bcasmart.com

DOCUMENT TECHNIQUE

SPÉCIFICATIONS FONCTIONNELLES DÉTAILLÉES DU CŒUR DE L'API DE SIGNATURE CERTIGNA NG

Référence :	2020-09-14-DT
Date d'émission :	20/01/2021
Révision	v1.1

BUSINESS CARD ASSOCIATES
SAS au capital de 260 148 € – SIRET 487 929 572 00070 – Code APE 6202A
75, rue du Général Leclerc – 92 270 BOIS-COLOMBES – FRANCE

SOMMAIRE

I. Objectifs

Le but de ce document est de définir précisément l'API de signature de nouvelle génération de la société CERTIGNA, appelée dans la suite du texte **API-NG**.

Le but de cette API-NG est d'offrir des fonctions de signature de documents de haut niveau permettant aux systèmes utilisateurs de cette API-NG d'organiser des workflows d'approbation et de signature de documents.

Le présent document prend comme référence la spécification fonctionnelle générale de l'API-NG qui est décrite dans le document de CERTIGNA : API_Signature_NG_v_0_9.pdf appelé [CERTIGNA_NG] dans la suite du texte.

Afin d'être aussi précis que possible dans la définition de l'API-NG et avant de lister de manière exhaustive l'ensemble des ressources et fonctions disponibles, plusieurs principes doivent être également précisés, expliqués voire modifiés par rapport aux spécifications générales initiales :

- le mode de fonctionnement et les protocoles définis pour utiliser l'API-NG,
- l'authentification des usagers de l'API et leurs rôles,
- l'architecture technique de l'API,
- le fonctionnement global du workflow et son impact sur le rôle des acteurs de la signature,
- les interactions entre le fonctionnement du workflow et les types de signatures attendues.

II. Principes généraux

1. Forme de l'API-NG

L'API-NG est une API RESTful et on y accède via le protocole HTTP/S.

Chaque ressource de l'API-NG est donc accessible selon une URL unique, versionnée et comprenant potentiellement des parties variables lorsqu'il s'agit de requêtes potentiellement multicritères (fonction de type GET).

Dans ce schéma chaque requête est envoyée via une requête HTTP sur une URL décomposable en cinq parties :

Le protocole	Le Host	La Racine	Le versionning	La ressource
`https://`	`myServer.com`	`/myService/mySubservice`	`/1.5`	`/session/25`

Le host et la racine de l'URL sont du ressort de la gestion du DNS[1].

Le versionning de l'API est une chaîne de caractère au format `"x"` ou `"x.y"`, `x` et `y` étant des nombres entiers positifs.

La ressource est représentée par le chemin d'accès vers l'objet ou les objets que l'on souhaite rechercher, lire, modifier ou ajouter. Le chemin est constitué des noms des ressources accédées et de leurs identifiants sous la forme de nombres entiers positifs.

Les noms des ressources peuvent être au singulier ou au pluriel. L'usage d'un nom au singulier correspond à l'accès à une ressource particulière et doit être immédiatement suivi de l'identifiant de la ressource en question. L'usage du pluriel correspond à la recherche, la lecture ou la modification d'un ensemble de ressources.

Exemples de requêtes sur l'API-NG illustrant les règles de constitution des chemins des ressources et des fonctions :

Action	Partie ressource/fonction de l'URL	Signification
`GET`	`/session/25`	Lecture du contenu de la session d'identifiant 25
`GET`	`/sessions`	Lecture de toutes les sessions de l'usager connecté
`PUT`	`/sessions`	Création d'une nouvelle session
`GET`	`/session/25/scenario/100`	Lecture du scenario d'ID 100 de la session d'ID 25
`GET`	`/session/25/scenarios`	Liste des IDs des scenarios de la session d'ID 25

La définition complète de l'API est donnée plus loin dans ce document.

Chaque requête HTTP est constituée de headers dédiés et éventuellement d'un BODY HTTP au format JSON, ou éventuellement binaire lorsqu'il s'agit d'uploader des documents par exemple.

[1] l'URL peut aissi contenir un numéro de port de communication non mentionné dans l'exemple.

Chaque réponse à une requête est une message HTTP avec un code d'erreur, des headers et éventuellement un BODY HTTP au format JSON ou binaire selon les cas.

L'API étant éminemment dynamique, le retour de ces requêtes ne doit jamais être caché. À chaque renvoi d'informations, l'API-NG renvoie donc le header `Cache-Control: no-store` afin d'indiquer à l'appelant de ne pas faire de caching.

Sauf en ce qui concerne le transfert de fichiers, le body HTTP est toujours au format JSON en UTF-8 contenant une partie imposée par le type de ressource transmise ou reçu et une partie libre que l'usager du service peut ajouter ou récupérer par ses requêtes (partie `"user-data"` dans l'exemple ci-après).

Tout envoi d'un BODY non conforme provoquera le renvoi d'une erreur de type 400 BAD REQUEST.

Les dates transmises dans les headers HTTP, dans les variables post-URL des requêtes de recherche et au sein des contenus JSON sont exprimées en GMT sous la forme de chaînes de caractères conformes au format ISO 8601.

Attention : l'API-NG ne gère pas du tout la notion de Time zone, toutes les dates qu'elle manipule et stocke sont en GMT et il appartient au système appelant de faire les transformations nécessaires pour exprimer les dates dans sa Time zone cible.

Les identifiants (de sessions, d'acteurs, de documents…) retournés par les requêtes peuvent être soit des URL sans le versionning (forme courte) soit des URL avec le versionning (forme longue).

Le choix entre les deux se fait de manière globale dans le fichier de configuration de l'API à l'aide de la variable booléenne `long-identifiers`.

La forme courte est proposée par défaut.

Exemple de requête HTTP envoyée à un serveur host de l'API-NG:

```
GET /v1.5/session/25
DefaultLanguage "fr"
```

Exemple de retour HTTP de la requête avec les identifiants sous forme courte :

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Date: 2020-04-21T08:12:02.252Z
Last-Modified: 2020-04-21T14:03:02.720Z
Expires: 2020-04-22T08:13:00Z
...
{
  "id": 25,
  "status": 1,
  "actors": [
    "/session/25/actor/100",
    "/session/25/actor/20",
    "/session/25/actor/35"
  ],
  "documents": [
    "/session/25/document/1000",
    "/session/25/document/300",
    "/session/25/document/500"
  ],
  "scenarios": ["/session/25/scenario/1232"],
  "ttl": 86400,
  "user-data": {
    "label": "Signature des documents du pacte A452",
    "internal-id": "52DDF244-FC3E-40EE-BCC4-D5A75E686032"
  }
}
```

Le même retour de requête avec les identifiants sous leur forme longue :

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Date: 2020-04-21T08:12:02.252Z
Last-Modified: 2020-04-21T14:03:02.720Z
Expires: 2020-04-22T08:13:00Z
...
{
  "id": 25,
  "status": 1,
  "actors": [
    "/v1.5/session/25/actor/100",
    "/v1.5/session/25/actor/20",
    "/v1.5/session/25/actor/35"
  ],
  "documents": [
    "/v1.5/session/25/document/1000",
    "/v1.5/session/25/document/300",
    "/v1.5/session/25/document/500"
  ],
  "scenarios": ["/v1.5/session/25/scenario/1232"],
  "ttl": 86400,
  "user-data": {
    "label": "Signature des documents du pacte A452",
    "internal-id": "52DDF244-FC3E-40EE-BCC4-D5A75E686032"
  }
}
```

2. Architecture de l'API-NG et authentification

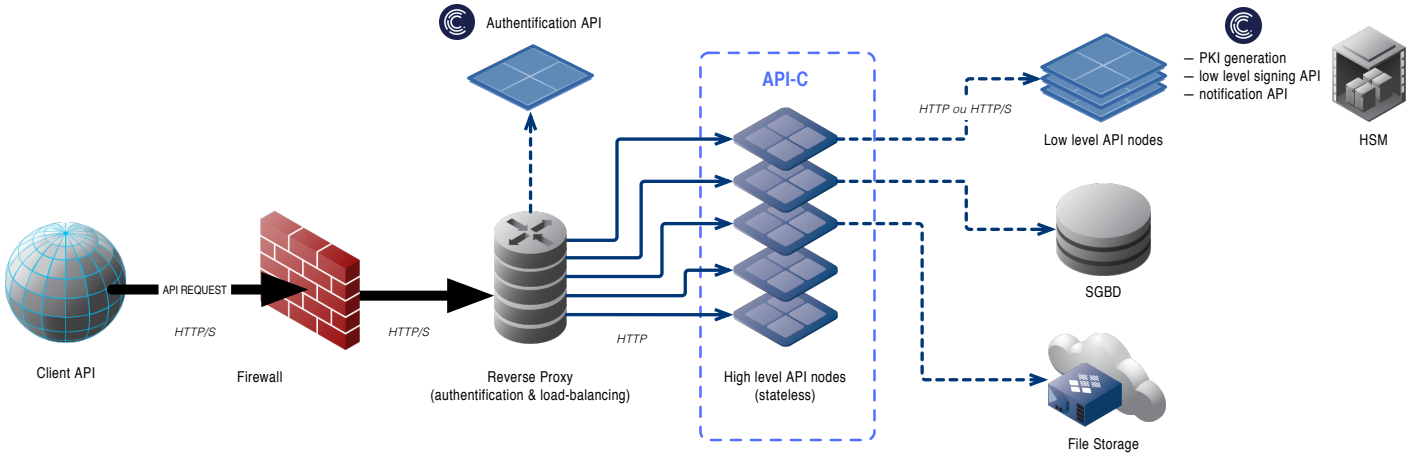
L'API-NG de CERTIGNA est destinée à être opérée sous forme SAAS ou en interne chez des clients voulant l'intégrer sur leur propre réseau.

Étant une API de haut niveau, en interne elle utilise d'autres services pour :

- dispatcher les requêtes reçues,
- authentifier les usagers et connaître leurs droits,
- sauvegarder ses sessions de workflow de signature et les fichiers manipulés,
- opérer effectivement les opérations cryptographiques de génération des clés et de signature,
- notifier les usagers.

L'API-NG s'inscrit dans un schéma opérationnel traditionnel autour de plusieurs composants :

- le système de load balancing et d'authentification qui vient en aval du cœur de service et qui est connecté à l'API d'authentification bas niveau de CERTIGNA ;
- l'**API-C**, le cœur de service de l'API-NG lui-même, qui est un composant RESTFul et stateless, appelé par le système de load balancing une fois l'appelant authentifié ;
- l'API de bas niveau de signature de CERTIGNA venant en amont de l'**API-C** dont le rôle est d'effectuer les signatures ;
- l'API de bas niveau de création de certificats à la volée venant en amont de l'**API-C** ;
- l'API de bas niveau permettant l'envoi de notifications ;
- une base de données intégrant la notion de transaction et assurant le stockage du contexte géré par l'**API-C**.
- un système de stockage des fichiers manipulés par l'**API-C**.



Du point de vue de l'utilisateur final (système client de l'API), l'API-NG est la seule interface qu'il connaît.

Du point de vue du cœur de l'API-NG (API-C), le client appelant est déjà authentifié et qualifié et les URL appelées ne contiennent plus que les parties utiles au traitement (versionning + accès aux ressources).

L'authentification de l'appelant est réalisée par le reverse proxy qui se connecte pour cela à l'API d'authentification de CERTIGNA dont le rôle est d'accepter ou de rejeter la connexion et, en cas de succès, de renvoyer le rôle de l'appelant (voir section sur l'authentification).

Le principe est que l'API-C reçoit dans les headers des requêtes qui lui sont envoyées des informations d'authentification issue du processus d'authentification mis en place par le système d'authentification et de load balancing et ce quelque soit la forme d'authentification utilisée par ce système.

En pratique, cela signifie que le système d'authentification et de load balancing ajoute trois headers HTTP spécifiques dans les requêtes qu'il envoie effectivement au composant API-C pour préciser :

- l'identifiant de l'usager via le header `CertignaUser`,
- le hash du password de l'usager en base64 via le header `CertignaHash`,
- le rôle de l'usager sous la forme d'un chiffre via le header `CertignaRole` (voir tableau ci-après).

Les rôles possibles d'un usager connecté sont :

Code rôle	Rôle	Description
1	Requêteur	L'appelant peut requêter et consulter l'ensemble des ressources mais ne peut ni créer, ni supprimer, ni modifier de ressources
2	Acteur	L'appelant peut créer de nouvelles sessions ou modifier des sessions ou des sous-ressources des sessions qu'il a précédemment créées. Il ne peut en aucun cas requêter ou lire le contenu de sessions qu'il n'a pas créées. L'appelant peut également réaliser toutes opération d'approbation de documents, ou de signatures.
3	Mainteneur	L'appelant peut requêter et consulter l'ensemble des ressources. Il ne peut pas créer de nouvelles sessions. Il peut modifier des sessions ou des sous-ressources des sessions. L'appelant ne peut pas réaliser d'opérations d'approbation de documents, ou de signatures.
4	Système	Ce rôle est le même que celui de mainteneur. La différence est que l'acteur est considéré comme étant le système de l'API-NG alors que le mainteneur est un client comme les autres avec des pouvoirs particuliers. Il n'existe pas *a priori* de différences de droits entre le l'appelant "Mainteneur" et "Système". *La présence de ce quatrième rôle est donc une anticipation des besoins de maintenance système qui ne manqueront pas d'apparaître dans la vie de l'API-NG.*

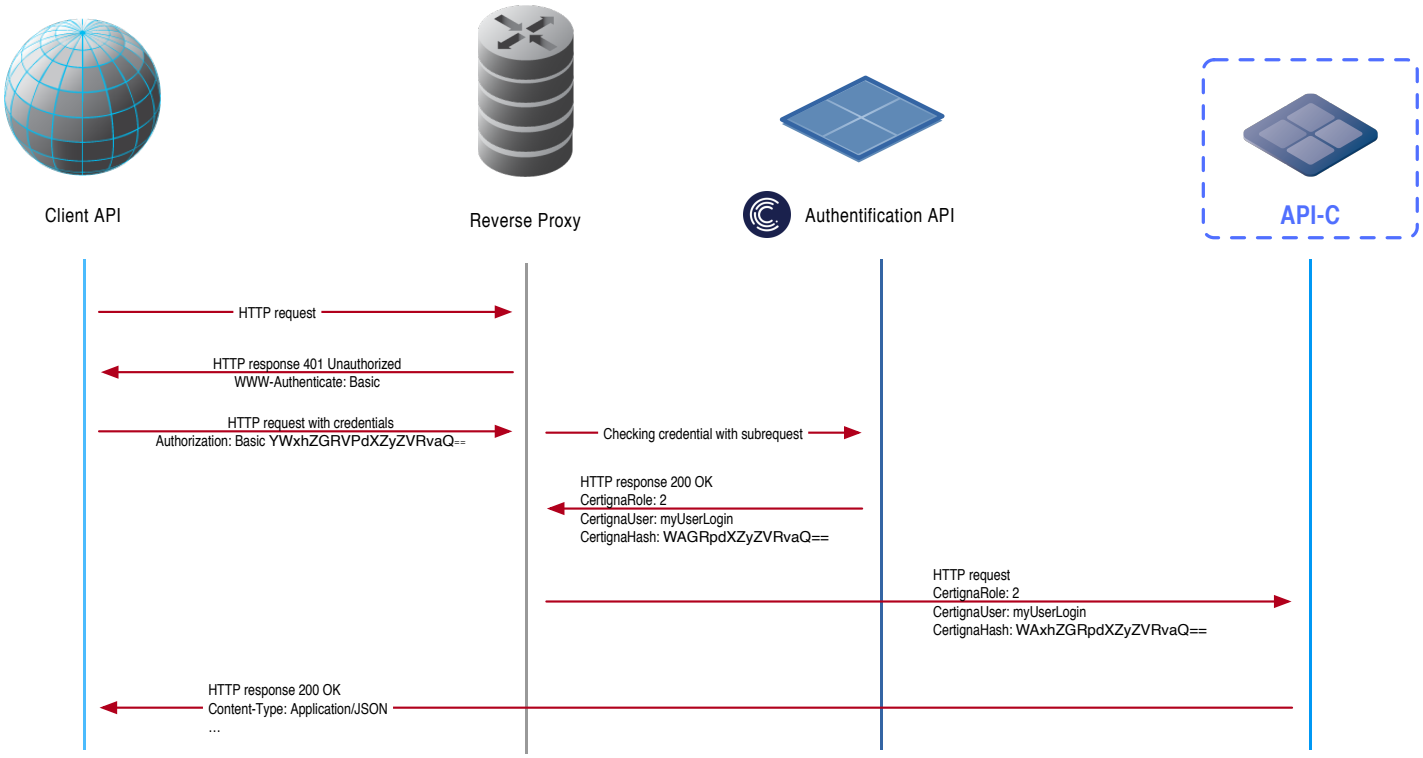
Exemple de requête envoyée à l'API-C avec les headers d'identifications ajoutés :

```
GET /v1.5/session/25
DefaultLanguage "fr"
CertignaUser: myuserlogin
CertignaHash: WAxhZGRpdXZyZVRvaQ==
CertignaRole: 2
```

En pratique, l'API-C devrait être placée derrière un système de proxy comme ngynx et utiliser des sous-requêtes pour réaliser l'authentification sur le système interne de CERTIGNA puis positionner les headers d'authentification et de rôle attendus.

Ainsi le cœur de l'API (API-C) devient indépendant de tout moyen d'authentification préalable même s'il est prévu que la première version de l'API-NG fonctionne avec une authentification de type HTTP basique.

Dans un tel schéma d'authentification en HTTP basique, une requête envoyée à l'API suivrait le schéma suivant :



De fait, l'API-C est donc une API passe-plats dont le rôle est de gérer le contexte de signature et qui utilise d'autres services pour réaliser des opérations cryptographiques de bas niveau pour :

- effectuer le scellement, l'approbation et la signature de documents : les opérations de signature seront effectuées par le service de signature de bas niveau de CERTIGNA ;
- générer des certificats à la volée : un service de gestion de PKI de bas niveau sera fourni par CERTIGNA. Ce service sera appelé par le cœur du service de l'API-NG et générera des biclés sur un boîtier cryptographique (HSM), génère une CSR (Certificate Signing Request), la signe et la transmet à l'Autorité de Certification qui fournit en retour le certificat émis ;
- notifier : pour transmettre aux acteurs de la session les OTP générés ainsi que les notifications d'appel à signature, de disponibilité de documents signés, etc.

