REPOBLIKAN'I MADAGASIKARA
Fitiavana - Tanindrazana - Fandrosoana

UNITE DE GOUVERNANCE DIGITALE

NORMES ET STANDARDS DE L'INTEROPÉRABILITÉ

Unité de Gouvernance Digitale - UGD

https://digital.gov.mg

Etat	Version	Date de la version
Validé	1	Août 2022

VERSIONS DU DOCUMENT

Version	Date	Auteur Fonction / Nom	Objet de la mise à jour
0.1	Juin 2021	HRA Architecte SI	Création
0.2	Février 2022	AMOA, PO, PMOs	chantier interopérabilité
0.3	Juin 2022	AMOA, PO	Mise en forme
1	Août 2022	AMOA, PO, PMOs	Prise en compte des retours venant des PMOs Harmonisation des affichages avec les autres documents de référence

DESTINATAIRES DU DOCUMENT

Diffusion pour action	Tous les services publics
Diffusion pour information	Secteur privé

VALIDATION DU DOCUMENT

Version	Valideurs/ Fonction/ Nom	Date	Valideurs/ Fonction/Nom	Date
0.1	Chief Digital Officer Architecte Urbaniste DGU			
1	Tous les PMOs (Semaine Interopérabilité Fianarantsoa)	Aout 2022		

DOCUMENTS DE RÉFÉRENCE

N°	Classement	Titre
1		Cadre d'interopérabilité
2		Manuel des standards des services
3		

LISTE DES ABREVIATIONS

AES : Advanced Encryption Standard
AMOA : Assistant à la Maîtrise d'Ouvrage
API : Application Programming Interface
AWS : Amazon Web Services
CA : Certification Authority
CNaPS : Caisse Nationale de Prévoyance Sociale
CSV : Comma-Separated Values
HTTP : HyperText Transfer Protocol
HTTPS : HyperText Transfer Protocol Secure
IPSEC : Internet Protocol Security
JSON : JavaScript Object Notation
JWT : JSON Web Token
MEF : Ministère de l'Economie et des Finances
MEN : Ministère de l'Education Nationale
MID : Ministère de l'Intérieur et de la Décentralisation
MINJUS : Ministère de la Justice
MINSanP : Ministère de la Santé Publique
MNDPT : Ministère du développement Numérique, de la transformation Digitale, des Postes et des Télécommunications
NTP : Network Time Protocol
OCSP : Online Certificate Service Protocol
OIDC : OpenID Connect
OSI : Open Systems Interconnection
PKI : Public Key Infrastructure
PMO : Partenaire de Mise en Oeuvre
PO : Product Owner
REST : REpresentational State Transfer
RGPD : Règlement Général sur la Protection des Données
RSA : Règlement Général sur la Protection des Données
SALM : Security Assertion Markup Language
SI : Système d'Informations
SOAP : Simple Object Access Protocol
SSL : Secure Sockets Layer
SSO : Single Sign-On
TLS : Transport Layer Security
TSA : Time Stamping Authority
UGD : Unité de Gouvernance Digitale
URI : Uniform Resource Identifier
VPN : Virtual Private Network
W3C : World Wide Web Consortium
WSDL : Web Services Description Language
XML : eXtensible Markup Language
X-ROAD SS : X-ROAD Security Server

Table des matières

Normes de développement des Services Web et des API's **6**

I.1. Introduction **6**

I.2. Définition des paramètres d'entrée/sortie. 6

I.3. Les bonnes pratiques. 6

I.4. Validation des messages. 6

I.5. Gestion des versions. 6

I.6. Gestion des erreurs. 7

I.7. Codage. 7

I.8. Fuseau horaire. 7

I.9. Fichier de services. 7

Normes de sécurité **8**

II.1 Introduction 8

II.2 Sécurité de la couche de transport. 8

II.2.1 TLS (Transport Layer Security). 8

II.2.2 VPN (Virtual Private Network ou réseau privé virtuel). 9

II.2.3. Recommandations générales. 9

II.3 Sécurité des données. 10

II.3.1 Authentification et autorisation. 10

II.3.1.1 Authentification de base 10

II.3.1.2 Authentification avec des certificats électroniques. 10

II.3.1.3 Authentification avec JWT (Jetons Web JSON). 11

II.3.1.4 Authentification et autorisation avec OAuth (OAuth 1.0a et OAuth2). 11

II.3.1.5. Authentification et autorisation avec OpenID Connect 12

II.3.1.6 Recommandations générales. 13

II.3.2 Intégrité. 14

II.3.2.1. Registre centralisé des hachages. 14

II.3.2.2. Signature électronique. 14

II.3.2.3 Horodatage. 15

II.3.2.4 Blockchain (Enchaînement de blocs). 15

II.3.2.5 Recommandations générales. 15

II.3.3 Confidentialité. 16

II.3.3.1 Cryptage symétrique. 16

II.3.3.2 Cryptage asymétrique. 17

II.3.3.3 Recommandations générales. 17
II.3.4 Le journal d'Audit. 17
Annexes. 18
III.1 Types de formats de représentation des données. 18
III.1.1. JSON (JavaScript Object Notation ou JavaScript Object Notation). 18
III.1.2. XML (langage de balisage extensible ou langage de balisage extensible). 19
III.2. Types de services Web et API's. 19
III.2.1. REST 19
III.2.1.1. Les bonnes pratiques. 20
III.2.1.2 Structures d'URI valides 20
III.2.1.3. Structures d'URI non valides 21
III.2.1.4. Éléments d'une liste 21
III.2.1.5. Filtrer la liste des éléments 21
III.2.1.6. Vérifier un élément 21
III.2.1.7. Pagination 22
III.2.1.8. Order 22
III.2.1.9. Limite de requête 22
III.2.1.9. Création d'un élément 23
III.2.1.10. Modifier un élément 23
III.2.1.11. Supprimer un élément 24
III.2.1.12. Validation des messages 24
III.2.1.12. Gestion des versions. 25
III.2.1.13. Gestion des erreurs. 25
III.2.1.14. Codage. 27
III.2.2. SOAP 27
III.2.2.1. Les bonnes pratiques. 27
III.2.2.2. Validation des messages 28
III.2.2.3. Gestion des versions. 29
III.2.2.4. La gestion des erreurs 29
III.2.2.5. Codage. 30
III.2.3 Verbes HTTP 30
III.2.4 Codes de réponse HTTP 31
Références 32

I. Normes de développement des Services Web et des API's

I.1. Introduction

Sur la base des normes et des meilleures pratiques utilisées à l'échelle internationale, l'utilisation des directives suivantes pour la construction de services Web et API[1]'s est suggérée.

I.2. Définition des paramètres d'entrée/sortie.

Il est recommandé que l'institution établisse les paramètres d'entrée et de sortie du service selon leur nature. Chaque paramètre sera clairement défini et décrira son objectif. Cette définition facilitera, plus tard, la génération de la documentation respective.

I.3. Les bonnes pratiques.

Lors de la mise en œuvre d'un service Web et d'une API, il est important de prendre en compte les expériences de mise en œuvre avec des résultats positifs qui aident à améliorer ou à résoudre les problèmes lors de la mise en service des services. La systématisation de ces expériences est ce qu'on appelle les bonnes pratiques.

Il est recommandé d'être cohérent dans la mise en œuvre des services Web et API's, en maintenant les bonnes pratiques dans tous les services en fonction de la technologie utilisée.

I.4. Validation des messages.

La validation des messages est effectuée pour vérifier que la structure des objets utilisés dans le service Web et l'API est correcte.

La validation des messages permet d'éviter des dysfonctionnements imprévus lorsque les clients consomment le service.

Il est recommandé de valider la structure, le format et la qualité des messages, à la fois des paramètres d'entrée et de sortie.

I.5. Gestion des versions[2].

Parfois, il est nécessaire d'avoir plusieurs versions d'un même service. Cela peut arriver lorsque le service présente de nouvelles caractéristiques de fonctionnement (ou en cas de besoin d'autres paramètres d'entrée/sortie).

---

[1] En informatique, API est l'acronyme d'Application Programming Interface, que l'on traduit en français par interface de programmation applicative ou interface de programmation d'application. L'API est une solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données.

