. Ainsi, l'augmentation de la taille du manifeste réduit la taille de l'ensemble de l'actif et vice-versa, et comme les hachages des sous-ensembles individuels sont répliqués entre les sous-ensembles selon les besoins pour dériver un hachage spécifié dans le manifeste, le compromis n'est pas de 1 pour 1.

Le choix de ce compromis en termes de taille est laissé à la discrétion de l'implémentation qui crée le manifeste ; cette spécification n'impose ni ne recommande le stockage d'une ligne spécifique de l'arbre de Merkle dans le manifeste. Cela dit, étant donné que le cas le plus simple de stockage de tous les hachages de sous-ensembles dans le manifeste équivaut à l'utilisation d'un arbre de Merkle dont les nœuds feuilles sont stockés dans le manifeste, la même construction d'arbre de Merkle est utilisée pour plusieurs hachages dans tous les cas. Cette construction est définie comme suit.

La partie du manifeste contenant le hachage BMFF doit inclure le champ `merkle`. Pour plus d'informations, reportez-vous à la section 9.2.3, « Hachage d'un actif au format BMFF ».

A.5.4.1.1. Ressource non fragmentée pouvant être validée par morceaux

Si le manifeste contient une ligne non terminale de l'arbre Merkle, deux ou plusieurs cases C2PA « uuid » auxiliaires avec `box_purpose` défini sur « merkle » comme décrit ci-dessous doivent être incluses dans le fichier. Elles ne sont pas obligatoires si le manifeste contient la ligne terminale de l'arbre Merkle. Si elles existent, elles doivent suivre la dernière case « mdat » du fichier.

Le hachage utilisé pour un nœud feuille donné dans l'arbre Merkle doit être calculé à partir du sous-ensemble de la charge utile du « mdat ». Le « mdat » est divisé en tailles définies par « fixedBlockSize » ou le tableau « variableBlockSizes » figurant dans le `bmff-merkle-map`, et la somme des « variableBlockSizes » doit être égale à la taille de la charge utile du « mdat ».

Toutes ces boîtes C2PA « uuid » auxiliaires doivent répondre aux exigences suivantes.

- Elles doivent être dans le même ordre que les sous-ensembles qu'elles hachent, comme spécifié par le champ « variableBlockSizes ». .
- Elles doivent être regroupées de manière à ce que les boîtes C2PA « uuid » auxiliaires d'un seul arbre Merkle soient séquentielles, sans boîtes intermédiaires.
- La valeur de localisation dans la première boîte doit être réglée sur 0, dans la deuxième boîte sur 1, et doit augmenter

par la suite de manière séquentielle.

A.5.4.1.2. Ressource fragmentée

Pour les actifs fMP4 qui sont répartis sur plusieurs fichiers :

- Une case C2PA « uuid » auxiliaire avec box_purpose défini sur « merkle » comme décrit ci-dessous doit être incluse dans chaque fichier fragmenté immédiatement avant la case « moof ».
- Le hachage utilisé pour un nœud feuille donné dans l'arbre Merkle doit couvrir toutes les données contenues dans le fichier fragment unique qui le contient, à l'exception des données exclues par la liste d'exclusion.

REMARQUE Cette spécification ne permet pas la prise en charge des ressources fMP4 qui sont réparties sur plusieurs fichiers où les fichiers de fragments individuels contiennent plus d'une boîte « moof » ou « mdat », ou les deux.

Pour les ressources fMP4 qui sont stockées sous la forme d'un seul fichier MP4 plat avec un seul « moov » pour toutes les pistes, puis une paire « moof »
/« mdat » pour chaque fragment :

- Une boîte C2PA « uuid » auxiliaire avec box_purpose défini sur « merkle » comme décrit ci-dessous doit être incluse immédiatement avant chaque boîte « moof ».
- Le hachage utilisé pour un nœud feuille donné dans l'arbre Merkle doit porter sur cette boîte « moof » plus toutes les données précédant la boîte « moof » suivante ou sur toutes les données jusqu'à la fin du fichier s'il n'y a pas d'autre boîte « moof ». Le hachage ne doit pas couvrir les données exclues par la liste d'exclusion.

IMPORTANT Prendre un actif fMP4 conforme à la norme C2PA qui est réparti sur plusieurs fichiers (c'est-à-dire qui comporte des boîtes « c2pa » de types « manifest » et « merkle ») et joindre les fichiers individuels ne produira pas un fichier unique conforme à la norme C2PA (et vice-versa). En effet,
boîtes sont incluses dans chaque hachage « merkle » sera différente dans les deux cas. Si les deux formes sont souhaitables, la deuxième forme doit considérer la première forme comme un ingrédient et le nouveau manifeste doit inclure à la fois une assertion d'ingrédient avec la relation parentOf et une assertion d'actions qui inclut une action de type c2pa.repackaged.