Par construction, l'API-C fonctionne de manière stateless, c'est-à-dire qu'aucun contexte n'est gardé par ce module logiciel : l'intégralité du contexte est gardée dans la base de données qui garantit son intégrité au travers de transactions.

Sur cette base, l'ensemble des opérations réalisées par l'API-C ne nécessitent ni compétences cryptographiques ni moyens intrinsèques de communication, car ces fonctions sont réalisées par les API dites « low level » qui sont appelées par l'API-C.

Par ailleurs l'API-C étant sensée fonctionner dans un environnement sécurisé, elle publie l'ensemble de ses API en HTTP et non pas en HTTP/S[2].

[2] on rappelle que quelque soit la forme d'authentification, l'API-C dispose du compte du client connecté au travers les headers spécifques CertignaUser et CertignaHash ce qui permet d'accéder à l'API de générer ou utiliser des certificats liés au compte.

3. Organiser le workflow de signature

3.1. Principes

L'organisation du workflow est basée sur un système utilisant des sessions. Les principes définis dans ce document régissant l'organisation des workflows de l'API-NG complètent, parfois en les modifiant, ceux définis dans le document de référence [CERTIGNA_NG].

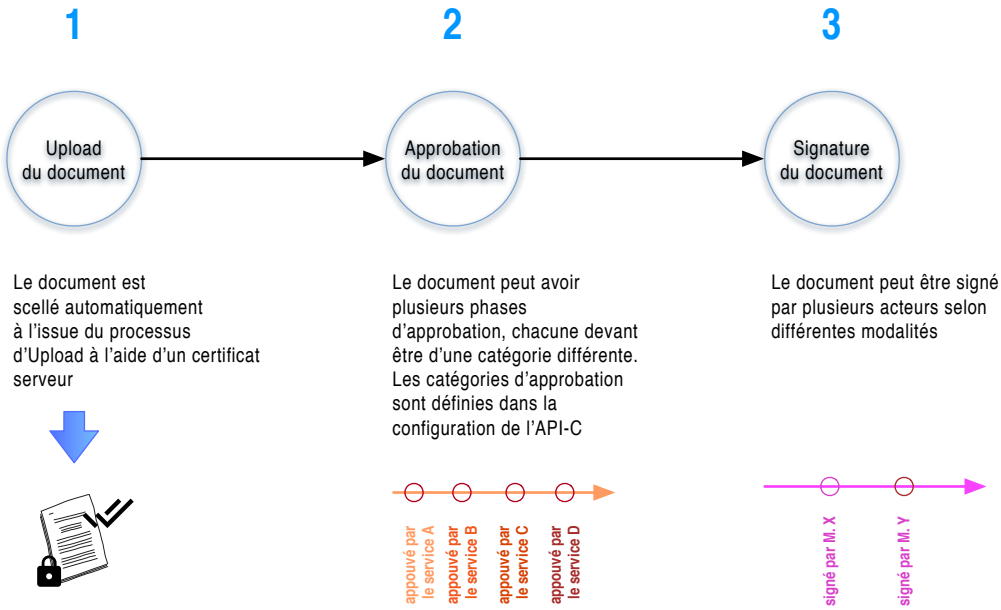
Un usager voulant organiser un workflow d'approbation et de signature de documents doit donc tout d'abord créer une session (valable pour une durée fixée) puis y ajouter les documents concernés et définir les acteurs ainsi que leur rôle dans ce processus.

Il organise ensuite, à l'intérieur de cette session, le workflow souhaité grâce à un système de scénarios lui-même constitué d'étapes d'approbation et de signature. Il reste ensuite à l'usager de jouer le scénario défini.

L'ensemble de ces actions peut faire partie d'un enchaînement plus complexe : on peut par exemple ajouter des documents et des acteurs, définir puis jouer un scénario et une fois celui-ci terminé rajouter des documents ou des acteurs, redéfinir un nouveau scénario, le jouer, etc.

En pratique, la session peut être le cadre d'une série de scénarios plus ou moins complexes et qui évoluent au cours du temps en fonction de critères dont seul l'usager a le contrôle.

Cela étant dit, dans tous les cas et pour toutes les sessions, le cycle de vie d'un document signé comporte trois phases :



Le scellement des documents uploadés est réalisé par la signature d'un jeton d'horodatage : le document est haché pour en obtenir un condensé qui, associé à la date et l'heure courante et d'autres métadonnées (dont l'URL du document), définit un faisceau de preuves qui sert à placer dans le temps l'existence et l'intégrité dudit fichier. Le fichier constituant le jeton de présence est ensuite signé par un certificat unique indiqué dans la configuration de l'API-C.

À noter qu'aucun document uploadé n'est chiffré par la plateforme.

D'un point de vue fonctionnel, et dans un contexte d'une utilisation réelle en entreprise, il faut s'attendre a avoir des sessions qui définissent, au travers leurs scénarios, plusieurs phases d'approbation correspondant à la validation des documents par différents services et une seule phase de signature relativement simple.

Dans tous les cas, la finalité de tous les documents de la session est d'être signés et ce sont les scénarios qui définissent l'organisation et le type de signatures attendues (voir section suivante).

Comme précisé dans le document de référence [CERTIGNA_NG], les documents peuvent être signés selon les formats PAdES, XAdES et CAdES dans tous leurs niveaux (B, T, LT et LTA) et selon tous les types possibles (signature détachée, enveloppée ou enveloppante).

Il est également précisé qu'il doit être possible de co-signer et de contresigner un ensemble de documents ou de faire de la signature individuelle.

Les deux tableaux suivants présentent la compatibilité intrinsèque de l'ensemble de ces possibilités :

Types \ Formats	PAdES	XAdES	CAdES
Signature détachée	**NON**	**OUI**	**OUI**
Signature enveloppée	**OUI**	**OUI**	**NON**
Signature enveloppante	**NON**	**OUI**	**OUI**

Processus \ Formats	PAdES	XAdES	CAdES
Co-signature	**OUI** (rétro-compatibilité)	**OUI** (sign. non enveloppée)	**OUI**
Contre-signature	**OUI**	**OUI** (signature enveloppante)	**OUI** (signature enveloppante)
Signature individuelle	**OUI**	**OUI**	**OUI**

Remarque 1 : la co-signature est toujours atteignable par rétrocompatibilité avec la contre-signature : on co-signe en contresignant sans vraiment tenir compte de l'ordre des signataires.

Remarque 2 : en théorie, il serait possible de faire de la co-signature et de la contre-signature au format XAdES avec la première signature enveloppée et les suivantes enveloppantes. Cette possibilité n'est pas retenue dans l'API-NG.

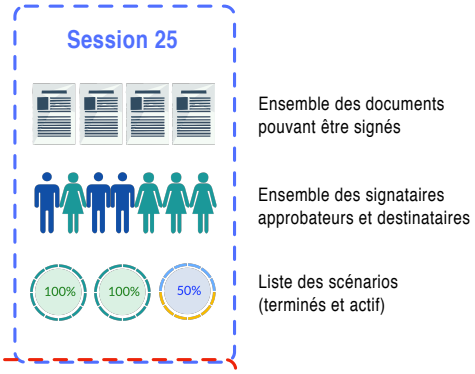
Ces compatibilités entre les formats, les types et les processus de signatures nous imposent une règle forte lors de la signature de plusieurs documents en même temps : **tous ces documents doivent être signés dans le même format**.

Impossible par exemple de proposer à la signature simultanée un PDF en PAdES et un XML en XAdES. Dans ce cas précis, seul le format XAdES pourra être utilisé.

3.2. Scénarios et étapes de scénarios

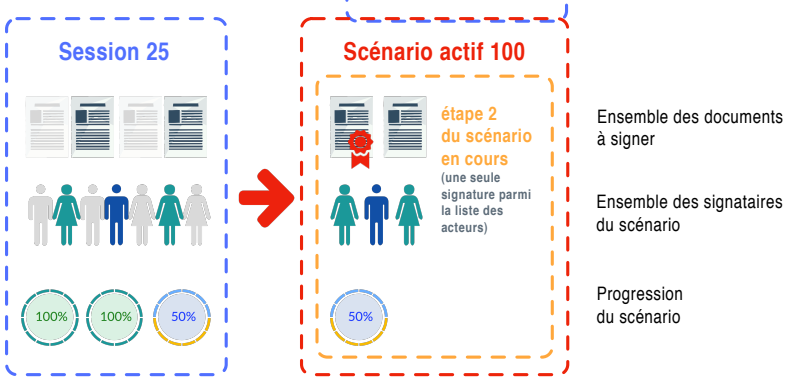
Une session comprend une liste de documents à approuver ou signer et d'acteurs pouvant approuver ou signer lesdits documents.

Une session est organisée en scénarios dont un seul (le dernier) peut être actif.



Un scénario est défini par :

- un ensemble de documents à approuver ou à signer choisis parmi ceux de la session,
- le format de signature des documents
- la définition d'étapes d'approbation ou de signatures par des acteurs de la session ayant le droit d'approuver ou de signer les documents sélectionnés.



Une étape de scénario consiste :

- en l'application d'un processus d'approbation ou de signature aux documents du scénario ;
- par une liste d'acteurs également choisis parmi ceux de la session.

Nous avons vu jusque-là trois processus de signatures. L'API-NG devrait offrir une liste plus complète de processus applicables à l'ensemble des documents d'un scénario et permettant d'organiser des workflows plus complexes :

- une approbation de catégorie C[3] par M acteurs parmi N ($1 \leq M \leq N$) ;
- la co-signature par M acteurs parmi N ($1 \leq M \leq N$) ;
- la contre-signature de l'ensemble des N acteurs ordonnés ;
- la co-signature de l'ensemble des N acteurs ordonnés[4] ;
- la signature individuelle par M acteurs parmi N ($1 \leq M \leq N$).

À noter que le fait de faire resigner un document déjà signé dans une étape ou un scénario précédent correspond toujours à une contre-signature.

Ainsi défini, un scénario est une simple succession d'étapes avec pour chacune d'entre elles un processus d'approbation ou de signature potentiellement différent.

Il convient néanmoins d'aller plus loin dans la logique si l'on veut pouvoir modéliser des processus complexes comme la délégation de signature.

---

[3] la liste des catégories d'approbation est définie dans le fichier de configuration de l'API-C

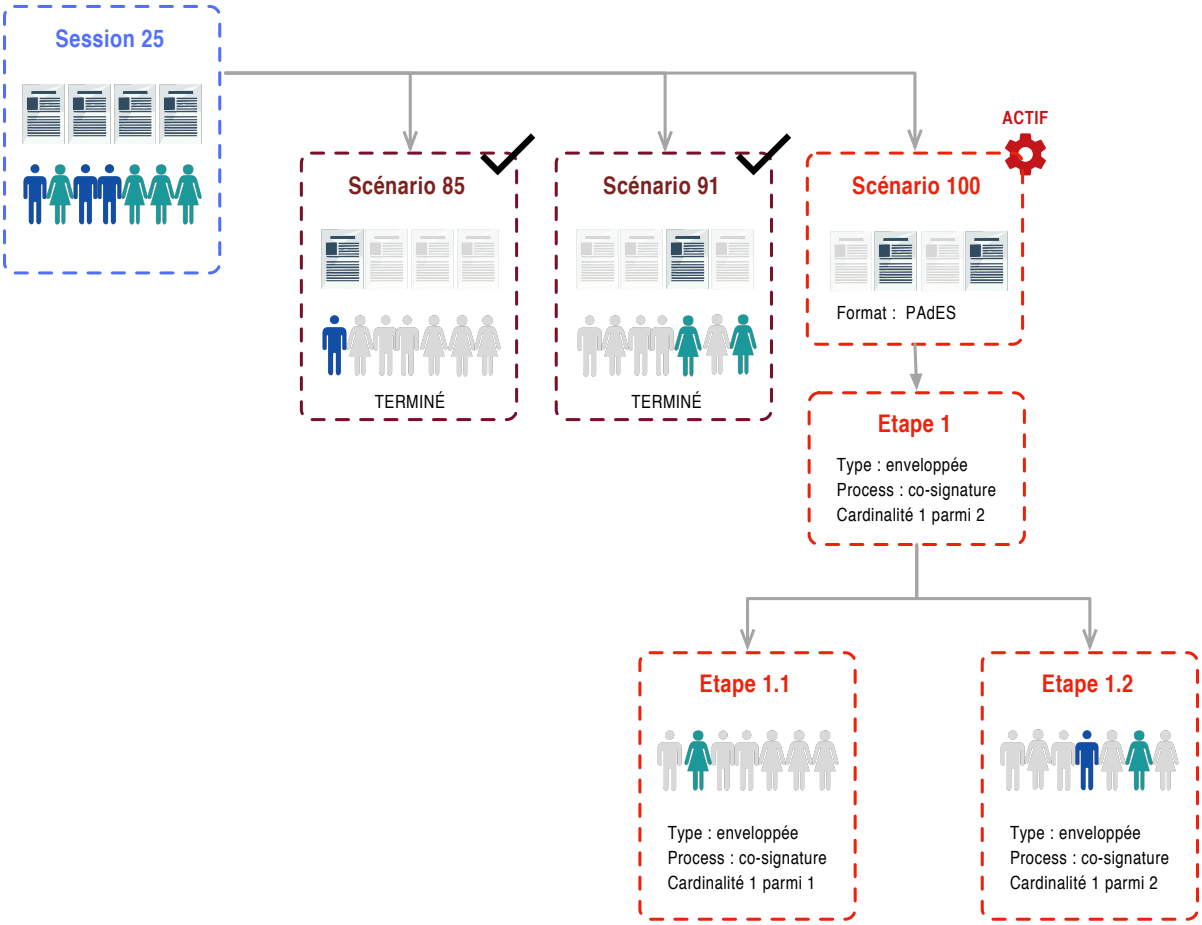
[4] ce processus de signature est proposé uniquement pour des raisons de workflow : il s'agit en fait d'ordonner les signatures sans que la signature suivante soit une approbation des signatures précédentes.

Admettons que nous souhaitions qu'un ensemble de documents soient signés par Madame X ou, si elle est absente par Monsieur Y ou Madame Z. Nous avons là une étape qui se défini comme (co-)signature d'une personne parmi deux ensembles de personnes qui sont {Madame X} et {Monsieur Y, Madame Z}.

Rien ne nous permet, dans la définition linéaire et successive des étapes de scénarios que nous avons vus jusqu'ici de modéliser ce scénario qui demande une définition arborescente des étapes.

Dans cette logique, le scénario porte l'ensemble des documents ainsi que le format de leur signature et les étapes définissent les acteurs et le processus de leur signature.

Si nous modélisons sous cette forme l'exemple de la signature de deux documents avec délégation dans le troisième scénario actif de notre session en exemple, nous obtenons le schéma suivant :



Dans ce schéma, le scénario actif 100 est constitué de trois étapes de signature. L'étape 1 qui est l'unique étape de rang un du scénario détermine que l'on souhaite une co-signature d'une personne parmi 2. L'étape 1.1 correspond à la signature de Madame X et l'étape 1.2 à la signature de Monsieur Y ou de Madame Z.

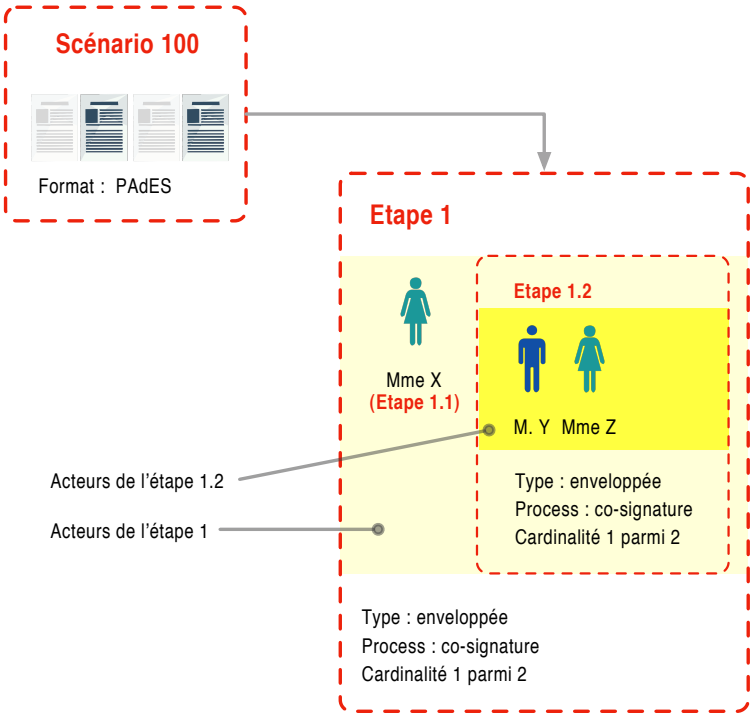
Une architecture arborescente comme celle-ci est articulée autour du fait que les étapes 1.1 et 1.2 sont l'équivalent d'acteurs de la signature demandée dans l'étape racine du scénario. C'est notamment le cas de l'étape 1.1 qui n'est effectivement que l'acteur Madame X devant signer et rien de plus.

D'un point de vue logique, cela signifie que la notion de scénario, d'étape et d'acteurs sont très imbriquées :

- un scénario définit l'ensemble des documents à signer et le format de leur signature ainsi qu'une liste d'étape de premier rang ;
- chacune des étapes comporte des acteurs qui sont soit directement des personnes signataires ou approbateurs soit des (sous-)étapes[5] organisant le scénario en une forêt d'arbres de signature.

La notion d'acteur des étapes de scénario est donc polymorphique puisqu'un acteur peut être un signataire (ou un approbateur) ou une sous-étape de scénario jouant le même rôle.

