

Architecture API de la Confédération

Classification	Non classifié
Statut	usage approuvé
Nom du programme	Stratégie numérique de la Confédération Initiative stratégique 3 « principe <i>once only</i> »
Responsable de l'initiative	Wüst Jürg, TNI
Version	1.0
Date	17 janvier 2022
Mandant	Transformation numérique et gouvernance de l'informatique (TNI)
Auteur	Michael Glavitsch, Detecon (Schweiz) AG
Groupes spécialisés	Groupe Architecture API de la Confédération, Conseil de l'architecture de la Confédération (ABB)
Approuvé par	Andreas Spichiger, TNI

Tableau des modifications

Version	Date	Modifications	Auteur
0.1	02.03.2021	Version initiale	Michael Glavitsch
0.2	26.03.2021	Première ébauche de la structure du contenu	Michael Glavitsch
0.3	05.05.2021	Version pour atelier du 18.05.2021	Michael Glavitsch
0.4	07.05.2021	Version pour atelier du 18.05.2021, avec remarques internes	Michael Glavitsch
0.5	23.06.2021	Version pour atelier du 29.06.2021	Michael Glavitsch
0.6	25.06.2021	Version pour atelier du 06.07.2021	Matthias Dyer Michael Glavitsch
0.7	07.07.2021	Recommandation de conception IAM, contexte, classification et objectif	Michael Glavitsch Jürg Wüst
0.8	21.07.2021	Formulation des contenus	Fabian Brüning Michael Glavitsch
0.9	23.07.2021	Traitement des résultats de l'examen interne	Michael Glavitsch
0.95	26.10.2021	Intégration des résultats de l'examen	Fabian Brüning Michael Glavitsch
0.96	01.12.2021	Assurance qualité après réception par l'ABB	Fabian Brüning Michael Glavitsch
0.97	02.12.2021	Modifications après séance du 02.12.2021 ChF TNI Detecon	Michael Glavitsch
1.0	17.01.2021	Version validée	Andreas Spichiger

Table des matières

1	Management Summary.....	1
2	Contexte	3
2.1	Origine du mandat	3
2.1.1	Motions parlementaires	3
2.1.2	Administration numérique suisse (Cyberadministration suisse)	3
2.2	Place dans la stratégie numérique de la Confédération 2020-2023 (ancienne Stratégie informatique)	4
2.2.1	Aperçu de la stratégie numérique de la Confédération 2020-2023.....	4
2.2.2	Domaine d'action C2 Fournir des portails et des interfaces.....	5
2.2.3	Objectif C2-1 Fournir des portails et des interfaces.....	5
3	But de l'architecture API de la Confédération	5
4	Bases légales et champ d'application	6
4.1	Notions	6
4.2	Partenaires	6
4.3	Intermédiaires	7
4.4	Types d'API	7
4.5	Processus d'intégration	8
4.6	Cas d'intégration API.....	10
4.7	Architecture API de la Confédération, OGD et I14Y.....	12
5	Structure.....	13
5.1	Structure de l'architecture API de la Confédération	13
5.2	Architecture de référence API.....	14
6	Principes architecturaux.....	15
7	Architecture d'entreprise.....	19
7.1	Capacité API management	19
7.1.1	Aperçu.....	19
7.1.2	Description des capacités d'entreprise	22
7.2	Recommandation de conception Transparence	25
7.2.1	API documentation management	26
7.2.2	API cataloging & searching	26
7.2.3	API publication & revocation	28
7.3	Recommandation de conception <i>API monetization</i>	29
7.3.1	Cadre juridique	29
7.3.2	Modèles de monétisation	29
7.3.3	Modèles tarifaires	30
7.3.4	Conditions.....	30
7.4	Description des rôles	31
7.4.1	Rôles du fournisseur.....	31
7.4.2	Rôles du partenaire	32
7.5	Gouvernance API.....	32
7.5.1	Modèle de gouvernance API.....	32
7.5.2	Développement de l'architecture API de la Confédération	33
8	Architecture de données.....	34
8.1	Modèle d'information.....	34
8.2	Description des objets d'information	37
8.3	Recommandation de conception <i>API Design</i>	38
8.3.1	Exigences d'entreprise	38
8.3.2	Conventions d'échange de données	38
8.3.3	Traitement des erreurs	41
8.4	Recommandation de conception <i>Identity & access management</i>	41
8.4.1	Introduction.....	41
8.4.2	Accès à l'API par jeton sécurisé	42
8.4.3	Architecture cadre IAM de la Confédération.....	42

8.4.4	Formes d'architecture API pertinente pour l'IAM	44
8.4.5	Single sign on	46
9	Architecture d'application.....	47
9.1	Système de l'architecture API de référence	47
9.1.1	Aperçu	47
9.1.2	Infrastructure API	47
9.1.3	Instances API.....	48
9.2	<i>API gateway</i> et <i>message exchange pattern</i>	48
9.3	Description des composants système par le mappage des niveaux d'architecture	49
9.4	Recommandation de conception Intégration API	51
9.5	Recommandation de conception <i>API versioning</i>	52
9.5.1	Versionnage des types d'objets livrés et des normes spécifiques aux API	52
9.5.2	Versionnage des <i>API releases</i>	52
9.5.3	Degrés de liberté	53
9.5.4	Représentation technique du numéro de version	53
9.5.5	Migration forcée.....	54
9.5.6	<i>API gateway</i> et service spécialisé	54
10	Annexe.....	55
10.1	Planification basée sur les capacités selon TOGAF	55
10.2	Description des objets d'information	56
10.2.1	<i>Manage API lifecycle</i>	56
10.2.2	Discover APIs	57
10.2.3	Register client	57
10.2.4	Operate API.....	57
10.2.5	Analyse API operation & usage.....	58
10.3	Conventions d'échange de données	59
10.3.1	Types d'interface	59
10.3.2	Formats de données.....	59
10.3.3	<i>Message exchange patterns</i> (MEP)	60
10.3.4	Types de messages	61
10.3.5	Protocoles API.....	61
10.4	<i>Resource description framework / linked data</i>	62
10.5	Table des abréviations	63
10.6	Table des figures	64
10.7	Table des tableaux	64

1 Management Summary

L'objectif de l'architecture API de la Confédération est de normaliser et de promouvoir l'accès numérique aux services publics dans l'environnement fédéral afin que les entreprises, l'administration et les personnes puissent bénéficier de la communication de machine à machine. Cet objectif correspond à une demande des milieux politiques et économiques et aux exigences de la stratégie numérique de la Confédération 2020-2023. La vision correspondante peut être formulée comme suit :

« Nous sommes une administration moderne qui simplifie l'accès de ses partenaires aux services publics en les rendant utilisables par de nombreux moyens électroniques. »¹

La vision s'articule autour de plusieurs dimensions présentées dans le tableau 1, sur lesquelles le groupe convoqué pour élaborer l'architecture s'est appuyé dans ses réflexions. L'architecture a été établie par des représentants de nombreux départements et de nombreuses UA sous la conduite du secteur TNI, ce qui lui confère une large assise dans l'environnement fédéral.

Dimension	Vision
Utilité	Il faut encourager l'accès numérique aux services publics (ouverts) dans le cadre de la communication M2M pour les entreprises, l'administration et les personnes. Les machines qui accèdent aux services publics numériques peuvent se présenter sous forme d'application locale, de service spécialisé ou d'application de portail. Il faut gérer les API, les faire connaître et les réutiliser à chaque fois que l'occasion se présente.
Groupe cible	Le groupe cible est constitué d'architectes d'entreprise, d'architectes informatiques, de spécialistes, de gestionnaires et de personnes issues des affaires et de l'informatique en recherche d'orientation qui soutiennent des projets visant l'utilisation numérique des services publics, qu'ils travaillent pour l'administration fédérale ou pour des partenaires externes.
Champ d'application	L'architecture API de la Confédération est un cadre réglementaire qui donne des recommandations pour le développement et l'utilisation d'API visant le traitement numérique des démarches administratives dans l'administration fédérale. Il s'agit en particulier de recommandations de conception pour les API et la gestion des identités et des accès, et de la tenue d'un registre des API. La conception des architectures de solution doit être largement libre de restrictions.
Principes architecturaux	Les principes architecturaux donnent des lignes de conduite pour le développement et le fonctionnement des API employées pour les services publics.
Normes et bonnes pratiques	L'architecture API repose sur des normes reconnues (par ex., eCH, Tallinn, EIF, W3C) et sur les bonnes pratiques dans l'environnement fédéral. Elle est revue périodiquement.

Tableau 1 Dimensions de la vision

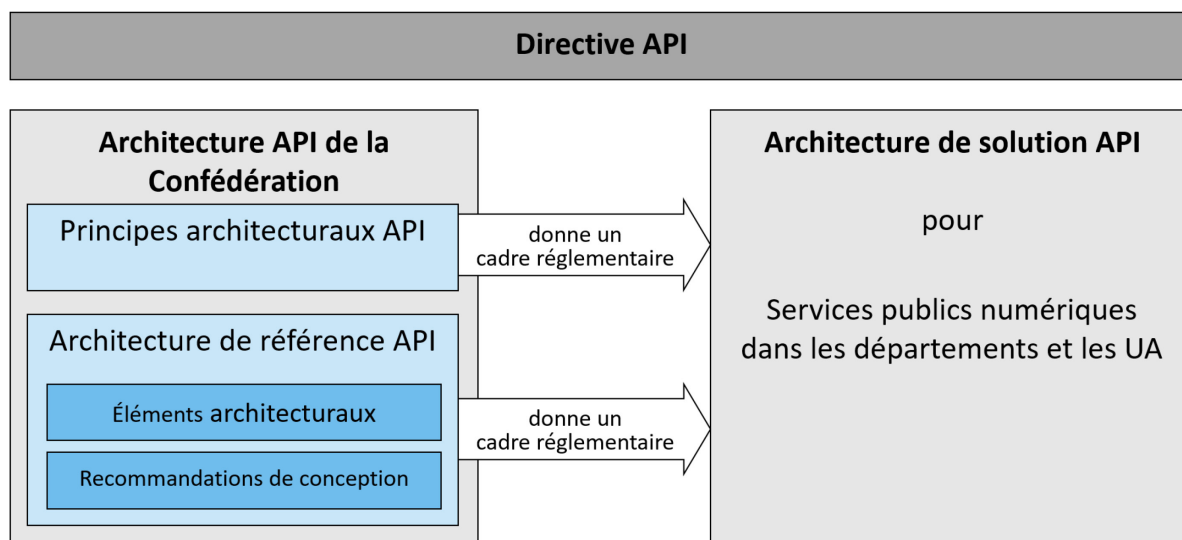


Figure 1 Position de l'architecture API de la Confédération

¹ Vision de l'architecture API de la Confédération

L'architecture API de la Confédération comprend une série de principes architecturaux et une architecture de référence inspirée des domaines architecturaux (BDAT) prévus par la norme TOGAF² ; toutefois, seuls les niveaux métier, données et applications y sont décrits et aucune recommandation n'est formulée pour l'architecture technologique. Ces principes et cette architecture offrent un cadre réglementaire et un langage commun en vue de la conception des architectures de solution API permettant un accès numérique aux services publics dans les départements et les UA (voir figure 1).

L'architecture API de la Confédération donne des recommandations et constitue une aide pour l'élaboration de solutions d'architecture API. Des directives indépendantes de l'architecture peuvent toutefois conférer un caractère contraignant à certains de ses éléments.

Pour décrire l'accès numérique aux prestations des autorités dans le contexte de la communication M2M, l'architecture API de la Confédération utilise le terme de services publics numériques. Un service public numérique peut reposer sur une seule API ou sur un ensemble d'API. Le terme API est utilisé pour désigner les deux.

Les principales caractéristiques de l'architecture API de la Confédération sont les suivantes.

- La Confédération permet à ses partenaires d'obtenir des services publics numériques sur la base des dispositions de la loi fédérale sur l'utilisation des moyens électroniques pour l'exécution des tâches des autorités (LMETA)³. Conformément à la LMETA, les départements et les UA publient les métadonnées relatives aux API qui relèvent de leur compétence dans un registre des API.
- D'après la planification basée sur les capacités selon TOGAF, l'architecture API de la Confédération définit les capacités d'entreprise requises pour le développement et l'utilisation des services publics numériques.
- Ces capacités sont associées à la capacité principale *API management* et attribuées à l'une des cinq étapes de processus : *manage API lifecycle*, *discover API*, *register client*, *operate API* et *analyse API operation & usage*. Les capacités spécifiques à l'API représentent quant à elles des sous-capacités de l'*API management*. S'il existe déjà des capacités non spécifiques à l'API, l'*API management* les utilise.
- L'architecture API de la Confédération repose sur une architecture axée sur les services grâce à laquelle des services spécialisés peuvent appeler des clients API au moyen de passerelles API (*API gateway*). Si les services publics numériques à intégrer dans les services spécialisés ne disposent pas encore d'API, il faut en mettre en place pour rendre leurs prestations accessibles.
- Afin de respecter le principe *once only* (voir ch. 2.2.1) et de proposer des services publics de la Confédération entièrement numériques, les services spécialisés peuvent eux-mêmes intégrer d'autres services spécialisés au moyen d'API. On parle alors d'« intégration d'API dans le service spécialisé de la Confédération ». Mais il est aussi possible de mettre en place l'« intégration d'API dans le client API du partenaire ». C'est ce qui est fait lorsque le service public de la Confédération ne représente qu'un sous-service numérique du partenaire.
- Pour développer et fournir des services publics numériques, il est prévu de centraliser ou de fédérer pour tout le pays certaines capacités des étapes de processus *discover API* et *register client*, en mettant en place, par exemple, un registre central des API ou l'utilisation d'un service standard IAM central.
- Dans l'architecture API de la Confédération, la question de la sécurité des données et de l'information (SIPD) n'est pas abordée. Dans l'environnement fédéral, cette question est réglée dans des bases juridiques et de prescriptions générales, qui viennent compléter le cas échéant des dispositions prises à l'échelon des départements ou des offices. L'architecture relève de l'OTNI⁴.

² <https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap02.html>

³ FF 2022 805 (la LMETA a été publiée dans la Feuille fédérale le 06.04.2022 mais elle n'est pas encore en vigueur, état le 01.08.2022)

⁴ <https://www.bk.admin.ch/bk/fr/home/digitale-transformation-ikt-lenkung/ikt-vorgaben/grundlagen.html>

2 Contexte

2.1 Origine du mandat

Les interfaces électroniques (API) constituent la base des services publics que l'administration met à disposition sous forme numérique. C'est pourquoi le Conseil fédéral a chargé l'administration fédérale de proposer ses services sous cette forme dans le cadre de la stratégie numérique de la Confédération 2020-2023 (voir ch. 2.2).

L'accès numérique aux données et aux services de l'administration fédérale s'effectue au moyen de portails de cyberadministration ou d'application d'humain à machine (H2M), ou alors directement par une interaction M2M en passant par des API. Les connexions M2M doivent être considérées comme des canaux d'accès aux services publics numériques qui peuvent être utilisés également par les portails de cyberadministration et les applications (H2M).

Ces dernières années, l'administration fédérale a mis en place de nombreux portails de cyberadministration (EasyGov.swiss, portail électronique du DFF, portail de cyberadministration du DETEC, portail Agate, etc.). À présent, les milieux politiques et économiques et la société civile demandent de plus en plus la création de possibilités de connexion directe M2M. Quelques départements et UA ont déjà franchi le pas. Les principaux facteurs de développement de l'architecture API de la Confédération sont présentés ci-après.

2.1.1 Motions parlementaires

Au total, trois motions du Parlement demandent la mise en place ou le développement d'interfaces électroniques.

Il s'agit de la motion 18.4276 « Faciliter l'échange d'informations en créant des interfaces électroniques au sein de l'administration fédérale », déposée par le conseiller aux États Beat Vonlanthen et de la motion 18.4238 « Mettre en place des interfaces électroniques au sein de l'administration fédérale pour simplifier l'échange d'informations », déposée par le conseiller national Franz Grütter. Ces deux motions réclament la possibilité d'échange de données direct entre les entreprises et les personnes physiques. La motion 20.4260 déposée par la Commission des finances du Conseil national complète les deux précédentes et demande ce qui suit : « Des interfaces en temps réel (microservices et interfaces de programmation d'applications, API) interopérables, lisibles par machine et basées sur des normes ouvertes doivent permettre d'améliorer les échanges sous forme numérique entre les autorités fédérales et les autorités d'autres niveaux de l'État, l'économie et la société civile. »

Le Parlement et le Conseil fédéral ont accepté les trois motions. L'administration fédérale est chargée de leur mise en œuvre.

2.1.2 Administration numérique suisse (Cyberadministration suisse)

Dans l'Agenda Infrastructures nationales et services de base, l'Administration numérique suisse présente notamment l'ambition ci-après en ce qui concerne la transformation numérique de l'État fédéral :

« Exploiter pleinement le potentiel d'automatisation et de simplification au profit de l'économie : **le fardeau administratif de l'économie diminue, grâce à l'échange automatisé des données et aux interfaces avec l'administration.** Les normes, les infrastructures et les bases institutionnelles nécessaires à cet effet sont créées avec cohérence sur le plan suisse d'ici à 2026. »

L'objectif de mise en œuvre 1 de la stratégie suisse de cyberadministration 2020-2023, « Développer le portail EasyGov.swiss », prévoit de mettre en place une architecture fédérale des portails. Le projet, dirigé par le SECO, est accompagné par un groupe de représentants de l'administration fédérale et de diverses administrations cantonales. L'architecture fédérale mettra l'accent sur l'interopérabilité entre les différents portails qui lui seront rattachés. Les API joueront un rôle essentiel, car elles serviront d'interface entre les portails et les services spécialisés.

2.2 Place dans la stratégie numérique de la Confédération 2020-2023 (ancienne Stratégie informatique)

2.2.1 Aperçu de la stratégie numérique de la Confédération 2020-2023

La stratégie⁵ définit des domaines d'action qui s'articulent autour des quatre axes stratégiques suivants.

- A. Gestion des informations, des données et des processus
- B. Gestion de l'innovation et du changement
- C. Focalisation sur les clients et sur les services
- D. Coopération entre la gestion des affaires et l'informatique

Ces axes sont coordonnés avec la stratégie Suisse numérique, la stratégie nationale de protection de la Suisse contre les cyberrisques, la stratégie suisse de cyberadministration et le rapport final du DFF et de la CdC sur le projet « Administration numérique ».