A.5.4.1.3. Boîte contenant l'auxiliaire merkle

Quelle que soit la structure de l'actif, les champs de la boîte correspondante décrite ci-dessus doivent être définis comme suit.

box_purpose

Pour une boîte C2PA « uuid » auxiliaire, cette valeur doit être merkle.

data

Lorsque box_purpose est merkle, cette valeur doit contenir des octets CBOR bruts indiquant comment valider une partie de l'actif, comme défini ci-dessous. S'il existe plusieurs boîtes C2PA « uuid » auxiliaires avec box_purpose merkle pour un arbre Merkle donné dans un seul fichier, chacune doit être suivie d'un nombre suffisant d'octets de remplissage (zéro ou plus) pour que toutes les boîtes C2PA « uuid » auxiliaires de cet arbre Merkle aient une taille fixe.

REMARQUE Lorsqu'il y a plusieurs de ces boîtes dans un seul fichier, c'est-à-dire dans le cas où il y a de grandes

Les « mdat » étant validés au fur et à mesure, une taille fixe est nécessaire afin de permettre à un client effectuant un téléchargement progressif de ne télécharger que les boîtes dont il a besoin pour commencer la validation plutôt que l'arborescence Merkle entière. Un tel client peut télécharger suffisamment de la première de ces boîtes en fonction du décalage absolu en octets du fichier dans le `manifeste actif` pour déterminer si son uniqueId et son localId correspondent au « mdat » qu'il tente de valider. Si c'est le cas, il peut déterminer le décalage absolu en octets du fichier par rapport à la boîte qu'il doit valider en multipliant le numéro du sous-ensemble par cette taille, puis télécharger uniquement cette boîte. Sinon, il peut déterminer le décalage absolu en octets du fichier par rapport au début de l'arbre Merkle suivant en multipliant cette taille fixe par le nombre total de nœuds feuilles de l'arbre Merkle actuel, et il peut répéter ce processus jusqu'à ce qu'il localise la boîte dont il a besoin. La taille totale du téléchargement pour ce sous-ensemble de boîtes est très petite par rapport à la taille d'un seul sous-ensemble.

A.5.4.2. Schéma et exemple

Le schéma de ce type est défini par la règle `bmff-merkle-map` dans la définition CDDL suivante :

```
; La structure de données utilisée pour stocker suffisamment d'informations pour valider une seule boîte « mdat »
ou

; une partie d'une boîte « mdat » lorsqu'un arbre Merkle est utilisé
», bmff-merkle-map = {
  « uniqueId » : int, ; Un entier unique utilisé pour différencier les identifiants
  locaux « localId » : int, ; Un identifiant local indiquant l'arbre Merkle.
  « location » : int, ; Index basé sur zéro dans la ligne la plus basse de l'arbre Merkle correspondant à
cette boîte « mdat » ou à une partie de cette boîte « mdat ».
  ? « hashes » : [1* bstr], ; Tableau ordonné représentant l'ensemble des hachages supplémentaires requis
pour atteindre un hachage dans l'arbre Merkle spécifié dans le manifeste, depuis la feuille la plus basse
(pair de ce nœud) jusqu'à la racine la plus haute (enfant du nœud dans le manifeste). Notez que ce tableau
peut ne pas être présent, par exemple si le manifeste lui-même contient la ligne la plus à gauche de l'arbre
Merkle. Les hachages nuls ne sont pas inclus dans ce tableau. L'algorithme utilisé est déterminé à l'aide du
champ « alg » de l'entrée correspondante dans le tableau de champs « merkle » de la structure de hachage
BMFF.
}
```

Un exemple en notation diagnostique CBOR (RFC 8949, clause 8) est présenté ci-dessous :

```
{
  « hachages » : [
    b64'TWVub3JhaA=='
  ],
  « localId » : 4402,
  « location » : 2203,
  « uniqueId » : 1339
}
```

Pour les ressources non fragmentées, le champ `localId` dans le `bmff-merkle-map` doit indiquer la case « `mdat` ». Il s'agit d'un index basé sur zéro indiquant l'ordre de « mdat » dans le fichier. Pour les ressources fragmentées, le champ `localId` dans le `bmff-merkle-map` doit être défini sur le champ `track_id` de la case « `tkhd` » correspondant au « `mdat` » haché.

A.5.5. Génération dynamique de flux

De nombreuses implémentations de streaming à débit adaptatif (ABR) stockent une seule version d'un actif, par exemple sous forme de MP4 plat ou dans

un autre format intermédiaire, et générer des flux d'actifs individuels à l'aide de divers codecs, débits binaires, etc. au moment de la consommation. En conséquence, un tel serveur doit soit hacher lesdits flux et créer un manifeste C2PA chaque fois que le contenu est consommé, soit, si la génération est déterministe, créer et mettre en cache les hachages et les manifestes C2PA une seule fois, puis les intégrer au moment de la consommation.