Dans les faits, la représentation du scénario 100 du schéma que nous venons de présenter devrait plutôt prendre la forme suivante dans laquelle l'étape 1 possède comme acteurs Madame X et la sous-étape 1.2 qui devient l'acteur d'une signature unique issue d'une cosignature de 1 parmi 2 acteurs, Monsieur Y et Madame Z :



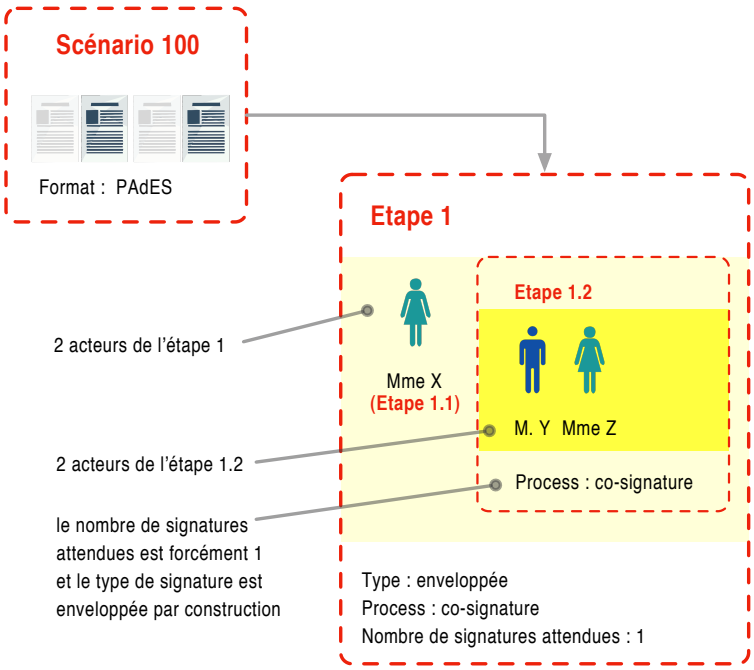
Soyons encore plus précis. La définition de la cardinalité des étapes dépend des acteurs impliqués et des étapes de rang supérieur. Dans notre exemple l'étape 1 demande une signature unique parmi les deux acteurs définis dans le tableau des acteurs et l'étape. Idem pour l'étape 1.2.

Inversement le fait que l'étape 1.2 délivre une et une seule signature dépend de ce qu'attend l'étape 1. Dans la même logique, le type de signature de l'étape 1.2 est forcément le même que celui de l'étape 1.

L'étape 1 est donc en partie définie par la liste de ses acteurs et l'étape 1.2 « hérite » de l'étape 1 le nombre de signatures à faire et le type de signature. Nous pouvons donc simplifier notre schéma d'exemple de la façon suivante :

En résumé :

[5] nous la nommons sous-étape pour la différencier de l'étape racine qu'est le scénario



- un scénario comprend un ensemble de documents à signer, le format de la signature desdits documents ainsi que la liste des étapes à mener pour les approuver et les signer ;
- chaque étape de la liste des étapes d'un scénario précise le type de signature, le processus de signature ainsi que le nombre de signatures attendues et la liste des acteurs impliqués dans ce process ;
- les acteurs d'une étape peuvent être soit des signataires (ou des approbateurs) soit des étapes dont le rôle est de produire le même nombre et la même forme de signature que si elles étaient de véritables acteurs.

Remarque 1 : dans la première version de l'API-NG les acteurs des étapes seront exclusivement des approbateurs ou des signataires (il n'y aura pas de sous-étapes).

Remarque 2 : comme nous allons le voir dans la section suivante, les approbateurs et signataires peuvent également être des destinataires des documents signés et ces rôles sont définis lors de l'ajout dans la session.

3.3. Rôles des acteurs

Les acteurs des sessions utilisés par les étapes des scénarios peuvent y jouer plusieurs rôles.

S'ils veulent pouvoir signer des documents, il faut qu'ils aient un rôle de signataire `sign` qui recouvre toutes les formes de signataires ou `cosign`, `countersign`, `ordered-cosign`, `individual-sign`, qui représente chacun des types de signatures proposé par l'API.

Si l'on souhaite les rendre destinataires des documents signés, il faut qu'ils aient soit le de primo destinataire `to` ou `cc` c'est-à-dire celui à qui l'on envoie les documents en copie[6].

[6] la distinction entre les deux rôles n'existe pas dans la première version de l'API. Ils sont tous deux des destinataires.

S'ils veulent pouvoir approuver des documents, il faut que les acteurs possèdent le rôle approval qui les autorise à approuver tous les documents lors de toutes les catégories d'approbation ou l'un des rôles d'approbation définis à travers les catégories d'approbation dans le fichier de configuration de l'API-C à l'aide de la variable `document-approval-categories`.

Cette variable est un dictionnaire dont les clés sont les rôles d'approbation et les valeurs des dictionnaires contenant le nom des catégories dans une ou plusieurs langues.

Les tags doivent être des chaînes de caractères en ASCII pur, ne comportant que des lettres, des chiffres ou les caractères tiret ("-") ou souligné ("_"). Les tags commencent forcément par une lettre et tous les noms de ressource de l'API sont interdits (actor, actors, document, documents, manifest, session, sessions…).

Exemple de définition JSON du dictionnaire de configuration des rôles :

```
{
  ...
  "document-approval-categories": {
    "legal": {
      "fr": "Approbation par le service juridique",
      "en": "Legal approval"
    },
    "comm": {
      "fr": "Approbation par le service commercial",
      "en": "Commercial approval"
    },
    "tech": {
      "fr": "Approbation technique",
      "en": "Technical approval"
    },
    "head": {
      "fr": "Approbation par la direction",
      "en": "Direction approval"
    }
  },
  "languages": ["fr", "en],
  ...
}
```

Les clés des langues pour les descriptions des catégories d'approbation sont celles utilisées pour définir le header HTTP `HTTP_ACCEPT_LANGUAGE` limité aux langues racines.

La priorité des langues utilisées est fixée à l'usage par ce header et à défaut par la variable de configuration de l'API-C `languages` qui définit l'usage des langues selon leur priorité. Si cette variable n'est pas définie, deux langues sont définies : en et fr (dans cet ordre).

Dans notre exemple quatre rôles d'approbation sont définis : legal, comm, tech et head.

Si la variable `document-approval-categories` est absente, seul le rôle approval peut être utilisé.

Du point de vue de l'API-NG, les rôles des acteurs sont donc des tags accolés aux acteurs lors de leur ajout dans les sessions.

L'usage de ces tags est également présent dans la définition des processus des étapes. Ainsi, on définira une étape d'approbation juridique en qualifiant le process en question par le tag legal.

Le tableau suivant récapitule l'usage des tags dans la définition des différents processus de signature :

Processus de signature	Qualification du processus	Rôles que doivent posséder les acteurs
Approbation	approval ou un tag défini dans la configuration (head, legal ...)	approval ou le tag défini dans la configuration (head, legal ...)
Co-signature M parmi N	cosign	sign ou cosign
Contre-signature M parmi N	countersign	sign ou countersign
Co-signature ordonnée	ordered-cosign	sign ou ordered-cosign
Signature individuelle (un fichier signé par individu)	individual-sign	sign ou individual-sign

En résumé :

- un acteur possède un ensemble de rôles définis par des tags système (sign, cosign, countersign, ordered-cosign, individual-sign, to, cc ou approval) ou ceux issus du paramétrage des catégories d'approbation (dans l'exemple legal, comm, tech et head) ;
- la définition des étapes d'approbation des documents utilise soit le tag système approval si aucune catégorie d'approbation n'est définie soit le tag d'une des catégories définies par le paramétrage de l'API-C et il faut que les usagers impliqués dans ces étapes possèdent soit le tag utilisé pour la définition soit le tag approval qui correspond à une autorisation globale d'approbation ;
- la définition des étapes de signature utilise les tags systèmes cosign, countersign, ordered-cosign ou individual-sign suivant le processus de signature choisi et les usagers impliqués doivent posséder le tag sign ou le tag spécifique du processus pour effectuer les signatures demandées ;
- pour être destinataires des documents signés lors d'une session, les usagers concernés doivent disposer du tag to ou cc ;
- il est bien évidemment possible d'avoir des usagers uniquement approbateurs, signataires ou destinataires. Ceux qui sont uniquement destinataires ne seront pas impliqués dans les processus d'approbation ou de signature.

3.4. Non-unicité de la définition des workflows

Même si la possibilité n'est pas ouverte dans la première version de l'API-NG, avoir permis, à terme, la définition de scénarios par une forêt d'étapes[7] donne une grande flexibilité aux usagers pour pouvoir leur workflow.

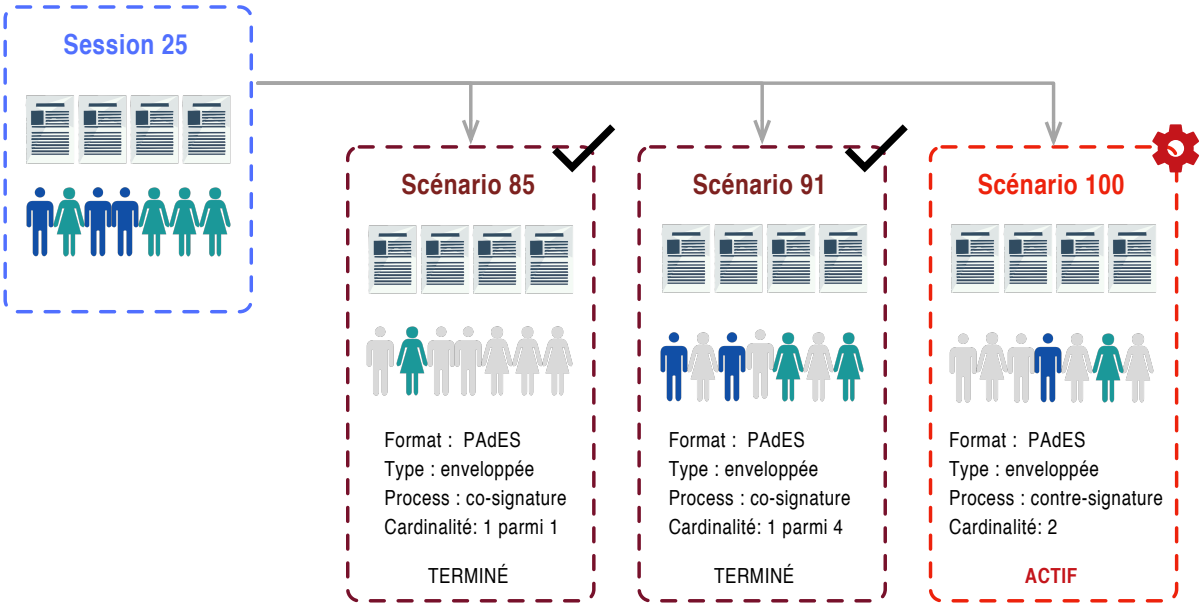
Cela implique aussi accepter que la définition d'un workflow puisse prendre plusieurs formes et qu'il n'existe pas de forme canonique pour opérer un processus complexe d'approbations et de signatures.

L'usager, suivant sa compréhension, l'évolution dans le temps de ses processus de signature (simples au début et plus complexes à la fin) pourra aboutir au même résultat avec des scénarios et des étapes différentes.

Prenons par exemple le cas de trois processus de signature successifs des mêmes documents (ce qui est donc équivalent à une contre-signature) : signature par Madame X puis signature par n'importe qui sauf Monsieur X et Madame Z et enfin signature par Monsieur Y contresignée par Madame Z.

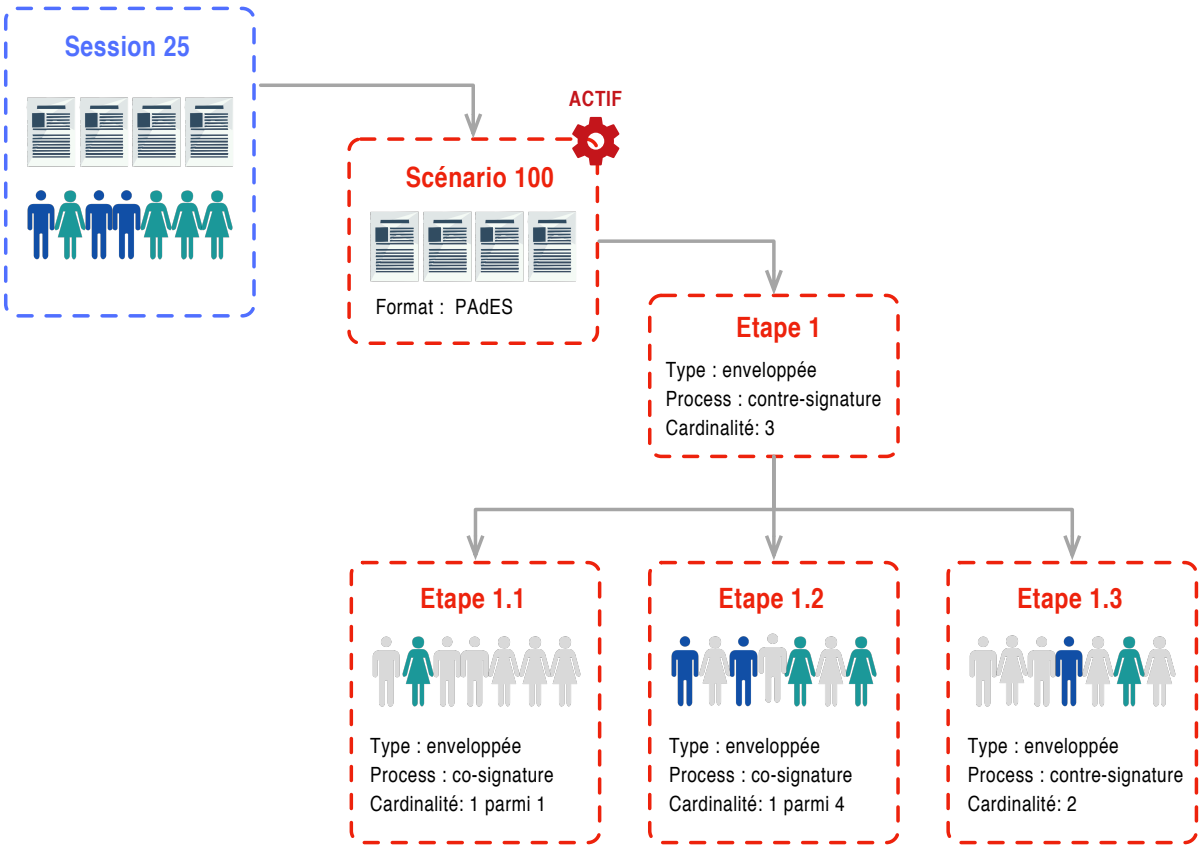
Ce processus peut se modéliser d'au moins trois façons différentes.

Premièrement, puisque toute nouvelle signature sur un document correspond à une contre-signature, en définissant trois scénarios successifs :

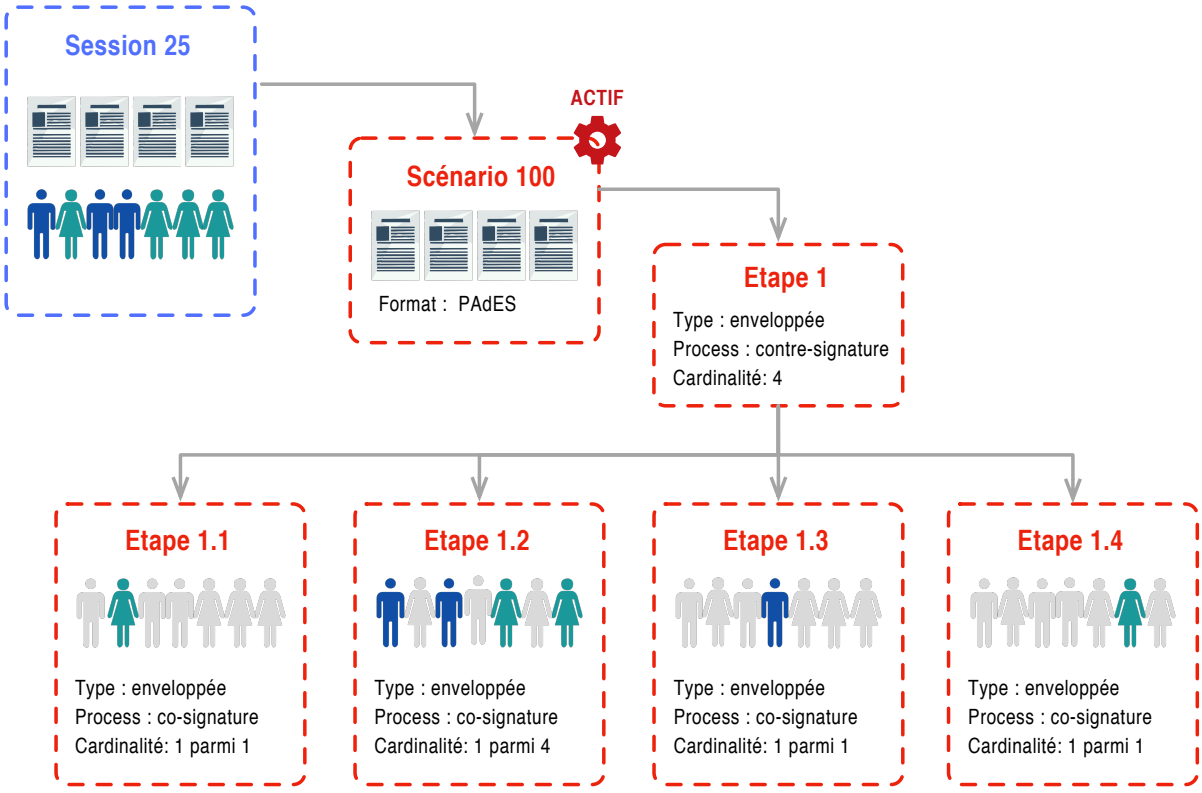


Ensuite en définissant un seul scénario et trois (sous) étapes :

[7] liste ordonnée d'étapes, chacune pouvant être définie par un arbre de sous-étapes.