[2] La gestion de versions (en anglais : version control ou revision control) consiste à gérer l'ensemble des versions d'un ou plusieurs fichiers (généralement en texte). Essentiellement utilisée dans le domaine de la création de logiciels, elle concerne surtout la gestion des codes source.

Il est toujours recommandé de versionner les services Web et API's pour éviter des désagréments aux consommateurs qui utilisent déjà le service. La gestion des versions doit être envisagée dès le début du déploiement.

Les changements ou modifications entre les versions sont documentés et publiés conformément aux dispositions de sa Documentation.

I.6. Gestion des erreurs.

Tous les problèmes présentés par les services Web et API's doivent être identifiés, collectés et retenus dans la piste d'audit / journal, afin que l'institution consommatrice (organisation cliente) interprète les raisons des échecs.

Il est recommandé que le message d'erreur comporte au moins un code et une description permettant de le comprendre clairement.

I.7. Codage.

Les institutions qui participent à l'échange de données à partir d'un service web ou d'un API (même s'il était dans une autre langue) doivent interpréter de la même manière les caractères utilisés pour la transmission des messages.

Pour cette raison, il est toujours recommandé d'utiliser l'encodage UTF-8[3] dans tous les services Web et API's.

I.8. Fuseau horaire.

Il est recommandé que tous les services Web et API's d'interopérabilité utilisent le même fuseau horaire, afin que toutes les institutions conçoivent les heures de la même manière. Pour notre cas GMT+3[4].

I.9. Fichier de services.

Il est nécessaire de documenter les services offerts par chaque institution. Pour chaque service, on doit détailler les données telles que l'url pour effectuer la consommation, les paramètres d'entrée et les données de sortie.

	Information technique	
Adresse physique du service	URL de consommation de service	

[3] UTF-8 (abréviation de l'anglais Universal Character Set Transformation Format - 8 bits) est un codage de caractères informatiques conçu pour coder l'ensemble des caractères du « répertoire universel de caractères codés », initialement développé par l'ISO dans la norme internationale ISO/CEI 10646, aujourd'hui totalement compatible avec le standard Unicode, en restant compatible avec la norme ASCII limitée à l'anglais de base, mais très largement répandue depuis des décennies.

[4] UTC+3 est un fuseau horaire, en avance de 3 heures sur le temps universel coordonné ou UTC.

	Information technique	
Type	Technologie utilisée pour obtenir des informations de service (SOAP[5], REST[6] ou autre)	Obligatoire
Environnement	Environnement de développement, de test, de production ou autre	Obligatoire
Données d'entrée	Détail des paramètres d'entrée du service ou des modificateurs	Obligatoire
Données de sortie	Détail de la structure de réponse du service	Obligatoire
Type de connexion	Moyens par lesquels les données du fournisseur des services sont transmises aux clients	Obligatoire
Logiciels associés	Liens logiciels liés au service	

II. Normes de sécurité

II.1 Introduction

La sécurité de l'information est la préservation de la confidentialité, de l'intégrité et de la disponibilité de l'information (D : Disponibilité, C : Confidentialité, I : Intégrité, P : Preuve); D'autres propriétés telles que l'authenticité, la responsabilité, la non-répudiation et la fiabilité peuvent également être impliquées.

II.2 Sécurité de la couche de transport.

Il est nécessaire d'avoir des protocoles de sécurité dans le canal, couche d'échange ou le support par lequel les informations seront transmises, ceci pour empêcher les autres de les voir ou de les modifier. Il existe différentes technologies à cet effet, telles que :

II.2.1 TLS (Transport Layer Security).

Le protocole TLS[7] permet l'identification et l'authentification des établissements publiques, des organisations ou entités au niveau du protocole de transport, garantissant que la communication est confidentielle et que les informations envoyées et/ou reçues sont complètes.

[5] SOAP (ancien acronyme de Simple Object Access Protocol) est un protocole d'échange d'information structurée dans l'implémentation de services web bâti sur XML.

[6] REST (representational state transfer) est un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web.

[7] La Transport Layer Security (TLS) ou « Sécurité de la couche de transport », et son prédécesseur la Secure Sockets Layer (SSL) ou « Couche de sockets sécurisée », sont des protocoles de sécurisation des échanges par réseau informatique, notamment par Internet.

Ce protocole fournit un moyen de communication sécurisé, son objectif principal étant la confidentialité des informations transférées. L'intégrité des informations échangées est obtenue en authentifiant les messages à chaque transmission.

Il est toujours recommandé d'utiliser TLS dans les processus d'interopérabilité qui nécessitent un certain type d'authentification et / ou de confidentialité, soit avec des clés obtenues auprès d'un organisme de certification, soit avec des clés autogénérées, car sa mise en œuvre est en général simple à réaliser, par rapport à d'autres les technologies.

II.2.2 VPN (Virtual Private Network ou réseau privé virtuel).

Un VPN est un canal de communication privé crypté. Il s'agit d'un environnement de communication où l'entrée est restreinte et autorisée uniquement pour les parties qui souhaitent communiquer ; c'est-à-dire que le canal ne peut pas être vu ou compris par des personnes extérieures.

Un VPN offre un canal de transmission sécurisé sur un réseau non sécurisé, avec les propriétés suivantes : confidentialité, intégrité et authentification. Toutes ces propriétés sont ajoutées à la propre sécurité du service Web (le service Web peut avoir une authentification et une autorisation).

Cette technologie est utilisée lorsqu'il est nécessaire de transmettre des informations sensibles, puisqu'il existe différentes technologies pour la mise en place de VPN, la connexion entre les produits de différents fournisseurs est souvent complexe.

Il est possible de créer un VPN en utilisant des mécanismes qui opèrent dans les premières couches du modèle OSI[8] comme IPSEC[9]. Ces tunnels permettent l'encapsulation et la connexion de deux réseaux physiquement séparés.
Parfois, il est inutile ou peu pratique de connecter deux réseaux, surtout si nous avons seulement besoin de partager un service Web entre institutions. Dans le cas le plus courant, un service Web peut être protégé et encapsulé à l'aide d'une identification mutuelle TLS.

Enfin, MPLS est une technologie d'étiquetage de trafic qui permet de créer des canaux privés sur un réseau public sans chiffrer les données, il est donc essentiel d'utiliser TLS.

II.2.3. Recommandations générales.

Il est toujours recommandé d'utiliser TLS pour protéger et partager des services Web. Cependant, si les données nécessitent un degré de sécurité plus élevé dans le transport, et qu'il est souhaitable d'avoir deux réseaux connectés entre les institutions, un VPN (IPSEC) peut être utilisé.

Il est important de prendre en compte que l'utilisation de ces technologies n'est pas exclusive, c'est-à-dire qu'un tunnel TLS peut être utilisé au sein d'un canal VPN/IPSEC.

---

[8] Le modèle OSI (de l'anglais Open Systems Interconnection) est une norme de communication, en réseau, de tous les systèmes informatiques. C'est un modèle de communications entre ordinateurs proposé par l'ISO (Organisation internationale de normalisation) qui décrit les fonctionnalités nécessaires à la communication et l'organisation de ces fonctions.

[9] IPSec (Internet Protocol Security) est un ensemble de protocoles (couche 3 modèle OSI) utilisant des algorithmes permettant le transport de données sécurisées sur un réseau IP.

II.3 Sécurité des données.

II.3.1 Authentification et autorisation.

Lors de la publication de données via un service Web, il est nécessaire d'identifier qui peut accéder au service, sauf dans les services Web accessibles au public. L'authentification est la vérification de l'institution de l'institution consommatrice (Organisation cliente), tandis que l'autorisation est la vérification des autorisations que ladite institution possède sur une ressource spécifique.

L'authentification doit toujours être effectuée en premier, puis l'autorisation. Cela doit être vérifié dans chaque demande de l'institution de consommation envers le service.

L'authentification et l'autorisation peuvent s'effectuer de différentes manières, dont les principales sont détaillées ci-dessous :

II.3.1.1 Authentification de base

L'authentification de base est la méthode la plus simple et utilise un nom d'utilisateur et un mot de passe pour identifier l'institution consommatrice.

Il est recommandé d'utiliser ce type d'authentification dans les services Web lorsque les données à échanger n'ont pas beaucoup de restrictions d'utilisation. De plus, ce type d'authentification permettra à l'utilisateur de l'établissement consommateur d'être identifié au minimum. A noter que ce type d'authentification peut être utilisé aussi bien avec un service web de type REST que SOAP.