L'architecture API de la Confédération est élaborée dans le cadre du domaine d'action C2 - Fournir des portails et des interfaces. Ce domaine fait partie de l'axe C, défini comme suit.

But et effet

Les prestations de l'administration reposent sur les principes de la légalité et de la compétence. Il s'ensuit une division du travail au sein de l'administration qui peut être en contradiction avec l'optique du client. L'axe C aborde cet aspect et doit permettre d'atteindre les buts fixés en termes d'innovation dans les domaines des services et des processus.

Intention

L'axe C (focalisation sur les clients et sur les services) vise à

- asseoir la transformation numérique sur le service à la clientèle, et partant sur une réflexion qui aille de l'extérieur vers l'intérieur plutôt que le contraire ;
- harmoniser les interactions numériques de toute l'administration fédérale avec les cantons, les communes, les organisations internationales ou étrangères, les associations et plus particulièrement les entreprises et les particuliers ;
- décharger les organisations informatiques et les utilisateurs des questions d'infrastructure et de plateforme afin de libérer des ressources humaines et financières pour la transformation numérique de la gestion des affaires.

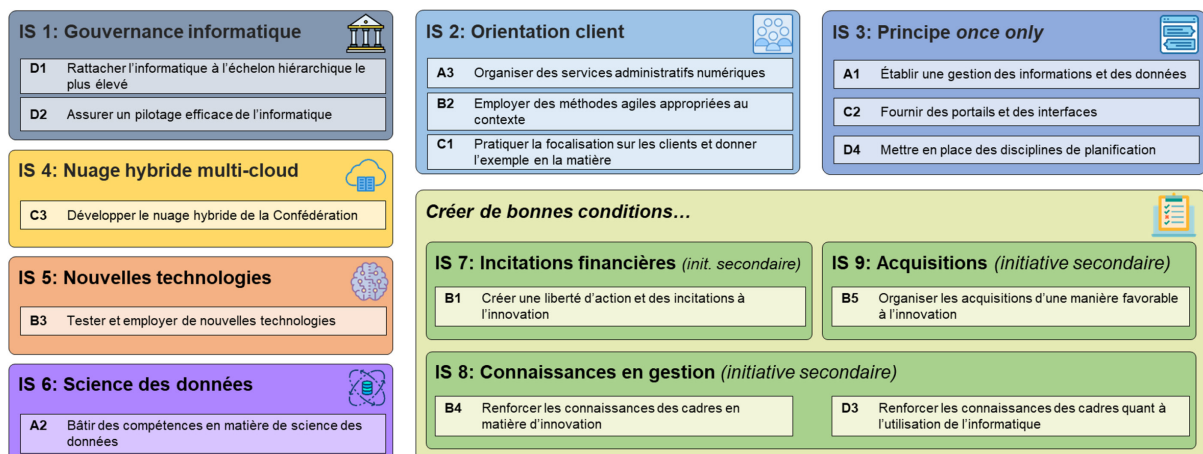


Figure 2 Aperçu des initiatives de la stratégie numérique de la Confédération

⁵ [Stratégie informatique de la Confédération 2020 à 2023 et plan directeur](#)

2.2.2 Domaine d'action C2 Fournir des portails et des interfaces

Les services numériques de l'administration peuvent être fournis par le biais de portails ou d'interfaces électroniques, en fonction de l'utilisateur et du type de prestation. Pour les services de l'administration qui impliquent un grand nombre de transactions récurrentes, les interfaces logicielles offrent une plus grande plus-value pour le client que les portails.

Un portefeuille⁶ des portails et des interfaces pouvant être exploités conjointement sera piloté et géré à l'échelon de l'administration fédérale, en fonction de l'utilité des services fournis par ce biais. Pour plus de clarté, un inventaire centralisé des services électroniques proposés par les autorités, des groupes d'utilisateurs⁷ correspondants et des responsabilités sera établi. Les UA définiront quels services seront proposés via les portails ou les interfaces, et pour quels groupes d'utilisateurs, et formuleront leurs exigences à l'égard de ces outils.

2.2.3 Objectif C2-1 Fournir des portails et des interfaces

Les autorités fournissent leurs services au moyen de portails (communication entre l'homme et la machine) et d'interfaces électroniques (communication de machine à machine). Des portails et des interfaces sont exploités en commun afin de pouvoir fournir les services plus rapidement, à moindres frais et à moindres risques.

3 But de l'architecture API de la Confédération

« Nous sommes une administration moderne qui simplifie l'accès de ses partenaires aux services publics en les rendant utilisables par de nombreux moyens électroniques. »⁸

Conformément aux objectifs définis dans la stratégie numérique de l'administration fédérale et aux exigences des milieux politiques, l'architecture API de la Confédération se concentre sur la mise à disposition de services publics numérique pour les partenaires de l'administration fédérale. Il faut soit rendre les services accessibles directement au moyen d'API (M2M), soit faire de l'API un service répondant aux critères d'interopérabilité sur la plateforme de cyberadministration d'une application de portail conforme à l'architecture fédérale des portails.

Au sein de l'administration fédérale, l'élaboration de l'architecture API est accompagnée par un groupe de représentants de nombreux départements et de nombreuses UA (FP et BP). Le groupe a développé le modèle cible et la vision. Le tableau 2 montre les dimensions de la vision.

Dimension	Vision
Utilité	Il faut encourager l'accès numérique aux services publics (ouverts) dans le cadre de la communication M2M pour les entreprises, l'administration et les personnes. Les machines qui accèdent aux services publics numériques peuvent se présenter sous forme d'application locale, de service spécialisé ou d'application de portail. Il faut gérer les API, les faire connaître et les réutiliser à chaque fois que l'occasion se présente.
Groupe cible	Le groupe cible est constitué d'architectes d'entreprise, d'architectes informatiques, de spécialistes, de gestionnaires et de personnes issues des affaires et de l'informatique en recherche d'orientation qui soutiennent des projets visant l'utilisation numérique des services publics, qu'ils travaillent pour l'administration fédérale ou pour des partenaires externes.
Champ d'application	L'architecture API de la Confédération est un cadre réglementaire qui donne des recommandations pour le développement et l'utilisation d'API visant le traitement numérique des démarches administratives dans le contexte de l'administration fédérale. Il s'agit en particulier de recommandations de conception pour les API et la gestion des identités et des accès, et de la tenue d'un registre des API. La conception des architectures de solution doit être largement libre de restrictions.
Principes architecturaux	Les principes architecturaux donnent des lignes de conduite pour le développement et le fonctionnement des API employées pour les services publics.
Normes et bonnes pratiques	L'architecture API repose sur des normes reconnues (par ex., eCH, Tallinn, EIF, W3C) et sur les bonnes pratiques de la Confédération. Elle est revue périodiquement.

Tableau 2 Dimensions de la vision

⁶ Dans l'architecture API de la Confédération, l'instrument de gestion du portefeuille est le registre des API.

⁷ Dans l'architecture API de la Confédération, les groupes d'utilisateurs sont répartis en trois types de partenaires (voir ch. 4.2).

⁸ Vision de l'architecture API de la Confédération

4 Bases légales et champ d'application

4.1 Notions

Le tableau 3 définit quelques notions applicables à l'ensemble de l'architecture.

Terme	Description
Service public numérique	Un service public numérique est l'accès numérique à une prestation administrative. Il peut reposer sur une seule API ou sur un ensemble d'API. Les services accessibles directement au moyen d'une API ou au moyen d'une interface numérique sur un portail de l'administration sont considérés comme tels.
<i>Application programming interface</i> (API)	Interface de programmation pour la communication entre des composants logiciels en général. Dans l'architecture API de la Confédération, le terme API désigne la communication entre des composants logiciels dans un réseau.
Partenaire	Terme générique qui désigne tout sujet utilisant un service public numérique.
Fournisseur	Terme générique qui désigne toute autorité proposant un service public numérique.
Utilisateur final	Personne physique qui utilise des produits informatiques et des logiciels qui accèdent à des services publics numériques au moyen d'API.
Application locale	Application exécutée localement chez l'utilisateur final pour accéder à un service public numérique. Il peut s'agir d'une application de bureau, d'une application mobile ou d'un code d'accès API exécuté dans un navigateur (par ex. JavaScript). Les navigateurs ne sont pas considérés comme des applications locales, car HTML n'est pas capable d'accéder aux API et d'analyser les réponses sans code d'accès (voir ch. 8.3.2.1).
Service spécialisé	Composant logiciel représentant le caractère spécialisé d'un service public numérique et le proposant dans une interface numérique.
<i>Cluster</i> de service spécialisé	Ensemble des instances API identiques d'un service spécialisé. La création de <i>clusters</i> augmente la performance et la stabilité. Les <i>clusters</i> peuvent être répartis sur plusieurs sites.
Application de portail	Application web accessible par un navigateur. La notion est définie dans l'architecture fédérale des portails élaborée dans le cadre de l'objectif de mise en œuvre 1 de la stratégie suisse de cyberadministration 2020-2023. Une application de portail est un cas particulier de service spécialisé et peut donc se présenter sous forme de client API.
Client API	Composant logiciel capable de commander des interfaces électroniques et de traiter des réponses. Il peut se présenter sous forme d'application locale, de service spécialisé ou d'application de portail.
<i>API gateway</i>	Aussi appelé passerelle, l' <i>API gateway</i> est un composant intermédiaire entre le client API et le service spécialisé, qu'il protège contre les dangers du monde extérieur. Il s'agit d'un <i>reverse proxy</i> doté de capacités supplémentaires telles que la gestion du trafic (<i>traffic management</i>). Les sites hébergeant un <i>cluster</i> de service spécialisé peuvent disposer de leurs propres instances <i>API gateway</i> , qui peuvent être interrogées individuellement.

Tableau 3 Notions

4.2 Partenaires

L'administration fédérale propose des services publics numériques utilisés par ses partenaires. Ces partenaires interviennent dans trois contextes : *government* (G), *business* (B) et *citizen* (C), définis dans le tableau 4. Il y a donc trois types de relations entre les UA et les partenaires : G2G, G2B et G2C. Dans l'architecture API de la Confédération, la notion de partenaire couvre les trois types de relation, y compris les collaborateurs désignés pour agir sur mandat d'un partenaire.

Partenaire	Description
<i>Government</i>	Niveaux administratifs internationaux, UA de l'administration fédérale (inter-unité), cantons et communes
<i>Business</i>	Personnes morales, c'est-à-dire entreprises, mais aussi universités, instituts de recherche, ONG ou associations
<i>Citizen</i>	Personnes physiques qui utilisent des applications locales et des applications de portail pour accéder à des services publics numériques

Tableau 4 Définition des partenaires

4.3 Intermédiaires

Des intermédiaires peuvent représenter les partenaires (par ex. Swissdec⁹) lorsqu'il s'agit d'interroger les services publics numériques. L'organisation qui fait le lien entre la prestation et le partenaire est qualifiée d'intermédiaire. Les intermédiaires sont des FP qui ne font pas partie de l'administration fédérale. Leurs tâches, qui peuvent aussi être combinées, sont en général les suivantes.

- Développer et gérer une plateforme intermédiaire (*hub*)
- Développer et gérer des mécanismes d'authentification couvrant la communication du début à la fin, c'est-à-dire du client API jusqu'à l'*API gateway* ou au service spécialisé
- Développer des normes visant à uniformiser la transmission de données par voie électronique des entreprises aux autorités des différents niveaux administratifs fédéraux et aux assurances sociales
- Développer et gérer des applications locales agissant comme des clients API
- Contrôler des logiciels standard agissant comme des clients API (par ex. logiciels ERP) en vue de les certifier conformes aux normes suscitées

4.4 Types d'API

L'architecture API de la Confédération compte trois types d'API : API publiques, API partenaires et API privées (voir tableau 5). Elle se concentre principalement sur les deux premiers types, les API privées n'étant pas sa priorité. Elle peut toutefois s'appliquer à des API privées, en particulier si ces dernières ont le potentiel de devenir des API publiques ou partenaires. Les API publiques ou partenaires peuvent être utilisées par les trois types de partenaires définis au ch. 4.3.

Types d'API	Description
API publique	<ul style="list-style-type: none"> - Les API publiques facilitent l'accès interne et externe aux données et aux services proposés par les UA. - Les partenaires peuvent les utiliser AVEC ou SANS enregistrement d'une identité, sachant qu'en l'absence d'enregistrement, l'utilisation est anonyme. - L'utilisation avec enregistrement présuppose de fournir une identité. L'accès est ensuite possible au moyen d'un identifiant et d'un jeton sécurisé. - Les conditions d'utilisation (ou convention d'utilisation) sont définies, communiquées et doivent être respectées. En cas d'utilisation avec enregistrement, l'acceptation explicite de ces conditions peut être demandée. - En cas d'utilisation avec enregistrement, toute demande d'accès est acceptée dès lors que l'identité peut être authentifiée. En général, l'utilisateur s'enregistre lui-même.
API partenaire	<ul style="list-style-type: none"> - Les API partenaires exigent SYSTÉMATIQUEMENT l'enregistrement de l'identité du partenaire. - Elles ne peuvent être utilisées que par les partenaires autorisés qui ont des relations avec l'administration fédérale. Les demandes d'accès sont donc toujours vérifiées et peuvent être acceptées ou refusées. - Elles requièrent l'enregistrement d'une identité. L'accès est ensuite possible au moyen d'un identifiant et d'un jeton sécurisé. - Un système d'autorisation est obligatoire pour piloter individuellement les accès. L'authentification exige une procédure conforme au besoin de protection des données, qui confirme, par vérification, que l'identité est fiable.
API privée	<ul style="list-style-type: none"> - Les API privées sont conçues pour être utilisées au sein des UA. Elles ne sont pas accessibles de l'extérieur.

Tableau 5 Définition des types d'API

⁹ Swissdec est un projet commun à but non lucratif regroupant plusieurs partenaires indépendants. Les organismes responsables et les membres de l'association sont la Confédération suisse des impôts (CSI), l'association eAHV/AI, qui représente les caisses de compensation, l'Office fédéral de la statistique (OFS), qui représente la Confédération, l'Association Suisse d'Assurances (ASA) et la Caisse nationale suisse d'assurance en cas d'accidents (CNA, Suva). Swissdec règle et organise la transmission de données salariales des entreprises aux assurances sociales et aux autorités et le traitement des événements (accident ou maladie). Swissdec gère une plateforme intermédiaire appelée Distributor.

4.5 Processus d'intégration

Pour fournir des services publics numériques, il faut intégrer de différentes manières des services spécialisés dans des clients API ou dans d'autres services spécialisés. Il existe à cet effet plusieurs processus d'intégration. Cinq d'entre eux sont décrits dans le chapitre, qui explique par ailleurs leur importance pour l'architecture API de la Confédération et montre les types d'API qui y sont liés (voir tableau 6). La figure 3 montre les cinq processus d'intégration en rapport avec les partenaires présentés au ch. 4.2 (*government*, *business* et *citizen*) et fournit une vue de tous les cas d'intégration API, où la Confédération propose des services publics numériques. Le ch. 4.6 donne une vue d'ensemble de six cas d'application représentatifs des services publics numériques dans l'environnement fédéral.

L'architecture API de la Confédération fournit un cadre réglementaire pour les processus d'intégration IP3 et IP4, indiqués en bleu sur la figure 3. Les processus IP1, IP2 et IP5 ne sont pas pris en compte. L'IP3 est important, car il concerne l'utilisation des services par les UA et que les appels d'API qui y sont liés ne quittent pas les zones de réseaux de la Confédération. L'IP4 concerne quant à lui les appels d'API qui dépassent les limites des zones de réseau de la Confédération. L'IP1 et l'IP2 ne sont pas traitées, car ils concernent des API privées de même que l'IP5 qui ne concerne que des partenaires de communication appartenant à des zones de réseau hors de la Confédération.

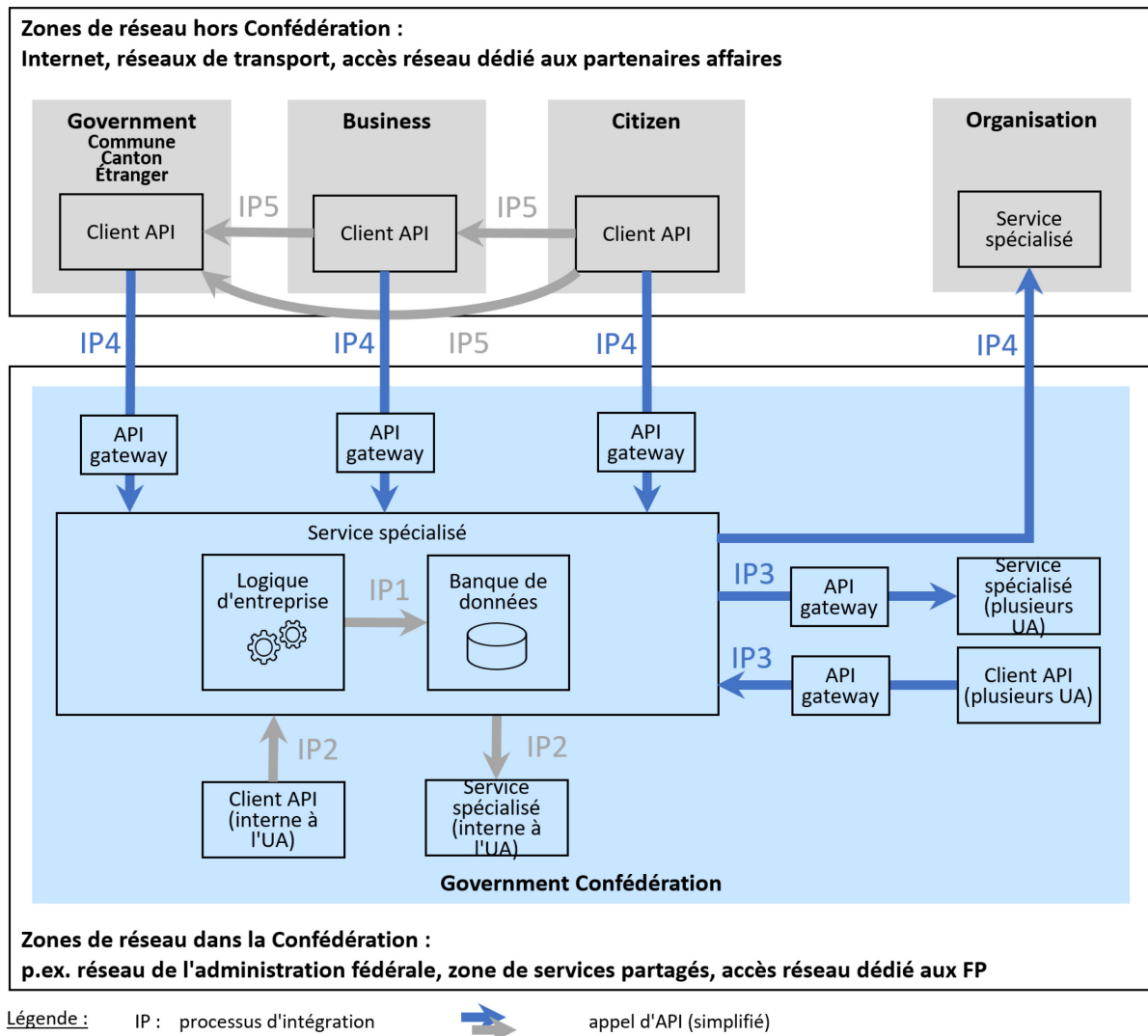


Figure 3 Processus d'intégration

La figure 3 montre qu'une organisation peut fournir à une UA un service numérique dont celle-ci a besoin pour fournir à son tour un service public numérique à un partenaire. L'organisation en question peut être attribuée à l'un des partenaires mentionnés dans le tableau 4, mais elle est représentée séparément sur la figure 3 pour

indiquer qu’elle fournit des services numériques tandis que les partenaires ne sont que bénéficiaires de ces services. Dans tous les cas, l’intégration d’un service numérique fourni par une organisation dans un service public doit reposer sur une base légale (*legal compliance*).

