Liberté • Égalité • Fraternité

RÉPUBLIQUE FRANÇAISE

MINISTÈRE DU TRAVAIL, DE L'EMPLOI, DE LA FORMATION PROFESSIONNELLE ET DU DIALOGUE SOCIAL

MINISTÈRE DES AFFAIRES SOCIALES ET DE LA SANTÉ

MINISTÈRE DE L'ÉCONOMIE ET DES FINANCES

# Spécifications Interops-R

Standard d'interopérabilité entre organismes de la sphère sociale et l'administration

Réf. : Standard_Interops-R1.0_Specifications

Version 1.0 du 21/04/2016

| | |
|---|---|
| Référence :<br>Version :<br>Date de dernière mise à jour : | Standard_Interops-R1.0_Specifications<br>1.0<br>21/04/2016 |
| Niveau de confidentialité : | Public |

# Diffusion

| Destinataires | Objet de la diffusion |
|---|---|
| | |

# Table des mises à jour du document

| N° de version | Etat[1] | Date | Auteur | Objet de la mise à jour |
|---|---|---|---|---|
| 0.1 | T | 29/02/16 | Y. Béot | Création |
| 0.2 | T | 15/03/16 | Y. Béot | Prise en compte des remarques de l'atelier du 02/03/2016 |
| 0.3 | T | 17/03/16 | Y. Béot | Prise en compte des remarques de l'atelier du 16/03/2016 |
| 0.4 | T | 13/04/16 | Y. Béot | Prise en compte des remarques de l'atelier du 24/03/2016, de la MSA, du GIP MDS et de la CNAM-TS |
| 1.0 | V | 21/04/16 | GIP-MDS | Document validé |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

[1] **T** : En cours de modification ; **V** : Validé

# SOMMAIRE

# 1. INTRODUCTION

## 1.1 Objet du document

Ce document présente les spécifications fonctionnelles et techniques du Standard d'Interopérabilité des Organismes de la Sphère Sociale pour des échanges **basés sur REST** dans un **mode « application à application »** : **Interops-R**.

## 1.2 Relation avec d'autres documents

Ce document dérive des principes édictés dans [R1] mais reste indépendant par rapport à ce document et auto-portant.

## 1.3 Organisation et structure du document

La structure du présent document est, en sus de la présente introduction, organisée comme suit :

- Le chapitre 2 « Description des mécanismes » présente les différents mécanismes mis en œuvre dans le cadre d'Interops-R
- Le chapitre 3 « Spécifications techniques » précise pour l'architecture, les éléments de sécurité ainsi que le format des différents échanges à mettre en place dans le cadre d'Interops-R
- Le chapitre 4 « Impacts sur les traces » énumère les différents éléments à consolider lors de la génération des traces.
- Le chapitre 5 « Gestion de la convention » préciser le contenu de la convention

## 1.4 Références

### 1.4.1 Références internes

| | Référence | Titre | Auteur | Ver. | Date |
|---|---|---|---|---|---|
| [R1] | Standard Interops-A2.0_SpécificationsDétaillées | Spécifications détaillées du mode « application à application » | Groupe de travail Interops | 2.0 | 05/04/2012 |
| [R2] | Standard Interops2.0_FormatEchangeTraces | Format d'échange des traces | Groupe de travail Interops | 2.0 | 05/04/2012 |

### 1.4.2 Références externes

| Réf | Titre | Auteur | Date |
|---|---|---|---|
| [HTTPAuth] | RFC 2617 - HTTP Authentication: | P. Hallam-Baker, J. | Juin 1999 |

| | | | |
|---|---|---|---|
| | Basic and Digest Access Authentication | Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart | |
| [JSON] | RFC 7159 - The JavaScript Object Notation (JSON) Data Interchange Format | T. Bray, Ed. | Mars 2014 |
| [JWE] | RFC 7516 - JSON Web Encryption (JWE) | M. Jones, J. Hildebrand | Mai 2015 |
| [JWS] | RFC 7515 - JSON Web Signature (JWS) | M. Jones, J. Bradley, N. Sakimura | Mai 2015 |
| [JWT] | RFC 7519 - JSON Web Token (JWT) | M. Jones, J. Bradley, N. Sakimura | Mai 2015 |
| [OAUTH2] | RFC 6749 - The OAuth 2.0 Authorization Framework | D. Hardt, Ed. | Octobre 2012 |
| [OAUTH2Bearer] | RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage | M. Jones, D. Hardt | Octobre 2012 |
| [RGS] | Référentiel Général de Sécurité<br>Version 2.0 | ANSSI/ SGMAP | 13/06/2014 |
| [RGS_A_4] | Référentiel Général de Sécurité version 2.0<br>Annexe A4 - Profils de Certificats / LCR / OCSP et Algorithmes Cryptographiques – version 3.0 | ANSSI/ SGMAP | 27/02/2014 |
| [RGS_B_1] | Référentiel Général de Sécurité version 2.0<br>Annexe B1 : Mécanismes cryptographiques - version 2.03 | ANSSI/ SGMAP | 21/02/2014 |
| [TIMESTAMP] | RFC 3339 - Date and Time on the Internet: Timestamps | G. Klyne, C. Newman | Juillet 2002 |
| [TLS11] | RFC 4346 - The Transport Layer Security (TLS) Protocol Version 1.1 | T. Dierks, E. Rescorla | Avril 2006 |
| [TLS12] | RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2 | T. Dierks, E. Rescorla | Août 2008 |

## 1.5 Conventions

Sauf indication contraire, toutes les spécifications précisées par ce document sont OBLIGATOIRES (« MUST »).

# 2. DESCRIPTION DES MECANISMES

Le standard d'interopérabilité entre organismes de la sphère sociale et l'administration dans le mode « Interops-R » repose sur deux principes :

- L'utilisation d'architecture REST pour les Web Services
- L'utilisation d'OAUTH 2.0 pour la sécurisation des flux [OAUTH2, OAUTH2Bearer].

Interops-R ne concerne pas le cas d'une intégration avec la Plate-forme d'état (France Connect). En effet, France Connect définit déjà les principes d'intégration.

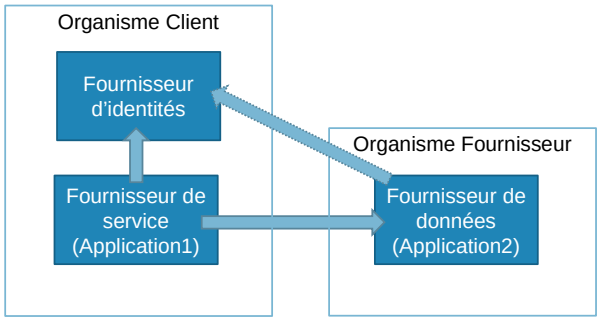
Un **Vecteur d'identification** (VI) est un ensemble de caractéristiques portant sur une application ou un utilisateur final.