Il est recommandé de mettre en place des procédures d'utilisation et de cycle de vie des mots de passe.

II.3.1.2 Authentification avec des certificats électroniques.

L'authentification par certificats est effectuée en demandant le certificat numérique du consommateur du service Web (client ou organisation cliente) et en vérifiant chaque message envoyé par rapport au dit certificat, ce qui garantit que le consommateur du service est bien celui qu'il prétend être. Ce mécanisme est fastidieux si l'on authentifie des personnes/navigateurs mais il est surtout utile pour authentifier un autre service web puisqu'il est intégré au protocole HTTPS[10] et assure ainsi l'identification, l'authentification, la confidentialité et l'intégrité. Ce mode est conçu comme une authentification TLS mutuelle

Si l'authentification avec des certificats numériques doit être mise en œuvre, il est nécessaire de vérifier que le certificat a été délivré par une autorité de certification de confiance (CA),

[10] HTTPS veut dire HyperText Transfer Protocol Secure (protocole de transfert hypertextuel sécurisé), et c'est la version chiffrée du HTTP. Il est utilisé pour sécuriser les communications sur internet ou sur un réseau. Le protocole de communication est chiffré au moyen de la TLS (Transport Layer Security) ou, auparavant, de la SSL (Secure Sockets Layer)

qu'il n'a pas expiré et n'a pas été révoqué (on peut utiliser OCSP[11] ou Online Certificate Service Protocol pour se faire).

II.3.1.3 Authentification avec JWT (Jetons Web JSON).

JWT est le principal moyen d'authentification dans les services de type REST, c'est un moyen compact et autonome d'envoyer des données en toute sécurité entre les parties au format JSON.

Un JWT peut être signé avec un algorithme symétrique (AES[12]) et asymétrique (RSA[13]).

Les JWT se composent de trois parties séparées par des points (.), qui sont : l'en-tête (en-tête), le corps (charge utile) et la signature (signature).

Le cryptage des données corporelles doit être envisagé s'il est considéré comme sensible, voir point (Confidentialité).

En raison de sa facilité de mise en œuvre lors de l'utilisation d'un service REST, il est toujours recommandé d'utiliser JWT, sans oublier que plusieurs fois il est nécessaire de révoquer un jeton, pour lequel il est suggéré qu'au moins cette implémentation supporte la révocation en plus du délai d'expiration du JWT.

II.3.1.4 Authentification et autorisation avec OAuth[14] (OAuth 1.0a et OAuth2).

Il existe deux versions : OAuth 1.0a et OAuth2, cette dernière étant la plus utilisée.

OAuth2 est un protocole d'autorisation et le mode d'authentification dépasse le cadre de la spécification, de sorte que tout processus d'authentification peut être mis en œuvre. Une grande majorité des implémentations de la norme proposent déjà des mécanismes d'authentification.

Dans OAuth2 il y a plusieurs rôles : le client (ou l'application qui essaie d'accéder aux informations de l'utilisateur), le serveur de ressources, le serveur d'autorisation (c'est celui qui demande à l'utilisateur s'il veut vraiment donner les permissions au client) et le Nom d'utilisateur. Il existe également plusieurs flux ou moyens d'obtenir un token, les principaux sont :

Octroi d'informations d'identification client (Client Credentials Grant) : Cette méthode est couramment utilisée pour les communications de serveur à serveur, le client envoie ses

[11] OCSP signifie Online Certificate Status Protocol. En français, “protocole de vérification de certificat en ligne”. Il s'agit d'un protocole de navigation qui permet de vérifier la validité d'un certificat SSL à l'aide d'une liste blanche.

[12] Advanced Encryption Standard ou AES, aussi connu sous le nom de Rijndael, est un algorithme de chiffrement symétrique.

[13] RSA est un système cryptographique, ou cryptosystème, pour le chiffrement à clé publique. Il est souvent utilisé pour la sécurisation des données confidentielles, en particulier lorsqu'elles sont transmises sur un réseau peu sûr comme Internet.

[14] OAuth est un protocole libre qui permet d'autoriser un site web, un logiciel ou une application (dite « consommateur ») à utiliser l'API sécurisée d'un autre site web (dit « fournisseur ») pour le compte d'un utilisateur.

informations d'identification au serveur d'autorisation et renvoie un jeton signé (le client et l'utilisateur sont les mêmes). Octroi de code d'autorisation (Authorization Code Grant) : couramment utilisé dans les applications Web lorsque le client souhaite accéder à des ressources protégées en faveur de l'utilisateur.

Le flux est le suivant : le client redirige l'utilisateur vers le serveur d'autorisation (Serveur d'identité SSO[15] / SAML[16]); puis il est demandé à l'utilisateur de saisir ses identifiants dans le serveur d'autorisation et d'approuver la demande du client ; puis le client avec l'autorisation déjà donnée par l'utilisateur négocie un jeton d'accès avec le serveur d'autorisation (c'est un appel en faveur d'un utilisateur, l'utilisateur et le client sont différents). [Keycloack[17], IdentityServer[18]]

OAuth est complexe à mettre en œuvre ; cependant, il se démarque lorsqu'on s'attend à disposer d'un grand nombre de services Web, car comme il s'agit d'un service centralisé, son administration est plus facile à appliquer à tous les niveaux.

Quel flux d'autorisation doit être utilisé ? :

II.3.1.5. Authentification et autorisation avec OpenID Connect

OpenID Connect[19] (OIDC) remplace OpenID2 et est un protocole qui augmente le protocole OAuth2 en ajoutant des données d'identité utilisateur stockées dans des jetons signés (JWT); cela lui permet d'offrir à la fois l'authentification et l'autorisation,

c'est pourquoi il est considéré comme plus complet.

OpenID Connect est basé sur le concept d'un fournisseur d'identité de confiance, qui est chargé de fournir un ensemble d'attributs qui identifient de manière unique les utilisateurs et qui permettent aux applications clientes de s'appuyer sur l'authentification pour vérifier un utilisateur. Cela élimine le besoin pour les services de connaître ou de stocker les informations d'identification de l'utilisateur final.

OpenID Connect utilise des messages JSON sur HTTPS et est recommandé par rapport aux versions précédentes d'OpenID qui ont été marquées comme obsolètes. Plus de détails sur la norme sont disponibles sur ce lien

---

[15] L'authentification unique, souvent désignée par le sigle anglais SSO (de single sign-on) est une méthode permettant à un utilisateur d'accéder à plusieurs applications informatiques (ou sites web sécurisés) en ne procédant qu'à une seule authentification.

[16] Security assertion markup language (SAML) est un standard informatique définissant un protocole pour échanger des informations liées à la sécurité. Basé sur le langage XML, SAML a été développé par OASIS.

[17] Keycloak est un logiciel à code source ouvert permettant d'instaurer une méthode d'authentification unique à travers la gestion par identité et par accès.

[18] IdentityServer est le framework de référence OpenID Connect et OAuth 2.0 pour .NET.

[19] OpenID Connect (OIDC) est une simple couche d'identification basée sur OAuth 2.0, un dispositif d'autorisation1. Ce standard est géré par la fondation OpenID.

	SAML 2.0	OAUTH 2.0	Connexion OpenID
Qu'est que c'est ?	Norme ouverte pour l'authentification et l'autorisation	Norme ouverte à autoriser	Norme ouverte qui ajoute l'authentification à OAUTH2.0
Histoire	Développé par OASIS en 2001	Développé par Twitter et Google en 2004	Développé par la Fondation OpenID en 2014
Utilisation principale	SSO d'application client/serveur	Autoriser les services Web / API	SSO pour les utilisateurs finaux
Format	SOAP/XML	SOAP / XML - REST / JSON	REST / JSON

II.3.1.6 Recommandations générales.

Si l'administration, l'entité ou une organisation ne dispose pas d'un mécanisme d'authentification et d'autorisation déjà défini, il est recommandé d'utiliser au minimum l'authentification mutuelle TLS avec JWT pour les services REST et BASIC pour SOAP (On a besoin donc d'une plateforme tel que X-Road[20] pour faciliter l'utilisation).