Dans les IP3 et IP4, l’utilisation d’une *API gateway* est prévue comme composant système intermédiaire entre le client API et le service spécialisé. Dans l’IP4, un intermédiaire peut représenter le partenaire et donc jouer le rôle de client API (voir figure 4). Les plateformes d’intermédiation peuvent être utilisées aussi bien pour les API publiques que pour les API partenaires.

Un service public numérique peut exiger qu’un service spécialisé appelle l’API d’un partenaire sans requête préalable immédiate de ce dernier (par ex. rappel d’une obligation administrative à remplir). Les IP3 et IP4 comprennent cette communication en sens inverse même si elle n’est pas représentée explicitement sur la figure 3.

La figure 4 représente en revanche ce chemin inverse. Le chemin inverse ne passe pas nécessairement par l’*API gateway* (attribué au service spécialisé), car la fonction première de ce dernier est de protéger les ressources du service spécialisé et non celles du client API (qui propose une API). Le client API peut protéger ses ressources à l’aide de son propre *API gateway*, tout comme l’organisation peut protéger son service spécialisé à l’aide d’une telle passerelle. Les *API gateway* ne sont pas représentés sur la figure 3 ni sur la figure 4, car ils ne sont pas au centre des préoccupations dans l’architecture API de la Confédération. Les appels API dirigés vers l’extérieur doivent cependant dans tous les cas respecter les directives sur la sécurité de l’information et la protection des données en vigueur dans l’environnement fédéral.

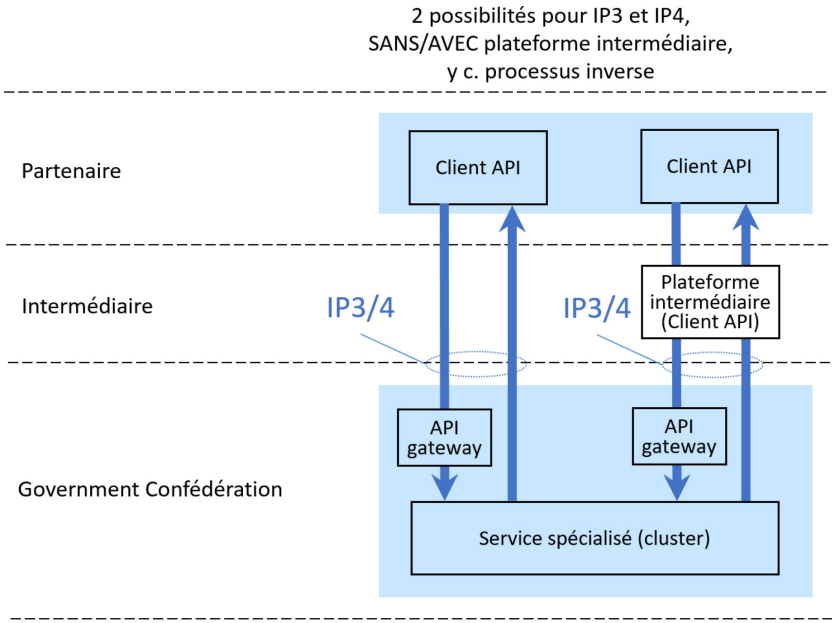


Figure 4 Deux expressions des processus d’intégration IP3 et IP4, avec représentation du chemin inverse

Processus	Désignation	Description	Types d’API
IP1	Service spécialisé interne	Communication à l’intérieur du service spécialisé	API privée
IP2	UA interne	Communication à l’intérieur de l’UA entre deux services spécialisés ou entre un client API et un service spécialisé	API privée
IP3	Zones de réseau internes de la Confédération	Communication dans plusieurs UA entre deux services spécialisés, sachant qu’un client API peut représenter un service spécialisé.	API publique API partenaire
IP4	Dépassement des zones de réseau de la Confédération	Communication entre une application locale ou un service spécialisés hors de l’administration fédérale et un service spécialisé interne.	API publique API partenaire
IP5	Zones de réseau hors de la Confédération	Communication entre applications dans des zones de réseau hors de l’administration fédérale	API publique API partenaire

Tableau 6 Processus d’intégration

4.6 Cas d'intégration API

Six cas d'intégration représentatifs des services publics numériques dans l'environnement fédéral sont définis. Ils sont répartis dans deux groupes selon que l'intégration API passe par un client API ou qu'elle est effectuée dans un service spécialisé. La figure 5 montre ces six cas et le tableau 7 donne les explications correspondantes.

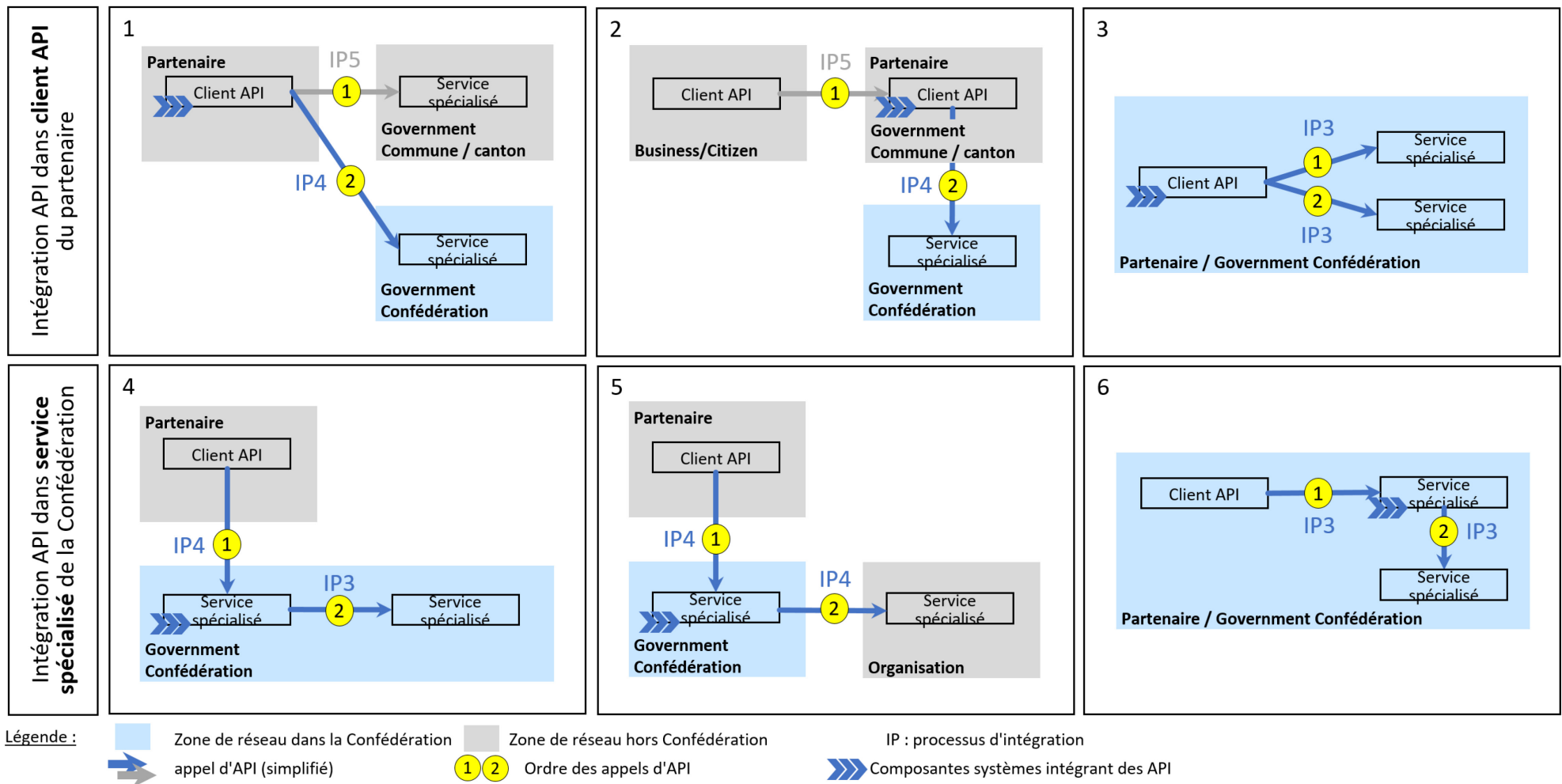


Figure 5 Cas d'intégration API

Intégration d'API dans le client API du partenaire		Intégration API dans le service spécialisé de la Confédération	
1	Un partenaire hors de la Confédération intègre dans un client API 1-N appels de services spécialisés internes et externes.	4	Le service spécialisé d'une UA est appelé par un partenaire externe et intègre 1-N API d'autres UA de la Confédération.
2	Un partenaire externe est appelé par un système tiers et intègre 1-N-API d'UA de la Confédération pour fournir des prestations au système tiers.	5	Le service spécialisé d'une UA est appelé par un partenaire externe et intègre 1-N API d'autres organisations hors de la Confédération.
3	Le partenaire est une UA de la Confédération dont le client API intègre 1-N API d'autres UA de la Confédération.	6	Le service spécialisé d'une UA est appelé par un partenaire interne et intègre 1-N API d'autres UA de la Confédération.

Tableau 7 Cas d'intégration API

Les six cas représentés sur la figure 5 utilisent les processus d'intégration IP3 ou IP4 sur lesquels se concentre l'architecture API de la Confédération. On les retrouve également sur la figure 3 d'où sont tirés les extraits correspondants. Ils peuvent être combinés, sachant qu'un client API peut représenter un service spécialisé.

La figure 5 indique à chaque fois dans quel composant du système se trouve la prestation d'intégration. Cette prestation peut également se présenter sous la forme d'un flux de travail à plusieurs niveaux d'appels API.

La figure 5 illustre également le fait qu'un cas d'utilisation est normalement lié à un certain ordre d'appels API. Si l'ordre des appels API représentés sur la figure 5 doit être respecté dans les cas 2, 4, 5 et 6, il ne doit pas obligatoirement l'être dans les cas 1 et 3. En fonction des exigences techniques, les appels peuvent aussi être lancés en parallèle.

Les partenaires sont chargés de concevoir l'intégration d'API dans le client API alors que cette tâche incombe toujours aux UA pour l'intégration d'API dans les services spécialisés. Dans les cas 3 et 6, la Confédération est elle-même un partenaire.

4.7 Architecture API de la Confédération, OGD et I14Y

Le Conseil fédéral a approuvé le 30 novembre 2018 la deuxième stratégie en matière de libre accès aux données publiques en Suisse pour les années 2019 à 2023 (stratégie OGD)¹⁰. Cette stratégie vise à mettre à la disposition du public un ensemble de données ouvertes de l'administration publique en libre accès sur le portail opendata.swiss¹¹ géré par l'Office fédéral de la statistique (OFS). Lorsque le portail est en fait une application de portail qui appelle des API de la Confédération, l'architecture API de la Confédération s'applique également aux clients API intégrés dans le portail opendata.swiss.

La plateforme d'interopérabilité I14Y¹² dresse le catalogue de données national suisse. Sa fonction est d'assurer l'efficacité des échanges de données entre les autorités, les entreprises et les citoyens. Elle offre une vue d'ensemble – développée en permanence – des fichiers de données et des interfaces de la Confédération, des cantons et des communes tout en centralisant la mise à disposition des métadonnées correspondantes. La plateforme d'interopérabilité I14Y peut être utilisée pour mettre en place des registres des API.

¹⁰ <https://www.bfs.admin.ch/bfs/fr/home/services/ogd.html>

¹¹ <https://opendata.swiss>

¹² <https://www.i14y.admin.ch>

5 Structure

5.1 Structure de l'architecture API de la Confédération

L'objectif de l'architecture API de la Confédération est de normaliser et de promouvoir l'accès numérique aux services publics dans l'environnement fédéral afin que les entreprises, l'administration et les personnes puissent bénéficier de la communication de M2M.

Elle comprend donc une série de principes architecturaux et une architecture de référence inspirée des domaines architecturaux (BDAT) prévus par la norme TOGAF¹³. Ces principes et cette architecture offrent un cadre réglementaire et un langage commun en vue de la conception des architectures de solution API permettant un accès numérique aux services publics dans les départements et les UA. L'architecture de référence API est divisée en éléments architecturaux et recommandations de conception qui s'articulent autour des domaines architecturaux.

L'architecture API de la Confédération en soi n'a pas de caractère contraignant. Des directives indépendantes de l'architecture peuvent toutefois conférer un caractère contraignant à certains de ses éléments. Le ch. 7.5 décrit les modalités selon lesquelles des prescriptions sur les API peuvent être édictées et la manière de les gérer. La figure 6 montre les liens entre l'architecture API de la Confédération et les architectures de solution et directives API.

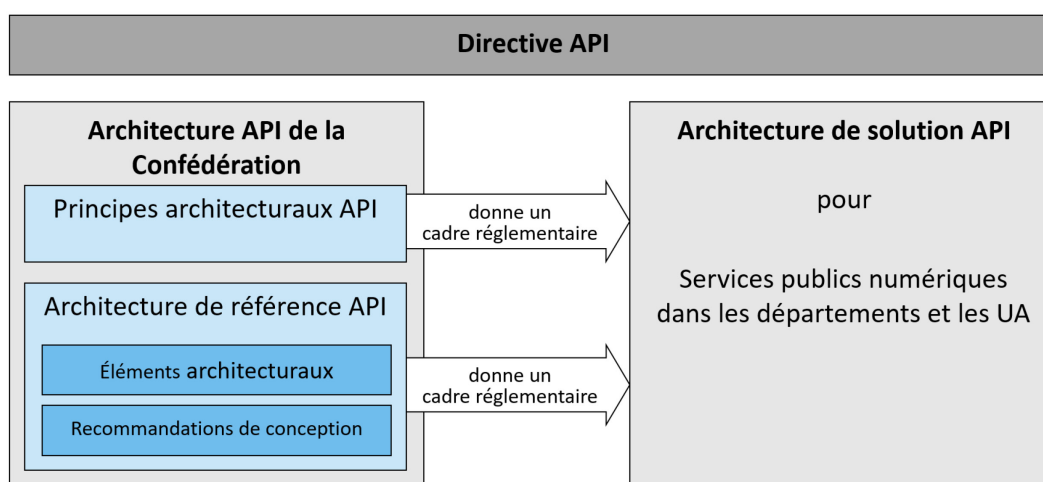


Figure 6 Position de l'architecture API de la Confédération

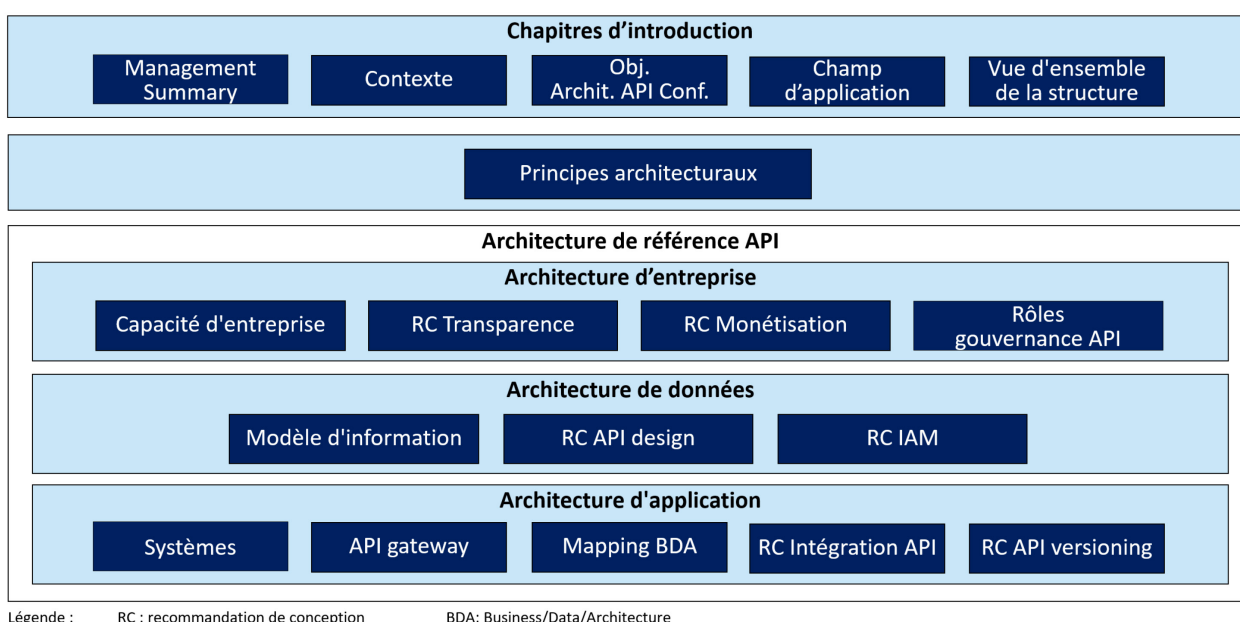


Figure 7 Structure de l'architecture API de la Confédération et domaines architecturaux selon la norme TOGAF

¹³ <https://www.opengroup.org/togaf>

La figure 7 montre la structure de l'architecture API de la Confédération dans son ensemble et propose un aperçu du contenu. Un chapitre est consacré à chaque domaine architectural, à l'exception du domaine technologique. L'architecture technologique n'est pas prise en considération et ne fait l'objet d'aucune recommandation de la part de la Confédération.