Interops-R définit les rôles suivant :

- **Fournisseur d'identités :** entité authentifiant l'application ou l'utilisateur final. Il est chargé de produire un VI.
- **Fournisseur de service :** entité faisant appel au Fournisseur de données. Il peut s'agir par exemple d'une application cliente de type batch ou d'une application Web exposée aux utilisateurs finaux.
- **Fournisseur de données :** service exposé. Il vérifie et consomme le VI pour contrôler l'accès au service.

Interops-R définit le cas d'usage d'un échange entre un **Organisme Client** et un **Organisme Fournisseur**. Dans ce cas d'usage, les rôles sont répartis comme suit :

- les rôles de Fournisseur d'identités et de Fournisseur de service seront joués par l'Organisme Client
- Le rôle de Fournisseur de données sera joué par l'Organisme Fournisseur




Interops-R permet de répondre aux exigences émises par les organismes :

- Le modèle repose sur la confiance entre les organismes
- L'authentification de l'utilisateur ou de l'application n'est pas effectuée de bout en bout mais est réalisée par l'Organisme Client
- L'habilitation est attribuée par l'Organisme Client à ses utilisateurs ou applications clientes en respectant les règles établies avec l'organisme fournisseur (Convention)
- L'habilitation est transmise à l'organisme fournisseur de manière sécurisée (par un Vecteur d'identification)
- Toute création de vecteur d'identification est auditable afin d'en permettre le contrôle « a posteriori »

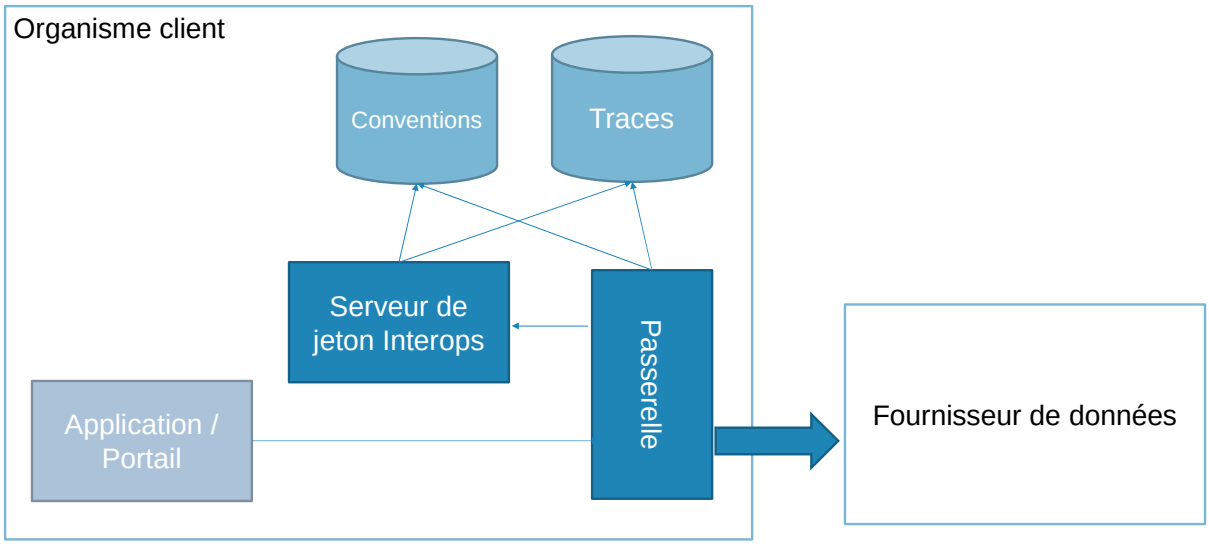
# 3. SPECIFICATIONS TECHNIQUES

## 3.1 Architecture fonctionnelle

Ce chapitre décrit l'architecture fonctionnelle dans le cas d'un échange entre un Organisme Client et un Organisme Fournisseur

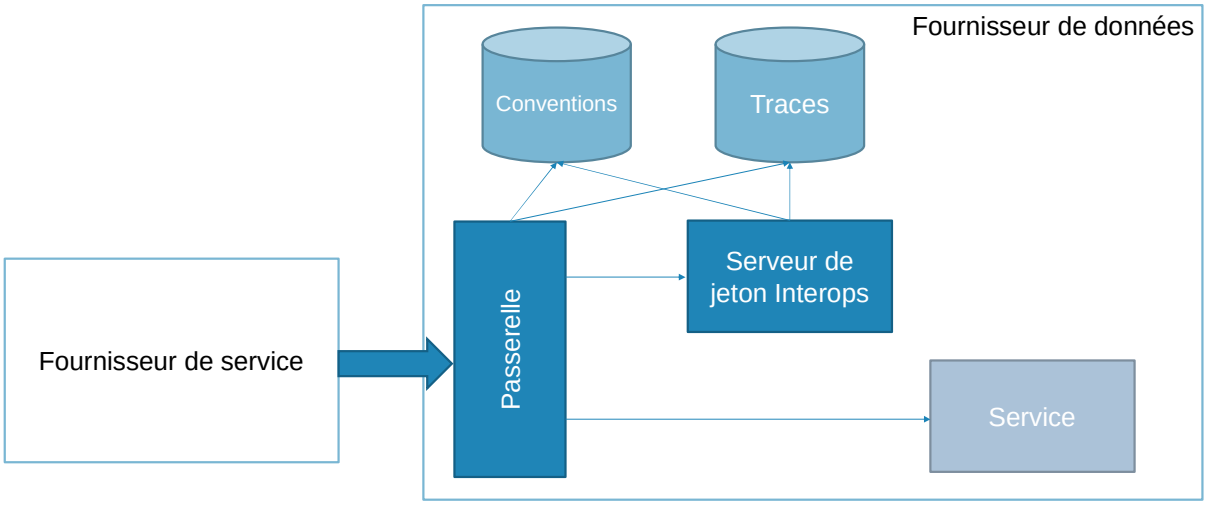
### 3.1.1 Organisme Client

Le schéma ci-dessous présente l'architecture fonctionnelle d'un Organisme Client jouant les rôles de Fournisseur d'identités et Fournisseur de service :




### 3.1.2 Organisme Fournisseur

Le schéma ci-dessous présente l'architecture fonctionnelle d'un Organisme Fournisseur jouant le rôle de Fournisseur de données :



## 3.2 Cinématique générale des échanges

Dans le cadre d'Interops-R, la cinématique générale suit les étapes suivantes :

1. **L'obtention d'un VI** auprès d'un Fournisseur d'identités
2. La **requête applicative** entre le Fournisseur de service et le Fournisseur de données
3. La **vérification du VI**

L'obtention du VI peut se faire auprès de France Connect ou d'un autre Fournisseur d'identités.