Si l'établissement dispose déjà d'un mécanisme d'authentification et d'autorisation (ex : X-Road, API Manager), il est recommandé de maintenir ce mécanisme et de n'effectuer que des actions de documentation et d'accompagnement de son fonctionnement.

La mise en œuvre d'OAuth2 ou d'OpenID Connect est recommandée à court et moyen terme. Il est suggéré d'utiliser un mécanisme d'authentification et d'autorisation centralisé pour les services Web, ceci afin d'éviter la complexité administrative à long terme.

Il est nécessaire de s'assurer que les fonctions administratives d'authentification et d'autorisation du service web ne sont accessibles que par les administrateurs du service et non par les consommateurs.

Pour REST, il est suggéré d'implémenter une couche d'échange sécurisée (X-Road security server). Cela permet de l'appliquer de manière transversale à tous les services Web (à la fois l'authentification, l'autorisation, signature des transactions et d'horodatage), en plus de fournir d'autres mécanismes de contrôle tels que les listes d'accès, les journaux, les limites de requêtes, etc.

Si la nature du service est transactionnelle, il est nécessairement recommandé d'appliquer un mécanisme d'autorisation (d'horodatage, non répudiation etc…). La plateforme X-Road supporte très bien ce genre de mécanisme d'autorisation.

[20] X-Road est une couche d'échange de données (Data Exchange Layer) entre des systèmes d'information. Les organisations peuvent échanger des informations par Internet à l'aide de X-Road pour garantir la confidentialité, l'intégrité et l'interopérabilité entre les parties prenantes de l'échange de données.

Il est recommandé de réaliser un inventaire des vulnérabilités techniques concernant la mise en œuvre du type d'authentification et d'autorisation adopté avant le déploiement en production. Ces vulnérabilités peuvent correspondre au type de cryptage utilisé, à la gestion des mots de passe, entre autres.

II.3.2 Intégrité.

Habituellement, les mêmes protocoles de transmission de données garantissent l'intégrité mais uniquement au niveau du transport (transport d'information), cela ne suffit pas toujours lors de la mise en œuvre d'un service Web, car il est seulement garanti que les informations n'ont pas été altérées dans le canal, bien qu'elles puissent avoir été altérées ultérieurement par d'autres processus.

Les services Web doivent garantir "l'exactitude" et "l'exhaustivité" des données en utilisant des mécanismes tels que la signature électronique, qui évitent l'altération des données à la fois dans le canal et dans le lieu où elles seront traitées ou stockées.

Les mécanismes qui garantissent l'intégrité des données sont détaillés ci-dessous :

II.3.2.1. Registre centralisé des hachages.

Une base de données centralisée peut être utilisée pour garantir l'intégrité des messages. Cela se fait en distribuant un résumé (hachage[21]) des données échangées à une base de données de confiance où elles sont stockées. Avec ces résumés, l'heure (horodatage[22]) est stockée afin de vérifier quand une demande a été faite.

Il est nécessaire que cette base de données soit accessible aux institutions participantes (sécurité serveur de X-Road, etc..) afin que chacun puisse en obtenir le soutien, s'il le souhaite.

Ce moyen de vérification de l'intégrité ne sera utilisé que si les institutions qui demandent l'échange de données établissent un accord entre elles (Contrat ou convention, ou également le respect de certaines procédures), où elles acceptent la validité dudit enregistrement.

II.3.2.2. Signature électronique.

La signature électronique associe le signataire à un document (un message en interopérabilité ou transaction) assurant l'authenticité, l'intégrité et la non-répudiation[23].

La signature électronique fait partie d'une infrastructure appelée PKI (Public Key Infrastructure ou Infrastructure à clé publique) qui permet d'identifier un certificat numérique

[21] Le hachage est la transformation d'une chaîne de caractères en valeur ou en clé de longueur fixe, généralement plus courte, représentant la chaîne d'origine.

[22] L'horodatage (en anglais timestamping) est un mécanisme qui consiste à associer une date et une heure à un événement, une information ou une donnée informatique

[23] La non-répudiation est l'une des propriétés de l'information considérées en cybersécurité (avec la confidentialité, l'intégrité et la disponibilité). Elle consiste en l'assurance qu'une action sur la donnée réalisée au nom d'un utilisateur (après authentification) ne saurait être répudiée par ce dernier. Tout comme la confidentialité et l'intégrité, la non-répudiation s'appuie sur des mécanismes de chiffrement.

avec l'institution d'une personne ou d'une institution. Avec ce mécanisme, il est possible de vérifier si les informations ont été modifiées, même si un seul caractère a été modifié.

Il est même possible d'avoir sa propre infrastructure PKI, si toutes les institutions concernées en acceptent la validité (CA privé, même open source).

Pour réaliser l'interopérabilité entre les parties, il est nécessaire qu'elles échangent les clés publiques, en conservant les clés privées en lieu sûr (ceci uniquement s'il s'agit de certificats auto-signés, car si les certificats sont générés par un Établissement Public de Certification, il est chargé de vérifier les clés).

Les certificats pour effectuer la signature électronique entre serveurs peuvent être obtenus de plusieurs manières. Ci-dessous nous détaillons les plus utilisés :

- ✔ Avec une Institution Publique de Certification (CA ou Certificate Authority) : l'institution auprès de laquelle on se rend pour obtenir des certificats numériques afin de les utiliser dans l'échange de données.
- ✔ Avec les certificats auto-signés : dans ce cas, les certificats sont générés par les institutions qui souhaitent échanger des données. L'échange de clés publiques peut s'effectuer par tout moyen entre les parties ; cependant, pour lui donner un caractère légal, l'échange de clés peut être effectué devant un notaire de foi publique.

II.3.2.3 Horodatage.

L'horodatage est utilisé pour spécifier le moment auquel la signature électronique a été appliquée. Généralement, l'horodatage fait partie de l'infrastructure PKI et l'autorité d'horodatage est appelée TSA (Time Stamping Authority).

L'horodatage indique que le contenu signé numériquement existait à un moment donné et n'a pas changé depuis. Sans l'existence d'un horodatage, il n'est pas possible de savoir quand la Signature électronique a été utilisée ou si elle a été appliquée avec un certificat valide au moment de la signature.

Il est suggéré d'utiliser l'horodatage en conjonction avec la signature électronique, car il permet de savoir quand les données ont été signées, si cela a été fait avec un mot de passe valide ou si l'information est récente.

II.3.2.4 Blockchain (Enchaînement de blocs).

La blockchain est un enregistrement public dans l'ordre chronologique des transactions qui ont lieu. Cette chaîne est partagée entre les utilisateurs, ce qui permet de vérifier qu'un événement s'est produit, garantissant l'intégrité. En d'autres termes, il s'agit d'un registre de transactions décentralisé et distribué, où chaque client peut avoir sa propre copie, ce qui évite les modifications par des tiers.

En interopérabilité, il pourrait être utilisé en générant un résumé (hash basé sur une fonction appelé hachage cryptographique qui est utilisé également par la plateforme ou technologie X-Road) du message qui contient les données à échanger et en insérant ce résumé dans la chaîne de blocs, qui lorsqu'il est distribué est très difficile à altérer.

II.3.2.5 Recommandations générales.

L'utilisation de la signature électronique est recommandée dans tous les services Web qui échangent des données sensibles ou données de l'état, compte tenu de la complexité que cela implique.

Si la signature électronique doit être utilisée, il convient d'établir des mécanismes sécurisés pour l'échange de clés publiques entre les institutions (si nécessaire) et le stockage des clés privées dans un endroit sûr, garantissant ainsi l'intégrité des données et empêchant les clés sont perdues en cas d'éventualité. Il est également nécessaire de vérifier que les certificats n'ont pas expiré et qu'ils n'ont pas été révoqués (nécessité d'utilisation de l'OCSP).

Dans SOAP, il est suggéré d'utiliser une signature XML selon le W3C[24] ou WS-Security[25]; dans REST, le JWS RFC 7515 (JSON Web Signature) pour la signature JSON.

II.3.3 Confidentialité.

Lorsque les données circulent d'un endroit à un autre, plusieurs fois par plusieurs intermédiaires, il faut éviter qu'elles soient comprises par des tiers ; de plus, il est nécessaire que ces données ne soient pas compréhensibles par les administrateurs d'infrastructure (ex : l'autorité de gouvernance) et autres ayant accès au même service web. La méthode la plus courante pour garder les données confidentielles est le cryptage (ex : il faut utiliser le protocole HTTPS).