Et aussi avec un seul scénario et quatre sous-étapes :



Tous ces scénarios parfaitement valides et équivalents dans le résultat qu'ils produisent.

Le premier d'entre eux est un processus incrémental qui se construit au fil du temps, les autres sont construits *ab initio* et l'évolution de leur contexte sera donc plus facilement maîtrisée par le système usager.

3.5. Règles générales de gestion

L'API-NG a été conçue pour offrir le plus de liberté possible aux usagers, y compris comme nous venons de le voir dans la forme qu'ils choisissent pour définir un workflow.

Offrir des possibilités n'impliquant pas absence de contrôle, l'API-NG est contrainte par un certain nombre de règles générales auxquelles les sessions et les scénarios doivent se conformer pour assurer la cohérence du contexte.

Elles sont listées dans le tableau suivant :

But	Règles
Fonctionnel	Aucune étape de signature n'est obligatoire dans un scénario, il est donc possible de définir des scénario ne comportant que des processus d'approbation. En revanche, tous les documents de la session ayant vocation a être signés, Il est obligatoire que la définition globale de l'ensemble des scénarios de la session puisse aboutir à la signature de tous les documents.
Fonctionnel	Aucune étape d'approbation n'est obligatoire, il est donc possible de définir des sessions qui ne comportent que des étapes de signature (on rappelle que les documents sont scellés lors de leur téléversement dans la session). Dans ce cas, la cohérence voudrait que les acteurs inscrits sur la cession ne portent aucun tag d'approbation.
Cohérence	Un document qui a été signé dans le cadre d'une session, ne peut plus être approuvé : l'ensemble des phases d'approbation d'un document doivent se dérouler avant toute phase de signature.
Intégrité	Un seul scénario à la fois est actif dans une session et c'est forcément le dernier de la liste.
Intégrité	Il n'est pas possible d'ajouter des documents, des acteurs ou des scénarios à une session lorsque celle-ci est active.
Cohérence	Quels que soient les scénarios et les étapes définies, un acteur ne peut aprouver un document qu'une fois par catégorie d'approbation dans une même session.
Cohérence	Quels que soient les scénarios et les étapes définies un même acteur ne peut signer un document qu'une seule fois.

4. Automates de gestion du contexte

Le rôle de l'API-NG est de gérer le contexte de signature de documents par des acteurs selon un workflow organisé selon un système de sessions comprenant des scénarios eux-mêmes composés d'étapes.

Pour ce faire, l'API-C, c'est-à-dire le cœur de l'API-NG, doit maintenir un contexte session par session qui s'assure :

- de la cohérence à chaque instant de la session et de ses scénarios ;
- de l'intégrité de la session, y compris pendant la phase active d'exécution d'un scénario ;
- du respect des règles de fonctionnement de l'API ;
- du respect de l'exécution des scénarios définis par l'usager.

De manière plus précise, la gestion du contexte global d'une session suit le déroulement de trois automates différents, mais interdépendants, chacun ayant son propre état.

Le premier est l'automate global de la gestion des sessions.

Le deuxième l'automate de gestion des scénarios lorsqu'ils sont dans une phase non active.

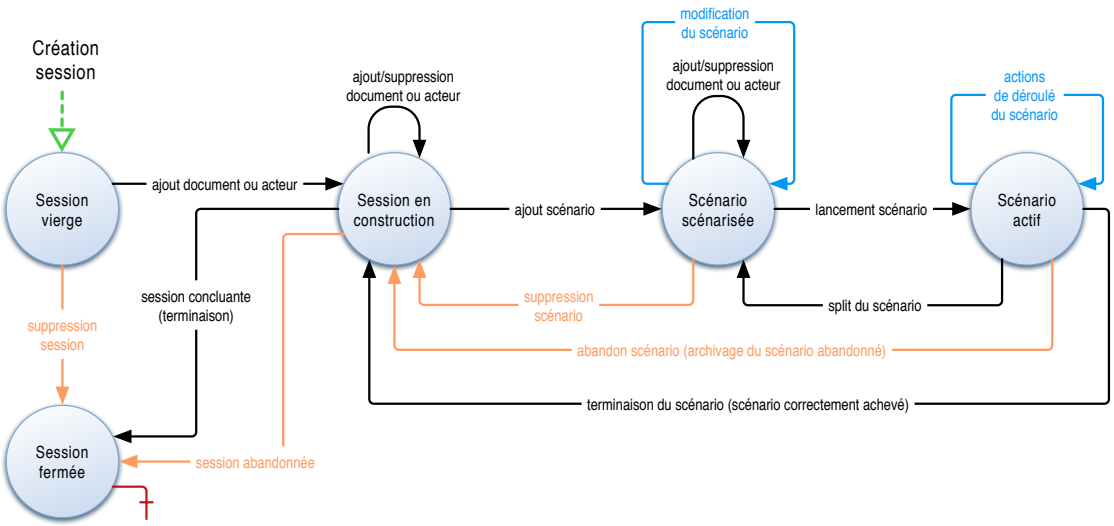
Le dernier l'automate est formé par les étapes d'exécution d'un scénario actif.

En pratique, ce dernier automate peut-être considéré comme une sous-partie non figée du précédent ; l'automate résultant de la définition des étapes d'un scénario étant une forêt d'étapes à parcourir jusqu'à épuisement des nœuds nécessaires à l'obtention des approbations ou signatures demandées.

Ainsi, de manière logique, une session, quelle que soit sa définition est équivalente à un automate à états (qui en regroupe trois) et le rôle de l'API-C sera de suivre le contexte de la session à travers les différents états possibles de l'automate en question.

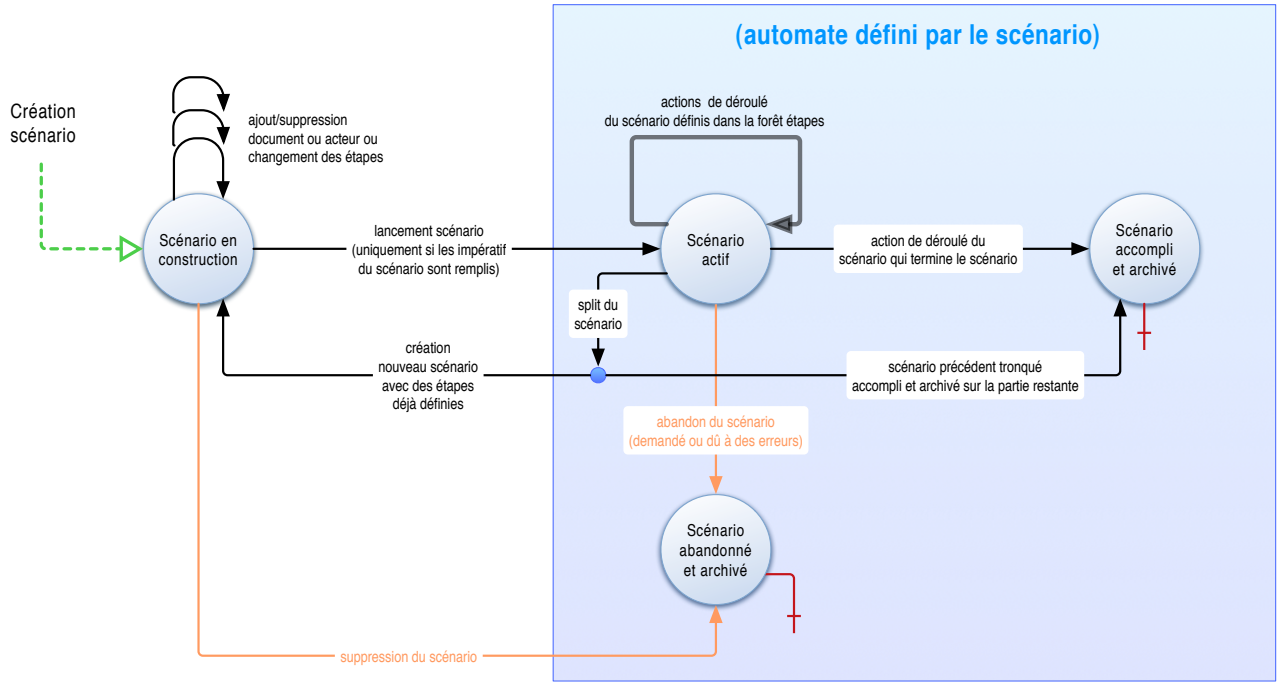
4.1. Automate de gestion des sessions

L'automate de gestion des sessions permet d'assurer la construction et l'évolution des sessions pendant ses phases d'inactivité et de constater qu'elle est « figée » lorsqu'elle a un scénario actif :



4.2. Automate de gestion des scénarios

L'automate de gestion des scénarios assure les parties représentées en bleu sur l'automate de gestion des sessions :



Le parcours du déroulé des actions est un parcours des arbres de la forêt définissant les étapes du workflow de l'utilisateur.

III. Ressources et fonctions de l'API-C

1. Présentation des fonctions

Les ressources et fonctions de l'API-C sont présentées ici dans leur version `1.0`.

Les appels qui sont décrits dans ce chapitre lui sont envoyés sont réalisé en HTTP (pas de HTTP/S pour l'API-C qui est considéré comme devant pouvoir fonctionner dans une zone sécurisée)

La liste des ressources et fonctions `disponibles` dans la version `1.2` de l'API-C est la suivante :

Verbe	URL	Description
POST	`/v{version}/sessions`	Création d'une nouvelle session
GET	`/v{version}/sessions?...`	Liste des identifiants des sessions accessibles à l'usager connecté (appel sans paramètres) ou requête paramétrique pour rechercher des sessions selon des critères.
GET	`/v{version}/session/{id}`	Accès à une session particulière
PUT	`/v{version}/session/{id}/extend`	Augmente la durée de vie d'une session
PUT	`/v{version}/session/{id}/close`	Fermeture de la session
GET	`/v{version}/session/{id}/manifest`	Récupération de l'URL du manifeste de preuve de la session
POST	`/v{version}/uploads`	Upload d'un nouveau document sur la plateforme
GET	`/v{version}/uploads?...`	Liste des documents uploadés sur la plateforme non encore utilisés (appel sans paramètres) ou requête paramétrique pour rechercher des uploads selon des critères
DELETE	`/v{version}/upload/{uid}`	Suppression d'un upload de la plateforme
GET	`/v{version}/uploads/accepted-extensions?type=`	Liste des extensions de fichiers acceptés pour l'upload
POST	`/v{version}/uploads/purge`	Supression de tous les uploads expirés
POST	`/v{version}/session/{id}/documents`	Ajout d'un document à la session
GET	`/v{version}/session/{id}/documents?...`	Liste des identifiants des documents de la session ou suivant requête, liste des documents à approuvé ou à signer dans l'étape en cours et selon le tag de processus spécifié
GET	`/v{version}/session/{id}/document/{did}`	Accès à un document particulier

GET	`/v{version}/session/{id}/document/{did}/current`
	`/v{version}/session/{id}/document/{did}/genuine`
	`/v{version}/session/{id}/document/{did}/all`
		Accès au lien de téléchargement d'un document dans sa version actuelle, orginal ou toutes les versions

DELETE	`/v{version}/session/{id}/document/{did}`	Suppression d'un document

Verbe	URL	Description
PUT	`/v{version}/session/{id}/generate-otp`	Génération et envoi d'un OTP pour un acteur et un ou plusieurs documents à approuver ou signer
PUT	`/v{version}/session/{id}/approve-documents`	Approbation d'un ensemble de documents par un acteur
PUT	`/v{version}/session/{id}/sign-documents`	Signature d'un ensemble de documents par un acteur
POST	`/v{version}/session/{id}/actors`	Ajout d'un acteur à la session
GET	`/v{version}/session/{id}/actors?...`	Liste des identifiants des acteurs de la session ou suivant requête la liste des acteurs impliqués dans l'étape de signature ou d'approbation courante, selon le tag de processus spécifié
GET	`/v{version}/session/{id}/actor/{aid}`	Accès à un acteur particulier
DELETE	`/v{version}/session/{id}/actor/{aid}`	Suppression d'un acteur
POST	`/v{version}/session/{id}/scenarios`	Ajout d'un scenario à la session
GET	`/v{version}/session/{id}/scenarios`	Liste des identifiants des scénarios de la session
GET	`/v{version}/session/{id}/scenario/{sid}`	Accès à un scenario particulier
PATCH	`/v{version}/session/{id}/scenario/{sid}`	Modifier un scénario
PUT	`/v{version}/session/{id}/scenario/{sid}/activate`	Activer un scénario
PUT	`/v{version}/session/{id}/scenario/{sid}/split`	Split d'un scénario
PUT	`/v{version}/session/{id}/scenario/{sid}/cancel`	Abandon d'un scénario
DELETE	`/v{version}/session/{id}/scenario/{sid}`	Suppression d'un scénario
GET	`/v{version}/cas`	Liste des AC émettrices disponibles
GET	`/v{version}/ca/{caid}`	Accéder aux informations d'une AC
GET	`/v{version}/ca/{caid}/cgu?session=...&author=...`	Récupération des CGU de l'AC émettrice
POST	`/v{version}/session/{id}/certificates`	Génération d'un certificat à la volée
GET	`/v{version}/session/{id}/certificates?...`	Liste de certificats générés de la session
GET	`/v{version}/session/{id}/certificate/{cid}`	Informations sur un certificat généré à la volée
DELETE	`/v{version}/session/{id}/certificate/{cid}`	Invalidation d'un certificat
POST	`/v{version}/downloads/purge`	Suppression de tous les documents à downloader qui ont expiré

Toutes les fonctions sont donc appelées selon le schema classique :

```
<VERB> <SERVEUR>/<PREFIX>/v{version}/<URL>[?<QUERY-STRING>]
```

- `<VERB>` peut prendre les valeurs `GET`, `POST`, `PUT`, `DELETE` et `PATCH`.
- `<SERVEUR>` est l'adresse du serveur, contenant éventuellement un numéro de port
- `<PREFIX>` est une racine d'url fixe dépendant de la configuration de l'API sur le serveur
- `{version}` est la version de l'API
- `<URL>` est la partie de l'URL pointant effectivement sur la fonction appelée
- `[?<QUERY-STRING>]` est la chaîne optionnelle permettant de passer des paramètres aux requêtes qui le nécessitent

Toutes les requêtes incluent donc au minimum :

Le paramètre de version commun à toutes les URL :

Type	Nom	Description	Schéma
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Les headers d'authentification communs à tous les URLs :

Nom	Description
`CertignaUser`	Identifiant de l'usager connecté
`CertignaHash`	Hash du mot de passe de l'usager connecté
`CertignaRole`	Role de l'usager connecté (nombre entier positif)
`DefaultLanguage`	Langue préférée de l'appelant (header HTTP standard)

Concernant maintenant les réponses renvoyées par l'API-C :

Si la fonction appelée de l'API-C fonctionne correctement, la réponse HTTP aux requêtes renvoie l'un des codes suivants :

Code HTTP	Statut	Description réponse
200	OK	Le contenu du retour dépend du type de fonction appelée
201	Created	Un body est renvoyé. Son contenu dépend du type de fonction appelée L'URL de la nouvelle entité est renvoyée dans le header **Location** La date de sa création est dans le header **Date** L'éventuelle date d'expiration est dans le header **Expires**

Si la fonction appelée de l'API-C ne fonctionne pas correctement, la réponse HTTP aux requêtes peut toujours renvoyer l'un des codes suivants :

Code HTTP	Statut	Description réponse
400	OK	Pas de centenu. La requête n'est pas conforme à ce qui est attendu, que cela soit dans ses paramètres ou l'éventuel body passé
403	Forbidden	Pas de contenu. La requête essaie d'accéder à des informations ou un fonction interdite dans le contexte d'usage et avec les crédentiels passés.
404	Not Found	La ressource demandée n'a pas été trouvée ou l'un des paramètres de la requête (query string ou body) pointe vers une ressource non trouvée
500	Internal Error	La fonction demandée a produit une erreur sur le serveur. Ces erreurs peuvent appartenir à quatre catégories : – erreur de fichier (lecture ou écriture) – erreur de base de données (lecture ou écriture) – erreur l'ors de l'appel des API cryptographiques de CERTIGNA – erreur interne générique (non catégorisée)
501	Not Implemented	La fonction appelée n'est pas implémentée

2. Ressources génériques renvoyées par les fonctions

URLResource :

Resource renvoyée par toutes les requêtes qui renvoient une URL (sauf cas particulier défini dans la description d'une fonction) :

Clé	Description	Schéma
url	Url renvoyé	String
date	Date de génération du download du manifest *(facultatif)*	String

ExpiringURLResource :

Resource renvoyée par toutes les requêtes qui renvoient une URL qui n'est valide que pendant un certain temps (sauf cas particulier défini dans la description d'une fonction) :

Clé	Description	Schéma
url	Url renvoyé	String
date	Date de génération du download du manifest *(facultatif)*	String
expires	Date d'expiration du download du manifest *(facultatif)*	String

DeletedURLResource :

Resource renvoyée par toutes les requêtes qui renvoient une URL (sauf cas particulier défini dans la description d'une fonction) :

Clé	Description	Schéma
deleted	Url de la ressource qui vient d'être détruite	String

PurgedResourcesCount :

Resource renvoyée par toutes les requêtes de purge de ressources expirées (uploads et downloads) :

Clé	Description	Schéma
deleted-count	Nombre de ressources supprimées	Entier positif

3. Ressource Session

3.1. Créer une session

```
POST /v(version}/sessions
```

Description

Permet de créer une session de signature.