Cette version des spécifications détaille le cas d'échange entre un Organisme Client et un Organisme Fournisseur dans lequel :

- L'Organisme Client joue le rôle de Fournisseur d'identités et Fournisseur de service
- Le VI porte sur le Fournisseur de service

## 3.3 Obtention du VI

### 3.3.1 Principe

Ce chapitre décrit le cas où le VI porte sur un Fournisseur de service et non un utilisateur final. De ce fait, il est possible d'utiliser la cinématique « Client Credential Grant » d'OAUTH 2.0 ([OAUTH2]).

Dans cette version des spécifications d'Interops-R, il n'y a pas d'interaction avec un utilisateur final. Le cas où le VI porte sur un utilisateur final peut être fait au travers d'appels propriétaires d'un Fournisseur de service vers le Fournisseur d'identités qui sont hébergés au sein du même organisme. Le format de ces appels sort donc du périmètre de ce document.

Le diagramme ci-dessous présente une cinématique entre le Fournisseur de service et le Fournisseur d'identités.

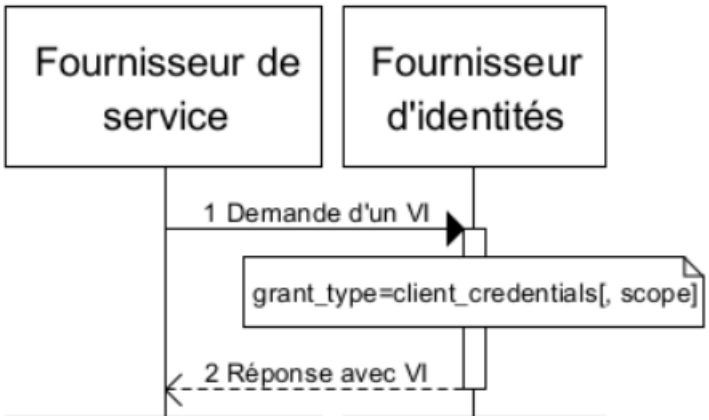



**Figure 1 : Obtention d'un VI**

1. Le Fournisseur de service s'authentifie auprès du Fournisseur d'identités pour demander un VI pour un ensemble de scope donné. Pour le format des scopes, se référer au paragraphe 3.8 p23.

2. Si le Fournisseur de service est correctement authentifié, le Fournisseur d'identités retourne une structure JSON contenant le VI

> ***Cette cinématique ne s'applique qu'à des Fournisseurs de service pouvant conserver un moyen d'authentification de manière sécurisée. En effet, le Fournisseur de service doit être en mesure de s'authentifier auprès du Fournisseur d'identités***
>
> ***Une application mobile ne pouvant conserver de manière secrète un moyen d'authentification, cette cinématique de génération de VI ne peut être utilisée dans le cas d'une application mobile.***

### 3.3.2 Cinématique détaillée

Ce paragraphe détaille :

- L'appel du Fournisseur de service
- La réponse du Fournisseur d'identités
- La gestion des erreurs éventuelles

#### 3.3.2.1 Demande de VI

La demande de VI s'effectue à l'aide d'une requête HTTP POST dont l'entête « Content-Type » est « application/x-www-form-urlencoded ». Paramètres et valeur du corps du message doivent être encodés en UTF-8.

Le tableau suivant donne les paramètres du corps du message :

| Paramètre | Description | Format | Exemple |
|---|---|---|---|
| `grant_type` | Définit le type de cinématique OAUTH | Chaîne de caractères sensible à la casse.<br>Dans ce cas, la valeur doit être « client_credentials ». | `client_credentials` |
| `scope` | Liste de scope demandé par le Fournisseur de service.<br>Ce paramètre est optionnel. | Liste délimités par des espaces de chaînes de caractère sensibles à la casse | `scope1 scope2` |

Le Fournisseur de service doit s'authentifier auprès du Fournisseur d'identités.

#### 3.3.2.2 Authentification du Fournisseur de service

L'authentification du Fournisseur de service peut se faire suivant plusieurs mécanismes :

- HTTP Basic
- Certificat
- Etc.

La méthode d'authentification du Fournisseur de service doit être définie par convention entre le Fournisseur de service et le Fournisseur d'identités.

Il est recommandé que les implémentations supportent à minima HTTP Basic.

Pour envoyer ses informations d'authentification, le Fournisseur de service utilise l'entête HTTP « Authorization ». La valeur de l'entête « Authorization » est construit comme suit :

1. Le nom d'utilisateur et le mot de passe sont concaténés en une chaîne séparés par le caractère ' :'. Par exemple : « Login:pwd »
2. La chaîne résultante est encodée en Base64 (sans passage à la ligne après 76)
3. La méthode d'autorisation (« Basic » et un espace sont alors mis avant la chaîne codée.

Ainsi, pour un login « Login » et un mot de passe « pwd », l'entête résultante est :

```
Authorization: Basic TG9naW46cHdk
```

#### 3.3.2.3 Réponse du Fournisseur d'identités

Le Fournisseur d'identité doit :

1. Authentifier la requête
2. S'assurer que les scopes demandés par le Fournisseur de service sont bien autorisés pour le Fournisseur de service.

Si la demande est autorisée, le Fournisseur d'identités retourne avec un statut HTTP 200 une réponse de type « application/json » contenant dans le corps du message une structure JSON. Cette structure JSON contient des éléments au plus haut niveau les paramètres décrit dans le tableau ci-après.

Le Fournisseur d'identité doit également utiliser les entêtes HTTP suivant pour que la réponse ne soit pas mise en cache :

- « Cache-Control » avec la valeur « no-store »
- « Pragma » avec la valeur « no-cache »

| Paramètre | Description | Format | Exemple |
|---|---|---|---|
| `access_token` | Le VI | Chaîne de caractères sensible à la casse.<br>Le format du VI est défini au §3.5.1 p15 | `mF_9.B5f-4.1JqM` |
| `token_type` | Type de token | Chaîne de caractères insensible à la casse<br>Doit être égal à « Bearer » | `Bearer` |
| `expires_in` | La durée de validité du VI en seconde | Nombre | `3600` |
| `scope` | Liste de scope inclus effectivement dans le VI. | Liste délimités par des espaces de chaînes de caractère sensibles à la casse | `scope1 scope2` |

Un exemple de réponse est donné ci-dessous :

L'ordre des paramètres n'a aucune importance. Tout paramètre non supporté par le Fournisseur de service doit simplement être ignoré.