A.5.6. Exigences relatives à la liste d'exclusion

Pour toutes les assertions `c2pa.hash.bmff.v2` (obsolètes) et `c2pa.hash.bmff.v3`, les entrées de l'exemple 18, « Boîtes toujours exclues », doivent toujours figurer sur la liste d'exclusion. D'autres entrées sont autorisées mais non obligatoires.

L'ensemble de la boîte C2PA « `uuid` » doit être exclu. (Le champ « `data` » garantit que les autres boîtes « `uuid` » ne sont pas exclues.)

Exemple 18. Boîtes toujours exclues

```
xpath = "/uuid"
data = [ { offset = 8, data = b64'2P7D1hsOSDyS11goh37EgQ==' } ]
```

Les cases « `ftyp` » et « `mfra` » doivent être exclues dans leur intégralité.

```
xpath = "/ftyp"
```

```
xpath = "/mfra"
```

REMARQUE Les versions précédentes de cette spécification incluaient des exclusions obligatoires supplémentaires, mais il a été découvert que leur exclusion n'était pas sécurisée.

Pour toutes les assertions `c2pa.hash.bmff.v2` (obsolètes) et `c2pa.hash.bmff.v3` où la bmff-hash-map comprend à la fois le champ `hash` et les champs `merkle`, l'entrée de l'exemple 19, « Boîtes supplémentaires toujours exclues », doit figurer sur la liste d'exclusion.

Exemple 19. Boîtes supplémentaires toujours exclues

```
xpath = "/mdat"
sous-ensemble = { { 16, 0 } }
```

REMARQUE Comme indiqué dans la définition CDDL ci-dessus, l'assertion `c2pa.hash.bmff` exclut l'intégralité de la boîte « `mdat` » dans ce cas, mais il a été découvert que cette exclusion n'était pas sécurisée.

Comme indiqué dans la définition CDDL ci-dessus, un décalage relatif en octets ou un décalage relatif en octets plus une longueur qui dépasse la longueur de la boîte est autorisé ; les octets au-delà de la fin de la boîte ne doivent jamais être hachés. Par exemple, si la boîte `mdat` ne fait que 12 octets de long, elle est entièrement hachée et l'entrée d'exclusion obligatoire susmentionnée n'a aucun effet, bien qu'elle soit

toujours requise.

A.5.7. Flux multimédias synchronisés qui ne sont ni audio ni vidéo

Les flux multimédias synchronisés qui ne sont ni audio ni vidéo, tels que les flux de texte pour les sous-titres, que le générateur de revendications souhaite rendre inviolables, doivent être traités de la même manière que les flux audio et vidéo.

A.5.8. Références externes

Le contenu référencé en externe déclaré dans les boîtes BMFF, par exemple dans une boîte « `dref` », « `url` » ou « `urn` », que le générateur de revendications souhaite rendre inviolable **ne** doit **pas** exclure la boîte de référence et doit inclure une assertion de données cloud distincte pour chaque référence externe à hacher.

A.5.9. Exigences en matière de taille

Si un actif basé sur BMFF utilise des tailles ou des décalages de 32 bits dans une ou plusieurs boîtes, par exemple la boîte « `stco` », et que l'ajout de boîtes pour se conformer à cette spécification fait passer la taille du fichier au-delà de 4 gigaoctets, il incombe au créateur du manifeste de modifier le fichier afin d'utiliser des tailles et des décalages appropriés, par exemple en remplaçant la boîte « `stco` » par une boîte « `co64` », avant de créer le manifeste.

A.6. Intégration de manifestes dans des formats ZIP

A.6.1. Généralités

En raison de leur longévité et du fait qu'il s'agit de spécifications publiées ouvertement, de nombreux formats de fichiers de commande sont en réalité des archives ZIP, mais avec une organisation spécifique des fichiers de contenu. Cela inclut des formats tels que EPUB, Office Open XML, Open Document et OpenXPS.

A.6.2. Hachage

A.6.2.1. Hachage des fichiers

Un format de fichier ZIP doit être haché à l'aide d'un hachage de données de collection, dans lequel chaque fichier contenu dans le ZIP (à l'exception du manifeste C2PA lui-même) doit être inclus. Le hachage de chaque fichier de la collection est calculé à partir de `l'en-tête du fichier local`, suivi du contenu compressé et/ou crypté, et de toute description de données, le cas échéant. L'algorithme de hachage utilisé doit être spécifié dans le champ `alg` de la structure de hachage des données de collection.

REMARQUE La raison pour laquelle le hachage est effectué sur le contenu compressé/crypté est de permettre la validation sans la nécessité de décompresser ou de disposer de la clé de déchiffrement. Ceci est important pour les formats pouvant être chiffrés, tels que EPUB.