Paramètres URL

Type	Nom	Description	Schéma
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
ttl	Time to Live en secondes *(requis)*	UInt32
user-data	Données propres au client	JSON Dictionary
manifest-data	Données liées à la constitution du manifeste du client (voir plus loin chapitre « Resource Manifest »)	JSON Dictionary contenant les clés définies dans le paramètre général `session-manifest-data`

Réponses

Code HTTP	Statut	Description réponse
201	Created	Contenu Application/JSON : `ExpiringURLResource`. L'URL de la nouvelle session est renvoyée dans le header **Location** La date de création est dans le header **Date** La date d'expiration dans le header **Expires**
403	Forbidden	Pas de contenu. L'utilisateur connecté n'a pas le droit de création.
409	Conflict	Pas de contenu. Erreur renvoyée si la ttl passée dans le body de la requête est inférieure à la variable de configuration `ttl-min` ou supérieure à la variable de configuration `ttl-max`.
422	Unprocessable	Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `session-manifest-data`.

Le code HTTP 200 n'est pas retourné par cette fonction.

3.2. Récupérer une liste d'identifiants de sessions

```
GET /v(version}/sessions?... parameters...
```

Description

Permet de récupérer une liste d'URL ou d'identifiants de sessions correspondant aux critères de recherche passés dans les post-variables de l'URL. La liste est renvoyée dans la variable `sessions`.

Si aucun critère n'est précisé, cette API renvoie l'ensemble des identifiants de session de l'usager appelant.

Paramètres URL

Type	Nom	Description	Schéma
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Variables post URL

Nom	Description	Schéma
ttlmin	durée de vie totale minimale(*)	entier positif ou nul
ttlmax	durée de vie totale maximale(*)	entier strictement positif
dynttlmin	durée de vie restante minimale de la session(*). Ce paramètre est prioritaire sur expirationstatus	entier positif ou nul
dynttlmax	durée de vie restante maximale de la session(*). Ce paramètre est prioritaire sur expirationstatus	entier. si le nombre est négatif, permet de chercher des sessions expirées depuis un certain temps
userids	liste d'identifiant d'usagers séparés par des virgules	liste de strings
expirationstatus	remplace dynttlmin et dynttlmax. Par défaut si ni dynttlmin ni dynttlmax n'est utilisé et que rien n'est spécifié, la requête renvoie les sessions valides.	all = toutes valid = les non expirées expired = les expirées
status_mask	masque de recherche du statut de la session. Exemple status_mask = 3096 retourne les sessions correctement terminées soit par clôture explicite soit par terminaison après expiration(**)	entier strictement positif

(*) la recherche de temps exacte se fait en positionnant la même valeur pour le min et le max
(**) status_mask est un bitmask, chaque bit représentant un statut

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body contenant un tableau JSON de la liste des identifiants des sessions trouvées correspondant au critères (le tableau peut être vide).
403	Forbidden	Pas de contenu

Exemple d'un body de réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "sessions": [
    "/session/1240",
    "/session/1247,
    "/session/1524",
    "/session/1620",
    "/session/1750",
    "/session/1900",
    "/session/2150",
    "/session/3025",
    "/session/9540",
   ]
}
```

3.3. Récupérer une session

```
GET /v{version}/session/{id}
```

Description

Permet de récupérer le contenu d'une session de signature.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body : `SessionResource`. **Last-Modified** header donne la date de dernière modification **Date** header donne la date de création de la session de signature **Expires** header donne la date de dépréciation de la session de signature
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

SessionResource :

Clé	Description	Schéma
date	Date de création de la session	String
expires	Date d'expiration de la session	String
id	Identifiant public de la session	Entier strictement positif
status	Statut de la session	`SessionStatus`
ttl	Time To Live de la session	Entier strictement positif
actors	Liste des URLS des acteurs de la session *(facultatif)*	Tableau de String
documents	Liste des URLS des documents de la session *(facultatif)*	Tableau de String
manifest-data	Données liées à la constitution du manifeste *(facultatif)*	JSON Dictionary
scenarios	Liste des URLS des scenarios de la session *(facultatif)*	Tableau de String
user-data	Données utilisateur	JSON Dictionary

SessionStatus :

Status	Description
1	Session vierge (vient d'être créée)
2	Session en construction
3	Session Idle : potentiellement terminée mais peut-être en construction après la terminaison correcte du scénario actif.
4	Session active (le dernier scénario de la session est actif)
10	Session correctement terminée
20	Session supprimée (abandonnée alors qu'elle était vierge)
21	Session abandonnée (fermeture forcée par exemple)

Exemple d'un body de réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Date: 2020-04-21T08:12:02.252Z
Last-Modified: 2020-04-21T14:03:02.720Z
Expires: 2020-04-22T08:13:00Z
...
{
  "id": 25,
  "status": 10,
  "actors": [
    "/session/25/actor/100",
    "/session/25/actor/20",
    "/session/25/actor/35"
  ],
  "documents": [
    "/session/25/document/1000",
    "/session/25/document/300",
    "/session/25/document/500"
  ],
  "scenarios": ["/session/25/scenario/1232"],
  "ttl": 86400,
  "user-data": {
    "label": "Signature des documents du pacte A452",
    "internal-id": "52DDF244-FC3E-40EE-BCC4-D5A75E686032"
  }
  "manifest-data": {
    "iid": "001C2F16-E4D8-45D6-BE03-4DA29399B1DC"
  }
}
```

3.4. Étendre la durée de vie d'une session

```
PUT /v(version}/session/(id}/extend
```

Description

Permet d'ajouter du temps à une session avant que celle-ci n'expire.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
ttl	Nouvelle durée de vie en secondes *(requis: remplace l'ancienne)*	UInt32

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `ExpiringURLResource`. L'URL de la session est renvoyée dans le header **Location** La date de création est dans le header **Date** La nouvelle date d'expiration dans le header **Expires**
403	Forbidden	Pas de contenu. La session était déjà fermée ou expirée.
404	Not Found	Pas de contenu
409	Conflict	Pas de contenu. Erreur renvoyée si `ttl ≤ ancienne ttl` ou si `ttl > ttl-max`.

3.5. Clore une session

```
PUT /v(version}/session/(id}/close
```

Description

Permet de fermer une session de signature. La fermeture peut être forcée ou non. Dans le cas où la fermeture est forcée, la session est fermée même si un scénario est actif.

Dans le cas où la fermeture n'est pas forcée ou si la fermeture forcée est demandée alors que la variable de configuration `accept-forced-closure` est positionnée à `false`, la méthode ne permet pas de fermer une session avec un scénario actif et une erreur HTTP 403 est renvoyée. Cette erreur est également renvoyée si la session est déjà fermée.

Une fois fermée une session est de facto considérée comme une ressource read-only et tout appel d'une fonction de modification de la session et de ces sous-objets provoque une erreur.

Suivant la valeur de la variable `manifest-on-closure`, la fermeture de session peut systématiquement provoquer ou non la génération du manifeste de preuves qui devient partie intégrante de la session (voir section 2 pour une description du manifeste et de son usage).

Si le manifeste a été généré, le code HTTP 201 est renvoyé. Si le manifeste devait être généré et qu'il ne l'est pas, la fonction effectue néanmoins la clôture de la session, ne renvoie pas d'erreur, mais un code HTTP 200 est renvoyé au lieu du code HTTP 201 si le manifeste avait été généré. Si la génération du manifest n'était pas prévue, le code HTTP 200 est renvoyé.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
force	Force la clôture même si un scénario est actif *(requis)*	Boolean
reason	Raison en clair de la clôture de session *(requis)*	String
manifest-data	Données liées à la constitution du manifeste du client (voir plus loin chapitre « Resource Manifest »)	JSON Dictionary contenant les clés définies dans le paramètre général `closure-manifest-data`

Réponses spécifiques :

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `ClosingSessionResource`
201	Created	Contenu Application/JSON : `ClosingSessionResource` L'URL du manifeste créé est de plus renvoyée dans le header **Location**. Sa date de création dans le header **Date**. Sa date d'expiration dans le header **Expires**.
403	Forbidden	Pas de contenu. Erreur renvoyée s'il est impossible de fermer la session dans le contexte où elle se trouve.
404	Not Found	Pas de contenu. Session non trouvée
422	Unprocessable	Pas de contenu. Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `closure-manifest-data`.
504	Gateway Timeout	S'il a été demandé, l'éventuelle requête de scellement du manifeste de preuve anvoyée au système interne de CERTIGNA a pris trop de temps

ClosingSessionResource :

Clé	Description	Schéma
status	Nouveau status de la session	Entier
url	Url du download du manifest *(facultatif)*	String
date	Date de génération du download du manifest *(facultatif)*	String
expires	Date d'expiration du download du manifest *(facultatif)*	String

4. Ressource Manifest

La ressource Manifest ou manifeste de preuve d'une session de signature est une ressource très particulière. En effet, il ne s'agit pas d'une ressource de gestion, mais d'un document PDF produit à la demande ou lors de la fermeture d'une session des signature.

Par définition, cette ressource est read-only et elle est de plus signée par un certificat serveur lors de sa génération (définit dans la configuration de l'API-C par la variable `manifest-certificate`)[8].

4.1. Constitution du manifeste de preuve

Le manifeste de preuve est constitué d'éléments constituant les sessions, leurs scénarios, leurs acteurs et leurs documents ainsi que d'éléments dynamiques décrivant le déroulement des phases d'approbation, de signature et de clôture des documents.

L'utilisateur peut ajouter des données à insérer à l'intérieur du manifeste par l'intermédiaire de dictionnaires JSON qu'il peut ajouter à l'aide de la clé `manifest-data` dans certaines requêtes.

Le contenu du dictionnaire JSON introduit par la clé `manifest-data` dépend de variables de configurations globales de l'API-C qui sont définies dans son fichier JSON de configuration.

Pour chacune des requêtes acceptant la clé `manifest-data`, il existe une variable de configuration déterminant les clés que peut contenir le dictionnaire envoyé.

Le tableau suivant liste les requêtes où la clé `manifest-data` peut être ajoutée avec en regard la variable de configuration déterminant l'ensemble des clés possibles du dictionnaire JSON transmis par le biais de cette clé :

Verbe	URL de la requête	Variable globale de configuration
POST	`/v(version}/sessions`	`session-manifest-data`
PUT	`/v(version}/session/(id}/close`	`closure-manifest-data`
POST	`/v(version}/session/(id}/documents`	`document-manifest-data`
PUT	`/v(version}/session/(id}/approve-documents`	`approve-manifest-data`
PUT	`/v(version}/session/(id}/sign-documents`	`signature-manifest-data`
POST	`/v(version}/session/(id}/actors`	`actor-manifest-data`
POST	`/v(version}/session/(id}/scenarios`	`scenario-manifest-data`
PUT	`/v(version}/session/(id}/scenario/(sid}/activate`	`activate-manifest-data`
PUT	`/v(version}/session/(id}/scenario/(sid}/cancel`	`cancel-manifest-data`

Toutes les variables de configuration `*-manifest-data` contiennent un dictionnaire constitué des clés acceptées et pour chacune d'entre-elle, le label de l'information dans les langues acceptées de l'API-C.

[8] dans une version suivante de l'API, il sera possible d'utiliser un cachet serveur lié au compte de la personne connectée.

Exemple de définition JSON de la variable `session-manifest-data` :

```
{
  ...
  "session-manifest-data": {
   "origin": { "fr": "Service demandeur", "en": "Origin office" },
    "filing-code": { "fr": "Code d'archivage", "en": "Filing code" },
  },
 ...
}
```

Ainsi définie la variable de configuration `document-manifest-data` permet l'insertion d'une section `manifest-data` contenant au plus trois clés dans l'ajout de documents.

Exemple :

```
POST /v1.5/sessions
...
{
  ...
  // contenu normal pour la création d'une nouvelle session
  ...
  "manifest-data": {
    "origin": "Service juridique",
    "filing-code": "A4513-12",
  },
 ...
}
```

L'ensemble des données des dictionnaires `manifest-data` doivent être des chaînes de caractères qui seront ajoutées en regard du label ad hoc dans le manifeste de preuves.

4.2. Récupération de l'URL du manifeste

```
GET /v(version}/session/(id}/manifest
```

Description

Cette fonction permet de récupérer l'URL publique du document PDF signé comprenant le manifeste de preuves de la session. Ce document peut-être automatiquement généré et signé lors de la clôture de la session et être disponible dès que celle-ci est fermée.

Dans le cas où il n'a pas été généré à la clôture de la session, l'appel de cette fonction provoque la création et la signature du manifeste avant que de renvoyer son URL. La signature du manifest est réalisée par un cachet serveur (certificat paramétré dans le fichier de configuration).

Utiliser cette fonction sur une session ouverte provoque le renvoi d'une erreur HTTP 403.

Pour récupérer le manifeste, soit les codes HTTP 201 ou 302 sont renvoyés et le programmeur n'a ensuite qu'à télécharger le contenu de l'URL pointée par le header **Location**.

Il est recommandé de lancer le téléchargement immédiatement après la réception de l'URL dont la durée de vie peut être courte.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
201	Created	Contenu Application/JSON : `ExpiringURLResource`. L'URL du download du manifeste est renvoyée dans le header **Location** La date de création est dans le header **Date** La date d'expiration dans le header **Expires**
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
504	Gateway Timeout	La requête de scellement du manifeste de preuve anvoyée au système interne de CERTIGNA a pris trop de temps

Le code HTTP 200 n'est pas retourné par cette fonction.

5. Ressource Upload

Un upload est un fichier que l'on ajoute à la plateforme destiné à devenir un document rattachée à une session dans le but d'être signé soit ou utilisé pour un autre usage comme la transmission de justificatifs lors de la création de certificats à la volée.

Attention : un upload n'est pas rattaché à une session. Ce sont les sessions qui utillisent ensuite les uploads pour disposer des fichiers uploadés.

C'est la raison pour laquelle tous les HTT sont anonymes (les fichiers ne sont pas nommés) et ont une durée de vie limitée. Il est également prévu que le nombre d'upload et la taille des fichiers soient limités (voir la définition des variables globales `upload-size-max`, `session-upload-size-max`, `session-upload-number-max`, `client-upload-size-max`, `client-upload-number-max` et `upload-bandwidth-max`)[9].

Dans tous les cas, l'usage d'un fichier transmis à la plateforme nécessite deux phases.

Il faut tout d'abord uploader le document en question sur la plateforme qui héberge l'API. Cela se fait via la requête `POST /uploads`.

Cette requête d'upload enregistre le document, l'horodate et le scelle (signature détachée) à l'aide de son certificat dédié et renvoie une URL temporaire d'upload valable un temps fixé par la variable globale `upload-ttl`.

Il est ensuite possible d'utiliser cette URL temporaire soit pour ajouter un document à une session pour signature soit pour un autre usage.

Lorsque le fichier uploadé est utilisé par une autre API, le fichier choisi est retiré de la liste des uploads et l'URL temporaire n'est bien évidemment plus valable.

[9] toutes ces variables encadrent l'upload de fichiers de manière statique. Dans une version ultérieure de l'API, le système d'authentification en amont de l'API-C pourrait transmettre un identifiant de session permettant de demander au système d'authentification bas niveau l'ensemble des informations liés au client connecté et nottamment ses ratios de téléchargement, la place qui lui est allouée sur disque, etc.

5.1. Uploader un document

```
POST /v{version}/uploads
```

Description

Permet d'uploader un document sur la plateforme et de le sceller pour usage ultérieur.

Paramètres URL

Type	Nom	Description	Schéma
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Le contenu binaire du document.

Réponses

Code HTTP	Statut	Description réponse
201	Created	Contenu Application/JSON : `ExpiringURLResource`. L'URL du nouvel upload est renvoyée dans le header **Location** La date de création est dans le header **Date** La date d'expiration dans le header **Expires**
403	Forbidden	Pas de contenu
409	Conflict	Mauvais mime type, fichier trop grand, inexistant, quota d'upload dépassé en quantité ou en bande passante...
504	Gateway Timeout	La requête de scellement du token décrivant le fichiers envoyée vers le système interne de CERTIGNA a pris trop de temps

Le code HTTP 200 n'est pas retourné par cette fonction. Toute erreur de signature ou d'horodatage du document (ce qui ne devrait pas se produire) renvoie une erreur 500 Internal Server Error.

5.2. Récupérer la liste des uploads

```
GET /v(version}/uploads? ... parameters
```

Description

Sans paramètres, permet de récupérer la liste des URL temporaires de tous les documents uploadés sur la plateforme et non encore utilisés (non ajoutés à une session).

En ajoutant des paramètres dans les variables post URL, on peut lister l'ensemble des uploads expirés ou en passe de l'être ou encore les uploads fait par un usager ou un ensemble d'usagers.

Paramètres URL

Type	Nom	Description	Schéma
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Variables post URL

Nom	Description	Schéma
ttlmin	durée de vie totale minimale du fichier uploadé(*)	entier positif ou nul
ttlmax	durée de vie totale maximale du fichier uploadé(*)	entier strictement positif
dynttlmin	durée de vie restante minimale du fichier uploadé(*). Ce paramètre est prioritaire sur expirationstatus	entier positif ou nul
dynttlmax	durée de vie restante maximale du fichier uploadé(*). Ce paramètre est prioritaire sur expirationstatus	entier. si le nombre est négatif, permet de chercher des uploads expirés depuis un certain temps
userids	liste d'identifiant d'usagers séparés par des virgules	liste de strings
expirationstatus	remplace dynttlmin et dynttlmax. Par défaut si ni dynttlmin ni dynttlmax n'est utilisé et que rien n'est spécifié, la requête renvoie les uploads valides.	all = tous valid = les non expirés expired = les expirés

(*) la recherche de temps exacte se fait en positionnant la même valeur pour le min et le max

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body contenant un tableau JSON de la liste des URL des uploads.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

Exemple d'un body de réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "uploads": [
    "/uploads/3712",
    "/uploads/3952",
    "/uploads/4001"
  ]
}
```

5.3. Renvoyer la liste des extensions des fichiers uploadables

```
GET /v(version}/uploads/accepted-extensions?type=…
```

Description

Renvoie la liste des extensions de fichiers acceptés par le système d'upload de l'API-C.

Paramètres URL

Type	Nom	Description	Schéma
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Variables post URL

Nom	Description	Schéma
type	indique si l'on souhaite récupérer tous les types de document uploadables ("all") ou seulement les documents uploadables pour signature ("signing")	string ("all" ou "signing")

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : Renvoie un dictionnaire avec comme clés les extensions de fichier autorisées (en minuscules) et en valeur correspondante le type-mime à utiliser pour cette extension
403	Forbidden	Pas de contenu

Exemple d'un body de réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "accepted-extensions": {
    "pdf": "application/pdf",
    "xml": "application/xml",
    "jpeg": "image/jpeg",
    "jpg": "image/jpg",
    "png": "image/png"

  }
}
```

5.4. Purger les uploads expirés

```
POST /v{version}/uploads/purge
```

Description

Supprime un upload de la plateforme.