La durée de vie du VI doit être déterminée par convention entre le Fournisseur d'identités et le Fournisseur de données.

Si le paramètre « scope » est absent de la demande, alors le Fournisseur d'identités peut retourner un VI avec un ensemble de scopes défini par défaut.

Dans le cas où le Fournisseur de scope demande des scopes qui ne font pas partie de sa convention, c'est-à-dire si ces scopes ne sont pas autorisés pour le client, le Fournisseur d'identités peut supprimer ces scopes du VI et laisser les scopes autorisés. S'il ne reste plus de scope à la fin de ce processus, le Fournisseur d'identités doit remonter une erreur de type `invalid_scope` (cf. paragraphe ci-dessous).

Afin de pouvoir générer le VI, la convention doit être déterminée sans ambigüité. 2 cas se présentent :

- Le Fournisseur de service ne fournit aucun scope dans sa demande.

  Dans ce cas, le Fournisseur de service ne peut être associé qu'à une et une seule convention et le Fournisseur d'identités retournera les scopes par défaut de la convention. Si le Fournisseur de service est associé à plusieurs conventions, une erreur « `invalid_request` » est retournée indiquant que des scopes doivent être fournis.
- Le Fournisseur de service fournit un ou des scopes dans sa demande.

  Dans ce cas, tous les scopes doivent être autorisés pour une et une seule convention.

#### 3.3.2.4 Gestion des erreurs

En cas d'erreur fonctionnelle, le Fournisseur d'identités peut retourner une réponse HTTP avec un statut 400 dont le type est « application/json » et contenant dans le corps du message une structure JSON avec au plus haut niveau les paramètres décrit dans le tableau ci-après.

| Paramètre | Description | Format | Exemple |
|---|---|---|---|
| `error` | Code d'erreur | Chaîne de caractères « lisible » en US ASCII.<br>Les valeurs possibles sont décrites ci-dessous. | `invalid_client` |
| `error_description` | Champ optionnel pouvant donner des indications plus précises concernant l'erreur | Chaîne de caractères « lisible » en US ASCII | `Please provide a valid grant_type, supported types are: authorization_code, client_credentials, password.` |

L'ordre des paramètres n'a aucune importance. Tout paramètre non supporté par le Fournisseur de service doit simplement être ignoré.

Les codes d'erreur pouvant être utilisés sont :

- `invalid_request` : paramètre manquant ou contenant une valeur invalide (en dehors du paramètre `grant_type`) ou paramètre répété ou requête contenant plusieurs paramètres d'authentification
- `invalid_client` : l'authentification du Fournisseur de service a échoué: login inconnu, méthode d'authentification incorrecte ou non supportée. Dans le cadre d'une authentification HTTP Basic, le Fournisseur d'identité doit retourner un code d'erreur HTTP 401 et préciser dans l'entête « WWW-Authenticate » la méthode d'authentification utilisée par le Fournisseur de service
- `invalid_grant` : Le paramètre `grant_type` est invalide
- `unauthorized_client` : Le fournisseur de service n'est pas autorisé pour cette cinématique
- `unsupported_grant_type` : cette cinématique n'est pas supportée par le Fournisseur d'identité
- `invalid_scope` : Les scopes demandés sont invalides, inconnus, mal formés ou non autorisés

## 3.4 Requête applicative

### 3.4.1 Principe

Lorsque le Fournisseur de service a obtenu un VI, il peut alors réaliser des requêtes applicatives au Fournisseur de données.

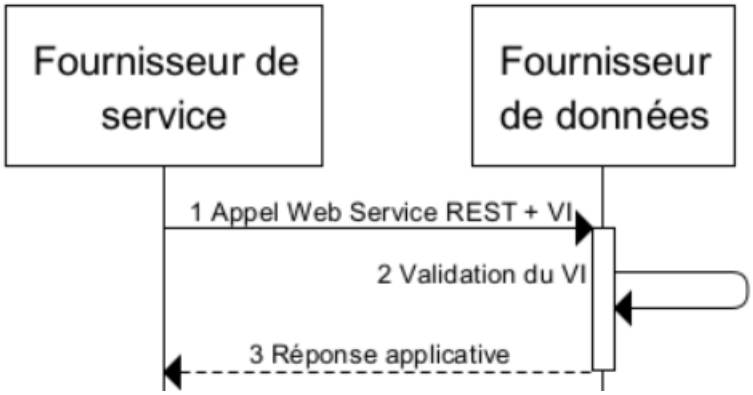



**Figure 2 : Requête applicative**

Le transport du VI dans la requête applicative est détaillé dans le paragraphe suivant.

La validation du VI par le Fournisseur de données est décrite au paragraphe 3.5 p15.

Le format de la réponse applicative sort du scope de ce document.

### 3.4.2 Transport du VI

Le standard OAUTH 2.0 [OAUTH2Bearer] spécifie 3 façons de transporter un jeton d'accès :

1. Au travers de l'entête « Authorization » de la requête
2. Dans le corps de la requête (ceci exclut le cas d'un GET HTTP)
3. Dans les paramètres d'URL de la requête

Dans le contexte Interops-R, le jeton d'accès correspond au VI.

Interops-R **exclut** l'utilisation de paramètres d'URL ou le passage dans le corps du message du VI.

Ainsi, un Organisme Client doit faire des requêtes applicatives authentifiées à un Organisme Fournisseur en passant le VI dans l'entête « Authorization » de la requête avec la méthode d'authentification HTTP « Bearer » tel que défini par HTTP [HTTPAuth].

La valeur de l'entête « Authorization » est exactement la concaténation de la chaîne « Bearer » (sans les guillemets), un espace et la valeur du VI. Pour se conformer au standard HTTP, le VI ne doit être composé que :

- De lettres
- De chiffres
- De caractères « - », « . », « _ », « ~ », « + », « / » et d'un éventuel caractère « = » terminal

Dans tous les cas, le format du VI est décrit dans le paragraphe 3.5 « Format du VI et validation » p15.