5.2 Architecture de référence API

Afin de définir le cadre réglementaire de l'architecture API de la Confédération, il faut commencer par définir une architecture de référence API (ch. 7 à 9), qui décrit tous les éléments architecturaux pertinents. L'architecture de référence API réunit ces éléments en une image globale, en définissant les éléments considérés comme nécessaires pour la fourniture de services publics numériques et pour une interopérabilité efficace.

Selon le cas, dans l'environnement fédéral, seuls les éléments d'architecture qui contribuent à répondre aux exigences d'entreprise sont utilisés pour développer une architecture de solution API. L'architecture de référence API est conçue de manière à laisser une grande liberté lors de la conception des éléments de la solution. Afin d'exploiter les synergies, il est recommandé d'utiliser des services informatiques standard ou des applications centrales, lorsque cela est possible et judicieux.

L'architecture API n'impose rien. Elle comprend toutefois des recommandations de conception. Une distinction y est faite entre les éléments pour lesquels la conception relève des FP et des UA, et pour lesquels il n'y a donc pas de recommandation, et les éléments pour lesquels des recommandations concrètes sont formulées.

Afin de garantir l'interopérabilité et la durabilité, les recommandations reposent systématiquement sur les normes ou les bonnes pratiques reconnues.

6 Principes architecturaux

Les principes de l'architecture API de la Confédération s'inscrivent dans une hiérarchie à trois niveaux :

1. Les principes (architecturaux) généraux applicables à l'administration fédérale tels que (liste non exhaustive)
 - la *Tallinn Declaration on eGovernment*, qui définit cinq principes centraux pour le domaine de la cyberadministration¹⁴
 - les *SOA policies*¹⁵, qui sont des directives pour l'utilisation de services selon des concepts SOA dans l'administration fédérale
 - les principes découlant de l'architecture cadre IAM de la Confédération (voir ch. 8.4)
2. Les principes de l'architecture API de la Confédération (qui font l'objet de ce chapitre), qui servent d'architecture de référence API lors de la conception d'architectures de solution.
3. Les recommandations de conception attribuées aux trois domaines architecturaux dans l'architecture de référence API (voir ch. 5.1).

D'autres directives pour les REST API et les API orientés événements ont été publiées par la société Zalando (*Zalando RESTful API and Event Guidelines*¹⁶) et par les CFF (*API Principles*¹⁷). Les directives des CFF reposent sur celles de Zalando. Certains principes de l'architecture API de la Confédération proviennent de ces directives. La plupart du temps, les directives API sont très détaillées et peuvent donc être utilisées comme complément aux principes architecturaux formulés ci-après.

Les principes architecturaux sont au nombre de dix et sont regroupés en trois domaines : orientation client et service, *API design* et interopérabilité, publication et communication sur les API (voir figure 8). Le tableau 8 décrit les principes architecturaux en détail.

Orientation client et service

AP1	API traitées comme des produits
AP5	Définition et garantie du niveau de service
AP6	Définition de l'utilisation des données et des prestations

Publication / communication sur les API

AP3	Transparence des API disponibles
AP4	Documentation homogène et exhaustive

Conception d'API / interopérabilité

AP2	API first
AP7	Garantie de la rétrocompatibilité
AP8	Utilisation de technologies API reconnues
AP9	Protection du client face aux détails d'implémentation
AP10	Garantie de l'idempotence

Figure 8 Groupes de principes architecturaux dans l'architecture API de la Confédération

¹⁴ <https://www.admin.ch/gov/de/start/dokumentation/medienmitteilungen.msg-id-68342.html>

¹⁵ <https://www.bk.admin.ch/bk/fr/home/digitale-transformation-ikt-lenkung/ikt-vorgaben/architekturen/r016-soa-policies.html>

¹⁶ <https://opensource.zalando.com/restful-api-guidelines>

¹⁷ <https://schweizerischebundesbahnen.github.io/api-principles>

Réf.	Nom	Énoncé	Motif	Effet
AP1	API traitées comme des produits	<i>Les API sont proposées aux partenaires comme des produits auxquels sont associées des prestations. Chaque API est gérée par un propriétaire d'API (API product owner) rattaché à une UA. L'API est un sous-produit de l'ensemble des données et des services auxquels elle permet d'accéder.</i>	La fourniture de services publics numériques est orientée client et axée sur les données et les prestations. Traiter les API comme des produits permet aux autorités d'accomplir efficacement leurs devoirs et de répondre au mieux aux exigences des partenaires. Grâce à cette approche, les partenaires acceptent et utilisent plus volontiers les API.	<p>Traiter une API comme un produit garantit l'accomplissement de tâches spécifiques à la gestion des produits, telles que :</p> <ul style="list-style-type: none"> - garantir une utilisation simple et conviviale - gérer les relations avec les partenaires - mettre à disposition de la documentation - fournir un soutien spécialisé et technique - mettre à disposition une version test - assurer des procédures d'enregistrement simples - analyser l'utilisation - poursuivre le développement <p>Lorsque les API, de même que les données et les prestations auxquelles elles sont subordonnées, sont considérées comme des produits, les <i>API product owner</i> et maîtres de données (<i>data owner</i>) sont garants de leur exactitude. Le <i>data owner</i> définit le cadre juridique pour l'utilisation des données et l'<i>API product owner</i> s'assure que les API mises à disposition sont conformes aux directives.</p>
AP2	API first	<i>La conception fonctionnelle des API orientées vers l'extérieur est définie comme élément central d'un service public numérique. L'API est conçue en fonction de la démarche couverte par le service public numérique et des données et prestations requises. Les documents relatifs à la conception sont établis selon un format de spécification normalisé.</i>	<p>Dans une approche <i>API first</i>, une planification préalable des API avec une définition claire et cohérente des spécifications garantit que la démarche administrative visée est couverte au mieux et que les exigences fonctionnelles posées à l'interface sont connues et documentées. Le service public numérique proposé sera ainsi d'autant plus efficace.</p> <p>Une approche <i>API first</i> garantit la stabilité des API, en particulier lorsque l'architecture d'application ou l'implémentation du service sous-jacente est remplacée. Cette approche assure l'interopérabilité avec tous les partenaires qui utilisent l'API. Elle est donc à conforme la volonté développer davantage les architectures orientées vers les services.</p>	<p>La spécification standardisée de l'API est la documentation actuelle et valable de l'API. Elle comprend les accords concernant le type d'interface, le format des données, les MEP, les types de messages et les protocoles API orientés sur la démarche administrative visée.</p> <p>Les partenaires et les développeurs sont invités à donner leur avis sur les spécifications API à un stade précoce afin que celui-ci soit pris en compte.</p> <p>Dans les cas où des services spécialisés existent déjà, l'architecture technologique de l'API est conçue de manière aussi indépendante que possible du service spécialisé en question. Cela entraîne, le cas échéant, de nouvelles exigences pour le service spécialisé.</p> <p>Dans les cas où les services spécialisés n'existent pas, leur conception s'aligne sur celle des API.</p> <p>Derrière chaque API orientée vers l'extérieur se trouve un service spécialisé, mais tous les services spécialisés ne disposent pas d'API orientées vers l'extérieur.</p>
AP3	Transparence des API disponibles	<i>Les métadonnées d'API et les interlocuteurs responsables des API orientées vers l'extérieur de l'administration fédérale sont publiés dans</i>	La publication encourage l'utilisation et la réutilisation des API disponibles dans l'administration fédérale et orientées vers l'extérieur. La transparence devient ainsi un moyen d'encourager l'orientation client.	Il est nécessaire de bien gérer les métadonnées d'API afin que les informations soient accessibles à tout moment aux groupes cibles de l'architecture API de la Confédération.

Réf.	Nom	Énoncé	Motif	Effet
		<i>un registre central des API conformément aux dispositions de la LMETA.</i>		
AP4	Documentation homogène et exhaustive	<i>La documentation des API releases dans l'administration fédérale est homogène et complète.</i>	La qualité de l'expérience faite lors de l'utilisation et du développement d'une API dépend de la documentation API. Une documentation complète et homogène facilite le travail avec l'API et assure que les partenaires l'acceptent et sont satisfaits.	La documentation API correspond à la documentation d'un <i>API release</i> (publié). Elle comprend toutes les informations techniques nécessaires pour pouvoir décider d'utiliser l'API et pour s'y adresser correctement à partir d'un client API. La documentation est la plus possible générée automatiquement. Ce principe ne s'applique pas aux API dont les données ne sont pas publiées selon la LMETA, par exemple en raison de dispositions relatives à la protection des données ou des informations.
AP5	Définition et garantie du niveau de service	<i>Chaque API dispose d'objectifs de niveau de service (SLO) définis et transparents. Si la documentation de l'API ou, dans le cas d'API payantes, le contrat de licence garantit des SLO, l'API product owner API s'assure de leur respect.</i>	Ce principe garantit le niveau de service des services publics numériques proposés. Pour cela, des SLO sont définis aussi bien pour les API publiques que pour les API partenaires.	Les SLO sont définis et communiqués de manière transparente. Des paramètres tels que la disponibilité, le temps de réponse, le débit ou le temps de récupération y sont inscrits. Si la documentation de l'API ou, dans le cas d'API payantes, le contrat de licence garantit des SLO, ils doivent être mesurables afin de vérifier leur respect. Le niveau de la qualité de service communiqué aux clients repose sur l'héritage de tous les SLO sous-jacents, même en cas d'intégration API dans le service spécialisé de la Confédération. Les SLO les plus faibles s'appliquent. En principe, les SLO valables pour le service spécialisé définissent les SLO qui s'appliquent à l'API. Un département ou une UA peut définir des SLO uniformes pour ses services publics numériques, par exemple « <i>best effort</i> ».
AP6	Définition de l'utilisation des données et des prestations	<i>Les données et les prestations de chaque API font l'objet d'une convention d'utilisation qui définit la manière dont elles peuvent être employées. La convention fait partie de l'ensemble des données et des services auxquels l'API permet d'accéder.</i>	Le cadre juridique pour l'utilisation des données échangées dans les services publics numériques doit être défini. L'utilisation est régie dans des ordonnances. La convention d'utilisation peut y faire référence et comble en outre les éventuelles lacunes. Les bases juridiques pour l'utilisation des services publics numériques et de leurs données sont ainsi créées.	L'utilisation des données définie dans la convention d'utilisation prévoit les modalités selon lesquelles les services qui proposent des API et leurs partenaires peuvent utiliser les données échangées, dans la mesure où celles-ci font partie du service public numérique. Les règles d'utilisation des données dépendent de la démarche administrative visée et des prestations et données qui en font partie.
AP7	Garantie de la rétrocompatibilité	<i>Lors du développement des API, la rétrocompatibilité est assurée.</i>	Les API sont développées en permanence. La rétrocompatibilité entre les différents <i>API releases</i> évite aux partenaires de devoir à chaque fois modifier leurs clients API.	Si la rétrocompatibilité est assurée, le partenaire n'a pas besoin de modifier son client API lors de la publication d'un nouveau <i>release</i> . En cas d'interruption de la rétrocompatibilité (par ex. <i>legal compliance</i>), le partenaire doit bénéficier d'un délai adéquat pour adapter son client API à un <i>API release</i> opérationnel. Une migration forcée entraîne systématiquement l'utilisation parallèle de plusieurs <i>releases</i> . Pour maintenir les coûts aussi bas que possible,

Réf.	Nom	Énoncé	Motif	Effet
				il faut utiliser en même temps aussi peu de <i>releases</i> que nécessaire.
AP8	Utilisation de technologies API reconnues	<i>Les API de l'administration fédérale n'utilisent que des technologies API largement reconnues.</i>	Les API ont des fonctionnalités spécialisées, mais elles représentent aussi le langage qu'utilisent deux systèmes ou les systèmes d'un même ensemble pour communiquer. Pour favoriser l'interopérabilité, seules les technologies API largement reconnues sont utilisées. De cette manière, les API sont plus facilement réutilisables.	La technologie API est concrétisée dans les accords relatifs à l'échange de données pour lesquels l'architecture API de la Confédération donne des recommandations. Il s'agit notamment des éléments suivants. <ul style="list-style-type: none"> - Protocole API - Type d'interface - MEP - Format de données - Type de message
AP9	Épargner au client API les détails d'implémentation	<i>Les partenaires ne voient pas les détails d'implémentation des services spécialisés.</i>	Conformément au principe <i>API first</i> , la spécification de l'API reste constante vis-à-vis de l'extérieur. L'API met en œuvre cette spécification.	Les API ne doivent pas révéler de détails sur la technique d'implémentation sous-jacente. Celle-ci peut changer d'un <i>release</i> d'API à l'autre ou d'un service spécialisé à l'autre, sans que cela soit visible de l'extérieur.
AP10	Garantie de l'idempotence	<i>Plusieurs appels d'API successifs et identiques ont toujours le même effet dans le service spécialisé.</i>	Il se peut qu'intentionnellement ou non, un client API lance plusieurs appels d'écriture successifs et identiques. Ces appels ne doivent produire un effet qu'une seule fois dans le service spécialisé. Comme l' <i>API gateway</i> représente l'aspect technique du service spécialisé, il faut également lui appliquer le principe d'idempotence.	Les demandes de lecture successives et identiques peuvent être exécutées plusieurs fois. En revanche les demandes d'écriture successives et identiques ne doivent l'être qu'une seule fois. L'API reconnaît ces dernières et informe correctement le client API du résultat de traitement laissé dans le service spécialisé.

Tableau 8 Principes de l'architecture API de la Confédération

7 Architecture d'entreprise

7.1 Capacité API management

7.1.1 Aperçu

Dans l'architecture API de la Confédération, les capacités d'entreprises sont définies au moyen de la planification en fonction des capacités selon la norme TOGAF (voir ch 10.1). La capacité **API management** est définie comme une capacité spécifique de l'API de premier ordre. Cette capacité utilise en outre d'autres capacités, qui sont également requises dans le contexte de l'architecture API et peuvent être importantes pour des projets informatiques d'une autre nature. Afin de dresser un tableau complet des capacités nécessaires à une architecture API, l'ensemble des capacités est décrit dans l'architecture API Confédération. Elles sont au nombre de 20 et sont attribuées aux 5 étapes de processus relatifs aux API.

La mise en œuvre de la capacité **API management** consiste à appliquer les capacités spécifiques à l'API dans l'architecture de solution API et à y intégrer les capacités non spécifiques. Tous les acteurs y participent, qu'il s'agisse du fournisseur d'API, du partenaire ou des FP.

Les étapes de la modélisation des processus sont également définies à ce moment, bien que la planification de la modélisation intervienne après la planification en fonction des capacités. Ces étapes de processus servent à regrouper les capacités et permettent de modéliser un déroulement séquentiel. Une initiative de développement, de modification ou de révocation d'un service public numérique est donc toujours à l'origine d'une itération de ce déroulement qui débute par l'étape *Manage API lifecycle* et se poursuit selon le schéma présenté à la figure 9. L'étape *Register client* est ignorée dans certains cas.

Le chapitre décrit les étapes du processus. La figure 10 montre les capacités et l'attribution aux étapes de processus. Une capacité peut contribuer à plusieurs étapes de processus. Les capacités spécifiques aux API sont indiquées sur la figure 9.

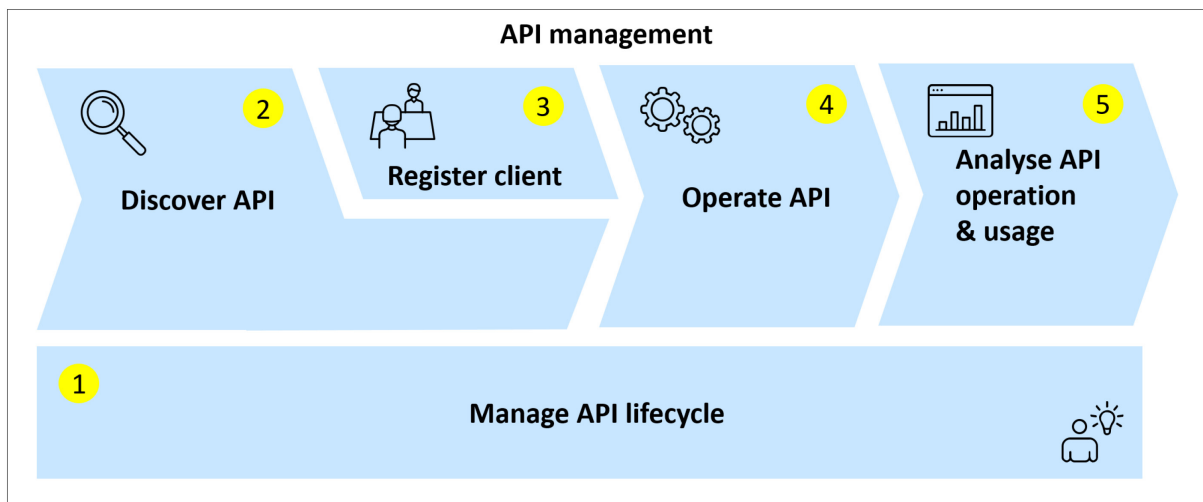


Figure 9 Étapes de processus en vue du regroupement des capacités

Le processus est divisé en plusieurs étapes qui sont en relation entre elles : *Manage API lifecycle* (1), *Discover API* (2), *Register client* (3), *Operate API* (4) et *Analyse API operation & usage* (5). Un processus est lancé lorsqu'une initiative visant le développement d'un service public numérique est formulée. L'initiative peut être motivée notamment par une initiative parlementaire, une prescription légale ou la volonté de proposer un service public existant sous forme numérique.