A.6.2.2. Hachage du répertoire central ZIP

Comme décrit dans la section 4.3.12 de la note d'application ZIP, le répertoire central est un tableau d'en-têtes de répertoire central, à raison d'un par fichier dans l'archive ZIP. Il est stocké à la fin de l'archive ZIP et sert à localiser les fichiers dans l'archive ZIP ainsi que les informations/métadonnées nécessaires les concernant.

informations/métadonnées les concernant. Il est immédiatement suivi de l'enregistrement de fin du répertoire central (ZIP AppNote, 4.3.16), qui contient des informations sur l'archive ZIP elle-même.

Afin d'empêcher toute altération du répertoire central ZIP, telle que l'ajout de nouveaux fichiers ou la modification d'informations sur les fichiers existants, chaque « en-tête du répertoire central » dans le répertoire central ZIP ainsi que la « fin de l'enregistrement du répertoire central » doivent être hachés. Le hachage est calculé sur la plage d'octets allant du premier octet de l'« en-tête du répertoire central » au dernier octet de la « fin de l'enregistrement du répertoire central » à l'aide de l'algorithme de hachage spécifié dans le champ `alg` de la structure de hachage des données de collection.

REMARQUE Les « en-têtes du répertoire central » sont stockées de manière contiguë et immédiatement suivies de la « fin de l'enregistrement du répertoire central ».

La valeur de hachage résultante doit être stockée dans le champ `zip_central_directory_hash` de la structure de hachage des données de collection.

REMARQUE L'utilisation d'un fichier spécialement nommé dans la liste des fichiers a été envisagée, mais n'a pas été acceptée en raison du scénario à deux passes décrit ci-dessous.

```
; Un tableau d'URI et leurs hachages associés
$collection-data-hash-map /= { "uris": [1*
  uri-hashed-data-map],
  « alg » : tstr .size (1..max-tstr-length), ; Une chaîne identifiant l'algorithme de hachage cryptographique
utilisé pour calculer le hachage de chaque entrée du tableau « uris », tirée de la liste des identifiants
d'algorithmes de hachage C2PA.
    ? « zip_central_directory_hash » : bstr,
}

; Structure de données utilisée pour stocker une référence à un URI et son hachage.
$uri-hashed-data-map /= {
  « uri » : type d'URL relative, ; référence URI relative « hash » :
  bstr, ; chaîne d'octets contenant la valeur de hachage
  ? « size » : type-taille, ; nombre d'octets de données
  ? « dc:format » : format-string, ; type de média IANA des données
  ? « data_types » : [1* $asset-type-map], ; informations supplémentaires sur le type des données
}

; avec CBOR La tête (#) et la queue ($) sont introduites dans l'expression régulière, elles ne sont donc pas
nécessaires explicitement type-url-relatif /= tstr .regexp « [-a-zA-Z0-9@:%._\\+~#=]{2,256}\\.[a-z]{2,6}\\b[-
a-zA-Z0-9@:%_\\+.~#?&//=]*"
```

Étant donné que le fichier ZIP doit être terminé avant l'achèvement du manifeste C2PA, une approche en deux étapes (comme décrit pour les formats JPEG, BMFF et PDF) doit être utilisée. Le premier passage crée un fichier ZIP avec un fichier `content_credential.c2pa` rempli de zéros et calcule le hachage du répertoire central ZIP. Le deuxième passage complète le manifeste C2PA, y compris en remplissant la valeur du champ `zip_central_directory_hash`.

Une implémentation possible de cette approche en deux étapes serait la suivante :

- créer un fichier ZIP avec un fichier C2PA Manifest Store rempli de zéros (suffisamment grand pour être remplacé) ;
- calculer le hachage du répertoire central ZIP ;
- ajouter le hachage au champ `zip_central_directory_hash` du `collection-data-hash-map` ;

- compléter le manifeste ;
- Remplacez le fichier `content_credential.c2pa` rempli de zéros par les données du manifeste complétées.

Lors de la création du fichier `content_credential.c2pa` dans l'archive ZIP, celui-ci doit être stocké (méthode de compression 0) et non crypté. Ses champs `bit flag` et `crc-32 à usage général` doivent être définis sur 0. Les champs date et heure peuvent être définis sur l'heure de création de l'archive ZIP ou sur 0. Il peut comporter un commentaire de fichier.

A.6.3. Emplacement du magasin de manifestes

Le magasin de manifeste C2PA doit être stocké dans le répertoire `META-INF` de l'archive ZIP avec un nom de fichier `content_credential.c2pa` et un type de média recommandé pour les manifestes externes. Le fichier doit être stocké (méthode de compression 0) et non crypté.