Un exemple non normatif est donné ci-dessous:

```
GET /resource HTTP/1.1
Host: server.example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

### 3.4.3 Gestion des erreurs

Ce paragraphe ne porte pas sur la gestion des erreurs applicatives mais sur la gestion des erreurs relatives au VI.

Si le Fournisseur de service tente d'accéder à une ressource protégée du Fournisseur de données sans VI ou avec un VI invalide, le Fournisseur de données doit répondre avec un statut 401 et un entête « WWW-Authenticate »(cf. [HTTPAuth]) avec pour challenge la valeur « Bearer » et les paramètre définis dans le tableau suivant :

| Nom du paramètre | O/R[1] | Description | Format | Exemple |
|---|---|---|---|---|
| `realm` | R | Définition d'un royaume tel que défini dans [HTTPAuth]) | Chaîne de caractères « lisible » en US ASCII. | `example` |
| `error` | O | Code d'erreur | Chaîne de caractères « lisible » en US ASCII.<br>Les valeurs possibles sont décrites ci-dessous. | `invalid_token` |
| `error_description` | O | Champ optionnel pouvant donner des indications | Chaîne de caractères « lisible » en US ASCII | `Please provide a valid` |

[1] Optionnel/Requis

| | | | | |
|---|---|---|---|---|
| | | | | `grant_type, supported types are: authorization_code, client_credentials, password.` |

Les codes d'erreur pouvant être utilisés sont :

- `invalid_request` : paramètre manquant ou contenant une valeur invalide ou VI soumis de façon répété ou de manière différente
- `invalid_token` : Le VI est invalide, expiré ou mal formé
- `insufficient_scope` : Le VI ne dispose pas des scopes nécessaires pour le service visé

Si la requête applicative ne contenait pas de VI, le paramètre « error » ne doit pas être utilisé.

Exemple de réponse pour un VI expiré :

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example",
                  error="invalid_token",
                  error_description="The access token expired"
```

## 3.5 Format du VI et validation

Le VI est au format JWT (cf. [JWT]). Le VI est alors signé et autoportant : le VI peut être validé par un Fournisseur de données sans appel au Fournisseur d'identités.

### 3.5.1 Format du VI

Dans le cadre d'Interops-R, Le VI est formaté suivant le standard JWT ([JWT]). JSON Web Token (JWT) est une structure JSON ([JSON]) qui est encodée dans une structure JWS ([JWS]) et/ou JWE ([JWE]), respectivement pour être signée et/ou chiffrée.

Le VI doit être obligatoirement signé. Il s'agira donc d'une structure JWS qui se décompose en 3 chaînes de caractères base64url-encodées :

- Un entête JOSE (JSON Object Signing and Encryption)
- La charge utile (structure JSON)
- La partie signature

#### 3.5.1.1 Entête JOSE

L'entête JOSE est une structure JSON au format clé/valeur permettant de définir :

- L'algorithme utilisé
- La clé utilisée
- Le type de contenu

Il se présente sous la forme d'une structure JSON avec au plus haut niveau les paramètres décrit dans le tableau ci-après.

| Nom du paramètre | O/R[2] | Description | Format | Exemple |
|---|---|---|---|---|
| `alg` | R | Algorithme de signature | Chaîne de caractères sensible à la casse.<br>Les valeurs supportées sont | `RS256` |
| `kid` | O | Identifiant de la clé de signature.<br>Sa valeur est donnée par convention. Cela peut être une suite de caractères décrivant la clé, une empreinte du certificat utilisé, etc. | Chaîne de caractères sensible à la casse | `rsa1` |
| `typ` | O | Type de contenu.<br>Cette revendication est optionnelle | Chaîne de caractères sensible à la casse.<br>La seule valeur possible aujourd'hui est « JWT ». | `JWT` |

#### 3.5.1.2 Structure JSON

Ce paragraphe décrit le contenu du VI correspondant à la charge utile de la structure JWS.

La structure JSON contient un ensemble de « clé/valeur » appelé **revendications** (claims en anglais) détaillées ci-après.

On pourra se reporter au paragraphe 6.1 p27 pour un exemple de VI conforme à la spécification.

[2] Optionnel/Requis

348

| Nom du paramètre | Description | Format | Exemple |
|---|---|---|---|
| `jti` | Identifiant unique du VI | Chaîne de caractères sensible à la casse.<br>Le format de l'identifiant doit suivre les recommandations de la RFC 4122 [RFC4122] afin d'assurer l'unicité de l'identifiant | `uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6` |
| `sub` | Identifiant unique de l'utilisateur.<br>Cet identifiant peut être impersonnifié ou non. | Chaîne de caractères sensible à la casse. | `NzbLsXh8uDCcd` |
| `iat` | Instant de génération du VI | Entier correspondant au nombre de secondes depuis la date de 1970-01-01T0:0:0Z jusqu'à la date et l'heure d'expiration en UTC | `1311281970` |
| `iss` | Identification de l'émetteur du VI, et donc de l'Organisme Client | Le format de l'identification est une URL sensible à la casse qui doit contenir :<br>• Le protocole (obligatoirement HTTPS)<br>• Le nom d'hôte<br>• Optionnellement un numéro de port<br>• Un chemin<br>L'URL ne doit contenir aucun paramètre ou fragment d'URL. | `https://oidc.cnaf.fr/` |
| `ver` | Numéro de version de l'accord d'interopérabilité | Chaîne de caractères permettant de représenter une version | `1.0` |
| `exp` | Date d'expiration du VI<br>La date d'expiration est dépendante de la durée de validité du VI et doit prendre en compte une dérive des horloges des systèmes de quelques minutes | Entier correspondant au nombre de secondes depuis la date de 1970-01-01T0:0:0Z jusqu'à la date et l'heure d'expiration en UTC | `1311281970` |
| `nbf` | Date de début de validité du VI | Entier correspondant au nombre de secondes depuis | `1311281970` |