Le début de l'initiative marque également le début de l'étape de processus **Manage API lifecycle**, par lequel commence le cycle de vie d'une API. L'API est conçue, développée, testée, documentée, puis publiée, utilisée et révoquée à la fin de son cycle de vie. Cette étape se superpose donc à toutes les autres, elle commence par la planification du développement d'un service public numérique et se termine après sa révocation. Les API sont améliorées en permanence. Des améliorations peuvent être nécessaires en raison de modifications du cadre légal, de nouvelles exigences de la part des clients ou de la mise en place de demandes d'optimisation cumulées au fil du

temps. L'*API lifecycle management* couvre toute les versions et tous les *releases* d'une API créés et publiés au cours du cycle de vie. Un *API release* est une version de l'API publiée dans le registre des API.

La publication vise à mettre un *API release* à la disposition du public cible pertinent, à savoir les architectes d'entreprise, les architectes informatiques, les spécialistes, les gestionnaires et les personnes issues des affaires et de l'informatique en recherche d'orientation. Elle comprend la mise à disposition de paquets logiciels à installer (*API build*) ou de configurations inscrites dans un *repository* ou un *registry*¹⁸ ainsi que le déploiement des instances API qui en découlent dans l'environnement d'exploitation. Les métadonnées relatives à l'*API release* sont en outre inscrites dans un registre des API, ce qui rend publique la disponibilité de l'API. Pour les API publiques, la publication est toujours liée au déploiement d'une instance API dans l'environnement d'exploitation. Pour les API partenaires, qui demandent par nature un enregistrement, le déploiement de l'instance API peut être lié à la validation de la demande d'enregistrement.

Le registre des API dans une organisation est un endroit où le groupe cible obtient de manière centralisée des informations sur les API publiées et où il peut rechercher de manière ciblée les API disponibles qu'il peut utiliser pour développer ses propres API. La recherche d'API réutilisables est une capacité clé de l'étape de processus **Discover API**. Pour les API qui demandent l'enregistrement d'une identité, le registre des API est le point de départ de l'étape de processus *Register client*.

L'étape de processus **Register client** comprend la demande, la validation et la planification de l'accès à une API. Si les partenaires sont pertinents pour les données de base, cela implique également l'enregistrement d'un partenaire dans la gestion des données de base. Cette étape est ignorée pour les API publiques qui ne demandent pas d'enregistrement.

Le cœur de l'API est l'*API gateway*, qui joue un rôle d'intermédiaire entre le client API et le service spécialisé (voir ch. 9.2). Il constitue la porte principale pour accéder au service public numérique, contrôle l'accès aux services spécialisés et les protège du monde extérieur. L'exploitation de l'*API gateway* s'effectue dans le cadre de l'étape **Operate API**. Les capacités *Logging* et *Monitoring* font le lien avec l'étape *Analyse API operation & usage*.

L'étape de processus **Analyse API operation & usage** comprend la mesure des indicateurs clés de performance (ICP), leur évaluation et la création de rapports. Les rapports peuvent être de nature opérationnelle ou spécialisée et se composer de simples listes ou d'évaluations complexes, voire d'éléments d'intelligence artificielle. L'étape de processus *Analyse API operation & usage* fournit également la base permettant de générer des revenus à partir de l'exploitation des API. Les capacités *API reporting* et *API monetization* peuvent avoir une influence sur le fonctionnement d'une API en créant automatiquement des profils de trafic de données et des plans d'utilisation, et en contrôlant ainsi de manière dynamique les instances API (voir ch. 9.1) dans l'environnement d'exploitation.

Les étapes du processus sont liées à différentes notions de temps. Comme certaines capacités contribuent à plusieurs étapes de processus, l'architecture API de la Confédération attribue à ces étapes différentes notions de temps (voir tableau 9), qui permettent de délimiter les prestations les unes par rapport aux autres. Ces notions de temps sont utilisées dans la description des capacités et du modèle d'information (voir ch. 7.1.2 et 8.2).

Étape du processus	Notions de temps
<i>Manage API lifecycle</i>	<ul style="list-style-type: none"> - Temps de développement (<i>Creation Time</i>, CT) - Temps de définition (<i>Definition Time</i>, DT) - Durée d'exécution (<i>Run Time</i>, RT)
<i>Discover API</i>	<ul style="list-style-type: none"> - Temps de développement
<i>Registre Client</i>	<ul style="list-style-type: none"> - Temps de définition
<i>Operate API</i>	<ul style="list-style-type: none"> - Durée d'exécution
<i>Analyse API operation & usage</i>	<ul style="list-style-type: none"> - Durée d'exécution

Tableau 9 Étapes du processus et notions de temps

¹⁸ Le code et les configurations de l'API sont généralement conservés dans un code *repository* avec contrôle de version, tandis que les *builds* de l'API sont conservés dans un *registry* (*software store*).

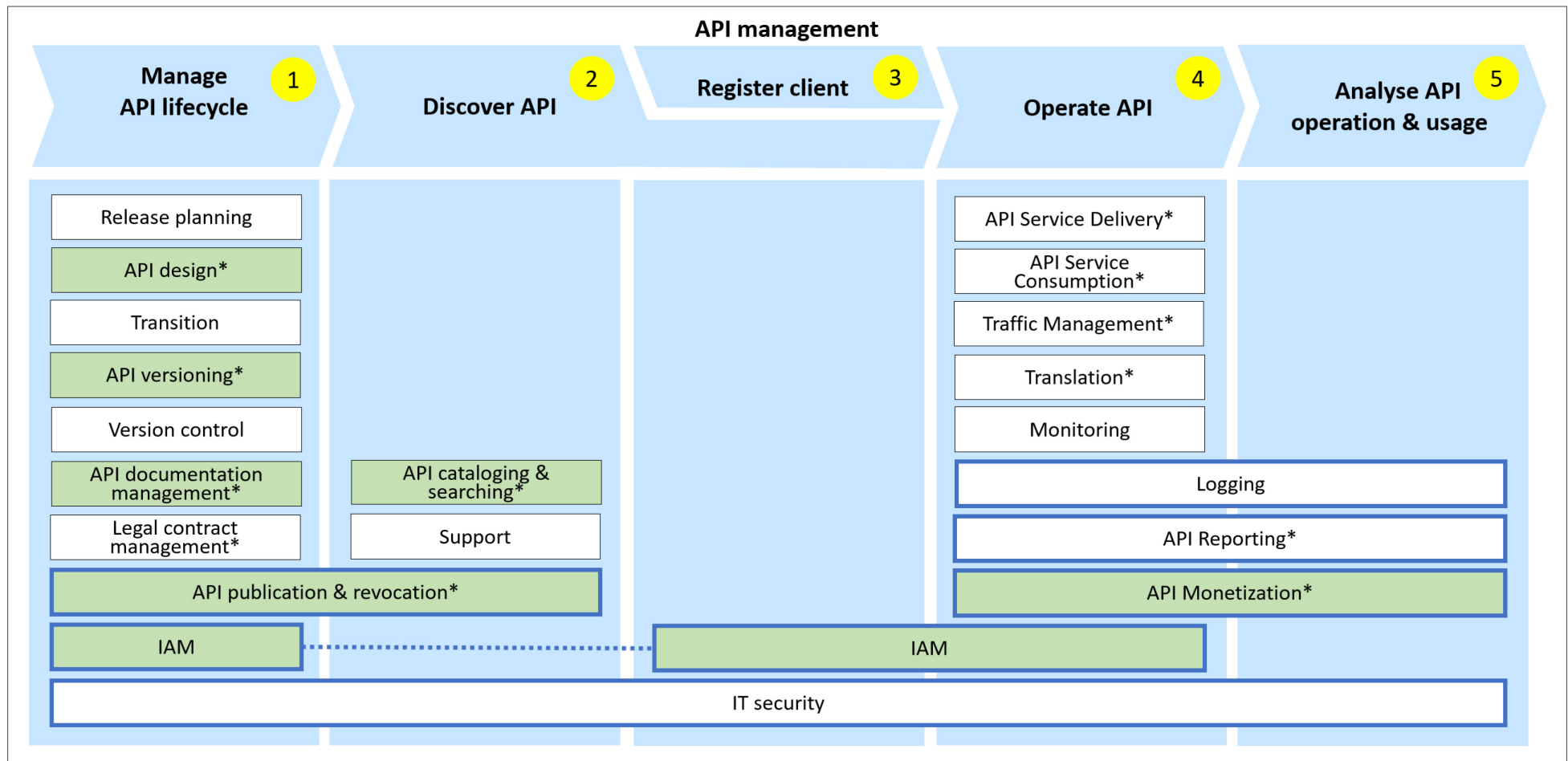


Figure 10 Carte des capacités

7.1.2 Description des capacités d'entreprise

7.1.2.1 Manage API lifecycle

Capacité	Définition
<i>Release planning</i>	Au cours du cycle de vie d'une API, il faut planifier des <i>releases</i> . Cette capacité comprend la collecte et le regroupement des changements à venir, la définition de la nouvelle version avec la communication qui l'accompagne (particulièrement pertinente en cas de migration forcée, voir ch. 9.5.5), la mise à disposition des ressources de développement et d'exploitation et la préparation de l'organisation de l'assistance.
<i>API design</i>	Dans toute architecture (API), il faut accorder une attention particulière à la capacité de conception des API (<i>API design</i>). Les quatre domaines architecturaux BDART comportent des éléments de conception. Dans ce contexte, l'architecture API de la Confédération propose un cadre structurel composé d'éléments architecturaux et de recommandations de conception pour les API. L'un des points centraux à prendre en considération lors de la conception d'une API est le respect du cadre l'égal dans lequel s'inscrit le service public numérique (<i>legal compliance</i>). La capacité <i>API design</i> comprend également la capacité d'utiliser des normes pour la spécification d'API, la documentation d'API et la génération automatisée de code (par ex. OpenAPI Specification pour les RESTful API ¹⁹ ou des outils basés sur le WSDL).
<i>Transition</i>	La <i>transition</i> est la capacité à mettre en œuvre et à tester la conception de l'API choisie en utilisant peu de ressources. Les API peuvent être le produit d'un développement interne ou correspondre à l'application de produits <i>open source</i> ou de produits commerciaux. Les directives de développement facilitent la collaboration au sein de l'équipe et contribuent à la qualité de l'API. Les tests ont pour but de faire des déclarations sur la qualité des API développées, c'est-à-dire de savoir si les API répondent aux exigences et à l'objectif. Ils peuvent être effectués manuellement ou automatiquement. Dans le cadre de la transformation numérique, les tests automatisés prennent de plus en plus d'importance. Dans les environnements de développement et d'exploitation modernes, les incréments d'API sont testés, intégrés et déployés automatiquement et en continu dans le cadre des processus DevOps (CI/CD).
<i>API versioning</i>	La stratégie relative aux versions sur laquelle repose l'architecture API de la Confédération est capitale pour les partenaires (voir ch. 9.5). Elle permet de savoir si une modification de l'API interrompt son intégration ou si un comportement cohérent de l'API est assuré d'une version à l'autre. Une stratégie bien pensée incite les partenaires à passer au <i>release</i> suivant.
<i>API documentation management</i>	La gestion des documents de l'API (<i>API documentation management</i>) comprend la création et la mise à jour de la documentation spécialisée, y compris les conventions d'utilisation, les métadonnées API et la documentation API dans le cadre de la gestion du cycle de vie. Elle comprend également l'utilisation de normes pour la génération automatique de la documentation API (voir capacité <i>API design</i>).
<i>API publication & revocation</i>	Une fois qu'un <i>API release</i> est développé, il est publié dans un registre des API (voir capacité <i>API cataloging & searching</i>), de sorte que les partenaires et les FP mandatés puissent s'informer sur les services publics numériques mis à disposition par l'administration fédérale et en demander l'accès pour les utiliser. La publication comprend toutes les informations pertinentes (voir ch. 7.2) pour prendre des décisions éclairées concernant l'obtention d'API. Elle comprend également le dépôt de l' <i>API build</i> et de la configuration de l'API dans le <i>repository</i> ou le <i>registry</i> . À la fin de son cycle de vie, l' <i>API release</i> est révoqué. La révocation a un impact important sur les partenaires et sur les ressources de développement et d'exploitation. Elle nécessite donc, tout comme la publication, une planification minutieuse.
<i>Version control</i>	Le contrôle des versions (<i>version control</i>) offre la possibilité de gérer les différentes versions d'une API de manière à assurer la persistance et à permettre la restauration de tous les fichiers et documents pertinents pour le service public numérique, avec les modifications qui y ont été apportées et l'identifiant de l'utilisateur. Les développeurs d'API travaillent en général localement avec des copies des fichiers et des documents et comparent leurs versions locales avec le <i>repository</i> à intervalles réguliers.
<i>Legal contract management</i>	La gestion des contrats juridiques (<i>legal contract management</i>) comprend la création et la gestion du contrat de licence, de la convention d'utilisation, des SLO et de l'accord de niveau de service (SLA). <ul style="list-style-type: none"> - Le contrat de licence est conclu entre le fournisseur et le partenaire. Il s'applique aux services publics numériques payants. - La convention d'utilisation régit l'utilisation des données sous-jacentes à l'API par le fournisseur et le partenaire. Dans le cas des API publiques, elle est publiée et implicitement acceptée par le partenaire lors de l'utilisation. Pour les API partenaires, elle est

¹⁹ <https://swagger.io>

Capacité	Définition
	<p>acceptée explicitement lors de la procédure d'enregistrement ou les identités autorisées à accéder à l'API appartiennent à un groupe d'utilisateurs qui l'a déjà acceptée globalement avant l'enregistrement.</p> <ul style="list-style-type: none"> - Le SLA est conclu entre le FP exploitant l'API et l'UA responsable de l'API, il définit le niveau de service et les SLO ainsi que le prix convenu pour l'exploitation de l'API. Les SLO sont publiés dans la documentation API et acceptés par le partenaire lors de l'utilisation.
<i>Identity & access management (IAM)</i>	<p>La gestion des identités et des accès (<i>identity & access management</i>) est particulièrement importante lorsque l'API demande l'enregistrement d'une identité. Elle intervient à trois étapes de processus :</p> <ul style="list-style-type: none"> - <i>Manage API lifecycle</i> - <i>Register client</i> - <i>Operate API</i> <p>Lors du développement, l'IAM offre la possibilité de définir et de saisir des rôles. Lors de la définition, elle permet d'émettre des identités, de leur attribuer des rôles (demande d'accès), de valider les demandes d'accès et même d'émettre des preuves d'identité. Lors de l'exécution, elle propose la fédération de services IAM et l'émission de jetons sécurisés. L'IAM a également pour but de permettre au partenaire de s'enregistrer lui-même.</p>
<i>IT security</i>	<p>La sécurité informatique (<i>IT security</i>) englobe toutes les étapes du processus et vise à garantir la protection contre les cyberattaques. Elle comprend la protection des systèmes informatiques, des réseaux et d'autres systèmes cyberphysiques, et donc à la fois la protection complète contre l'interruption ou l'utilisation abusive des services et fonctions proposés et la protection de la confidentialité, de l'intégrité et de la disponibilité des données traitées.</p> <p>Les prescriptions générales garantissant la sécurité de l'information et la protection des données (SIPD) dans l'environnement fédéral, qui sont définies dans les bases juridiques et les directives transversales en vigueur, s'appliquent, parmi lesquelles les directives de sécurité informatique du NCSC²⁰. La principale base juridique du NCSC est l'ordonnance du 27 mai 2020 sur les cyberrisques dans l'administration fédérale (OPCy)²¹, que viennent compléter le cas échéant des dispositions prises à l'échelon des départements ou des offices.</p>

Tableau 10 Capacités de l'étape de processus *Manage API lifecycle*

7.1.2.2 Discover API

Capacité	Définition
<i>API cataloging & searching</i>	<p>Le catalogage est la capacité à collecter les métadonnées relatives aux API mises à disposition par l'administration fédérale conformément aux dispositions de la LMETA, de les publier et d'offrir la possibilité d'obtenir des API. Il a pour but de fournir aux partenaires et aux FP mandatés un catalogue contenant toutes les informations pertinentes pour prendre des décisions concernant l'obtention d'API. Le groupe cible de l'architecture API de la Confédération peut consulter le catalogue pour y rechercher des API spécifiques. Les champs de métadonnées de l'API constituent alors les critères de recherche.</p>
<i>Support</i>	<p>L'assistance (<i>support</i>) consiste à aider les utilisateurs d'API à collecter des informations qui les aideront à prendre des décisions concernant les API, à développer leur client et à gérer son enregistrement. Elle apporte aussi une aide pour tout autre problème, lorsque les moyens du bord du système à disposition ou la documentation en ligne ne sont pas suffisants (voir ch. 7.2).</p>
<i>API publication & revocation</i>	Voir ch. 7.1.2.1
<i>IT security</i>	Voir ch. 7.1.2.1

Tableau 11 Capacités de l'étape de processus *Discover API*

²⁰ <https://www.ncsc.admin.ch/ncsc/fr/home/dokumentation/sicherheitsvorgaben-bund.html>

²¹ RS 120.73

7.1.2.3 Register Client

Capacité	Définition
<i>Identity & access management (IAM)</i>	La capacité IAM intervient à trois étapes de processus : <ul style="list-style-type: none"> - <i>Manage API lifecycle</i> - <i>Register client</i> - <i>Operate API</i> Voir ch. 7.1.2.1
<i>IT security</i>	Voir ch. 7.1.2.1

Tableau 12 Capacités de l'étape de processus *Register client*

7.1.2.4 Operate API

Capacité	Définition
<i>API service delivery</i>	L' <i>API service delivery</i> permet de proposer un service public numérique à l'extérieur. Il faut à cet effet exploiter les instances API et une architecture API dans plusieurs environnements d'exécution. Dans l'environnement fédéral, les environnements d'exécution sont normalement divisés en un environnement de référence, un environnement de réception et un environnement de production. La référence et la réception peuvent toutefois être réunies en un seul environnement d'exécution. Un environnement d'exécution peut être mis en place dans un nuage informatique, en local ou sous une forme hybride.
<i>API service consumption</i>	Le service de consommation API (<i>API service consumption</i>) est la capacité à consommer un service public numérique. Il faut à cet effet que les clients API et une infrastructure de clients API soient exploités dans plusieurs environnements d'exécution (voir capacité <i>API service delivery</i>). Mais les clients API sont également utilisés en dehors de l'environnement fédéral. Un environnement d'exécution peut être mis en place dans un nuage informatique, en local, sous une forme hybride ou encore sur un appareil mobile.
<i>Identity & access management (IAM)</i>	La capacité de gestion des identités et des accès (IAM) intervient à trois étapes de processus : <ul style="list-style-type: none"> - <i>Manage API lifecycle</i> - <i>Register client</i> - <i>Operate API</i> Voir ch. 7.1.2.1
<i>Traffic management</i>	La gestion du trafic correspond aux tâches suivantes : <ul style="list-style-type: none"> - l'équilibrage de charge qui consiste à répartir uniformément les accès aux instances API sur un même site ou à proposer des <i>API gateway</i> répartis sur plusieurs sites et affectés à un service spécialisé avec des points de sortie individuels²² ; - la priorisation, qui assure que les accès aux API sont priorisés selon un ordre défini ; - la limitation ou réduction, qui garantit des quotas par unité de temps pour les accès à un point de sortie de l'API ; - le changement d'échelle, qui met à disposition ou retire automatiquement des ressources en cours de fonctionnement en lançant ou en stoppant des instances ou des conteneurs. Ces tâches contribuent aussi à améliorer les temps de latence. Dans l'ensemble, elles garantissent la performance des API.
<i>Translation</i>	La <i>translation</i> est la capacité à traduire des protocoles API. Elle exclut clairement la conversion de formats de données (p. ex. JSON<>XML) et le traitement de charges utiles (p. ex. filtrage de données).
<i>Logging</i>	La capacité de collecte d'informations (<i>logging</i>) intervient à deux étapes de processus : <ul style="list-style-type: none"> - <i>Operate API</i> - <i>Analyse API operation & usage</i> Elle permet de journaliser des événements opérationnels ²³ et techniques en lien avec l'exploitation de services spécialisés et d'API. Les événements peuvent donc être générés aussi bien par le service spécialisé que par l' <i>API gateway</i> . La collecte d'informations couvre également la journalisation ²⁴ au sens à l'art. 10 de l'ordonnance du 14 juin 1993 relative à la loi fédérale sur la protection des données (OLPD) ²⁵ , qui a lieu exclusivement dans le service spécialisé.