Le cryptage est le processus de conversion des données dans un format qui n'est pas lisible par des tiers non autorisés (tiers observant le canal de transmission, administrateurs de services Web ou autres). Les données ne peuvent être comprises que par ceux qui ont la clé pour les déchiffrer.

Les méthodes utilisées pour le chiffrement sont symétriques et asymétriques, que nous détaillons ci-dessous :

II.3.3.1 Cryptage symétrique.

Le cryptage symétrique est effectué avec une clé secrète et il est sûr tant que cette clé est conservée en bonne garde. En interopérabilité, il suffit que les parties échangent la clé de manière sécurisée pour établir une communication avec un niveau de cryptage acceptable et à un coût de calcul (temps de traitement) assez faible.

L'algorithme le plus utilisé pour effectuer le chiffrement symétrique est AES (Advanced Encryption Standard ou standard de chiffrement avancé), qui est assez efficace et consomme peu de ressources lors du traitement. Il peut être utilisé avec des clés de longueur 128, 192, 256 ou d'un plus grand nombre de bits, la plus grande différence entre celles-ci étant le temps de décryptage des données.

---

[24] Le World Wide Web Consortium, abrégé par le sigle W3C, est un organisme de standardisation à but non lucratif, fondé en octobre 1994 chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML5, HTML, XHTML, XML, RDF, SPARQL, CSS, XSL, PNG, SVG, MathML et SOAP.

[25] WS-Security (Web Services Security) est un protocole de communications qui permet d'appliquer de la sécurité aux services web. Le 19 avril 2004, la spécification WS-Security 1.0 a été lancée par OASIS-Open.

L'un des principaux inconvénients de ce type de cryptage est l'échange de la clé, qui devient un risque lorsqu'elle n'est plus secrète (toute personne en possession de la clé peut déchiffrer les données).

II.3.3.2 Cryptage asymétrique.

Le chiffrement asymétrique utilise deux clés : une pour le chiffrement et une pour le déchiffrement. Une clé publique est utilisée pour appliquer le chiffrement et une clé privée est utilisée pour déchiffrer les données (le déchiffrement des données ne peut se faire qu'avec la clé privée).

Dans ce type de cryptage, il n'est pas nécessaire d'échanger les clés, évitant ainsi le problème de la distribution des clés.

L'utilisation de RSA avec une longueur de clé minimale de 2048 est recommandée (on utilise actuellement le longueur 4096 actuellement qui est déjà installé dans notre datacenter AWS[26]).

II.3.3.3 Recommandations générales.

En aucun cas, il n'est conseillé de mettre en œuvre notre propre algorithme de cryptage.

Il est recommandé d'utiliser le cryptage symétrique chaque fois qu'il n'est pas sur un canal sécurisé (TLS ou VPN) pour empêcher les tiers de comprendre les données envoyées.

Il est également recommandé d'utiliser un cryptage asymétrique si les données à échanger sont sensibles et ne peuvent être connues que par des utilisateurs spécifiques.

En général, si une sécurité des données plus élevée est requise, il est suggéré d'utiliser un cryptage asymétrique (RSA).

II.3.4 Le journal d'Audit.

Le journal d'audit est l'examen et la vérification des actions menées pour reconstituer une série d'événements qui ont généré un événement spécifique. Le moyen le plus courant d'effectuer cela consiste à utiliser des journaux d'événements sur des serveurs sécurisés et accessibles uniquement par le personnel autorisé.

Il est recommandé de toujours utiliser les journaux d'événements avec autant d'informations que possible (Principe de base de l'e-Gouvernance à propos de la transparence). Dans ce cadre, ils doivent contenir au minimum : des informations temporaires, des informations sur l'application qui enregistre l'événement, qui réalise l'événement (données client qui nécessite un système de journalisation), le type d'événement qui a eu lieu et la demande et la réponse du service.

Les informations client sensibles (mot de passe par exemple), les chemins de fichiers ou les chaînes de connexion à la base de données ne doivent pas être inclus dans les journaux d'événements, car cela constitue un risque de sécurité.

[26] Amazon Web Services (AWS) est une division du groupe américain de commerce électronique Amazon, spécialisée dans les services de cloud computing à la demande pour les entreprises et particuliers.

Les actions correctives qui ont été prises en cas d'événement affectant le service doivent être documentées dans des rapports, de sorte que si les mêmes événements se répètent, ils peuvent être facilement résolus.

Il est recommandé d'utiliser un centralisateur de journaux d'événements pour faciliter son administration et son analyse ultérieure (X-Road monitoring, ELK[27] qu'on a déjà proposé, Apache Kafka[28] ou autre).

On peut également effectuer le journal d'audit avec le registre centralisé des hachages si ce mécanisme a été mis en place; voir point (registre de hachage centralisé : déjà disponible dans X-Road).

De plus, des technologies de synchronisation d'horloge système (NTP) peuvent être utilisées afin de permettre un suivi chronologique des événements.

III. Annexes.

III.1 Types de formats de représentation des données.

Les principaux types de formats de données pour représenter les informations dans un service d'interopérabilité sont JSON et XML, et ils sont détaillés ci-dessous :

III.1.1. JSON (JavaScript Object Notation ou JavaScript Object Notation).

Il s'agit d'un format de représentation de données léger, simple à lire et à comprendre à la fois par les humains et les machines. C'est le principal type de format de données pour l'échange d'informations dans un service d'interopérabilité REST et, dans sa forme la plus simple, il se caractérise par l'utilisation d'une collection de paires nom/valeur pouvant être imbriquées. Sa structure est illustrée ci-dessous :

```
{
   "nom1": "valeur1", // paire nom/valeur
   "nom1": "valeur2",
   "nom3": {
      "nom4": "valeur4",
       "nom5": ["valeur1", "valeur2"] // paire nom/ valeur, dont les valeurs sont des collections
//(array)
   }
}
```

A titre d'exemple, l'objet « personne » est décrit au format JSON :

```
{
    " Id " : " 12345678-9 " ,
    " prenoms " : " Juan " ,
    " nom " : " Perez Gomez " ,
    " dateDeNaissance " : " 22/05/2017 "
```

[27] ELK est un outil d'analyse de logs composé de 3 logiciels open source, développés par la société Elastic : Elasticsearch, Logstach et Kibana.

[28] Apache Kafka est un projet à code source ouvert d'agent de messages développé par l'Apache Software Foundation et écrit en Scala.

}

III.1.2. XML (langage de balisage extensible ou langage de balisage extensible).

Il s'agit d'un format de représentation de données flexible qui définit un ensemble de règles pour former des documents compréhensibles par les humains et les machines. Ce format de type de données est généralement utilisé avec les services d'interopérabilité SOAP, bien qu'il soit également possible de l'utiliser avec un service d'interopérabilité REST.

La structure du document XML doit contenir la déclaration qui précise la version (version) et l'encodage (encodage) du document, ceci afin d'améliorer l'interopérabilité des données (connaître la version et l'encodage évite les erreurs d'interprétation).

A titre d'exemple, l'objet « personne » est détaillé au format XML :

```
<? xml version = " 1.0 " encodage = " UTF-8 " ?>
< personne >
   < Id > 012345678-9 </ Id>
   < nom > Juan </ nom>
   < prenoms > Perez Gomez </ prenoms >
   <dateDeNaissance > 22/05/2017 </ dateDeNaissance >
</ personne >
```

III.2. Types de services Web et API's.

Au moment de la rédaction, les types de services Web et API's les plus couramment utilisés lors de la mise en œuvre d'un processus d'interopérabilité sont REST et SOAP. Voici les définitions et les principales caractéristiques de chacun :

III.2.1. REST

REST est un style d'architecture pour les systèmes distribués et, bien qu'il ne dépende d'aucun protocole, il est principalement utilisé en dessus de HTTP. Ce style d'architecture définit un ensemble de principes pour sa mise en œuvre :

- Interface uniforme. Toutes les ressources sont identifiées par un URI[29].
- Interactions sans stockage d'état (stateless). Chaque message contient suffisamment d'informations pour être traité sans qu'il soit nécessaire d'enregistrer un état sur le serveur à partir des messages précédents.
- Permet l'utilisation du cache. Les réponses peuvent être mises en cache ou non par le client pour optimiser les requêtes.
- C'est une architecture client-serveur. Le serveur ne connaît pas les clients qui s'y connectent. Les deux parties peuvent se développer indépendamment.
- Architecture en couches. Le client ne sait pas s'il est connecté directement au service, à un cache ou à toute autre couche intermédiaire.