| | | | |
|---|---|---|---|
| | La date de début de validité du VI doit prendre en compte une dérive des horloges des systèmes. La date de début de validité doit donc être légèrement avancée par rapport à la date d'émission | la date de 1970-01-01T0:0:0Z jusqu'à la date et l'heure de début de validité en UTC | |
| `auth_time` | Instant d'authentification de l'usager sur le SI.<br>Si le VI porte sur une application et non un utilisateur, cette revendication peut être omise | Entier correspondant au nombre de secondes depuis la date de 1970-01-01T0:0:0Z jusqu'à la date et l'heure d'authentification en UTC | `1311281970` |
| `acr` | Méthode d'authentification de l'utilisateur sur le SI de l'organisme client.<br>Si le VI porte sur une application et non un utilisateur, cette revendication peut être omise | Chaîne de caractères sensible à la casse décrivant le niveau eIDAS qui a été utilisé par l'utilisateur pour s'authentifier<br>Les valeurs possibles sont :<br>• eidas1 : niveau standard (exemple : authentification par identifiant / mot de passe)<br>• eidas2 : niveau substantiel (exemple : authentification 2-facteurs)<br>• eidas3 : niveau fort (exemple : carte à puce) | `eidas1` |
| `aud` | Identifiant du Fournisseur de service | Chaîne de caractères sensible à la casse | `https://rniam.cnav.fr` |
| `scp` | Liste des scopes (cf. §3.8 p23) | Chaîne de caractères sensible à la casse contenant la liste des scopes autorisés séparés par un espace. | `urn:cnaf:rise:1.0:read`<br>`urn:cnaf:rise:1.0:write` |
| `env` | Environnement pour lequel est prévu le VI | Chaîne de caractères sensible à la casse | `prod` |
| `azp` | Identifiant du service visé | URI identifiant le service exposé par le Fournisseur de données | `https://rise.cnaf.fr` |

349

### ➢ Attributs complémentaires

D'autres Attributs peuvent être ajoutées au VI (élément 10) sous la forme de nouvelles revendications. Dans ce cas, les revendications doivent être ajoutées à la structure JSON sous la forme clé/valeur en respectant le format JSON au plus haut niveau de la structure, c'est-à-dire au même niveau que les revendications standards définis ci-dessus. Ces revendications peuvent être typées : chaîne de caractères, entier, tableau, etc.

#### 3.5.1.3 Algorithmes

Les algorithmes de signatures qui doivent être supportés à minima sont :

- RS256 : RSASSA-PKCS1-v1_5 avec SHA-256)
- ES256 (ECDSA avec les courbes P-256 et SHA-256) sont recommandés

L'utilisation d'ES256 est recommandée car la taille de la signature et donc du VI est moins importante que RS256.

Sont exclus les algorithmes suivants :

- HS256
- None

### 3.5.2 Validation du VI

Les mécanismes suivant doivent être mis en œuvre pour la validation du VI par le Fournisseur de données. Si une étape de validation échoue, le VI doit être rejetée.

1. Vérifier le format du VI en s'assurant qu'il existe 2 caractères '.'
2. Extraire la première partie du VI, avant le premier '.' et décoder l'entête JOSE
3. Valider que l'entête JOSE décodé est bien une structure JSON, conforme à [JSON], encodée en UTF-8. Pour être conforme à [JWS], il est également requis que les paramètres en double ne soient pas autorisés.
4. Valider que l'entête JOSE comporte bien les paramètres décrits au §3.5.1.1. Dans le cas où le paramètre « `typ` » est présent, s'assurer qu'il a la valeur « JWT ».
5. Extraire la deuxième partie du VI, entre les 2 '.' et décoder la charge utile du VI
6. Valider que la charge utile décodée est bien une structure JSON, conforme à [JSON], encodée en UTF-8. . Pour être conforme à [JWS], il est également requis que les paramètres en double ne soient pas autorisés.
7. Récupérer la convention à partir du Fournisseur d'identités (paramètre « `iss` »), du Fournisseur de service (paramètre « `aud` »), le service visé (paramètre « `azp` ») et la version la version (paramètre « `ver` »).
8. Valider que le VI est bien destiné au Fournisseur de données (paramètre « `azp` »).
9. Valider que les scopes font tous partie de la même convention.
10. Valider que le VI est dans sa période de validité (paramètre « `exp` » et « `nbf` ») en tenant compte de la dérive d'horloge autorisée, donnée par convention.
11. Si le VI porte sur un utilisateur, valider le niveau d'authentification (paramètre « `acr` »).
12. Valider les scopes (paramètre « `scp` ») par rapport à la convention.

13. Valider que l'environnement pour lequel est prévu le VI (paramètre « `env` ») correspond à celui de la convention.
14. Valider que l'algorithme de signature utilisé (paramètre « `alg` » de l'entête JOSE) est autorisé dans la convention.
15. Valider la signature à partir de la clé fournie dans la convention. Si plusieurs clés sont disponibles pour la même convention, il est possible d'utiliser le paramètre « kid » pour identifier la clé utilisée pour la signature.

#### ➢ Rejeu du VI et mise en cache

Dans le cadre d'Interops-R, l'utilisation multiple d'un VI est autorisée. L'utilisation multiple d'un VI doit être définie par convention entre les Fournisseurs d'identité, de service et de données au même titre que la durée de vie du VI. Cette dernière doit être adaptée au besoin.

Ainsi, si un VI ne doit être utilisé qu'une fois, une durée de quelques minutes suffit.

Si le VI peut être utilisé plusieurs fois, une durée de vie allant de quelques minutes à quelques heures est possible en fonction des besoins métiers.

Si l'utilisation multiple est possible, le Fournisseur de données peut alors mettre en cache pour la durée de vie du VI:

- Le VI (s'appuyer simplement sur l'identifiant du VI ne serait pas suffisant dans ce cas)
- Le statut de la validation
- Les revendications

Il est de la responsabilité du Fournisseur de service de renouveler son VI avant son expiration. En effet, le Fournisseur de service connait la durée de validité du VI soit par convention soit au travers du paramètre `expires_in` qui est retourné en même temps que le VI.

#### ➢ Validation des scopes

La validation des scopes lors de la validation du VI porte sur le fait que les scopes sont bien ceux définis dans la convention.

Il appartient à l'application de vérifier que les scopes portés par le VI sont bien autorisé pour la requête applicative.

## 3.6 Sécurité des échanges

> ***Dans le cas où les échanges devront être sécurisés en utilisant des mécanismes conformes au Référentiel Général de Sécurité, les moyens cryptographiques utilisés devront suivre les préconisations contenues dans le contenues dans la dernière version du [RGS]. En particulier pour la version 2.0, les tailles de clés et algorithmes utilisés devront respecter [RGS_B_1] et les profils de certificats devront s'appuyer sur [RGS_A_4].***

### 3.6.1 Certificats X.509

Les scénarios de distribution des certificats n'entrent pas dans les spécifications du standard.

Néanmoins, chaque organisme devra être à même de vérifier la validité du ou des certificats de son partenaire.