²² Dans le cas d'un *cluster* de service spécialisé réparti sur plusieurs sites, les accès d'écriture API demandés au moyen d'*API gateway* répartis sur plusieurs sites peuvent soulever des questions de sauvegarde en miroir et de sécurité des transactions, auxquelles il faut répondre lors de la conception de l'architecture de solution.

²³ Les événements opérationnels sont des événements exceptionnels pertinents pour le SLA.

²⁴ En anglais : *audit trail*

²⁵ RS 235.11

Capacité	Définition
	<p>Les événements techniques sont en particulier les accès API du public, les accès contrôlés, acceptés ou refusés par les services IAM aux points de sortie de l'API et les événements générés dans le cadre de la journalisation. Pour les API qui demandent un enregistrement, l'identité de l'appelant doit notamment être enregistrée lors des appels API, en plus d'autres paramètres de journalisation pertinents.</p> <p>Il est dans tous les cas impératif de respecter les dispositions sur la protection des données en vigueur. C'est pourquoi les données de <i>log</i> servant à la journalisation sont gérées dans des systèmes de <i>logging</i> dédiés.</p> <p>La capacité <i>Logging</i> constitue la base des capacités <i>Monitoring</i> et <i>Reporting</i>. Les sous-capacités Reconnaissance, Filtrage et Envoi d'événements y sont attribuées à l'étape de processus <i>Operate API</i> et les sous-capacités Réception et Persistance d'événements à l'étape de processus <i>Analyse API operation & usage</i>. La capacité <i>Logging</i> permet d'intégrer la collecte d'informations de l'infrastructure API à celle plus générale de l'UA.</p>
<i>Monitoring</i>	Le monitoring (<i>monitoring</i>) est la capacité de réagir aux événements opérationnels et de les transmettre aux systèmes cibles ou aux acteurs. Il avertit l'organisation d'exploitation en cas d'événements opérationnels. Cette capacité permet d'intégrer le <i>monitoring</i> de l'infrastructure API à celui plus général de l'UA.
<i>IT security</i>	Voir ch. 7.1.2.1

Tableau 13 Capacités de l'étape de processus *Operate API*

7.1.2.5 Analyse API operation & usage

Capacité	Définition
<i>API reporting</i>	<p>Le <i>reporting</i> est la capacité de collecter, d'évaluer et de traiter les événements opérationnels et techniques pertinents. Il peut comporter des éléments d'intelligence artificielle permettant, par exemple, de faire de la reconnaissance de formes dans les événements et de proposer des possibilités d'optimisation. Une mise à l'échelle automatique est par exemple possible en combinaison avec l'étape de processus <i>Operate API</i>. Les rapports sont destinés aux FP qui exploitent des API et aux responsables API des UA. Ils permettent</p> <ul style="list-style-type: none"> - à l'<i>API product owner</i> de l'UA de chiffrer la valeur commerciale, d'améliorer l'API à l'aide des ICP et de l'orienter vers les besoins du partenaire ; - aux FP qui exploitent des API d'indiquer l'état de réalisation des SLO.
<i>API monetization</i>	La monétisation est la capacité à générer des revenus par l'exploitation d'API (voir ch. 7.3). Le but est non seulement de relier la consommation d'API à des modèles tarifaires pour déterminer les recettes, mais aussi d'établir des plans d'utilisation rattachés à des modèles tarifaires ou de créer la liaison avec les capacités de l'étape <i>Operate API</i> , pour notamment imposer des quotas d'accès lorsque les limites sont dépassées. La capacité de monétisation se fonde sur les rapports établis dans le cadre de la capacité <i>Reporting</i> et sur les ICP calculés.
<i>Logging</i>	<p>La capacité <i>Logging</i> intervient à deux étapes de processus :</p> <ul style="list-style-type: none"> - <i>Operate API</i> - <i>Analyse API operation & usage</i> <p>Voir ch. 7.1.2.4</p>
<i>IT security</i>	Voir ch. 7.1.2.1

Tableau 14 Capacités de l'étape de processus *Analyse API operation & usage*

7.2 Recommandation de conception Transparence

La recommandation de transparence contient des recommandations pour les trois capacités suivantes :

- *API documentation management*
- *API cataloging & searching*
- *API publication & revocation*

Ces capacités ont été réunies, car la recommandation de transparence n'est utile pour le public cible que si elle les combine les trois.

7.2.1 API documentation management

L'architecture API de la Confédération recommande de développer la capacité comme suit :

- Les artéfacts de la **documentation** relatifs au service public numérique (voir figure 15, modèle d'information) sont **élaborés** et **mis à jour** régulièrement. L'*API product owner* (voir ch. 7.4.1) crée et assure la maintenance des métadonnées et de la documentation API. Le service spécialisé concerné est responsable de la documentation spécialisée.
- La **documentation API** comprend toutes les informations nécessaires pour pouvoir décider d'utiliser l'API et pour s'y adresser correctement à partir d'un client API. Elle contient en particulier les spécifications API, les notes d'*API release* et les SLO. Les spécifications sont rédigées en anglais.
- La **documentation spécialisée** constitue la base de la documentation API, qui s'y réfère pour décrire les aspects techniques couverts par un *API release*. Elle comprend également la convention d'utilisation. Toutes ses parties mises à la disposition des partenaires et des utilisateurs finaux doivent être fournies dans les langues locales, selon les besoins. Les contenus linguistiques et techniques transmis par l'API en font notamment partie.
- Les **métadonnées API** constituent une description structurée du profil de l'API, qui peut être traitée par une machine. Elles servent à publier **en ligne** dans des **registres d'API** des informations sur l'existence et les caractéristiques des API dans l'environnement fédéral et sur les UA qui les proposent. Les métadonnées sont publiées dans un registre des API conformément aux dispositions de la LMETA.
- La **documentation spécialisée** et la **documentation API** sont également gérées et mises à disposition **en ligne**. Cette documentation décentralisée est également mise à disposition par le fournisseur qui gère la page d'atterrissage ou *landing page* (définie comme élément *RDF Property dcat:landingPage*²⁶ dans la norme eCH-0200 ; voir ch. 10.4).
- Dans la mesure du possible, la documentation API est générée **automatiquement**. Pour ce faire, des spécifications d'interface sont créées à l'aide de langages de définition d'interface (IDL) indépendants des langages de programmation. Sur cette base, des documentations, des modèles de code serveur/client, des codes de test et des cas de test peuvent être générés automatiquement. Les normes reconnues pour la spécification de RESTful API sont l'OpenAPI Specification (OAS)²⁷ ou l'AsyncAPI²⁸ pour les API avec des MEP asynchrones (orientés événements) (voir ch. 8.3.2).

7.2.2 API cataloging & searching

7.2.2.1 Métadonnées API

La capacité *API cataloging & searching* consiste à mettre à disposition des métadonnées API sous une forme standardisée. Pour ce faire, l'architecture API de la Confédération se fonde sur les normes concernant les normes ISA²⁹ et sur les normes eCH. Ces normes servent à décrire les métadonnées API de manière uniforme à l'aide d'un modèle, de manière à pouvoir échanger les métadonnées API entre les registres API.

Dans l'architecture API de la Confédération, les métadonnées API sont décrites des deux points de vue présentés dans le tableau 15. Le registre des API contient à la fois un catalogue de données et un catalogue de services. Les deux catalogues contiennent des entrées qui suivent une norme de métadonnées API spécifique, décrite dans le tableau 16.

²⁶ <https://www.w3.org/TR/vocab-dcat-3>

²⁷ <https://swagger.io>

²⁸ <https://www.asyncapi.com>

²⁹ Norme de la Commission européenne.

Points de vue sur les métadonnées API	Description	Norme de métadonnées API recommandée
Données (<i>Dataset</i>)	Les partenaires qui veulent surtout obtenir des données de l'administration fédérale cherchent les API dans le catalogue des données. En général, les données sont alors simplement lues par les clients API.	eCH-0200 Profil d'application DCAT pour les portails de données en Suisse ^{30, 31}
Services publics (<i>Public Services</i>)	Les partenaires qui veulent obtenir un service public numérique de l'administration fédérale ou réaliser une démarche administrative de cette manière cherchent les API dans le catalogue des services publics numériques. En général, les clients API permettent de créer, de lire, de modifier et d'effacer des données.	Norme ISA CPSV-AP ³²

Tableau 15 Points de vue sur les métadonnées API

Les services publics numériques et leurs API peuvent figurer dans le catalogue de données, dans le catalogue des services, ou dans les deux à la fois, selon le caractère du service concerné. L'inscription dans un catalogue ne dépend pas du type d'API (public ou partenaire). Les modèles de données des deux normes de métadonnées API se recoupent et peuvent donc également être réunis.

Norme	Description
eCH-0200 Profil d'application DCAT pour les portails de données en Suisse	<ul style="list-style-type: none"> - Norme eCH qui repose sur la norme ISA DCAT-AP³³ - Norme eCH-0200 qui concrétise la norme DCAT-AP et ajoute des attributs au modèle de données.
Norme ISA CPSV-AP	<ul style="list-style-type: none"> - Norme pour les métadonnées API visant à standardiser les métadonnées des services publics numériques à tous les niveaux administratifs de l'État. - Il n'y a pas (encore) de norme eCH correspondante.

Tableau 16 Normes de métadonnées API

Les deux normes de métadonnées API comprennent leur propre modèle de métadonnées, dont les classes et les attributs sont décrits à l'aide d'un vocabulaire RDF (voir ch. 10.4). Le vocabulaire RDF définit la signification (sémantique) des classes et des champs, et les relations entre eux. L'utilisation de ces normes permet d'échanger des métadonnées API entre des registres API ou de rechercher des métadonnées API réparties dans des registres des API fédérés.

7.2.2.2 Registre central des API et *landing pages*

L'architecture API de la Confédération prévoit la tenue d'un registre central des API dans l'environnement fédéral. Ainsi, lors de la publication d'un service public numérique, les métadonnées API sont ajoutées dans le registre des API. Lorsqu'un service public numérique est révoqué, les métadonnées sont effacées du registre des API. Selon les normes mentionnées, les métadonnées API ne contiennent pas d'information sur les versions d'API. Au lieu de cela, ce sont les *landing pages* référencées dans les métadonnées API et tenues de manière décentralisée qui contiennent les documentations API où sont décrits les *API releases* concernant les différentes versions. Le registre des API est accessible au public. Les métadonnées API y sont publiées conformément aux dispositions de la LMETA.

Si la LMETA ne permet pas la publication des métadonnées d'une API, il est recommandé au fournisseur de gérer une *landing page* accessible uniquement à un cercle d'utilisateurs contrôlé. L'accès à ces pages est alors strictement réglementé, car la documentation API risque de contenir des informations sensibles sur le service concerné. Si des raisons économiques s'opposent à la gestion d'une *landing page*, le fournisseur peut y renoncer.

³⁰ <https://www.ech.ch/index.php/fr/standards/60609> (norme en cours de révision pour l'adapter à la norme ISA, état le 30.11.2021)

³¹ Le portail <https://opendata.swiss>, qui répertorie les données en libre accès (OGD) de l'administration publique suisse, utilise cette norme.

³² https://ec.europa.eu/isa2/solutions/core-public-service-vocabulary-application-profile-cpsv-ap_en

³³ https://ec.europa.eu/isa2/solutions/dcat-application-profile-data-portals-europe_en

Les partenaires et les FP mandatés accèdent aux *landing pages* en utilisant l'un des deux services standard IAM Confédération du secteur TNI (voir eIAM Définitions³⁴ « Qu'est-ce que le portail SSO DFJP et comment est-il lié à l'eIAM ? ») :

- l'eIAM³⁵ est le système central de gestion des accès et des autorisations de l'administration fédérale pour les applications web et les applications mobiles natives, géré par l'Office fédéral de l'informatique et de la télécommunication (OFIT)
- le portail SSO³⁶ est le portail à signature unique (*single sign on*) du Département fédéral de justice et police (DFJP) pour les applications des autorités judiciaires et des organisations affiliées, géré par le Centre de services informatiques du DFJP (CSI-DFJP)

Un système de rôles élaboré conjointement avec eIAM ou le portail SSO est intégré dans les *landing pages*, de sorte que les rôles décrits au ch. 7.4 puissent être assumés par les fournisseurs et les partenaires.

7.2.2.3 Registre des API fédéré

Si nécessaire, le fournisseur peut également gérer son propre registre des API décentralisé. Il est alors recommandé de le relier au registre central des API. Ainsi, le registre des API fédéré sera perçu par l'utilisateur comme un registre central.

Dans un registre des API, les services publics numériques sont recherchés en utilisant les métadonnées API. Chaque champ de métadonnées API peut alors servir de critère de recherche. Il faut dès lors s'assurer qu'une recherche de service public numérique lancée dans le registre fédéré soit effectuée dans toutes les métadonnées API de la Confédération. Deux actions permettent d'atteindre ce résultat.

- Le registre central des API reçoit périodiquement des copies des métadonnées API des registres API décentralisés.
- En partant d'un registre maître défini, la recherche est effectuée dans tous les registres des API décentralisés.

7.2.2.4 Renvoi à des portails de services publics numériques sur le web

Le modèle de données fourni par la norme CPSV-AP permet d'enregistrer l'adresse URL d'un portail de services publics numériques³⁷, de sorte que le registre des API met à disposition non seulement les informations requises pour utiliser des services publics numériques, mais aussi des informations sur les portails qui en proposent sur le web. Le catalogage de ces portails web doit répondre aux exigences de l'architecture fédérale des portails (voir ch.2.1.2).

7.2.3 API publication & revocation

L'architecture API de la Confédération recommande de concevoir la capacité de publication et révocation (*publication & revocation*) de la manière suivante.

- L'*API product owner* publie les métadonnées dans le registre des API et la documentation en ligne de l'*API release* (voir ch. 7.2.1) sur la *landing page*. Il peut déléguer cette tâche à un *publisher*, c'est-à-dire à une personne physique chargée de représenter son UA.
- L'*API product owner* s'assure, par l'attribution de rôles dans eIAM, que toutes les personnes physiques qui ont besoin d'accéder à la *landing page* obtiennent cet accès conformément à leur rôle (voir ch. 7.4).
- Lorsqu'un *API release* est publié ou révoqué, l'*API product owner* est responsable des tâches suivantes.
 - Il publie, modifie et supprime les métadonnées d'API. Les métadonnées d'API sont des données générales qui ne renvoient pas à une version de l'API (voir ch. 7.2.2.2), mais elles peuvent être modifiées lors de la publication d'un *release*.
 - Il publie, modifie et supprime la documentation en ligne de l'*API release* sur la *landing page*.
 - Lors de la publication, il assure le fonctionnement des instances API en concluant ou en adaptant des conventions d'utilisation. En cas de révocation, il met fin à ces conventions.

³⁴ https://www.eiam.admin.ch/pages/fleiamglossary!pub_fr.html?c=fleiamglossary!pub&l=fr&ll=1

³⁵ <https://www.eiam.admin.ch>

³⁶ <https://www.isc-ejpd.admin.ch/isc/fr/home/dienstleistungen/sso-portal.html>

³⁷ La classe optionnelle *Channel* représente les canaux d'utilisation alternatifs aux API, tels que les portails de services publics. L'attribut *Identifier* permet d'ajouter une URL.

- Lors de la publication, il assure le déploiement des instances API sur l'environnement de production et les supprime lors de la révocation.
- Il met en place, gère et dissout l'organisation d'assistance pour les rôles de développeur et d'utilisateur du client API (côté du partenaire) (voir ch. 7.4.2).