A.6.4. Signature numérique des formats basés sur ZIP

A.6.4.1. EPUB

Les signatures numériques EPUB sont basées sur W3C XML DigSig Core, où chaque fichier signé est répertorié dans l'élément élément `<Manifest>` de l'élément `<Signature>`. De plus, la signature du répertoire central ZIP n'est pas prise en charge. Par conséquent, la signature native EPUB doit avoir lieu avant l'introduction du manifeste C2PA.

A.6.4.2. Office Open XML

Les signatures numériques OOXML sont basées sur W3C XML DigSig Core, où chaque fichier signé est répertorié comme un élément élément `<Reference>` dans l'élément `<Manifest>` de l'élément `<Signature>`. De plus, la signature du répertoire central ZIP n'est pas prise en charge. Ainsi, la signature native OOXML doit avoir lieu avant l'introduction du manifeste C2PA.

REMARQUE OpenXPS est basé sur la même norme Open Packaging Convention (OPC) que OOXML, et la même approche s'applique donc.

Annexe B : Détails de mise en œuvre pour `c2pa.metadata`

L'assertion `c2pa.metadata` ne doit contenir que le sous-ensemble de schémas et leurs champs décrits ci-dessous. Cependant, les assertions de métadonnées personnalisées peuvent contenir n'importe quelle valeur provenant de ces schémas ou d'autres schémas.

REMARQUE Une liste lisible par machine de tous les schémas valides et de leurs champs est disponible sur le site Web des spécifications C2PA.

Les valeurs présentes dans une assertion `c2pa.metadata` peuvent être propres à l'assertion de métadonnées ou provenir des « blocs de métadonnées » standard du format de l'actif. Dans les deux cas, elles doivent être sérialisées selon les règles de sérialisation JSON-LD de XMP décrites ici.

B.1. Schémas entièrement pris en charge

Les schémas/espaces de noms suivants, répertoriés dans le tableau 15, « Schémas entièrement pris en charge », sont entièrement pris en charge par tous les signataires :

Tableau 15. Schémas entièrement pris en charge

Nom	Espace de noms
XMP Basic	http://ns.adobe.com/xap/1.0/
Gestion des médias XMP	http://ns.adobe.com/xap/1.0/mm/
XMP Paged-Text	http://ns.adobe.com/xap/1.0/t/pg/
Camera Raw	http://ns.adobe.com/camera-raw-settings/1.0/
PDF	http://ns.adobe.com/pdf/1.3/

B.2. Schémas partiellement pris en charge

Les schémas/espaces de noms suivants, dans le tableau 16, « Schémas partiellement pris en charge », ne sont pris en charge que partiellement.

Tableau 16. Schémas partiellement pris en charge

Nom	**Espace de noms**
Dublin Core (DC)	http://purl.org/dc/elements/1.1/
IPTC Core	http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/
Extension IPTC	http://iptc.org/std/Iptc4xmpExt/2008-02-29/
Exif	http://ns.adobe.com/exif/1.0/
ExifEx	http://cipa.jp/exif/1.0/exifEX
Nom	**Espace de noms**

Photoshop	http://ns.adobe.com/photoshop/1.0/
TIFF	http://ns.adobe.com/tiff/1.0/
XMP Dynamic Media	http://ns.adobe.com/xmp/1.0/DynamicMedia/
PLUS	http://ns.useplus.org/ldf/xmp/1.0/

B.2.1. Dublin Core (DC)

Seules les propriétés Dublin Core (`dc`) suivantes sont prises en charge :

- `dc:coverage`
- `dc:date`
- `dc:format`
- `dc:identifier`
- `dc:language`
- `dc:relation`
- `dc:type`

B.2.2. IPTC Core

Seules les propriétés IPTC Core (`Iptc4xmpCore`) suivantes sont prises en charge :

- `Iptc4xmpCore:Scene`

REMARQUE Certaines propriétés IPTC Core ont été remplacées par des versions plus récentes dans le schéma IPTC Extension.

B.2.3. Extension IPTC

Seules les propriétés IPTC Extension (`Iptc4xmpExt`) suivantes sont prises en charge :

- `Iptc4xmpExt:DigImageGUID`
- `Iptc4xmpExt:DigitalSourceType`
- `Iptc4xmpExt:EventId`
- `Iptc4xmpExt:Genre`
- `Iptc4xmpExt:ImageRating`
- `Iptc4xmpExt:Région de l'image`
- `Iptc4xmpExt:Identifiant du registre`
- `Iptc4xmpExt:Lieu de création`

- `Iptc4xmpExt:Emplacement affiché`
- `Iptc4xmpExt:Hauteur maximale disponible`
- `Iptc4xmpExt:Largeur maximale disponible`

Pour plus d' informations sur ceux-ci, consultez à https://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata#xmp-namespaces-and-identifiers-2.