Le service basé sur ces principes est appelé RESTful.

[29] Un URI, de l'anglais Uniform Resource Identifier, soit littéralement identifiant uniforme de ressource, est une courte chaîne de caractères identifiant une ressource sur un réseau (par exemple une ressource Web) physique ou abstraite, et dont la syntaxe respecte une norme d'Internet mise en place pour le World Wide Web.

Bien que techniquement un service REST puisse transférer des informations dans n'importe quel format, il est recommandé d'utiliser JSON pour l'échange de données.

III.2.1.1. Les bonnes pratiques.

Pour maintenir une cohérence dans le développement des services REST, il est suggéré de mettre en œuvre les bonnes pratiques suivantes :

- Avoir une gestion des réponses de type JSON par défaut, et seulement si nécessaire, d'autres types de contenu (XML, CSV ou autre).
- Pour indiquer le format de réponse, utilisez le champ content-type.
- Par exemple :
    - XML : Content-Type : application/xml
    - JSON : Content-Type : application/json ;
    - Encoding character = utf-8
- Un URI (Uniform Resource Identifier) identifie une ressource ;
- Les URI sont des noms ou des noms au pluriel et en minuscules, pas des verbes.
- Utilisez les verbes du protocole HTTP car il gère les actions correspondant à CRUD[30] (GET, POST, PATCH, PUT, DELETE, ou autres).
- Mettez le numéro de version dans l'URL,
  par exemple : http://example-digital.gov.mg/api/v1.0/resource-name

- Utilisez la notation « chameau » (camelCase), qui consiste à écrire des phrases composées de sorte que chaque mot au milieu de la phrase commence par une majuscule (par exemple, phraseInCamelCase). Utilisez cette notation pour nommer les ressources, les attributs et les paramètres.

- Utilisez des mots et pas de verbes dans votre URL. Exemple : pour récupérer la liste des utilisateurs, utilisez /users et non /getusers

- Utilisez l'anglais pour nommer vos ressources. L'informatique est un monde principalement anglophone, si vous souhaitez fournir des API universelles, la langue de Shakespeare s'impose d'elle-même.

- Utiliser le pluriel pour nommer vos ressources. Lorsque l'URL se rapporte à un groupe de ressources (ici nous parlons bien de plusieurs utilisateurs et non pas d'un seul), le pluriel s'impose pour indiquer aux développeurs que l'URL se rapporte à un « tableau » de ressources.

III.2.1.2 Structures d'URI valides

- Liste des services :
    - GET http://www.digital.gov.mg /api/v1.0/services

[30] L'acronyme informatique anglais CRUD (pour Create, Read, Update, Delete) (parfois appelé SCRUD avec un "S" pour Search) désigne les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données.

- Liste des services de filtrage avec chaîne de requête :
    - GET http://www. digital.gov.mg /api/v1.0/services?annee=2016&order=desc
- Rechercher un service par identifiant (1234):
    - GET http://www.digital.gov.mg /api/v1.0/services/1234
- Recherchez les commentaires du service (1234):
- GET http://www.digital.gov.mg /api/v1.0/services/1234/commentaires
- Spécifiez les champs facultatifs dans une liste séparée par des virgules :
    - GET http://www.digital.gov.mg /api/v1.0/services/1234?campos=name,date
- Ajouter un commentaire à un service spécifique :
    - POST http://www.digital.gov.mg/api/v1.0/services/1234/commentaires

III.2.1.3. Structures d'URI non valides

- Noms singuliers : http://www.digital.gov.mg /service

  Exemple : http://www.digital.gov.mg /service/1234
- Verbe dans l'URI : http://www.digital.gov.mg /service/1234/create
- Filtre en dehors de la chaîne de requête http://www.digital.gov.mg /service/2016/desc

III.2.1.4. Éléments d'une liste

Lorsqu'on consomme cette ressource et n'ajoute pas de filtre de recherche à l'URL, le service doit lister tous les éléments appartenant à la ressource en cours de consommation. Afin de ne pas surcharger l'obtention des éléments, le résultat peut être limité en ajoutant une pagination par défaut au service, qui peut être modifiée via des filtres dans l'URL.

```
GET /services
# HTTP 200 OK
```

III.2.1.5. Filtrer la liste des éléments

En utilisant le "?" pour filtrer les ressources et "&" pour ajouter plus de filtres sur l'URL.

```
GET /services?{filtre1}=valeur&{filtreN}=valeur

# Fitre par ID de l'entité avec son état
GET /services?idEntite=2345&etat=enAttente

# HTTP 200 OK
```

III.2.1.6. Vérifier un élément

La requête d'un élément spécifique doit être effectuée en envoyant via l'URL le paramètre qui servira de clé ou d'identifiant de l'élément à consulter et aucun autre paramètre de filtre, de pagination ou de classement.

GET /services/{id}

Obtention de l'élément 12345 du service
GET /services/12345

HTTP 200 OK

III.2.1.7. Pagination

Pour les paginations (dans le cas d'une liste avec une grande quantité de données), il est suggéré d'avoir les paramètres "page" (numéro de page du premier élément à retourner par la ressource) et "elementsParPage" (quantité de données à retourner par page par ressource) dans l'URL.

Obtention des registes 61 - 90 (elementsParPage=30 parDefaut)
GET /services?page=3
HTTP 200 OK

Obtention des registres 21 - 30
GET /services?pagina=3&elementsParPage=10
HTTP 200 OK

Si la pagination est utilisée, un nombre par défaut d'**elementsParPage** doit être défini (le nombre normalement utilisé est de 30 éléments). De plus, il est suggéré d'envoyer éventuellement le lien d'en-tête dans la réponse avec des informations concernant la pagination (« first » la première page, « prev » la page précédente, « next » la page suivante et « last » la dernière page).

Lien :
< https : // www.digital.gov.mg /services?limite=20&interval=0 > ; rel = "first",
< https : // www.digital.gov.mg /services?limite=20&interval=40 > ; rel = "prev",
< https: // www.digital.gov.mg /services?limite=20&interval=80 > ; rel = "next",
< https: // www.digital.gov.mg /services?limite=20&intervalle=180 > ; rel = "last"

III.2.1.8. Order

Pour trier les ressources, utilisez le paramètre "**order**" suivi du nom de l'élément selon lequel on souhaite trier ; par défaut, les ressources sont triées par ordre croissant. Les éléments par lesquels il est permis de commander doivent être définis dans la documentation technique de la ressource API.

GET /services?order[element1]=asc|desc&order[elementN]=asc|desc
Tri par état ascendant et descendant fermés
GET /services?order[etat]=asc&[enregistrementFerme]=desc
HTTP 200 OK

III.2.1.9. Limite de requête

Dans le cas de traitement d'une limite de requêtes par client (rate-limiting), il est nécessaire d'envoyer un entête HTTP supplémentaire dans la réponse, avec des informations sur le nombre de requêtes autorisées, le nombre de requêtes restantes et l'heure (timestamp) dans lequel ces limites reviendront à leur état initial.

```
X-LimitSolicitudes-Limit: 100
X-LimitSolicitudes-Remaining: 14
X-LimitSolicitudes-Reset: 1499875025
```

III.2.1.9. Création d'un élément

La création d'un élément implique l'envoi d'un enregistrement ou d'un ensemble de ceux-ci conformes aux directives établies dans la documentation technique de l'API à laquelle ladite action est demandée. Le journal parcourt le corps de la demande et l'utilisation du format JSON est recommandée pour les valeurs d'entrée.

```
POST /services

# Corps de la requête
[{
   "nom": "Nom",
   "description": "Description de la registre"
}]

# HTTP 201 CREATED
```

Pour la validation des données d'entrée, l'utilisation de json-schema est recommandée. Dans le cas de la méthode POST, l'envoi des enregistrements à créer doit se faire dans un tableau qu'un ou plusieurs éléments soient créés.