7.3 Recommandation de conception *API monetization*

7.3.1 Cadre juridique

Le Conseil fédéral a approuvé le 30 novembre 2018 la deuxième stratégie OGD suisse 2019-2023. Cette stratégie permet de mettre à disposition gratuitement les données ouvertes et librement utilisables de l'administration fédérale.

Pour certaines API qui prévoient le transfert d'importants volumes de données ou demandent une qualité de service élevée, des coûts peuvent être imputés au partenaire. Un tel transfert de coûts est soumis à une ordonnance du Conseil fédéral³⁸. Dans ce cas, les UA concernées peuvent demander un émolument pour la mise à disposition de services publics numériques conformément à l'ordonnance.

7.3.2 Modèles de monétisation

Dans l'architecture API Confédération, on distingue deux modèles de monétisation des API, le modèle direct et le modèle indirect³⁹ (voir tableau 17).

Modèle de monétisation API	Définition
Modèle indirect	Un modèle de monétisation de l'API indirect repose sur le principe que l'interaction API entre le fournisseur et le partenaire génère de la valeur pour les deux parties. Cependant, le fournisseur finance entièrement le développement et le fonctionnement (du moins dans la phase initiale de fonctionnement). La monétisation résulte des coûts économisés grâce à l'API mise à disposition pour fournir un service public.
Modèle direct	Un modèle de monétisation de l'API direct repose sur le fait que le partenaire est le premier bénéficiaire de la mise à disposition d'une API et qu'il prend donc entièrement en charge les coûts de développement et de fonctionnement.

Tableau 17 Modèle de monétisation de l'API direct et indirect

La capacité *Monetization* suit un processus de développement par étape. La première étape est la phase qui suit la mise en service de l'API, au cours de laquelle le fournisseur acquiert de l'expérience avec le modèle de monétisation indirect, tout en analysant les données d'utilisation et en contrôlant le respect des SLO. L'étape suivante est un mélange entre les modèles direct et indirect, au cours de laquelle une partie des coûts pour la fourniture de prestations est transférée au partenaire. Ce mélange peut évoluer peu à peu vers un modèle de monétisation direct. Dès que les revenus provenant de l'exploitation de l'API couvrent entièrement le prix de développement et de fonctionnement, l'étape de monétisation directe est atteinte. L'UA définit l'étape à atteindre en fonction du critère de la valeur partagée (voir ch. 7.3.3) et des ordonnances pertinentes.

³⁸ Exemple : ordonnance sur la géoinformation (OGéo, RS 510.620)

³⁹ Voir l'article sur les barrières payantes sur le site <https://www.gartner.com/en/documents/3903563/choose-the-right-api-monetization-and-pricing-model>

7.3.3 Modèles tarifaires

Différents modèles tarifaires peuvent être utilisés pour la monétisation de l'API. Le tableau 18 présente un aperçu des modèles tarifaires envisageables⁴⁰. Il est possible de combiner plusieurs modèles.

Modèle tarifaire	Description
Utilisation gratuite	<ul style="list-style-type: none">- L'utilisation des données publiques est souvent gratuite. Proposer des services publics numériques gratuits peut parfois servir à stimuler la vente de services publics payants (vente croisée).- Il faut dans tous les cas respecter la convention d'utilisation. En cas de surcharge, une réduction des accès par unité de temps peut être prévue.
Paiement par accès API	<ul style="list-style-type: none">- Un prix unitaire fixe est facturé pour chaque accès API.
Abonnement	<ul style="list-style-type: none">- Le partenaire souscrit un abonnement mensuel pour une utilisation forfaitaire. Il reçoit des alertes ou doit payer un supplément en cas de dépassement du forfait.- L'abonnement peut se fonder sur un nombre d'accès ou un volume de données.
Échelonnement	<ul style="list-style-type: none">- L'utilisation est libre jusqu'à une certaine limite. Ensuite, elle est payante (de manière échelonnée). Il peut y avoir plusieurs échelons.- La limite peut être définie en fonction d'un nombre d'accès ou d'un volume de données.- En combinant l'échelonnement avec le modèle de paiement par accès, il est possible d'obtenir un meilleur prix unitaire à l'échelon suivant.- Pour ce modèle, il faut mettre en place un tableau de bord permettant au partenaire de voir sa consommation par unité de temps.
Répartition des revenus	<ul style="list-style-type: none">- Les revenus provenant de l'utilisateur final sont répartis entre le fournisseur et le partenaire.- Le partenaire cède au fournisseur une partie des revenus perçus lorsque l'utilisateur final paie pour utiliser l'application client.- Le fournisseur cède au partenaire une partie des revenus perçus lorsque l'utilisateur final paie pour utiliser le service public numérique.

Tableau 18 Modèles tarifaires pour la monétisation de l'API

7.3.4 Conditions

La monétisation de l'API pose des exigences accrues à l'exploitation, car le prix payé pour l'utilisation de l'API implique de garantir la prestation. Dans tous les cas, il faut respecter les SLO annoncés. Toute UA qui entend mettre en place une monétisation d'API directe doit remplir les conditions de développement et de fonctionnement suivantes⁴¹ :

- **Les API sont conçues de manière à être bien acceptées par les partenaires**
L'API design est particulièrement important, car les API utilisées dans le cadre d'un service public numérique doivent permettre un processus axé de bout en bout sur partenaire et sur la démarche administrative concernée. L'accent est mis sur l'automatisation afin de garantir la facilité d'utilisation.
- **Les développeurs doivent pouvoir accéder aux API sans aucun obstacle**
L'accès aux API à des fins de développement est aussi simple que possible. À cet effet, les développeurs ont accès à la documentation API et peuvent enregistrer facilement des clients de test dans la boîte à sable contenant les versions tests et les modèles de monétisation des API, qui encouragent leur utilisation.
- **Les prestations des API sont clairement définies et réalisables**
La monétisation des API demande des SLO fiables à tout moment.
- **Des services d'assistance sont fournis pour l'utilisation de l'API**
Tout au long du cycle de vie d'une API, les partenaires ont la garantie de bénéficier d'une assistance individuelle lors de l'utilisation de l'API grâce à des canaux de communication appropriés, en particulier lors d'une migration forcée vers une nouvelle version de l'API en raison d'une rupture imminente de la rétrocompatibilité.

Les infrastructures API qui doivent répondre à l'exigence de monétisation directe doivent proposer les fonctions suivantes :

⁴⁰ Voir l'article sur les accès payants

⁴¹ Voir l'article sur les accès payants

- établissement de plan d'utilisation pour les API et affectation des listes de prix en fonction de l'utilisation
- mise à disposition de fonctions en libre-service pour l'enregistrement des clients et le choix du modèle tarifaire souhaité
- contrôle de l'utilisation pour chaque client dans un tableau de bord
- limitation de l'utilisation de l'API en cas de dépassement de la limite fixée dans le modèle tarifaire
- journalisation des accès API et modifications apportées aux modèles tarifaires et aux SLA
- facturation de l'utilisation selon le modèle tarifaire choisi

7.4 Description des rôles

Les chapitres suivants sont consacrés à la description des rôles d'*API product owner*, d'architecte API, de développeur API et d'exploitant API, et à leur attribution aux domaines infrastructure API, instance API et client API. Le domaine architecture API englobe tous les services et composants systèmes généraux (plateforme technique) mis à disposition des UA pour la gestion et l'exécution de leurs instances API. Les rôles concernant les données (propriétaire de données et détenteur local de données) ne sont pas décrits, car ils figurent dans le modèle de rôles du programme de gestion nationale des données (NaDB)⁴² et ne sont pas spécifiques aux API.

Selon le modèle de rôles, c'est par exemple de propriétaire de données qui décide, en se fondant sur une base légale, de publier des données sous forme d'OGD, alors que l'*API product owner* est chargé d'appliquer la décision. Le détenteur local de données est notamment responsable de la représentation correcte et complète des méta-données API dans le registre des API. Ces deux rôles travaillent donc étroitement avec l'*API product owner*.

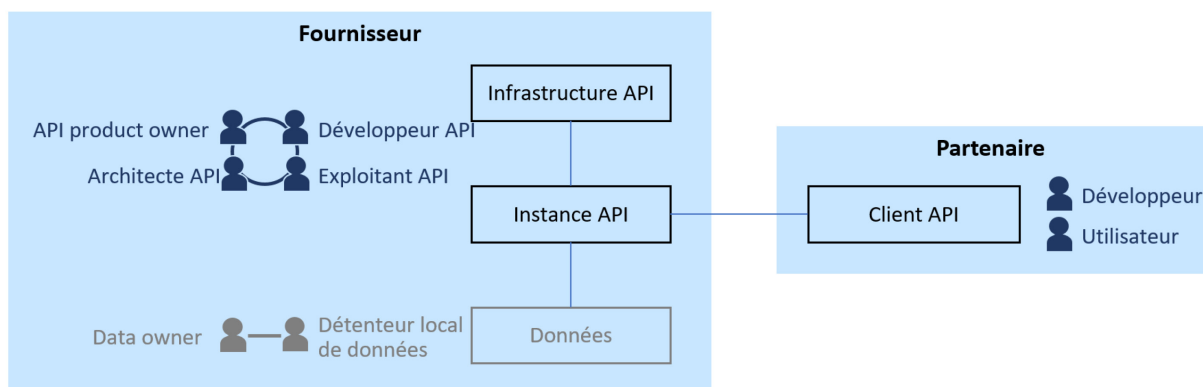


Figure 11 Rôles spécifiques aux API

Ces rôles peuvent être tenus à différents niveaux administratifs. Il peut ainsi y avoir des développeurs d'API dans les UA ou au niveau du département, par exemple dans le service du FP départemental. Les rôles spécifiques aux API ne sont en général pas assez importants pour qu'un poste spécifique leur soit dévolu entièrement. De plus, ils impliquent des tâches très similaires à celles du gestionnaire d'applications. Les rôles spécifiques aux API peuvent donc être occupés efficacement en les ajoutant aux attributions du gestionnaire des applications (un responsable d'applications peut par ex. également être *API product owner*).

7.4.1 Rôles du fournisseur

Les rôles dans le domaine de l'infrastructure API sont en général attribués au niveau du département ou de la Confédération, car il ne faut pas que chaque UA mette en place sa propre architecture. Les rôles dans le domaine des instances API sont reliés à des aspects spécialisés et aux applications. Ils se situent donc principalement au niveau des UA. Ces rôles peuvent être attribués pour chacune des instances API. Mais il est souvent plus judicieux, en fonction de la taille de l'UA, d'attribuer les rôles d'architecte API, de développeur API et d'exploitant API une seule fois pour toutes les instances de l'UA.

Rôle	Responsabilité, tâches, compétences
<i>API product owner</i> (infrastructure API, instance API)	- est responsable du développement durable et de l'exploitation de l'infrastructure API (plateforme technique) ou d'une ou plusieurs instances API. Cela comprend également la gestion d'application de l' <i>API gateway</i> .

⁴² <https://www.bfs.admin.ch/bfs/fr/home/nadb/nadb.assetdetail.14965606.html> (document en allemand)

Rôle	Responsabilité, tâches, compétences
	<ul style="list-style-type: none"> - est responsable du service numérique fourni aux partenaires. - est responsable de la description correcte et exhaustive des contenus et de la structure des API, et de leur qualité dans le registre des API et dans la documentation en ligne. - représente les intérêts des partenaires et des autres parties prenantes. - collabore avec l'architecte API et le développeur API. - identifie et formule les exigences posées à l'infrastructure API ou aux instances API. - priorise les exigences et décide de leur mise en œuvre (après consultation). - propose le budget annuel de développement et de fonctionnement de l'infrastructure API ou des instances API. - conclut des contrats (SLA) avec l'exploitant API. <p><i>Voir également le rôle de responsable de produit dans la norme P000 Processus informatiques de l'administration fédérale⁴³</i></p>
Architecte API (infrastructure API, instance API)	<ul style="list-style-type: none"> - développe et documente une architecture de solution API conforme. - participe et revoit des incréments de l'architecture API de la Confédération. <p><i>Voir également le rôle d'architecte de solution dans la norme P000 Processus informatiques de l'administration fédérale</i></p>
Développeur API (infrastructure API, instance API)	<ul style="list-style-type: none"> - développe l'infrastructure API ou des instances API conformes.
Exploitant API (API Infrastructure, instance API)	<ul style="list-style-type: none"> - assure l'exploitation informatique (y compris l'assistance) de l'infrastructure API ou des instances API <p><i>Voir également le processus P06 Exploiter l'infrastructure et les services informatiques dans la norme in P000 Processus informatiques de l'administration fédérale</i></p>

Tableau 19 Rôles du fournisseur

7.4.2 Rôles du partenaire

Les rôles décrits pour le partenaire sont les deux rôles qui impliquent un contact avec la Confédération.

Rôle	Responsabilité, tâches, compétences
Développeur de client API	- développe du côté du partenaire le client API et la connexion / l'intégration dans une API de l'administration fédérale.
Utilisateur de client API	- est une personne ou un système autorisé qui a accès aux services publics numériques ou aux données grâce à l'API.

Tableau 20 Rôles du partenaire

7.5 Gouvernance API

7.5.1 Modèle de gouvernance API

Le pilotage des API de l'administration fédérale se conforme au modèle de gouvernance informatique fondé sur l'OTNI⁴⁴. Les départements, leurs UA et la Chancellerie fédérale sont responsables de leurs propres infrastructures API et de leurs propres API, et respectent les directives générales. Le secteur TNI est responsable de l'architecture API de la Confédération et émet le cas échéant des directives (conformément à l'art. 17 OTNI), qui confèrent un caractère obligatoire à l'architecture API ou à certaines de ses parties.

⁴³ Intranet : https://intranet.dti.bk.admin.ch/isb_kp/fr/home/ikt-vorgaben/prozesse-methoden/p000-informatikprozesse_in_der_bundesverwaltung.html

⁴⁴RS 172.010.58

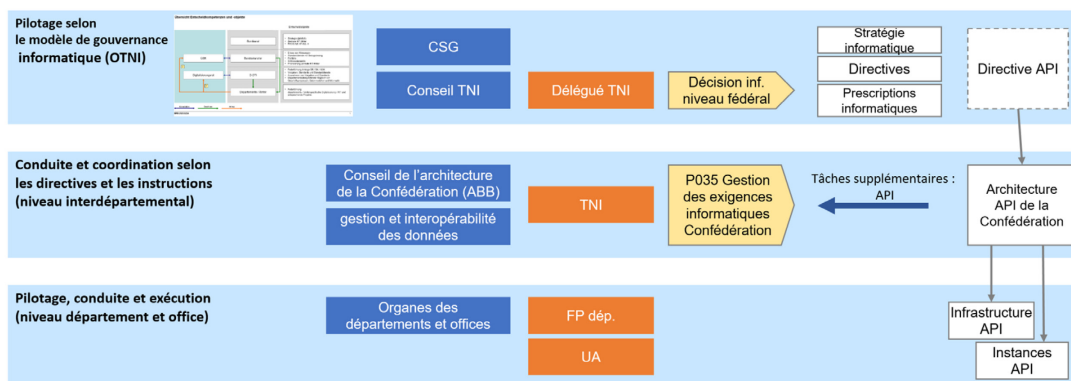


Figure 12 Gouvernance API selon le modèle de gouvernance informatique

La figure 12 décrit la gouvernance API de la Confédération à trois niveaux :

- Le **pilotage** selon le modèle de gouvernance informatique (fondé sur l'OTNI) : lorsqu'il faut prendre des décisions générales concernant l'architecture API de la Confédération et son application (notamment en raison d'un lien avec la stratégie informatique, le portefeuille, les ressources informatiques centrales, les directives informatiques, ou autre), le pilotage est assuré selon le modèle de gouvernance informatique grâce aux organes, aux processus et aux instances décisionnelles décrits dans l'OTNI.
- La **conduite et la coordination** selon les prescriptions et directives (niveau interdépartemental) : c'est à ce niveau qu'ont lieu la coordination technique et la consultation sur les problèmes interdépartementaux d'API (par ex. sur l'architecture API de la Confédération ou sur l'infrastructure API commune). Les organes et les processus en place sont utilisés, notamment le processus P035 Gestion des exigences et des directives au niveau de la Confédération, qu'utilisent les départements, la Chancellerie fédérale, les FP ainsi que les comités et organes spécialisés afin de soumettre leurs exigences informatiques pour approbation ou décision. Si nécessaire, l'ABB ou le comité d'experts sur la gestion et l'interopérabilité des données (en cours de création) sont consultés. Aucune organisation spécifique n'est mise en place sur le thème des API, qui est ajouté aux organes et aux processus existants.
- Le **pilotage, conduite et exécution (niveau département et office)** : Les départements et la Chancellerie fédérale règlent la gouvernance informatique dans leur sphère de compétence. Cela concerne également les infrastructures API et les API. Le développement et l'exploitation des architectures de solutions API sont ainsi délégués aux départements et aux UA. Pour favoriser les échanges entre les départements et l'interopérabilité, les rôles visés au ch 7.4 peuvent être utilisés à titre de recommandation.

7.5.2 Développement de l'architecture API de la Confédération

Les responsabilités pour le développement de l'architecture API de la Confédération sont attribuées comme suit :

- Le secteur TNI est responsable du développement de l'architecture API de la Confédération.
- Il se coordonne avec les *API product owner* et les architectes API des départements (y c. FP) pour créer de nouveaux incréments de l'architecture API de la Confédération.
- Il établit, le cas échéant, des directives qui confèrent un caractère obligatoire à l'architecture API ou à certaines de ses parties.
- L'ABB et le comité d'experts sur la gestion et l'interopérabilité des données sont consultés en cas de modification majeure de l'architecture API de la Confédération et le conseil TNI en cas de modification des directives.
- Les *API product owner* et les architectes API de tous les niveaux diffusent les informations sur l'architecture API de la Confédération et les modifications qui y sont apportées dans leur sphère de compétence.

L'architecture API de la Confédération ne doit être modifiée qu'avec prudence dans une perspective à long terme, afin de garantir la stabilité requise pour atteindre les objectifs architecturaux.

8 Architecture de données

8.1 Modèle d'information

L'architecture de données est décrite au moyen du modèle d'information. Le modèle d'information est représenté sous la forme d'un diagramme de classes UML simplifié. Il décrit les objets d'information de l'architecture de référence API et les relations entre ceux-ci. Les objets d'information sont répartis en trois catégories : acteurs, objets de service et objets de données. Le tableau 21 décrit les types d'objets d'information et le tableau 22 montre de quels types les objets d'information relèvent.