Paramètres URL

Type	Nom	Description	Schéma
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `PurgedResourcesCount`.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

5.5. Supprimer un upload

```
DELETE /v{version}/upload/{uid}
```

Description

Supprime un upload de la plateforme.

Paramètres URL

Type	Nom	Description	Schéma
Path	uid *(requis)*	Identifiant de l'upload	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `DeletedURLResource`.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

6. Ressource Document

Un document est un fichier que l'on ajoute à une session dans le but d'être potentiellement validé puis *in fine* signé.

Dans tous les cas, l'usage d'un document nécessite deux phases : il faut d'abord uploader le document en question sur la plateforme qui héberge l'API via la requête `POST /uploads` puis utiliser l'URL temporaire produite par cette requête pour ajouter le document à la session à l'aide de la requête `POST /session/{id}/documents`.

À l'issue de l'ajout à la session, le document choisi est retiré de la liste des uploads et l'URL temporaire n'est bien évidemment plus valable.

6.1. Ajouter un document à une session

```
POST /v{version}/session/{id}/documents
```

Description

Permet d'ajouter un document à une session de signature. L'ajout d'un document n'est possible que s'il n'y a pas de scénario actif. En cas de scénario actif, une erreur HTTP 403 est renvoyée.

L'ajout du document de la session provoque automatiquement le retrait de ce dernier de la liste des uploads.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
abstract	résumé de présentation du document	String
file-name	nom du fichier avec extension (dernier composant du path) *(requis)*	String
title	titre en clair du document *(requis)*	String
upload	URL de l'upload du document *(requis)*	String

Clé	Description	Schéma
user-data	Données propres au client	JSON Dictionary
manifest-data	Données liées à la constitution du manifeste du client (cf. chapitre « Resource Manifest »)	JSON Dictionary contenant les clés définies dans le paramètre général `document-manifest-data`

Réponses

Code HTTP	Statut	Description réponse
201	Created	Contenu Application/JSON : `URLResource`. L'URL du document est renvoyée dans le header **Location** La date de création est dans le header **Date**
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu (Upload non trouvé)
409	Conflict	passed 'file-name' is incompatible with given upload
422	Unprocessable	Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `document-manifest-data`.

Le renvoi d'un code HTTP 200 est considéré comme une erreur sur cette requête qui n'est valide que si un code HTTP 201 Created a été renvoyé et le header **Location** positionné.

6.2. Récupérer une liste d'identifiants des documents d'une session

```
GET /v(version}/session/(id}/documents? ... parameters
```

Description

Sans paramètres, permet de récupérer la liste des URL de tous les documents d'une session dans la variable `documents`.

En utilisant le paramètre `status_mask` dans les variables post URL, on peut filtrer les documents selon leur avancée dans le processus de signature. L'usage de `status_mask` est antagoniste avec l'usage des paramètres tags ou actor, l'usage de ceux-vi visant à récupérer exclusivement des documents à approuver ou à signer. Contrevenir à cette règle provoque le retour d'erreur `422 Unprocessable`.

En ajoutant les paramètres `tags` ou `actor` dans les variables post URL, on peut lister l'ensemble des documents actuellement à approuver ou à signer dans la session. Les documents sont alors renvoyés classés tag par tag.

L'usage de ces paramètres pour une session non active renvoie une liste vide de documents

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Variables post URL

Nom	Description	Schéma
actor	Identifiant d'un acteur de la session. permet de récupérer l'ensemble des documents à signer ou approuver pour l'acteur spécifié.	UInt32 (strictement positif)
status_mask	Permet de positionner les bits du filtre de status sur les document recherchés. Exemple : status_mask = 33 renvoie les documents ni signés ni approuvés ET les documents signés(*)	UInt32
tags	Liste des tags correspondant à types de signature ou des catégories d'approbation recherchés. Permet de récupérer les documents à signer ou approuver correspondant aux tags passés.	Liste de chaîne de caractères séparées par des virgules

(*) status_mask est un bitmask, chaque bit représentant un statut possible du document (voir tableau plus loin dans la section « Récupérer un document de session ».

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body contenant un tableau JSON de la liste des identifiants des documents trouvés dans la session dont l'identifiant est passé en paramètre de l'URL.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
409	Conflict	Pas de contenu. Impossible de positionner status_mask avec 'actor' ou 'tag' en même temps.
422	Unprocessable	Le paramètre status_mask est incorrect ou a été employé en même temps que le paramètre actor ou tags.

Exemple d'un body de réponse HTTP sur requête sans paramètres

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "documents": [
    "/session/25/documents/3712",
    "/session/25/documents/3713",
    "/session/25/documents/3716",
    "/session/25/documents/3952",
    "/session/25/documents/4001"
  ]
}
```

Exemple d'un body de réponse HTTP sur requête avec le paramètre actor

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "countersign": [
    "/session/25/documents/3712",
    "/session/25/documents/3713",
    "/session/25/documents/3716",
  ],
  "cosign": [
    "/session/25/documents/3952",
    "/session/25/documents/4001"
  ]
}
```

6.3. Récupérer un document d'une session

```
GET /v(version}/session/(id}/document/(did}
```

Description

Permet de récupérer la définition d'un document d'une session de signature. L'accès à cette ressource ne permet pas directement le téléchargement du contenu du document, signé ou non, mais à l'ensemble des paramètres qui le constituent.

Parmi ces paramètres, la variable `status` permet de savoir où le document en est dans son processus d'approbation et de signature.

Paramètres URL

Type	Nom	Description	Schéma
Path	did *(requis)*	Identifiant du document	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body : `DocumentResource`. **Last-Modified** header donne la date de dernière modification **Date** header donne la date de création de la session de signature
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

DocumentResource :

Clé	Description	Schéma
date	Date de création du document	String
did	identifiant du document	Entier strictement positif
file-name	nom du fichier	String
id	identifiant de la session	Entier strictement positif
status	Stade d'avancement di document dans son processus d'approbation et de signature	`DocumentStatus`

Clé	Description	Schéma
title	titre en clair du document	String
abstract	résumé de présentation du document *(facultatif)*	String
user-data	Données propres au client *(facultatif)*	JSON Dictionary
manifest-data	Données liées à la constitution du manifeste *(facultatif)*	JSON Dictionary

DocumentStatus :

Status	Description réponse
1	Document ni aprouvé ni signé (genuine)
2	Document en cours d'approbation
3	Document approuvé
4	Document en cours de signature
5	Document totalement signé

Exemple d'un body de la réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Date: 2020-04-21T08:12:02.252Z
Last-Modified: 2020-04-21T08:12:02.252Z
...
{
  "abstract:"2e contrat de l'année avec Zorgland\nDevrait être le dernier",
  "did": 1012,
  "id": 25,
  "status": 1,
  "title": "Contract Société Zorg",
  "user-data": { ... },
  "manifest-data": { ...},
}
```

6.4. Récupérer l'URL de téléchargement d'un document

```
GET /v(version}/session/(id}/document/($did}/current
GET /v(version}/session/(id}/document/($did}/genuine
```

Description

Permet de récupérer l'URL de téléchargement pointant vers le contenu du document. Attention l'URL a une durée de vie limitée et doit être utilisé dans les plus brefs délais. Il ne doit en aucun cas être caché ou stocké pour un usage ultérieur.

Avec l'URL `current` c'est l'état courant qui est téléchargé. Avec l'URL `genuine` c'est l'état lors de l'upload.

L'état courant du document lorsque le document a été complètement signé est l'état final du document.

Dans le cas où un document a été signé et que la signature est enveloppante ou détachée, il est possible que l'URL pointe vers un fichier ZIP contenant plusieurs fichiers[10].

Paramètres URL

Type	Nom	Description	Schéma
Path	did *(requis)*	Identifiant du document	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
201	Created	Contenu Application/JSON : `ExpiringURLResource`. L'URL du download du document est renvoyée dans le header **Location** La date de création est dans le header **Date** La date d'expiration dans le header **Expires**
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
504	Gateway Timeout	L'éventuelle requête de scellement du document à télécharger faite par le système interne de CERTIGNA a pris trop de temps

Le code HTTP 200 n'est pas retourné par cette fonction et est considéré comme une erreur.

[10] il sera étudié dans une version ultérieur de l'API, l'usage du format ASIC.

6.5. Enlever un document d'une session

```
DELETE /v(version}/session/(id}/document/($did}
```

Description

Permet d'enlever un document d'une session de signature. La suppression d'un document n'est possible que s'il n'y a pas de scénario actif et si le document n'a pas encore été utilisé. En cas de scénario actif, une erreur HTTP 403 est renvoyée. En cas de document déjà utilisé, une erreur 409 est renvoyée.

Paramètres URL

Type	Nom	Description	Schéma
Path	did *(requis)*	Identifiant du document	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `DeletedURLResource`.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

7. Approuver et signer des documents

Le processus d'approbation et de signature des documents fonctionne en deux étapes :

- Trouver les documents à approuver ou à signer :
  Appel de l'API `GET /session/{id}/documents?actor=...&tags=...` qui permet pour un acteur donné et une liste de tag de processus donné d'obtenir une liste d'URL de documents.
- Approuver ou signer toute ou partie des documents récupérés à l'aide des API :
  `PUT /session/{id}/approve-documents` ou
  `PUT /session/{id}/sign-documents`.

Le processus d'approbation ou celui de signature peut se faire par OTP. Dans ce cas, il est nécessaire de créer un nouvel OTP pour l'acteur de l'approbation et la signature avec comme paramètre la liste des documents à approuver ou à signer.

Les APIs d'approbation (API `PUT /session/{id}/approve-documents`) ou de signature (API `PUT /session/{id}/sign-documents`) renvoient toutes les deux la même ressource appelée `SigningResource` présentée ci-dessous :

SigningResource :

Clé	Description	Schéma
otp	OTP de validation de l'opération *(facultatif)*	String
signatures	Liste des approbations ou signatures effectuées	Tableau de `SignatureResource`
threadId	Identifiant unique de l'opération	String
token	token du certificat de l'opération *(facultatif)*	String

SignatureResource :

Clé	Description	Schéma
actor	URL de l'acteur signataire	String
document	URL du document approuvé ou signé	String
tag	Process de signature (`approval`, `cosign`, ...)	String
signatureId	Identificant unique de la signature	String

7.1. Générer token de type OTP

```
PUT /v{version}/session/{id}/generate-otp
```

Description

Génération d'un OTP pour un acteur et une liste de documents.

Si les documents passés à cette fonction d'approbation ne sont pas à approuver ou signer maintenant une erreur 409 Conflict est renvoyée.

Si l'acteur n'a pas le droit de les approuver ou s'il n'y a pas de scénario actif (donc rien à approuver), une erreur 403 Forbidden est renvoyée.

Il est possible de générer plusieurs OTP pour le même jeu de documents et le même acteur. Dans ce cas le dernier OTP généré est gardé et le précédent supprimé.

Toute approbation ou signature qui remettrait en cause la possibilité d'approuver ou de signer les documents passés en paramètre provoque la révocation de l'OTP généré.

Tous les OTP expirent. Le temps d'expiration des OTP est fixé par la variable globale `otp-ttl` mais peut être modifié par le paramètre ttl.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
actor	URL de l'acteur signataire ou approbateur *(requis)*	String
documents	liste des URL des documents à approuver ou à signer *(requis)*	Tableaux d'URL (de strings)
length	Taille de la chaîne OTP à générer (défaut 6)	UInt32 (valeur maximum 256)
numeric	Si vrai, l'OTP généré est constitué uniquement de chiffres décimaux (0 à 9)	Booléen
ttl	Temps d'expiration en seconde de l'OTP. Si omis la valeur de la variable système `otp-ttl` est utilisée.	UInt32

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body : `OTPResource`. **Date** header donne la date de création de l'OTP **Expires** header donne la date d'expiration de l'OTP
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
409	Conflict	Les documents ne sont pas à approuver ou à signer dans l'étape courante.

OTPResource :

Clé	Description	Schéma
date	Date de création de l'OTP	String
expires	Date d'expiration de l'OTP	String
otp	Chaîne de caractère de l'OTP	String

7.2. Vérifier la validité d'un OTP

```
PUT /v{version}/session/{id}/check-otp
```

Description

Vérification de l'existence d'un OTP potentiellement pour un acteur donné et une liste de documents.

Si l'OTP n'est pas trouvé, une erreur 404 Not Found est renvoyée.

S'il y a un acteur de spécifié et que cet acteur n'est pas trouvé ou n'a pas le droit d'accès aux documents à signer, une erreur 404 Not Found est renvoyée.

S'il y a des URLs des documents passés à cette fonction de vérification et qu'ils ne font pas partie des documents associés au token ou que ces documents ne sont pas ou plus à approuver ou signer maintenant, une erreur 404 Not Found est renvoyée.

S'il n'y a pas de scénario actif (donc rien à approuver ou à signer), une erreur 403 est renvoyée.

Si la suppression de l'OTP est demandée et que cette suppression ne peut pas intervenir, pour quelque raison que cela soit, une erreur 403 est renvoyée. Dans ce cas précis, le retour de la fonction ne précise pas si l'OTP était bien le bon ou pas, il faut refaire une requête sans la demande de suppression pour en être sûr.

À noter que lors d'une opération d'approbation ou de signature dans laquelle un OTP est passé, une vérifcation est faite de l'existence et de la validité de cet OTP est faite et, dans ce cas, la vérification implique forcément l'acteur signataire ou approbateur ainsi que le ou les documents à signer.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
otp	valeur de l'OTP recherché *(requis)*	String
actor	URL de l'acteur signataire ou approbateur associé à l'OTP *(requis si l'appelant veut checker l'acteur)*	String

Clé	Description	Schéma
documents	liste des URL des documents à approuver ou à signer associés à l'OTP *(requis si l'appelant veut checker les documents associés à l'OTP)*	Tableaux d'URL (de strings)
delete	Si vrai et que l'OTP passé est valide, l'OTP est supprimé par la requête.	Booléen

Réponses

Code HTTP	Statut	Description réponse
200	OK	Le statut OK est renvoyé uniquement si l'OTP checké est valide. **Application/JSON** body contenant contenant la clé `otp`, la clé `actor` contenant l'url de l'acteur et la clé `documents` contenant les urls des documents concernés. Si l'OTP est détruit, le JSON renvoie uniquement la clé `deleted` qui contient l'ancien OTP.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu : l'OTP recherché n'existe pas ou n'était pas pour le bon acteur ou le bon ensemble de documents. À noter que si l'acteur ou les documents ne sont pas passés dans la requête, le check sur ces ressources n'est pas fait.

7.3. Approuver des documents

```
PUT /v{version}/session/{id}/approve-documents
```

Description

Permet d'approuver une liste de documents.

Si les documents passés à cette fonction d'approbation ne sont pas à approuver maintenant une erreur 409 Conflict est renvoyée.

Si l'acteur n'a pas le droit de les approuver ou s'il n'y a pas de scénario actif (donc rien à approuver), une erreur 403 est renvoyée.

En cas de succès, l'approbation des documents passés à l'API peut provoquer la terminaison de l'étape courante et le passage à une autre étape ou la bonne terminaison du scénario actif.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
actor	URL de l'acteur signataire ou approbateur *(requis)*	String
documents	liste des URL des documents à approuver *(requis)*	Tableaux d'URL (de strings)
otp	valeur de l'OTP préalablement demandé pour ce jeu de documents et cet acteur *(requis)*	String
manifest-data	Données liées à la constitution du manifeste du client (cf. chapitre « Resource Manifest »)	JSON Dictionary contenant les clés définies dans le paramètre général `approve-manifest-data`
tag	Tag de l'opération d'approbation. Si omis vaut `approval`	String

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body : `SigningResource` (cf intro du chapitre).
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
409	Conflict	Les documents ne sont pas à approuver dans l'étape courante.
422	Unprocessable	Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `approve-manifest-data`.

7.4. Signer des documents

```
PUT /v(version}/session/(id}/sign-documents
```

Description

Permet de signer une liste de documents.

Si les documents passés à cette fonction de signature ne sont pas à signer dans le cadre de l'étape en cours du scénario actif, une erreur 409 Conflict est renvoyée.

Si l'acteur n'a pas le droit de les signer ou s'il n'y a pas de scénario actif (donc rien à signer),, une erreur 403 est renvoyée.

En cas de succès, la signature des documents passés à l'API peut provoquer la terminaison de l'étape courante et le passage à une autre étape ou la bonne terminaison du scénario actif.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
actor	URL de l'acteur signataire ou approbateur *(requis)*	String
certificate	URL du certificat généré à la volée *(requis)*	String
documents	liste des URL des documents à approuver *(requis)*	Tableaux d'objets décrivant les documents à signer (voir ci-après)
otp	valeur de l'OTP préalablement demandé pour ce jeu de documents et cet acteur	String
manifest-data	Données liées à la constitution du manifeste du client (cf. chapitre « Resource Manifest »).	JSON Dictionary contenant les clés définies dans le paramètre général `sign-manifest-data`
tag	Tag de l'opération d'approbation. *(requis)*	String

Il est possible de préciser à la fois une clé `certificate` et une clé `otp` : dans ce cas nous aurons une double confirmation, Il est en revanche impossible de signer un document juste avec un OTP. En

conséquence, si aucune clé `certificate` n'est fournie et si cela est possible, cette API utilisera un certificat serveur dédié lié au compte de l'appelant.

La clé `certificate` identifiant le certificat est une URL locale, liée à la plateforme et renvoyée par l'API de génération des certificats à la volée (voir plus loin ressources CA et Certificate). Cette URL n'a de sens que pour un usage avec la plateforme.

Le tableau des documents à signer est soit un tableau un tableau constitué d'URL des documents à signer ou de dictionnaires JSON contenant les clés suivantes (possibilité de mixité) :

Clé	Description	Schéma
document-url	URL du document à signer *(requis)*	String
visual-params	Structure permettant de déterminer le cadre de signature	Tableau d'objet

Les structures visual params contenues dans le tableau précédent contiennent les objets suivants :

Clé	Description	Schéma
font-size	Taille de la police en pixels	Number (si 0 taille proportionnelle au cadre de la signature)
height	Hauteur en points du cadre de signature	Number
image-content	Image encodée en base64	String
width	Largeur en points du cadre de signature	Number
page-number	Numéro de la page portant la signature	Nombre entier (négatif si on compte à partir de la fin)
text	Texte à insérer avec la signature	String
text-align	Alignement horizontal du texte à insérer (0 = à gauche, 1 = centré, 2 = à droite)	Nombre entier
x	Absisse du coin supérieur gauche du cadre de signature	Number
y	Ordonnée du coin supérieur gauche du cadre de signature	Number

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body : `SigningResource` (cf intro du chapitre).
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
409	Conflict	Les documents ne sont pas à approuver dans l'étape courante.
422	Unprocessable	Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `sign-manifest-data`.
504	Gateway Timeout	La requête de signature envoyée vers le système interne de CERTIGNA a pris trop de temps

8. Ressource Actor

Les acteurs sont les personnes qui approuvent et qui signent les documents.

8.1. Ajouter un acteur à une session