B.2.4. Exif

Seules les propriétés Exif suivantes, répertoriées dans le tableau 17 « Propriétés Exif prises en charge », sont prises en charge :

Tableau 17. Propriétés Exif prises en charge

- exif:ApertureValue
- exif:BrightnessValue
- exif:CFAPattern
- exif:ColorSpace
- exif:CompressedBitsPerPixel
- exif:Contraste
- exif:Rendu personnalisé
- exif:Date et heure de numérisation
- exif:Date et heure d'origine
- exif:Description des paramètres de l'appareil
- exif:TauxDeZoomNumérique
- exif:VersionExif
- exif:Valeur de correction d'exposition
- exif:Indice d'exposition
- exif:Mode d'exposition
- exif:Programme d'exposition
- exif:Temps d'exposition
- exif:Source du fichier
- exif:Flash
- exif:Énergie du flash
- exif:VersionFlashpix
- exif:FNumber
- exif:Longueur focale
- exif:Longueur focale en 35 mm
- exif:Résolution du plan focal
- exif:RésolutionXDuPlanFocal
- exif:RésolutionY du plan focal
- exif:Contrôle du gain
- exif:IdentifiantUniqueDeL'Image
- exif:Indices de sensibilité ISO
- exif:SourceLumière
- exif:Valeur d'ouverture maximale
- exif:Mode de mesure
- exif:OECF
- exif:Décalage horaire d'origine
- exif:DimensionPixelX
- exif:DimensionY en pixels
- exif:Fichier audio associé
- exif:Saturation
- exif:Type de capture de scène
- exif:Type de scène
- exif:Méthode de détection
- exif:Netteté
- exif:Valeur de vitesse d'obturation
- exif:Réponse en fréquence spatiale
- exif:Sensibilité spectrale
- exif:Zone du sujet
- exif:Distance du sujet
- exif:Distance du sujet
- exif:Emplacement du sujet
- exif:Balance des blancs
- exif:AltitudeGPS
- exif:RéférenceAltitudeGPS
- exif:DateGPS
- exif:DirectionGPSDestination
- exif:référence de direction GPS
- exif:DistanceGPSDestination
- exif:DistanceGPSDestinationRéférence
- exif:GPSDestLatitude
- exif:GPSDestLongitude
- exif:Différentiel GPS
- exif:GPSDOP
- exif:Erreur de positionnement GPS ou
- exif:Direction de l'image GPS
- exif:GPSImgDirectionRef
- exif:GPSLatitude
- exif:GPSLongitude
- exif:GPSMapDatum
- exif:Mode de mesure GPS
- exif:GPSProcessingMetho d
- exif:SatellitesGPS
- exif:VitesseGPS
- exif:Référence de vitesse GPS
- exif:StatutGPS
- exif:HorodatageGPS
- exif:GPSTrack
- exif:RéférenceTraceGPS
- exif:IDVersionGPS

B.2.5. ExifEx

Seules les propriétés ExifEx suivantes sont prises en charge :

- `exifEX:BodySerialNumber`
- `exifEX:Gamma`
- `exifEX:InteroperabilityIndex`
- `exifEX:ISOSpeed`
- `exifEX:ISOSpeedLatitudeyyy`
- `exifEX:LatitudeISOzzz`
- `exifEX:FabricantObjectif`
- `exifEX:Modèle d'objectif`
- `exifEX:Numéro de série de l'objectif`
- `exifEX:Spécifications de l'objectif`
- `exifEX:SensibilitéPhotographique`
- `exifEX:Indice d'exposition recommandé`
- `exifEX:TypeDeSensibilité`
- `exifEX:Sensibilité standard en sortie`

Pour plus d'informations à ce sujet, consultez https://www.cipa.jp/std/documents/download_e.html?DC-010-2020_E.

B.2.6. Photoshop

Seules les propriétés Photoshop suivantes sont prises en charge :

- `photoshop:Catégorie`
- `photoshop:City`
- `photoshop:ColorMode`
- `photoshop:Pays`
- `photoshop:DateCreated`
- `photoshop:DocumentsAncêtres`
- `photoshop:Historique`
- `photoshop:profil ICC`
- `photoshop:État`
- `photoshop:Catégories supplémentaires`

- photoshop:TextLayers
- photoshop:Référence de transmission
- photoshop:Urgence

B.2.7. TIFF

Seules les propriétés TIFF suivantes sont prises en charge :

- tiff : BitsParÉchantillon
- tiff:Compression
- tiff:DateTime
- tiff:ImageLength
- tiff:ImageWidth
- tiff:Fabricant
- tiff:Modèle
- tiff : Orientation
- tiff:Interprétation photométrique
- tiff:ConfigurationPlanaire
- tiff:Chromaticités primaires
- tiff:NoirBlancDeRéférence
- tiff:Unité de résolution
- tiff:ÉchantillonsParPixel
- tiff:Logiciel
- tiff:Fonction de transfert
- tiff:Point blanc
- tiff:RésolutionX
- tiff:RésolutionY
- tiff:CoefficientsYCbCr
- tiff:PositionnementYCbCr
- tiff:Sous-échantillonnage YCbCr