III.2.1.10. Modifier un élément

Les modifications des enregistrements peuvent être effectuées via les protocoles PUT ou PATCH, la différence entre ceux-ci est que PUT met à jour tous les champs de l'enregistrement, tandis que PATCH permet de mettre à jour un ou plusieurs champs de l'enregistrement. Afin d'effectuer la mise à jour, il est nécessaire de fournir la clé ou l'identifiant du registre à modifier via l'URL.

PUT

```
PUT /services/{id}

# Corps de la requête
{
   "nom": "Nom",
   "description": "Description de la registre"
}

# HTTP 200 OK
```

PATCH

```
PATCH /services/{id}

# Corps de la requête
```

```
{
   "description": "Description de la registre"
}
```

HTTP 200 OK
Pour la validation des données d'entrée, l'utilisation de json-schema est recommandée.

III.2.1.11. Supprimer un élément

Pour supprimer un enregistrement, il est nécessaire de fournir la clé ou l'identifiant de l'enregistrement à modifier via l'URL.

```
DELETE /services/{id}
```

HTTP 204 NO CONTENT
Utilisez les codes HTTP correspondants pour répondre aux requêtes faites par le client.

Envisagez de chiffrer les paramètres envoyés dans l'URI s'ils sont sensibles de quelque manière que ce soit.

III.2.1.12. Validation des messages

Utilisez JSON Schema pour la validation des messages dans REST comme décrit par l' IETF16.

Le schéma JSON est également décrit au format JSON ; cependant, il est utilisé pour décrire la structure d'autres données. À la base, schéma JSON définit les types de données de base suivants :

- string : utilisé pour les chaînes de texte.
- number ou integer: utilisés pour définir des valeurs numériques, la principale différence étant que "number" accepte à la fois des nombres entiers et des nombres décimaux, tandis que "integer" n'est utilisé que pour valider des nombres entiers.
- object : utilisé pour valider les structures JSON imbriquées (une structure dans une autre).
- array : utilisé pour définir un ensemble d'éléments.
- boolean : ce type de données est utilisé pour vérifier deux valeurs spéciales, true (valeur vraie) et false (valeur fausse).
- null - Cette valeur spéciale est généralement utilisée pour représenter l'absence d'une valeur.

Il est recommandé que chaque objet identifié dans la sémantique ait son schéma JSON respectif associé et publié.

Par exemple, pour valider la structure JSON, utilisez :

```
{
    " titre " : " Personne " ,
    " type " : " objet " ,
    " proprietes " : {
```

```
        " id " : {
            " type " : " integer"
        },
        " nom " : {
            " type " : " string"
        },
        " prenom " : {
            " type " : "string "
        },
        " dateDeNaissance " : {
            " type " : " string "
        }
    }
}
```

Outils de développement de schéma JSON :

- Exemples de schémas : http://json-schema.org/examples.html
- Validateur de schéma en ligne : https://www.jsonschemavalidator.net/
- Validation de schéma JSON pour Java, Python, PHP, Ruby, etc. : http://json-schema.org/implementations.html#validators

III.2.1.12. Gestion des versions.

La gestion des versions est appliquée à l'URL. On peut avoir jusqu'à deux versions en même temps.

```
GET /v1/services
GET /v2/services
```

III.2.1.13. Gestion des erreurs.

Tous les problèmes que présentent les services d'interopérabilité doivent être identifiés, afin que l'entité consommatrice de ceux-ci comprenne les raisons des échecs.

Pour la gestion des erreurs, il est recommandé d'utiliser la structure JSON minimale suivante :

```
{
    " code " : " Code d'erreur" ,
    " erreur " : " Description détaillée de l'erreur "
}
```

En cas de nombreuses erreurs, la structure JSON suivante peut être utilisée :

```
{
   "code": "Code d'erreur",
   "error": " Description détaillée de l'erreur ",
   "errors": [
      {
         "code": " Identifiant du contexte de l'erreur ",
         "error": " Description détaillée de l'erreur "
      },
```

```
        {
            "code": " Identifiant du contexte de l'erreur ",
            "error": " Description détaillée de l'erreur "
        },
        ...
    ]
}
```

Par exemple, dans un service d'insertion de documents qui se fait en masse et qui a présenté une erreur, la réponse attendue est :

```
{
    " code " : " 0003 " ,
    " error " : " Erreur dans le traitement de l'insertion de masse des services " ,
    " errors " : [
        {
            " code " : 1568548 ,
            " error " : " Le service n'a pas l'attribut montant "
        },
        {
            " code " : 4894564 ,
            " error " : " Le montant de la prestation ne peut pas être négatif "
        },
        {
            " code" : 8598568 ,
            " error " : " Le compte ne correspond pas à l' entité "
        },
    ]
}
```

Ces codes d'erreurs doivent être retournés en réponse au client à l'aide des codes HTTP correspondants.

POST /service

Réponse d'erreur du serveur

```
{
    "code": "0003",
    "error": " Erreur dans le traitement de l'insertion massive de services ",
    "error": [
        {
            "code": 1568548,
            "error": " Le service n'a pas l'attribut montant"
        },
        {
            "code": 4894564,
            "error": "fr
        }
    ]
}
```

HTTP 400 Bad Request

III.2.1.14. Codage.

L'encodage dans l'en-tête "content-type" du protocole HTTP sera défini comme suit :

content-type: application/json; charset=utf-8

III.2.2. SOAP

SOAP est un protocole créé pour effectuer l'échange structuré de données dans un environnement décentralisé (tous les participants à la communication ne sont pas au même endroit). Il s'agit d'un protocole d'accès aux services basé sur des normes qui définit un ensemble de règles pour structurer les messages en XML. Il est indépendant du protocole de transport, du langage d'implémentation, de la plate-forme et du système d'exploitation.

Il est recommandé d'utiliser la version 1.2 de SOAP dans la mesure du possible.

III.2.2.1. Les bonnes pratiques.

Pour maintenir une cohérence dans le développement des services SOAP, il est suggéré de mettre en œuvre les bonnes pratiques suivantes :

- Les opérations et messages sont définis selon ce qui est détaillé dans le fichier WSDL[31] (Web Service Description Language) qui a la structure suivante :

```
<? xml version = " 1.0 " encodage = " UTF-8 " ?>
< wsdl : definitions  xmlns : wsdl = " http://schemas.xmlsoap.org/wsdl/ " >
        < wsdl : types >
                <! - définitions des types de données (XSD) ->
        </ wsdl : types >
        < wsdl : message >
                <! - définitions des données à échanger ->
        </ wsdl : message >
        < wsdl : portType >
                <! - ensemble d'opérations prises en charge ->
        </ wsdl : portType >
        < wsdl : binding >
         <! - spécification du protocole et du format de données pour un portType ->
        </ wsdl : binding >
        < wsdl : service >
                <! - collection finale de ressources (points de terminaison) ->
        </ wsdl : service >
</ wsdl : definitions >
```

Il est important de noter que ce WSDL est généralement généré automatiquement à partir du code du langage de programmation qu'on souhaite utiliser.

Le fichier WSDL doit être publié et doit être accessible par l'entité consommatrice.

[31] Le WSDL ou Web Services Description Language est une grammaire XML permettant de décrire un service web.

- Il existe différentes manières de traduire le WSDL en message SOAP, cela est défini dans les styles. Il est recommandé d'utiliser le style document à usage littéral (document/literal), puisque tout ce qui est dans le corps du message SOAP est défini dans le schéma, ce qui permet de le valider. Ce style est conforme à la spécification WS-I pour l'interopérabilité.

Les styles de document sont définis dans l'élément du fichier WSDL comme on peut le voir ci-dessous :

```
< wsdl : binding  name = " HelloWorldBinding "  type = " ns: HelloWorldPortType " >
< soap : binding  transport = " http://schemas.xmlsoap.org/soap/http "  style = " document " />
< wsdl : operation name  = " getHelloWorld " >
   < soap : operation  soapAction = " urn: getHelloWorld "  style = " document " />
      < wsdl : input >
         < soap : body  use = " literal " />
      </ wsdl :input >
      < wsdl : output>
         < soap : body  use = " literal " />
      </ wsdl : output >
   </ wsdl : operation >
</ wsdl : binding >
```

- Il est recommandé d'utiliser la notation "camel" (camelCase) pour nommer les opérations du service web.
- Les noms des opérations sont des verbes descriptifs qui désignent les actions à effectuer.
- Il est recommandé que le fichier WSDL ait des opérations liées à un seul objet.
- Dans la mise en œuvre d'un service d'interopérabilité SOAP, il est recommandé de ne pas maintenir un état entre les requêtes au service (stateless), ce qui lui permet d'être réutilisable par différents consommateurs.