```
POST /v(version}/session/(id}/actors
```

Description

Permet d'ajouter un acteur à une session de signature. L'ajout d'un acteur n'est possible que s'il n'y a pas de scénario actif. En cas de scénario actif, une erreur HTTP 403 est renvoyée.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
adm-id	identifiant administratif (SIRET ou SIREN) *(requis si personne morale)*	String
country	code du pays de l'acteur *(requis)*	String (ISO 3166 ALPHA-2)
first-name	prénom d'une personne physique	String
login	identifiant sur le système de Certigna	String
email	Email de la personne *(requis)*	String
name	nom de la personne physique ou morale *(requis)*	String
mobile	Numéro de téléphone	String
roles	liste des rôles de signature et d'approbation autorisés *(requis)*	Tableau de string
type	0 = personne physique, 1 = personne morale	Entier positif
user-data	Données propres au client	JSON Dictionary

Clé	Description	Schéma
manifest-data	Données liées à la constitution du manifeste du client (cf. chapitre « Resource Manifest »)	JSON Dictionary contenant les clés définies dans le paramètre général `actor-manifest-data`

Réponses

Code HTTP	Statut	Description réponse
201	Created	Contenu Application/JSON : `URLResource`. L'URL du nouvel acteur est renvoyée dans le header **Location** La date de création est dans le header **Date**
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
409	Conflict	Prénom positionné sur une personne morale
422	Unprocessable	Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `actor-manifest-data`.

Le renvoi d'un code HTTP 200 est considéré comme une erreur sur cette requête qui n'est valide que si un code HTTP 201 a été renvoyé et le header **Location** positionné.

8.2. Récupérer une liste d'identifiants d'acteurs d'une session

```
GET /v(version}/session/(id}/actors? ... parameters
```

Description

Sans paramètres, permet de récupérer une liste d'URL des acteurs d'une session donnée. La liste est renvoyée dans la variable `actors`.

En utilisant le paramètre `tags`, permet de lister la liste des acteurs impliqués dans les opérations d'approbation ou de signature correspondant à la liste de tags fournis.

Dans ce cas, la réponse est un ensemble de clés correspondant aux différents tags. Un acteur peut alors apparaître dans plusieurs listes de tags.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Variables post URL

Nom	Description	Schéma
tags	Liste des tags correspondant à types de signature ou des catégories d'approbation recherchés	Liste de chaîne de caractères séparées par des virgules

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body contenant un tableau JSON de la liste des identifiants des acteurs trouvés dans la session dont l'identifiant est passé en paramètre de l'URL.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

Exemple d'un body de réponse HTTP sans paramètres

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "actors": [
    "/session/25/actors/8712",
    "/session/25/actors/10284",
    "/session/25/actors/12820"
  ]
}
```

Exemple d'un body de réponse HTTP avec paramètres

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "cosign": [
    "/session/25/actors/8712",
    "/session/25/actors/10284",
    "/session/25/actors/12820"
  ],
  "countersign": [
    "/session/25/actors/2011",
    "/session/25/actors/10284",
  ]
}
```

8.3. Récupérer un acteur d'une session

```
GET /v(version}/session/(id}/actor/(aid}
```

Description

Permet de récupérer le contenu d'un acteur d'une session de signature.

Paramètres URL

Type	Nom	Description	Schéma
Path	aid *(requis)*	Identifiant de l'acteur	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body : `ActorResource`. **Last-Modified** header donne la date de dernière modification **Date** header donne la date de création de la session de signature
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

ActorResource :

Clé	Description	Schéma
aid	identifiant du document	Entier strictement positif
adm-id	Indentifiant administratif (SIRET, SIREN, …)	String
country	Code du pays de l'acteur	String (ISO 3166 ALPHA-2)
date	Date de création de l'acteur	String
email	Mail de l'acteur	String
first-name	Prénom de l'acteur s'il s'agit d'une personne	String
id	identifiant de la session	Entier strictement positif
login	login (pour les acteurs connus du CMS de Certigna)	String
mobile	Numéro de téléphone mobile de la personne	String

Clé	Description	Schéma
name	Nom de la personne morale ou physique	String
roles	Rôles que peut prendre cet acteur	Tableau de Strings
type	Type d'acteur	0 = personne physique, 1 = personne morale
user-data	Données propres au client *(facultatif)*	JSON Dictionary
manifest-data	Données liées à la constitution du manifeste *(facultatif)*	JSON Dictionary

Exemple d'un body de la réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Date: 2020-04-21T08:12:02.252Z
Last-Modified: 2020-05-19T08:10:08.174Z

...
{
  "aid": 1000,
  "country": "FR",
  "email": "jeand.durand@gmail.com",
  "first-name" : "Jean"
  "id": 25,
  "name" : "Durand",
  "mobile": "+33685236912",
  "roles": ["sign", "cc", "to", "legal", "head"],
  "type": 0,
  "user-data": {
    "label": "Signature des documents du pacte A452",
    "internal-id": "52DDF244-FC3E-40EE-BCC4-D5A75E686032"
  },
  "manifest-data": { ... }
}
```

Les rôles d'un acteur déterminent les droits de ce dernier en tant qu'approbateur, signataire ou destinataire de documents.

Les rôles sont de deux types : en premier lieu les rôle systèmes, intrinsèques au fonctionnement de l'API (tags `sign`, `cosign`, `countersign`, `ordered-cosign`, `individual-sign`, `to`, `cc` ou `approval`) et en second lieu les rôles correspondant à des phases d'approbation définis dans le fichier de configuration de l'API (cf. chapitre II.3.3 rôle des acteurs pour plus de détail).

8.4. Enlever un acteur d'une session

```
DELETE /v(version}/session/(id}/actor/($aid}
```

Description

Permet de supprimer un acteur à une session de signature. La suppression d'un acteur n'est possible que s'il n'y a pas de scénario actif et si l'acteur n'est pas déjà utilisé dans le processus de signature.

En cas de scénario actif, une erreur HTTP 403 est renvoyée.

Paramètres URL

Type	Nom	Description	Schéma
Path	aid *(requis)*	Identifiant de l'acteur	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `DeletedURLResource`.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

9. Ressource Scenario

Les scénarios permettent la définition et le déroulement des workflows d'approbation et de signature.

Leur fonctionnement est largement expliqué dans les premiers chapitres de ce document.

La creation (`POST /session/{id}/scenarios`), l'intérogation (`GET /session/{id}/scenario/{sid}`) et l'update de scénario (`PATCH /session/{id}/scenario/{sid}`) contiennent toutes les trois dans leur corps de requête un tableau contenant les étapes appelée.

Ces étapes linéaires pour l'instant (elles pourront être hiérarchiques dans des évolutions à venir de l'API) sont constitué d'objets de type StepNode `StepNode` dont la composition est la suivante :

StepNode :

Clé	Description	Schéma
process	Tag du process d'approbation ou de signature *(requis)*	Chaîne de caractère qui doit être l'un des tags système (`sign`, `cosign`, `countersign`, `ordered-cosign`, `individual-sign` ou `approval`) ou ceux issus du paramétrage des catégories d'approbation
steps	liste des sous-étapes de l'étape	Tableau d'objets. (contenant uniquement des URLs d'acteurs dans la version actuelle de l'API)
type	Type de signature *(requis si étape de signature)*	`SignatureType`
user-data	Données propres au client	JSON Dictionary

SignatureType :

Type de signature	Description
1	Signature enveloppée
2	Signature enveloppante
3	Signature détachée

Par ailleurs, ces APIs utilise également des types communs pour définir le format (`SignatureFormat`) et le niveau de signature (`SignatureLevel`) exigés dans le cadre d'un scénario :

SignatureFormat :

Format	Acronyme	Description
1	PAdES	PDF Advanced Electronic Signature
2	XAdES	XML Advanced Electronic Signature
3	CAdES	CMS Advanced Electronic Signature

SignatureLevel :

Niveau	Acronyme	Description
1	B	Basic baseline profile
2	T	Timestamp baseline profile
3	LT	Long Term baseline profile
4	LTA	Long Term Archiving baseline profile

9.1. Ajouter un scénario à une session

```
POST /v{version}/session/{id}/scenarios
```

Description

Permet d'ajouter un scénario à une session de signature. L'ajout d'un scénario n'est possible que s'il n'y a pas déjà un scénario actif. En cas de scénario actif, une erreur HTTP 403 est renvoyée.

Dans la version actuelle de l'API de signature, seule les signatures de type PAdES enveloppée sont implémentées. Néanmoins, la forme des données gérées par les scénarios est prévue pour gérer toutes les formes de signatures.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
documents	liste des URL des documents à approuver ou à signer *(requis)*	Tableau d'URLs (de strings)
format	format de signature par défaut *(requis)*	`SignatureFormat`
level	niveau de signature (B, T, LT ou LTA) *(requis)*	`SignatureLevel`
steps	liste des étapes du scénario *(requis)*	Tableau d'objets `StepNode`
user-data	Données propres au client	JSON Dictionary
manifest-data	Données liées à la constitution du manifeste du client (cf. chapitre « Resource Manifest »)	JSON Dictionary contenant les clés définies dans le paramètre général `scenario-manifest-data`

Réponses

Code HTTP	Statut	Description réponse
201	Created	Contenu Application/JSON : `URLResource`. L'URL du nouveau scénario est renvoyée dans le header **Location** La date de création est dans le header **Date**
403	Forbidden	Pas de contenu. Renvoyé si les acteurs utilisés dans les étapes n'ont pas les droits nécessaires pour effectuer les opérations d'approbation ou de signature qui sont attendues d'eux.
404	Not Found	Pas de contenu
409	Conflit	Renvoyé si les formats de signature du scénario, les documents et la définition des étapes sont incompatibles
422	Unprocessable	Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `scenario-manifest-data`.

Le renvoi d'un code HTTP 200 est considéré comme une erreur sur cette requête.

9.2. Récupérer la liste d'identifiants des scénarios d'une session

```
GET /v(version}/session/(id}/scenarios
```

Description

Récupère dans la variable `scenarios` la liste d'URL des scénarios d'une session donnée.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body contenant les URLs des scenarios.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

Exemple d'un body de réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "scenarios": [
    "/session/25/actors/4033",
   "/session/25/actors/7965"
  ]
}
```

9.3. Récupérer un scénario d'une session

```
GET /v(version}/session/(id}/scenario/(sid}
```

Description

Permet de récupérer le contenu d'un scénario d'une session de signature.

Paramètres URL

Type	Nom	Description	Schéma
Path	sid *(requis)*	Identifiant du scenario	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body : `ScenarioResource`. **Last-Modified** header donne la date de dernière modification **Date** header donne la date de création de la session de signature
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

ScenarioResource

Clé	Description	Schéma
date	Date de création du scénario	String
documents	liste des URL des documents à approuver ou à signer	Tableau d'URLs (de strings)
format	format de signature	`SignatureFormat`
level	niveau de signature	`SignatureLevel`
id	identifiant de la session	Entier strictement positif
steps	liste des étapes du scénario	Tableau d'objets `StepNode`
sid	identifiant du scénario	Entier strictement positif
status	statut du scénario	`ScenarioStatus`

Clé	Description	Schéma
user-data	Données propres au client	JSON Dictionary
manifest-data	Données liées à la constitution du manifeste	JSON Dictionary

ScenarioStatus :

Status	Description
1	Scenario en construction
2	Scenario en construction après un split de scénario
4	Scénario actif
10	Scénario correctement terminé
11	Scénario correctement terminé après un split de scénario
20	Scénario supprimé
21	Scénario abandonné
22	Scénario abandonné à cause de l'expiration de la session (car non terminé)

9.4. Modifier un scenario d'une session

```
PATCH /v(version}/session/(id}/scenario/($sid}
```

Description

Permet de modifier un scénario de signature. Attention, seuls les scénarios non actifs et non terminés peuvent être modifiés.

En cas de scénario actif ou de tentative de modification d'un scénario déjà terminé, une erreur HTTP 403 est renvoyée.

Paramètres URL

Type	Nom	Description	Schéma
Path	sid *(requis)*	Identifiant du scénario	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les mêmes clés que pour l'API `POST /sessions/scenarios`.

Réponses

Code HTTP	Statut	Description réponse
200	OK : modifié	Contenu Application/JSON : `URLResource`. La date de création est dans le header **Date** La date de modification est dans le header **Last-Modified**
403	Forbidden	Pas de contenu. Renvoyé si les acteurs utilisés dans les étapes n'ont pas les droits nécessaires pour effectuer les opérations d'approbation ou de signature qui sont attendues d'eux.
404	Not Found	Pas de contenu
409	Conflit	Renvoyé si les formats de signature du scénario, les documents et la définition des étapes sont incompatibles
422	Unprocessable	Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `scenario-manifest-data`.

9.5. Activer un scénario

```
PUT /v{version}/session/{id}/scenario/{$sid}/activate
```

Description

Active un scénario. Ne peut se faire que sur un scénario non actif (c.-à-d. le dernier de la liste des scénarios de la session).

En cas de scénario actif, une erreur HTTP 403 est renvoyée.

Paramètres URL

Type	Nom	Description	Schéma
Path	sid *(requis)*	Identifiant du scénario	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
manifest-data	Données liées à la constitution du manifeste du client (cf. chapitre « Resource Manifest »)	JSON Dictionary contenant les clés définies dans le paramètre général `activate-manifest-data`

Réponses

Code HTTP	Statut	Description réponse
200	OK : activé	Contenu Application/JSON : `URLResource`. La date de création est dans le header **Date** La date de modification est dans le header **Last-Modified**
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
422	Unprocessable	Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `activate-manifest-data`.

9.6. Faire le split d'un scénario

```
PUT /v(version}/session/(id}/scenario/($sid}/split
```

Description

Permet de couper un scénario actif en deux. La partie correctement terminée devient un scénario correctement clos et un nouveau scénario est créé avec la partie non encore exécutée du scénario.

En cas de scénario non actif ou de scénario n'ayant aucune étape terminée une erreur HTTP 403 est renvoyée.

Paramètres URL

Type	Nom	Description	Schéma
Path	sid *(requis)*	Identifiant du scénario	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
reason	Raison en clair du split de scénario *(requis)*	String
user-data	Données propres au client. En l'absence de cette clé, les user-data du scénario splitté sont prises	JSON Dictionary. Le User data passé ici templace la clé éponyme dans le nouveau scénario
manifest-data	Données liées à la génération du nouveau sc du manifeste du client (cf. chapitre « Resource Manifest »). En l'absence de cette clé, le manifest-data du scénario splitté est pris	JSON Dictionary contenant les clés définies dans le paramètre général `scenario-manifest-data`

Réponses

Code HTTP	Statut	Description réponse
201	Created	Contenu Application/JSON : `URLResource`. L'URL du nouveau scénario généré par l'opération de split est renvoyée dans le header **Location** La date de création est dans le header **Date**
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
409	Conflict	Le scénario ne peut pas être splitté car il n'a pas été entammé.
422	Unprocessable	Erreur renvoyée si les données de `manifest-data` ne sont pas conformes à la définition de `scenario-manifest-data`.

Le renvoi d'un code HTTP 200 est considéré comme une erreur sur cette requête qui n'est valide que si un code HTTP 201 a été renvoyé et le header **Location** positionné.

9.7. Abandon d'un scénario actif

```
PUT /v(version}/session/(id}/scenario/($sid}/cancel
```

Description

Force l'abandon d'un scénario actif.

En cas de scénario non actif, une erreur HTTP 403 est renvoyée.

Paramètres URL

Type	Nom	Description	Schéma
Path	sid *(requis)*	Identifiant du scénario	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
reason	Raison en clair de l'abandon du scénario *(requis)*	String
manifest-data	Données liées à la constitution du manifeste du client (cf. chapitre « Resource Manifest »)	JSON Dictionary contenant les clés définies dans le paramètre général `cancel-manifest-data`

Réponses

Code HTTP	Statut	Description réponse
200	OK : cancelled	Contenu Application/JSON : `URLResource`. La date de création est dans le header **Date** La date de modification est dans le header **Last-Modified**
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

9.8. Enlever un scenario d'une session

```
DELETE /v(version}/session/(id}/scenario/($sid}
```

Description

Permet de supprimer un scénario d'une session de signature. La suppression d'un scénario n'est possible que s'il s'agit du dernier scénario et qu'il n'est pas actif.

En cas de scénario actif ou de tentative de suppression d'un scénario qui n'est pas le dernier, une erreur HTTP 403 est renvoyée.

Paramètres URL

Type	Nom	Description	Schéma
Path	sid *(requis)*	Identifiant du scénario	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `DeletedURLResource`.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

10.Ressources CA et Certificate

Cette section décrit les ressources permettant la génération de certificats à la volée qui seront ensuite utilisés par les acteurs dans les opérations de signature des documents.

La première chose à faire est de récupérer la liste des AC émettrices puis, pour chaque AC et chaque acteur concerné, demander les conditions générales de ventes et obtenir ainsi un token prouvant que ces dernières ont bien été récupérées par l'appelant et acceptées par l'acteur.

La récupération de la liste des AC émettrice se fait via l'API `GET /cas`

L'opération de récupération du token validant l'acceptation des CGU est ensuite réalisée par l'appel à l'API `GET /ca/(caid}/cgu?session=...&actor=...` ou `caid` est l'identifiant de l'AC émettrice, `session` celle de la session concernée et `actor` celle de l'acteur de la session pour lequel on demande les CGU.

Il est ensuite possible de générer un certificat pour un acteur choisi à l'aide de l'API `POST /session/(id}/certificates`.

Vous pouvez à tout moment lister l'ensemble des certificat générés d'une session ou pour un acteur à l'aide de l'API `GET /session/(id}/certificates?...`.

La validité de ce certificat en termes de durée de vie est ensuite possible grâce à l'API permettant de récupérer les informations standard du certificat généré à la volée à partir de son identifiant : `GET /session/(id}/certificate/(cid}`.

Il est enfin possible de le révoquer en utilisant l'API `DELETE /session/(id}/certificate/(cid}`.

10.1.Récupération de la liste des AC émettrices

```
GET /v(version}/cas
```

Description

Cette fonction permet récupérer la liste des AC émettrice (dans la première version de l'API, une seule AC émetrice est renvoyée).

Paramètres URL

Type	Nom	Description	Schéma
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Le Body est un dictionnaire JSON qui contient la liste des AC émettrices dans le tableau introduit par la clé `certification-authorities`.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

Exemple d'un body de la réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Date: 2020-04-21T08:12:02.252Z
...
{
  "certification-authorities": [
    "/ca/1201",
    "/ca/3104"
    ...
  ]
}
```

10.2.Accéder à la description d'une autorité de certification

```
GET /v(version}/ca/(caid}
```

Description

Cette fonction permet récupérer le contenu de l'information d'une autorité de certification.