B.2.8. XMP Dynamic Media

Seules les propriétés XMP Dynamic Media (xmpDM) suivantes, répertoriées dans le tableau 18, « Propriétés XMP Dynamic Media », sont prises en charge :

Tableau 18. Propriétés XMP Dynamic Media

- xmpDM:absPeakAudioFileP ath
- xmpDM:album
- xmpDM:altTapeName
- xmpDM:altTimecode
- xmpDM:audioChannelType
- xmpDM:compresseur audio
- xmpDM:fréquence d'échantillonnage audio
- xmpDM:type d'échantillon audio
- xmpDM:paramètres de raccordement des mesures
- xmpDM:angleCaméra
- xmpDM:cameraLabel
- xmpDM:modèle de caméra
- xmpDM:déplacement de caméra
- xmpDM:commentaire
- xmpDM:médias fournis
- xmpDM:durée
- xmpDM:débit binaire du fichier
- xmpDM:genre
- xmpDM:bon
- xmpDM:instrument
- xmpDM:durée d'introduction
- xmpDM:tonalité
- xmpDM:commentaire journal
- xmpDM:boucle
- xmpDM:nombreDeBattements
- xmpDM:marqueurs
- xmpDM:outCue
- xmpDM:nomDuProjet
- xmpDM:référenceProjet
- xmpDM:menu déroulant
- xmpDM:relativePeakAudio FilePath
- xmpDM:relativeTimestamp
- xmpDM:dateDeSortie
- xmpDM:paramètres de rééchantillonnage
- xmpDM:typeD'échelle
- xmpDM:scène
- xmpDM:dateDeTournage
- xmpDM:jourDeTournage
- xmpDM:lieu_de_prise_de_vue
- xmpDM:nomDeLaPriseDeVu
- xmpDM:numéro de prise de vue
- xmpDM:tailleDeLaPhoto
- xmpDM:placementEnceintes
- xmpDM:code temporel de début
- xmpDM:mode d'étirement
- xmpDM:numéroDePrise
- xmpDM:nomDeBande
- xmpDM:tempo
- xmpDM:paramètres d'échelle de temps
- xmpDM:signature
- xmpDM:numéroDePiste
- xmpDM:Pistes
- xmpDM:modeAlphaVidéo
- xmpDM:alphaPrémultipliéVidéo CouleurMultiple
- xmpDM:transparenceAlphaVidéo
- xmpDM:espaceCouleurVidéo
- xmpDM:compresseur vidéo
- xmpDM:ordre des champs vidéo
- xmpDM:fréquence d'images vidéo
- xmpDM:taille d'image vidéo
- xmpDM:rapport d'aspect vidéo
- xmpDM:profondeurPixelVidéo
- xmpDM:partieDeCompilation
- xmpDM:paroles
- xmpDM:numéro de disque

B.2.9. PLUS

Seules les propriétés PLUS suivantes sont prises en charge :

- plus:NomFichierTelQueLivré
- plus:DateDePremièrePublication

- `plus:FormatFichierImageTelQueLivré`
- `plus:TailleFichierImageTelQueLivré`
- `plus:TypeImage`
- `plus:Version`

Pour plus d'informations à ce sujet, consultez http://ns.useplus.org/LDF/ldf-XMPSpecification.

Annexe C : Considérations relatives à la dépréciation

C.1. Statut des constructions

Le tableau ci-dessous répertorie les constructions dont le statut a changé au fur et à mesure de l'évolution de cette spécification. Les valeurs de statut suivantes sont utilisées :

OBSOLÈTE

Construct est obsolète (les générateurs de revendications ne doivent plus le produire ; les validateurs sont encouragés à l'accepter).

INDÉFINI

La construction n'est pas définie (les validateurs sont tenus de l'ignorer).

<blank>

La construction est entièrement prise en charge (les validateurs sont tenus de l'accepter).

Tableau 19. Statut des constructions

Construction	Type	v1.3	v1.4	v2.0	v2.1	v2.2
Manifeste d'horodatage	Manifeste	INDÉFINI	INDÉFINI	INDÉFINI		INDÉFINI
`urn:uuid` espace de noms	Étiquette				OBSOLÈTE	OBSOLÈTE
`urn:c2pa` espace de noms	Étiquette	INDÉFINI	INDÉFINI	INDÉFINI		
`c2pa.data` (Boîte de données)	Étiquette					OBSOLÈTE
`c2pa.databoxes` (Magasin de boîtes de données)	Étiquette					OBSOLÈTE
`sigTst` timestamp	Horodatage				OBSOLÈTE	OBSOLÈTE
`sigTst2` horodatage	Horodatage	INDÉFINI	INDÉFINI	INDÉFINI		
`c2pa.claim`	Assertion			OBSOLÈTE	OBSOLÈTE	OBSOLÈTE
`c2pa.claim.v2`	Assertion	INDÉFINIE	INDÉFINIE			
`c2pa.actions`	Assertion					
Construction	**Type**	**v1.3**	**v1.4**	**v2.0**	**v2.1**	**v2.2**