III.2.2.2. Validation des messages

La validation de la structure XML et des contraintes de contenu se fait à l'aide de XSD (XML Schema Definition Language), qui est une recommandation du W3C. La version actuelle de cette recommandation est XML Schema 1.1, qui se compose de deux parties : XML Schema 1.1 Part 1 Structures et XML Schema 1.1 Part 2 Datatypes.

XSD est le langage de définition de schéma représenté par XML. Parmi ses principaux composants utilisés figurent : les définitions de types simples, les définitions de types complexes, la déclaration d'attribut et la déclaration d'élément.

Pour les restrictions sur le contenu des éléments XML, il existe les types de données les plus couramment utilisés :

- string : représente des chaînes de caractères.
- booléen : représente les valeurs logiques.
- dateTime : représente une instance de temps.
- décimal : représente un ensemble de nombres réels.

- base64binary : représente les données binaires codées en base-64.

Exemple XSD :

```
< xs : element name   = " personne " >
   < xs : complexeType >
      < xs : sequence >
         < xs : element name  = " id " type = " string " />
         < xs : element  name = " first name "  type = " string " />
         < xs : element  name = " last name "  type = " string " />
         < xs : element name  = " dateDeNaissance " type = " dateHeure " />
      </ xs : sequence >
   </ xs : complexeType >
</ xs : element >
```

Il est recommandé de séparer la définition des types de données du fichier WSDL, ce qu'on permet de rendre le fichier WSDL plus lisible ; travaillez séparément le fichier XSD (XML Schema Definition) et réutilisez les schémas et les espaces de nommage.

III.2.2.3. Gestion des versions.

Pour la gestion des versions dans SOAP, il existe trois options qui sont les plus utilisées :

- ✔ **Versionning de l'opération** : lorsqu'un service qui a plusieurs opérations veut en modifier une seule ; dans ce cas, il est recommandé d'appliquer la gestion des versions au nom de l'opération, ce qui évite des changements drastiques dans le WSDL.

```
< wsdl : operation name  = " operationV1 " > </ wsdl : operation >
< wsdl : operation name  = " operationV2 " > </ wsdl : operation >
```

- ✔ **Gestion des versions du service** : lorsqu'une entité modifie de nombreuses opérations dans un service, il est préférable de créer une nouvelle définition ; dans ce cas, un nouveau WSDL sera défini. La définition d'un nouveau WSDL implique qu'il y aura une nouvelle ressource finale (endpoint).

L'ajout de la propriété "version" au XML, de manière à ce que le consommateur puisse choisir la version dont il a besoin en fonction de la valeur envoyée dans le XML.

Il est recommandé de conserver le type de versionnage déjà utilisé, cependant pour les nouveaux services d'interopérabilité, il est suggéré d'effectuer le contrôle de version en ajoutant la propriété "version" au XML.

III.2.2.4. La gestion des erreurs

Tous les problèmes que présentent les services d'interopérabilité doivent être identifiés, afin que l'entité consommatrice du service comprenne la raison de l'échec. La structure XML suivante est recommandée pour la gestion des erreurs :

```
< response>
          < code > Code d' erreur </ code >
   < error > Description détaillée de l'erreur </ error >
</ response >
```

En cas de nombreuses erreurs, la structure XML suivante peut être utilisée :

```
< response >
            < code > Code d' erreur </ code >
            < error > Description détaillée de l'erreur </ error >
            < errors >
                        < error>
          < code > Identifiant du contexte d'erreur </ code >
          < description > Description détaillée de l'erreur </ description >
       </ error >
       < error >
          < code > Identifiant du contexte d'erreur </ code >
          < description > Description détaillée de l'erreur </ description >
       </ error >
       ...
    </ errors>
</ response >
```

Voici un exemple de gestion des erreurs dans SOAP :

```
< response >
   < code > 0003 </ code >
   < error > Erreur dans le traitement de l'insertion massive de procédures </ error >
   < errors >
      < error >
         < code > 15561561 </ code >
         < description > Le processus n'a pas l'attribut montant </ description >
      </ error>
      < error>
         < code > 4455621 </ code >
         < description > Le montant de la procédure ne peut être négatif </ description >
      </ error>
      < error >
         < code > 15267548 </ code >
            < description > Le compte ne correspond pas à l'entité </ description >
      </ error>
   </ errors >
</ response >
```

III.2.2.5. Codage.

On doit définir l'encodage dans l'en-tête du document XML comme suit :

```
<?xml version="1.0" encoding="UTF-8"?>
```

III.2.3 Verbes HTTP

Les verbes, ou méthodes HTTP, doivent être utilisés conformément à leurs définitions dans la norme 1.1/HTTP. Exemple d'utilisation des verbes HTTP pour créer, lire, mettre à jour et supprimer des opérations sur la ressource « services » :

Opération	Verbe HTTP	/service fournis	/fournis/1234
CRÉER	POST	Créer un nouveau ressource	Erreur
LIRE	GET	Obtenir la liste des ressources	Obtenir le roussource avec l'identifiant 1234
METTRE À JOUR	PATCH	Erreur	Mettre à jour le ressource avec l'identifiant 1234
TÉLÉVERSER UN OU DES FICHIER(S)	PUT	Téléverser un nouveau fichier	Mettre à jour ou remplacer le fichier avec l'identifiant 1234
SUPPRIMER	DELETE	Supprimer tous les ressources	Supprimer le ressource avec l'identifiant 1234

III.2.4 Codes de réponse HTTP

Bien qu'il existe un grand nombre de codes HTTP, les plus utilisés sont :

Code	Détail	Définition
200	Ok	Code de base du succès. Cela fonctionne pour les cas généraux. Utilisé en particulier dans une réponse GET réussie ou un contenu mis à jour.
201	Créé (Create)	Indique que la ressource a été créée. Il s'agit généralement de la réponse pour créer des requêtes PUT ou POST.
202	Accepté (Accepted)	Indique que la demande a été acceptée pour traitement. Il s'agit généralement de la réponse à un appel de traitement asynchrone.
204	Pas de contenu (No content)	La demande a réussi, mais il n'y a rien à montrer. Fréquemment envoyé après un DELETE réussi.
206	Contenu partiel (Partial content)	La ressource renvoyée est incomplète. Généralement utilisé avec des ressources paginées.
301	Rediriger définitivement (Redirect ou Moved permenantly)	L'URI demandé a été définitivement redirigé vers une autre ressource. Le consommateur doit acheminer ces requêtes vers un autre URI.

Code	Détail	Définition
302	Trouvé (Found)	L'URI demandé a été temporairement redirigée, le consommateur doit continuer à demander cette ressource à l'avenir.
400	Mauvaise demande (Bad request)	Erreur générale pour une demande qui ne peut pas être traitée (le consommateur ne doit pas répéter la demande sans la modifier).
401	Non autorisé (Unauthorized)	Indique que le consommateur n'a pas d'identité définie dans le service.
403	Interdit (Forbidden)	Indique que le consommateur a une identité définie dans le service, mais n'a pas les autorisations pour la demande qu'il a faite.
404	Pas trouvé (Not found)	La ressource demandée n'existe pas.
405	Méthode Non Autorisée (Method not allowed)	Soit la méthode n'est pas prise en charge, soit l'associé à cette ressource n'a pas l'autorisation.
406	Pas acceptable (Not acceptable)	La ressource n'existe pas dans le format demandé. Par exemple, une ressource est demandée en XML mais elle n'est disponible qu'en JSON.
500	Erreur Interne du Serveur (Internal server)	La requête semble correcte, mais un problème est survenu sur le serveur. Le client ne peut rien y faire.

Références

- Normes API de la Maison Blanche, États-Unis
- Normes API Gouvernement argentin
- Best Practices for Designing a Pragmatic RESTful API