Paramètres URL

Type	Nom	Description	Schéma
Path	caid *(requis)*	Identifiant de l'AC	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `AuthorityResource`.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

AuthorityResource :

Clé	Description	Schéma
caid	identifiant de l'autorité de certification	Entier strictement positif
cgu-version	version courante des conditions générales d'usages	String
long-name	description de l'autorité de certification	String
name	nom de l'autorité de certification	String

Exemple d'un body de la réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Date: 2020-04-21T08:12:02.252Z
...
{
  "caid": 1216,
  "cgu-version": "1.5.4",
  "long-name": "version longue du nom",
  "name": "Nom listable"
}
```

10.3.Récupération de l'URL de téléchargement des CGU de l'AC émettrice

```
GET /v(version}/ca/(caid}/cgu?session=... &actor=...
```

Description

Cette fonction permet récupérer la version courante des CGU d'une AC émettrice pour une session et acteur demandeur de certificat.

Pour récupérer le PDF des CGU, le programmeur peut ensuite, s'il le souhaite, télécharger le contenu de l'URL pointée par la variable `cgu-url`.

Il est recommandé de lancer le téléchargement immédiatement après la réception de l'URL dont la durée de vie peut être courte.

Le fait de demander les CGU vaut acceptation de celles-ci par le renvoie dans la réponse d'un `token` (une string sous la forme d'un UUID) prouvant cette acceptation.

Le token renvoyé est valable pour l'acteur pour toute la durée de la session. Néanmoins, il est possible d'appeler cette API plusieurs fois et, dans ce cas, un nouveau token peut être renvoyé si la version des CGU de l'AC a évoluée entre temps.

Paramètres URL

Type	Nom	Description	Schéma
Path	cid *(requis)*	Identifiant de l'AC émettrice	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Variables post URL

Nom	Description	Schéma
session	Identifiant de la session	Entier positif
actor	Identifiant de l'acteur demandant les CGU	Entier positif

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `CGUResource`.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
504	Gateway Timeout	La requête de récupération des CGU vers le système interne de CERTIGNA a pris trop de temps

CGUResource :

Clé	Description	Schéma
actor	URL de l'acteur pour lequel les CGU sont demandées	String
authority	URL de l'autorité de certification	String
download-url	URL de download des CGU	String
session	URL de la session	String
token	Token prouvant que les CGU ont bien été demandées	String
version	Version des CGU	String

Exemple d'un body de la réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "actor": "/session/1254/actor/100",
  "authority":"/v1.5/ca/1216",
  "download-url": "/myServer/anURL/file.pdf",
  "session: "/session/1254",
  "token": "00A9C459-539C-4411-9EB1-D82A60CB4278",
  "version": "1.3.12"
}
```

10.4.Générer un certificat à la volée

```
POST /v(version}/session/(id}/certificates
```

Description

Permet de générer un certificat à la volée pour une session donnée. Si on redemande plusieurs fois un certificat pour le même acteur alors qu'il existe un certificat valide pour le même token, dès le deuxième appel, l'erreur `422 Unprocessable` est renvoyée.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Request body

Dictionnaire JSON contenant les clés suivantes :

Clé	Description	Schéma
actor	URL de l'acteur pour lequel on demande le certificat *(requis)*	String
authority	URL de l'autorité de certification émétrice *(requis)*	String
token	token certifiant l'acceptation des CGU de l'AC émétrice par l'acteur *(requis)*	String
ttl	durée de vie en seconde du certificat. Si omis la valeur de la variable globale `certificate-ttl` est utilisée	UInt32
supporting-documents	Liste des d'objet conteant le nom (name) et l'url temporaire (url) d'upload des documents ayant servi à l'authentification	Tableau d'URLs (strings)

Réponses

Code HTTP	Statut	Description réponse
201	Created	Contenu Application/JSON : `ExpiringURLResource`. L'URL du certificat est renvoyée dans le header **Location** La date de création est dans le header **Date** La date d'expiration dans le header **Expires**
403	Forbidden	Pas de contenu

Code HTTP	Statut	Description réponse
404	Not Found	Pas de contenu
409	Conflict	L'acteur, l'autorité de certification et le token passés ne matchent pas (y compris mauvais token).
422	Unprocessable	Demande de certificat alors qu'il existe un certificat valide pour l'acteur et le token demandé
504	Gateway Timeout	La requête de génération de certificat vers le système interne de CERTIGNA a pris trop de temps

Le renvoi d'un code HTTP 200 est considéré comme une erreur sur cette requête qui n'est valide que si un code HTTP 201 a été renvoyé et le header **Location** positionné.

10.5.Récupérer une liste d'identifiants de certificats générés

```
GET /v(version}/session/(id}/certificates?... parameters...
```

Description

Permet de récupérer une liste d'URL de certificats générés par la plateforme pour une session et correspondant aux critères de recherche passés dans les post-variables de l'URL. La liste est renvoyée dans la variable `certificates`.

Si aucun critère n'est précisé, cette API renvoie l'ensemble des identifiants de certificats de l'usager appelant pour la session passée en paramètre.

Paramètres URL

Type	Nom	Description	Schéma
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Variables post URL

Nom	Description	Schéma
caid	identifiant d'une autorité de certification	entier positif
actorids	liste d'identifiants d'acteurs séparés par des virgules	liste de strings
ttlmin	durée de vie totale minimale(*)	entier positif ou nul
ttlmax	durée de vie totale maximale(*)	entier strictement positif
dynttlmin	durée de vie restante minimale du certificat(*). Ce paramètre est prioritaire sur expirationstatus	entier positif ou nul
dynttlmax	durée de vie restante maximale du certificat(*). Ce paramètre est prioritaire sur expirationstatus	entier. si le nombre est négatif, permet de chercher des certificats expirées depuis un certain temps
userids	liste d'identifiant d'usagers séparés par des virgules	liste de strings
expirationstatus	remplace dynttlmin et dynttlmax. Par défaut si ni dynttlmin ni dynttlmax n'est utilisé et que rien n'est spécifié, la requête renvoie les certificats valides.	all = tous valid = les non expirés expired = les expirés

(*) la recherche de temps exacte se fait en positionnant la même valeur pour le min et le max

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body contenant un tableau JSON de la liste des identifiants des certificats trouvées correspondant au critères (le tableau peut être vide).
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

Exemple d'un body de réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
...
{
  "certificates": [
    "/session/1620/certificate/7022",
    "/session/1620/certificate/8974",
   ]
}
```

10.6.Récupérer les informations d'un certificat

```
GET /v(version}/session/(id}/certificate/(cid}
```

Description

Permet de récupérer le contenu des informations d'un certificat généré à la volée par la plateforme.

Paramètres URL

Type	Nom	Description	Schéma
Path	cid *(requis)*	Identifiant du certificat	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	**Application/JSON** body : `CertificateResource`. **Last-Modified** header donne la date de dernière modification **Date** header donne la date de création du certificat **Expires** header donne la date de fin de validité du certificat
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu

CertificateResource :

Clé	Description	Schéma
actor	URL de l'acteur pour lequel les CGU sont demandées	String
authority	URL de l'autorité de certification	String
date	Date de création du certificat	String
expires	Date d'expiration du certificat	String
session	URL de la session	String
SN	Numéro de série du certificat	String
status	Statut du certificat	0 = expiré, 1 = valide
ttl	Durée de vie initiale du certificat (secondes)	Entier strictement positif

Exemple d'un body de réponse HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Date: 2020-04-21T08:12:02.252Z
Expires: 2020-04-22T08:13:00Z
...
{
  "actor": "/session/25/actor/100",
  "authority": "/ca/1245",
  "date": "2020-04-21T08:12:02.252Z",
  "expires": "2020-04-22T08:13:00Z",
  "id": 12,
  "cid": 1325,
  "SN": "10:e6:fc:62:b7:41:8a:d5:00:5e:45:b6:47:2a",
  "status": 1,
  "ttl": 7200,
}
```

On pourra utiliser cette API pour vérifier qu'un certificat est valide et que son temps de validité est suffisant pour les opérations projetées.

10.7.Invalider un certificat

```
DELETE /v(version}/session/(id}/certificate/(cid}
```

Description

Permet d'invalider un certificat précédemment généré à la volée.

Paramètres URL

Type	Nom	Description	Schéma
Path	cid *(requis)*	Identifiant du certificat	Entier positif
Path	id *(requis)*	Identifiant de la session	Entier positif
Path	version *(requis)*	Version de l'API sous la forme x.y	String

Request headers

Headers standard d'authentification.

Réponses

Code HTTP	Statut	Description réponse
200	OK	Contenu Application/JSON : `DeletedURLResource`.
403	Forbidden	Pas de contenu
404	Not Found	Pas de contenu
504	Gateway Timeout	La requête de d'invalidation de certificat vers le système interne de CERTIGNA a pris trop de temps

IV. Configuration de l'API-C

Le fonctionnement de l'API-C est régi par des variables de configuration qui sont référencées dans le tableau suivant (les variables de configuration sur fond orange ne sont pas implémentées dans la version actuelle de l'API) :

Variable de configuration	Description	Schéma
accept-forced-closure	Si vrai, accepte la clôture forcée des sessions	Boolean
activate-manifest-data	Liste d'objets définissants les informations que l'usager peut ajouter dans la clé manifest-data lors de l'activation d'un scénario	Dictionnaire d'objets
actor-manifest-data	Liste d'objets définissants les informations que l'usager peut ajouter dans la clé manifest-data lors de l'ajout d'un acteur dans une session	Dictionnaire d'objets
approve-manifest-data	Liste d'objets définissants les informations que l'usager peut ajouter dans la clé manifest-data lors de l'approbation de documents d'une session	Dictionnaire d'objets
cancel-manifest-data	Liste d'objets définissants les informations que l'usager peut ajouter dans la clé manifest-data lors de l'abandon d'un scénario actif	Dictionnaire d'objets
certificate-requests-timeout	Timeout des requêtes faites pour la génération des certificats à la volée en millisecondes (défaut 10000)	UInt32
certificate-ttl	Durée de vie par défaut en secondes des certificats autogénérés (si non précisé, 900 secondes soit 15 mn)	UInt32
certification-authorities	Liste des autorités de certification	Tableau de `CADefinition`
client-upload-number-max	Nombre maximal de fichiers uploadés par client (si non précisé, pas de limite, sinon client-upload-number-max ≥ session-upload-number-max)	UInt32
client-upload-size-max	Taille maximale de l'ensemble des fichiers uploadés par client en KB (si non précisé, pas de limite, sinon client-upload-size-max ≥ session-upload-size-max)	UInt32
closure-manifest-data	Liste d'objets définissants les informations que l'usager peut ajouter dans la clé manifest-data lors de la cloture des sessions	Dictionnaire d'objets
database-connection	Dictionnaire de conntection à la base de données	`DBConnection`
default-language	Langue par défaut de fonctionnement de l'API. Si ce paramètre est omis, l'anglais (EN) est la langue par défaut. Ce paramètre n'est utilile que dans le cas où les requêtes HTTP ne comprennent pas le header DefaultLanguage	String (2 charactères)
document-approval-categories	Liste d'objets définissants les catégories d'approbation des documents. Si cette variable n'est pas définie, le système comprend une seule catégorie d'approbation de document appelée "Approbation".	Dictionnaire d'objets

Variable de configuration	Description	Schéma
document-manifest-data	Liste d'objets définissants les informations que l'usager peut ajouter dans la clé manifest-data lors de l'ajout d'un document dans une session	Dictionnaire d'objets
downloads-path	Chemin du dossier qui contiendra les fichiers à downloader	String
download-ttl	Durée de vie en seconde des fichiers à downloader	UInt32
external-requests-timeout	Durée du timeout des requêtes externes (hors requêtes de signature et de génération de certificats). Temps donné en milliseconde (défaut 5000)	UInt32
long-identifiers	Si vrai, l'ensemble des identifiants renvoyés par les requêtes sont des URL complètes	Boolean
manifest-certificate	Chemin vers le fichier du certificat de signature des manifest de preuve sur la plateforme	String
manifest-generation-options	Options de génération du manifeste de preuve	`ManifestOptions`
manifest-on-closure	Si vrai, force la génération d'un manifest de preuves à la clôture d'une session de signature	Boolean
manifest-template-path	Chemin du fichier html de template pour la génération du manifeste de preuve	String
otp-ttl	Durée de vie en secondes des OTP générés par la plateforme. Si cette variable n'est pas précisée, le temps est fixé à 300 secondes (5 minutes)	UInt32
port	Port d'écoute du serveur de l'API. Défault 8008	Entier compris entre 80 et 65535
scenario-manifest-data	Liste d'objets définissants les informations que l'usager peut ajouter dans la clé manifest-data lors de l'ajout d'un scenario dans une session	Dictionnaire d'objets
session-documents-number-max	Nombre maximal de documents attachés à une session (si non précisé, pas de limite)	UInt32
session-documents-size-max	Taille maximale de l'ensemble des documents attachés à une session en KB (si non précisé, pas de limite, sinon session-documents-size-max ≥ upload-size-max)	
session-manifest-data	Liste d'objets définissants les informations que l'usager peut ajouter dans la clé manifest-data lors de la création des sessions	Dictionnaire d'objets
sign-server-login	Crédentiel de login au système de signature pour signer les manifeste	`SignServerLogin`
signature-manifest-data	Liste d'objets définissants les informations que l'usager peut ajouter dans la clé manifest-data lors de la signature de documents	Dictionnaire d'objets
signature-requests-timeout	Durée du timeout des requêtes externes de signature. Temps en milliseconde (défaut 30000)	UInt32
storage-path	Chemin du dossier de stockage des documents	String

Variable de configuration	Description	Schéma
ttl-max	Valeur maximale de la durée de vie des sessions (ttl-max > ttl-min)	UInt32
ttl-min	Valeur minimal de la durée de vie des sessions (0 < ttl-min < ttl-max)	UInt32
upload-bandwidth-max	Bande passante d'upload maximale en KB par seconde (si non précisié, pas de limite)	UInt32
upload-certificate	Chemin vers le dossier contenant les fichiers 'cert.pem' et 'key.pem' du certificat de scellement des fichiers uploadés sur la plateforme	String
upload-size-max	Taille maximale d'un fichier uploadé en KB (si non précisé 30000 KB soit un peu moins de 30MB)	UInt32
upload-ttl	Durée de vie en secondes des documents uploadés sur la plateforme avant leur ajout à une session. Si cette variable n'est pas précisée, le temps est fixé à 900 secondes (15 minutes)	UInt32
uploads-path	Chemin vers le dossier de téléversement des fichiers uploadés	String

CADefinition :

Clé	Description	Schéma
aki	Identifiant global de l'autorité de certification	String
uuid	Identifiant local de l'autorité de certification	String
name	Nom de l'autorité de certification	String
cgu-version	Version courante des CGU	String (optionnel)
long-name	Description de l'autorité de certification	String (optionnel)
cgu-path	Chemin vers un fichier local contenant les CGU	String (optionnel)

DBConnection :

Clé	Description	Schéma
host	Serveur hébergeant la base de données	String
database	Nom de la base de données	String
user	Login de connexion à la base de données	String
password	Password de connexion à la base de données	String

ManifestOptions :

Clé	Description	Schéma
bottom-margin	Taille de la marge du bas (défaut "15mm")	String ou nombre
format	Format de page ("a5", "a4", "a3", "letter", ...)	String
footer	Description du pied de page du manifeste	HTML String (format pupeteer)
header	Description de l'en-tête du manifeste	HTML String (format pupeteer)
left-margin	Taille de la marge de gauche (défaut "10mm")	String ou nombre
orientation	"portrait" ou "paysage" (ou "landscape")	String
right-margin	Taille de la marge droite (défaut "10mm")	String ou nombre
top-margin	Taille de la marge du haut (défaut "15mm")	String ou nombre

SignServerLogin :

Clé	Description	Schéma
login	login de connexion au serveur de signature	String
password	password de connexion au serveur de signature	String

* * *