c2pa.actions.v2	Assertion					
c2pa.asset-type	Assertion					OBSOLÈTE
c2pa.asset-type.v2	Assertion	INDÉFINIE	INDÉFINIE	INDÉFINIE	INDÉFINI	
c2pa.certificate-status	Assertion	INDÉFINI	INDÉFINI	INDÉFINI	INDÉFINI	
c2pa.embedded-data	Assertion	INDÉFINI	INDÉFINI	INDÉFINI	INDÉFINI	
c2pa.font.info	Assertion	INDÉFINIE		OBSOLÈTE	OBSOLÈTE	OBSOLÈTE
c2pa.hash.bmff	Assertion	OBSOLÈTE	OBSOLÈTE	INDÉFINI	INDÉFINI	INDÉFINI
c2pa.hash.bmff.v2	Assertion				OBSOLÈTE	OBSOLÈTE
c2pa.hash.bmff.v3	Assertion	INDÉFINIE	INDÉFINI	INDÉFINI		
c2pa.hash.collection.data	Assertion	INDÉFINIE				
c2pa.hash.multi-actifs	Assertion	INDÉFINIE	INDÉFINIE	INDÉFINI	INDÉFINI	
c2pa.ingredient	Assertion			OBSOLÈTE	OBSOLÈTE	OBSOLÈTE
c2pa.ingredient.v2	Assertion				OBSOLÈTE	OBSOLÈTE
c2pa.ingredient.v3	Assertion	INDÉFINIE	INDÉFINI	INDÉFINI		
stds.metadata	Assertion	INDÉFINIE		OBSOLÈTE	OBSOLÈTE	OBSOLÈTE
c2pa.metadata	Assertion	INDÉFINI	INDÉFINI			
c2pa.thumbnail.claim	Assertion	INDÉFINI	INDÉFINI	INDÉFINI	INDÉFINI	
c2pa.thumbnail.claim.*	Assertion					OBSOLÈTE
c2pa.thumbnail.ingredient	Assertion	INDÉFINIE	INDÉFINIE	INDÉFINI	INDÉFINI	
c2pa.thumbnail.ingredient.*	Assertion					OBSOLÈTE
Construction	**Type**	**v1.3**	**v1.4**	**v2.0**	**v2.1**	**v2.2**

c2pa.time-stamp	Assertion	INDÉFINIE	INDÉFINIE	INDÉFINI	INDÉFINI	
font.info	Assertion	INDÉFINI	INDÉFINI			
stds.iptc	Assertion		OBSOLÈTE	OBSOLÈTE	OBSOLÈTE	OBSOLÈTE
stds.exif	Assertion		OBSOLÈTE	OBSOLÈTE	OBSOLÈTE	OBSOLÈTE
stds.schema-org	Assertion		OBSOLÈTE	OBSOLÈTE	OBSOLÈTE	OBSOLÈTE
rôle dans carte-régionale	Champ				OBSOLÈTE	OBSOLÈTE
acteurs dans action-items-map-v2	Champ			OBSOLÈTE	OBSOLÈTE	OBSOLÈTE
softwareAgents dans actions-map-v2	Champ	INDÉFINIE	INDÉFINIE	INDÉFINI		
softwareAgentIndex dans action-common-map-v2	Champ	INDÉFINI	INDÉFINI			
modifié dans action-items-map-v2	Champ				OBSOLÈTE	OBSOLÈTE
modifications dans action-items-map-v2	Champ	INDÉFINI	INDÉFINI	INDÉFINI		
instanceID dans paramètres-map-v2	Champ				OBSOLÈTE	OBSOLÈTE
sourceLanguage in paramètres-map-v2	Champ	NON DÉFINI	INDÉFINI	INDÉFINI		
targetLanguage in paramètres-map-v2	Champ	NON DÉFINI	INDÉFINI	INDÉFINI		
c2pa.trainedAlgorithmicData	TypeSourceNumérique					OBSOLÈTE
Construct	**Type**	**v1.3**	**v1.4**	**v2.0**	**v2.1**	**v2.2**

http://c2pa.org/digitalsourcetype/trainedAlgorithmicData	TypeSourceNumérique	INDÉFINI	INDÉFINI	INDÉFINI	INDÉFINI	
http://c2pa.org/digitalsourcetype/ empty	TypeSourceNumérique	INDÉFINI	INDÉFINI	INDÉFINI	INDÉFINI	