La vérification d'un certificat comprend la validation des points suivants :

- La date de validité du certificat est correcte
- Le certificat a été émis par une chaîne de certification de confiance
- Le certificat n'a pas été révoqué
- L'usage du certificat correspond bien à l'emploi qui en est fait

### 3.6.2 Échanges entre les différents composants

Toutes les communications entre Organismes devront être protégées par TLS ([TLS12]) avec authentification serveur. Les communications seront alors protégées en confidentialité et en intégrité.

L'Organisme Fournisseur doit authentifier l'Organisme Client. Ceci peut être fait soit en réalisant une authentification mutuelle par certificat X.509 à la place de l'authentification serveur simple de la communication protégée par TLS soit en réalisant la communication entre l'Organisme Client et l'Organisme Fournisseur au travers d'un tunnel sécurisé de type VPN IPSec.

> ✍ ***L'authentification serveur est une fonction de sécurité faisant appel à des mécanismes cryptographiques qui peut nécessiter d'être conforme au RGS. Concernant cette conformité, se reporter à la note du paragraphe 3.4.3 p14***

### 3.6.3 Algorithmes

Pour garantir un niveau de sécurité suffisant, les implémentations doivent supporter au minimum (cf. [TLS12]). :

- TLS 1.2
- AES 128 bits ou 256 bits
- SHA-256

Pour des clés RSA, ceci correspond aux « ciphersuites » suivants :

- `TLS_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_RSA_WITH_AES_256_CBC_SHA256`

> ✍ ***Pour se conformer au RGS, TLS 1.2 est obligatoire.***

## 3.7 Principes REST

REST (representational state transfer) est un style d'architecture pour les systèmes hypermédia distribués, créé par Roy Fielding en 2000 dans le chapitre 5 de sa thèse de doctorat. Il trouve notamment des applications dans le World Wide Web. Il ne s'agit pas d'un protocole à proprement-parler.

Une architecture REST doit respecter les contraintes suivantes :

- Client-serveur : les responsabilités sont séparées entre le client et le serveur. L'interface utilisateur est séparée de celle du stockage des données. Cela permet aux deux d'évoluer indépendamment.
- Sans état : chaque requête d'un client vers un serveur doit contenir toute l'information nécessaire pour permettre au serveur de comprendre la requête, sans avoir à

dépendre d'un contexte conservé sur le serveur. Cela libère de nombreuses interactions entre le client et le serveur.

- Mise en cache : le serveur envoie une réponse qui donne l'information sur la propension de cette réponse à être mise en cache, comme la fraîcheur, sa date de création, si elle doit être conservée dans le futur. Cela permet à des serveurs mandataires de décharger les contraintes sur le serveur et aux clients de ne pas faire de requêtes inutiles. Cela permet également d'améliorer l'extensibilité des serveurs.
- Une interface uniforme ; cette contrainte agit selon quatre règles essentielles :
  - l'identification des ressources : chaque ressource est identifiée unitairement ;
  - la manipulation des ressources à travers des représentations : les ressources ont des représentations définies ;
  - un message auto-descriptif : les messages expliquent leur nature. Par exemple, si une représentation en HTML est codée en UTF-8, le message contient l'information nécessaire pour dire que c'est le cas ;
  - Hypermédia comme moteur d'état de l'application : chaque accès aux états suivants de l'application est décrit dans le message courant.
- Un système hiérarchisé par couche : les états de l'application sont identifiés par des ressources individuelles. Toute l'information n'est pas envoyée dans une seule ressource unique. Les requêtes/réponses entre le client et le serveur augmentent et donc peuvent faire baisser la performance d'où l'importance de la mise en cache, etc. Le bénéfice est que cela rend beaucoup plus flexible l'évolution du système.
- Code-On-Demand (facultatif) : la possibilité pour les clients d'exécuter des scripts obtenus depuis le serveur. Cela permet d'éviter que le traitement ne se fasse que du côté serveur et permet donc de faire évoluer les fonctionnalités du client au cours du temps. En revanche cela réduit la visibilité de l'organisation des ressources. Un état devient dépendant du client et non plus du serveur ce qui contredit la règle 2. Il faut donc être prudent en utilisant cette contrainte.

Le modèle de maturité de Richardson (Richardson Maturity Model) propose un fil conducteur pour passer d'un modèle RPC sur HTTP à un modèle RESTFul en appréhendant pas à pas les concepts sous-jacents.

Les niveaux du modèle de maturité sont :

- Le niveau 0 : Le RPC sur HTTP en POX (Plain Old XML). À ce niveau, on se contente d'utiliser HTTP comme système de transport pour interagir à distance avec un « service ».
- Le niveau 1 : L'utilisation de ressources différentiées. L'introduction des ressources permet de gérer la complexité du système cible, le Fournisseur de données
- Le niveau 2 : L'utilisation des verbes et des codes retours HTTP. En utilisant un jeu commun de verbes standards, il est possible d'homogénéiser la manipulation des différentes ressources.
- Le niveau 3 : L'utilisation des contrôles hypermédia. L'introduction de la notion de HATEOAS *(Hypertext As The Engine Of Application State*) permet d'adresser les derniers principes REST comme les transitions vers les états par des liens hypermédia mais aussi la « découvrabilité » des ressources (emplacements dynamiques des ressources, nouveaux liens, etc.)

> ✍ ***Si ces spécifications n'ont pas vocation à imposer un niveau de maturité pour un service REST s'appuyant sur Interops-R, il est recommandé de viser le niveau 2 dans le modèle de maturité de Richardson***

## 3.8 Format des scopes

Un scope est une chaîne de caractères sensible à la casse contenant des caractères imprimables ASCII, incluant des caractères comme '#', ':', '/', etc.[3]

Un Fournisseur de service fait la demande d'un VI pour un ou des scopes donnés. Les scopes matérialisent les habilitations attribuées à l'application cliente ou l'utilisateur final.

Le scope doit porter la notion de service visé, voire de version du service donné pour lever toute ambigüité sur le Fournisseur de données.

Ainsi, le Fournisseur de données peut s'assurer que le scope lui est bien destiné.

Il est donc recommandé de structurer les scopes sous formes d'URN ou d'URL comme suit :

- URN : <Identifiant du Fournisseur de données>:<Identifiant du Service visé>:<Version>:<Habilitation>
- URL : <Identifiant du Fournisseur de données>/<Identifiant du Service visé>/<Version>/<Habilitation>

  Il est possible de combiner l'identifiant du Fournisseur de service et Identifiant du Service visé>. D'autres séparateurs que le caractère '/' peuvent être utilisés comme le '.'

L'identifiant du Fournisseur de données peut être sous la forme d'un SIREN ou d'une chaîne de caractères non ambigüe.

Exemple d'URN :

```
urn:cnaf:rise:1.0:read
```

Exemple d'URL :

```
http://www.cnaf.fr/rise/1.0/read
```

## 3.9 Synchronisation temporelle

Pour faciliter le rapprochement des traces et limiter la déviation des horloges pour les périodes de validité des VI, les serveurs des organismes doivent se synchroniser sur un serveur NTP reconnu. Chacun communiquera alors le serveur NTP de référence choisi. S'il s'agit d'un serveur NTP interne, l'organisme devra préciser quelle méthode est utilisée pour synchroniser ce serveur (GPS, DCF77, etc.).

Dans certains cas, un serveur NTP pourra être mis à disposition par l'un ou l'autre des deux organismes.

Si la synchronisation temporelle des serveurs est obligatoire, le choix du serveur de temps est conventionnel.

[3] Il s'agit de l'ensemble de caractère %x21 / %x23-5B / %x5D-7E.

# 4. IMPACTS SUR LES TRACES

Les événements à tracer sont définis conventionnellement entre les organismes Client et Fournisseur.

Les événements liés à la génération et vérification d'un VI doivent être tracés respectivement par l'organisme Client et l'Organisme Fournisseur conformément au format d'échange des traces [R2].

## 4.1 Organisme Client

Afin de tracer l'ensemble des éléments relatifs à la connexion de l'utilisateur, l'opérateur doit tracer *a minima* :

- L'authentification de l'utilisateur
- La génération du VI

Afin de tracer l'ensemble des éléments relatifs aux requêtes d'une authentification cliente, l'opérateur doit tracer *a minima* la génération du VI

La trace d'une authentification de l'utilisateur final doit comporter les éléments suivants :

- Date de l'événement
- Identifiant local à l'opérateur d'authentification de l'utilisateur
- Méthode d'authentification
- Statut de l'authentification (succès et échec)

La trace de génération du VI doit comporter les éléments suivants :

- Date et heure de l'événement
- Identifiant unique du VI correspondant à l'attribut `jti` et l'identifiant de l'organisme client (attribut `iss`)
- Identifiant du service visé (attribut `azp`)
- Statut de la génération (échec ou réussite)

## 4.2 Organisme Fournisseur

Afin de tracer l'ensemble des éléments relatifs à la requête, l'Organisme Fournisseur doit tracer *a minima* :

- La réception et vérification du VI
- La transaction effectuée par l'application ou par un utilisateur

La trace de vérification du contexte applicatif doit comporter les éléments suivants :

- Date et heure de l'événement
- Identifiant unique du VI correspondant à l'attribut `jti` et l'identifiant de l'organisme client (attribut `iss`)

- Identifiant du service visé (attribut aud)
- VI vérifié, contenant la signature
- Statut de la vérification (échec ou réussite) avec éventuellement un détail en cas d'échec

La trace d'une transaction doit comporter les éléments suivants :

- Date de l'événement
- Identifiant local à l'Organisme Fournisseur de l'application cliente ou de l'utilisateur
- URL visée
- Action réalisée
- Statut de l'action réalisée (succès ou échec)

L'action réalisée peut contenir une description, le verbe HTTP utilisé, le code retour HTTP de la transaction, un identifiant de la ressource.

# 5. GESTION DE LA CONVENTION

Les éléments suivants doivent faire partie de la convention :

- Version
- Environnement
- Niveau d'authentification requis (eidas1, eidas2 ou eidas3)
- Scopes nécessaires
- Scopes par défaut
- Fournisseur d'identités
  - Identifiant du Fournisseur d'identités
  - Durée de vie du VI
  - Méthode de synchronisation de l'horloge
  - Algorithme de signature
  - URL pour la demande du VI
  - Méthode d'authentification (anonyme, HTTP Basic, etc.)
  - Attributs supplémentaires du VI
- Fournisseur de service
  - Identifiant du fournisseur de service
  - Authentification SSL mutuelle ou non
  - Adresses IP du ou des reverse-proxys
  - Dans le cas d'une authentification mutuelle, les informations de vérification du certificat :
    - Par émetteur et sujet
    - Chaîne de certification
    - Etc.
  - Méthode de synchronisation de l'horloge
- Fournisseur de données
  - Identifiant du service visé
  - Base URL exposée
  - Adresses IP du ou des reverse-proxys
  - Le certificat et la chaîne de certification
  - Dérive d'horloge autorisée
  - Méthode de synchronisation de l'horloge

# 6. ANNEXE

## 6.1 Exemple de VI

### 6.1.1 Structure encodée


```
eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IkNsZSBkXHUwMDI3ZXhlbXBsZSJ9.eyJz
dWIiOiJtci54QGNvbnRvc28uY29tIiwianRpIjoidXVpZDo1YmU5Y2U1Zi04MTAyLTRhMWQtOTczZ
C01OTIzNGM4MzlmNDMiLCJpYXQiOjE0NTgyMjQ5OTQsIm5iZiI6MTQ1ODIyNDkzNCwiZXhwIjoxND
U4MjI1Mjk0LCJpc3MiOiJodHRwczovL29pZGMuY25hZi5mci8iLCJ2ZXIiOiIxLjAiLCJhY3IiOiJ
laWRhczEiLCJhdWQiOiJodHRwczovL29pZGMuY25hZi5mci8iLCJzY3AiOiJ1cm46Y25hZjpyaXNl
OjEuMDpyZWFkIHVybjpjbmFmOnJpc2U6MS4wOndyaXRlIiwiZW52IjoicHJvZCIsImF6cCI6Imh0d
HBzOi8vcmlzZS5jbmFmLmZyIiwiYXV0aF90aW1lIjoxNDU4MjI0Mjg0fQ.n9vl6HUc15Aa0GmW1fH
4EIarJNTnEas_IRjorAs3SwAvVSTo1-96Orwab2NfkZgSQEZXb_DQaIrZJvIPzgp8ow
```


### 6.1.2 Charge utile décodée


```
{
  "sub": "mr.x@contoso.com",
  "jti": "uuid:5be9ce5f-8102-4a1d-973d-59234c839f43",
  "iat": 1458224994,
  "nbf": 1458224934,
  "exp": 1458225294,
  "iss": "https://oidc.cnaf.fr/",
  "ver": "1.0",
  "acr": "eidas1",
  "aud": "https://oidc.cnaf.fr/",
  "scp": "urn:cnaf:rise:1.0:read urn:cnaf:rise:1.0:write",
  "env": "prod",
  "azp": "https://rise.cnaf.fr",
  "auth_time": 1458224284
}
```