Type d'objet d'information	Description
Acteurs	<ul style="list-style-type: none"> - sont actifs - interagissent avec d'autres acteurs - participent à la définition, à la consommation et au traitement d'objets de données - utilisent des objets de service
Objets de service	<ul style="list-style-type: none"> - présentent un comportement clairement défini - produisent, consomment ou traitent des objets de données - utilisent d'autres objets de service
Objets de données	<ul style="list-style-type: none"> - sont passifs - peuvent être des agrégations d'objets de données - peuvent faire l'objet d'opérations

Tableau 21 Types d'objets d'information

Acteurs	Objets de service	Objets de données	
<ul style="list-style-type: none"> - Fournisseur API - Fournisseur FP - Partenaire API - Partenaire FP - Sujet 	<ul style="list-style-type: none"> - Service de catalogue - Service d'approvisionnement - Service Données de référence - Services IAM - Client - Service <i>Gatekeeper</i> - Service Ressources - Service <i>Logging</i> - Service <i>Monitoring</i> - Service <i>Reporting</i> - Service <i>Monetization</i> 	<ul style="list-style-type: none"> - Objet partenaire - Rôle partenaire - <i>API release</i> - <i>API version ID</i> - <i>API code</i> - Configuration API - <i>API build</i> - Métadonnées API - Documentation API - Documentation spécialisée - Convention d'utilisation - Identité 	<ul style="list-style-type: none"> - Rôle API - Jeton sécurisé - Ressource - Évènement - Évènement opérationnel - Évènement technique - Indicateur clé de performance (ICP) - Modèle tarifaire - Rapport - Contrôle de prestations - Rapport sur la valeur commerciale - Rapport ICP - Compte

Tableau 22 Objets d'information par type

Le modèle d'information attribue les objets d'information aux étapes du processus. La figure 13 et la figure 14 présentent le modèle d'information en deux parties. Les Figures sont reliées par l'étape de processus *Operate API*. Par souci de simplification, le modèle d'information présente le sous-ensemble des objets d'information qu'il faut connaître pour comprendre les relations entre les objets d'information⁴⁵. Le ch. 8.2 donne un aperçu des liens entre les objets d'information.

La figure 15 et la figure 16 complètent le modèle d'information en y ajoutant des objets de données des étapes de processus *Manage API lifecycle* et *Analyse API operation & usage*.

⁴⁵ Pour des raisons de clarté, les objets de données *API Build*, *API Configuration*, objet partenaire et rôle partenaire sont représentés deux fois dans la figure 13.

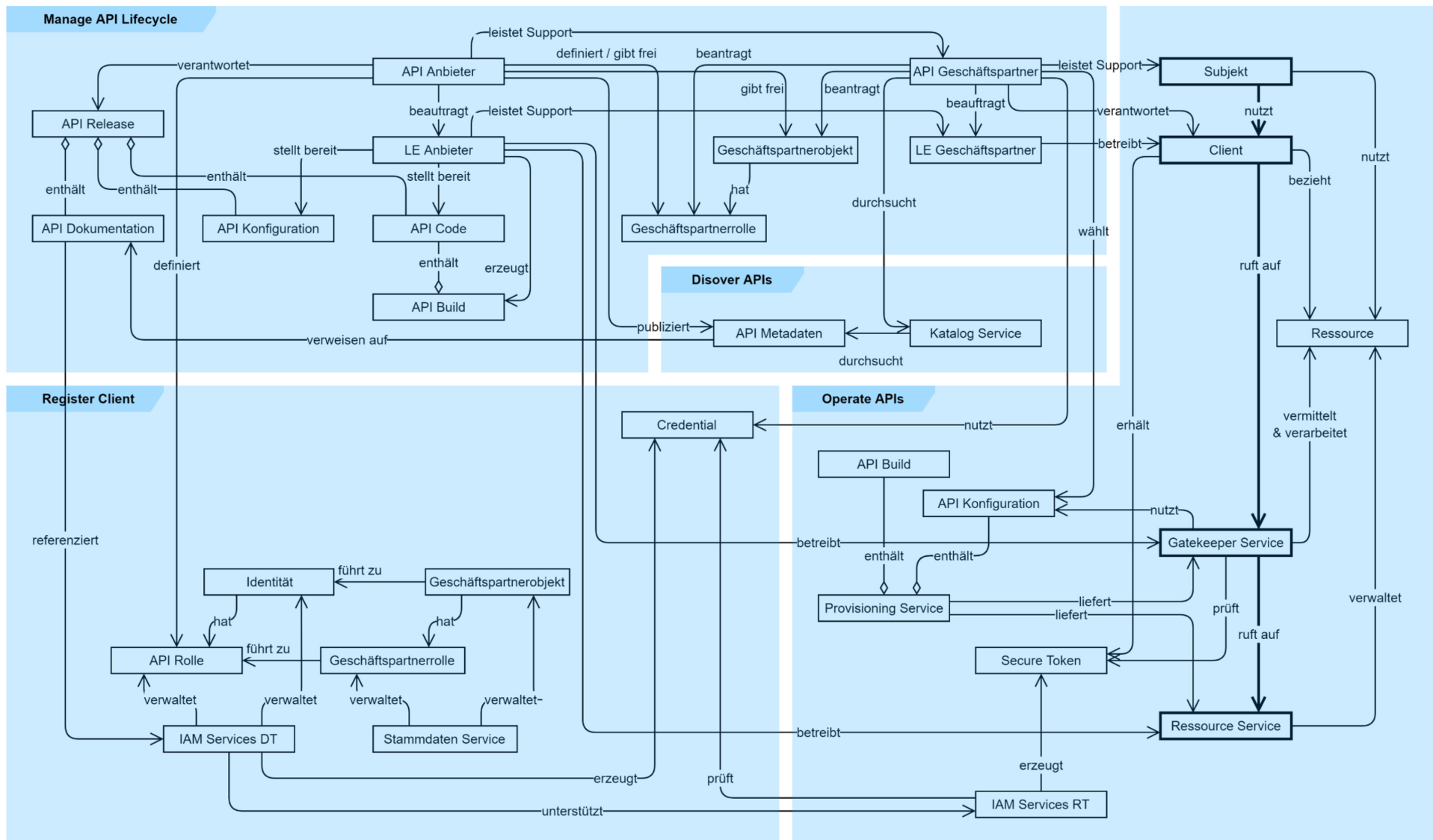


Figure 13 Modèle d'information -partie 1

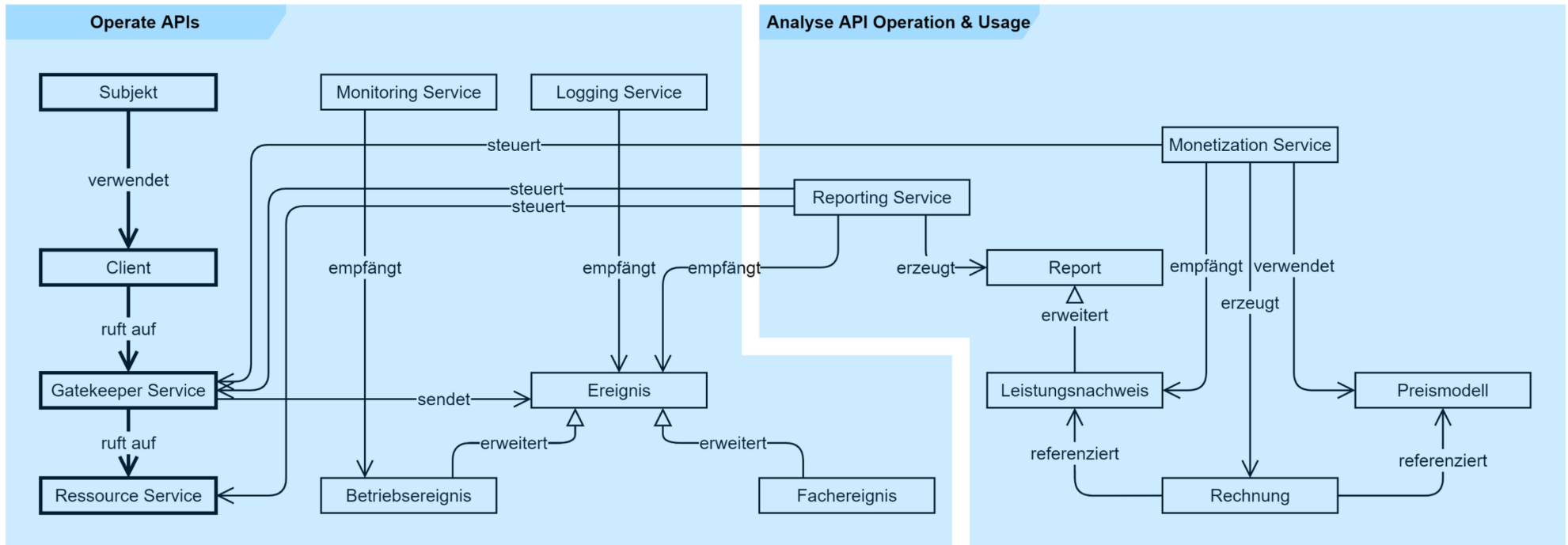


Figure 14 Modèle d'information -partie 2

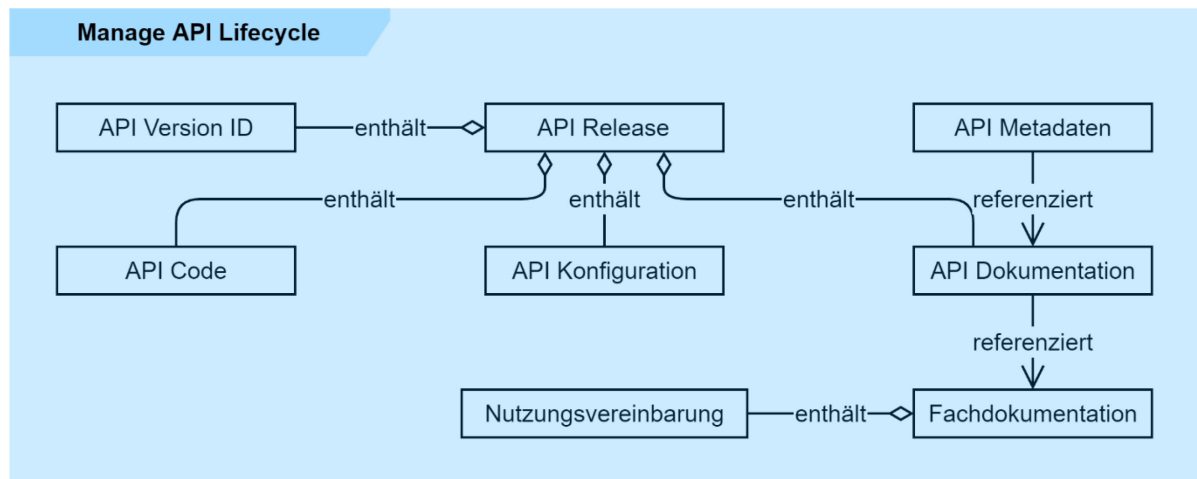


Figure 15 Modèle d'information avec objets de données de l'étape de processus *Manage API lifecycle*

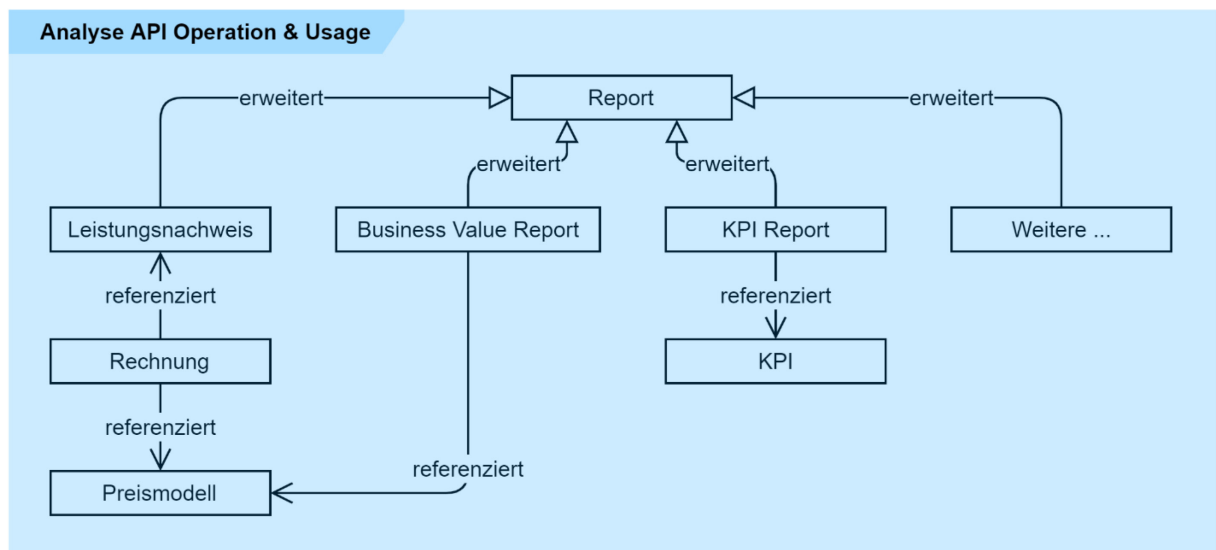


Figure 16 Modèle d'information avec objets de données de l'étape de processus *Analyse API operation & usage*

8.2 Description des objets d'information

Ce chapitre décrit les relations fondamentales entre les objets d'information de l'architecture API de la Confédération. En outre, le ch. 10.2 fournit une description détaillée de chaque objet d'information.

L'essence même d'un service public numérique est de permettre à un client d'accéder à un service Ressources proposant des prestations et des données. L'accès se fait au moyen d'un service de contrôle d'accès (*Gatekeeper*). Grâce aux services IAM, le service *Gatekeeper* protège les ressources du service Ressources⁴⁶.

Le fournisseur API est l'UA responsable du développement et du fonctionnement du service public numérique. Le fournisseur API peut déléguer le développement ou le fonctionnement du service public numérique à un FP. Le partenaire API est le partenaire qui utilise le service public numérique. Il peut s'agir soit d'une personne, soit d'une entreprise, soit d'une autre UA de la Confédération. Le partenaire API peut lui aussi déléguer le développement ou le fonctionnement du client à un FP.

Un *API release* est une version exécutable d'une API, publiée sur un service de catalogue et mise à disposition des partenaires API pour être utilisée. Les *API releases* comprennent habituellement les objets de données suivants : l'*API code*, la configuration API et la documentation API.

⁴⁶ Le service Ressources met à disposition des données et des prestations, ce qui, dans le contexte du modèle d'information, est considéré dans les deux cas comme une ressource. Il ne faut pas faire la confusion avec l'orientation vers les ressources et donc vers les données de la REST API.

Les API *builds* sont des progiciels livrables. Ils constituent la base de l'infrastructure API et, potentiellement, des instances API (voir ch. 9.1). Les API *builds* sont soit disponibles sur Internet soit gratuitement soit contre paiement ou générés à partir de l'API *code*. Une configuration API paramètre à la fois le service *Gatekeeper* et le service Ressources. Le service d'approvisionnement livre aussi bien les API *builds* que les configurations API. Si une couche API ou un composant système pour l'intégration API est livré pour le service spécialisé (voir figure 20), le service Ressources est également fourni avec une instance API ou une configuration API.

Le service de catalogue est l'élément central du registre des API. Il gère les métadonnées API publiées par les UA et offre une fonction de recherche.

Les API publiques avec enregistrement et les API partenaires exigent des clients qu'ils s'identifient au moyen d'identités. Des rôles API sont normalement attribués à ces identités pour accéder aux ressources. Dans le cas des API partenaires, les objets partenaires et les rôles partenaires, qui font partie de la gestion des données de référence, sont requis pour la création d'identités et l'attribution de rôles API à ces identités. Les identités et les attributions de rôles doivent être demandées et validées.

Les services IAM gèrent les identités, les rôles API et les attributions de rôles API aux identités. Ils génèrent des identifiants et des jetons sécurisés et contrôlent les accès.

Le service public numérique fonctionne avec des services *Logging* et *Monitoring*. Il fournit également des informations opérationnelles et techniques aux services *Reporting* et *Monetization*. Les services *Monetization* permettent de quantifier, à l'aide de rapports et de modèles tarifaires, les revenus générés par l'utilisation de l'API. Dans les systèmes d'exploitation matures, le service *Reporting* et le service *Monetization* permettent d'influer de manière dynamique sur la stabilité du fonctionnement du service public numérique.

8.3 Recommandation de conception *API Design*

8.3.1 Exigences d'entreprise

Lorsque l'on envisage de développer un service public numérique, la première étape consiste à développer l'architecture d'entreprise. Le plus souvent, l'architecture d'entreprise du service public numérique peut s'inspirer de celle du service proposé dans le monde réel. Il en résulte des artefacts d'architecture tels que la carte des capacités, les descriptions de processus, les systèmes de rôles et les exigences relatives à l'entreprise, ce qui permet ensuite d'élaborer des modèles de données et des processus pour l'utilisation des API ou de modéliser des rôles IAM.

Cette architecture, éventuellement associée à des API déjà réutilisables dans l'environnement fédéral, dictera le choix des conventions à conclure pour l'échange de données (voir ch. 8.3.2).

8.3.2 Conventions d'échange de données

Dans le cadre de la capacité *API design*, l'architecture API de la Confédération fournit un cadre réglementaire pour les conventions d'échange de données entre le client API et le service spécialisé. Ce cadre réglementaire couvre le champ des caractéristiques qu'une API peut avoir. Les conventions doivent être choisies en fonction des exigences d'entreprise et des normes techniques déterminantes du domaine d'activité⁴⁷ dans le cadre duquel le service sera fourni.

Dans le cadre de l'architecture API de la Confédération, les conventions s'articulent autour des axes suivants :

- Protocole API
- Type d'interface
- MEP
- Format de données
- Type de message

Dans la pratique, lors du développement d'API, certaines combinaisons de types d'interface, de formats de données, de types de messages, de MEP et de protocoles API sont plus répandues que d'autres. Il peut aussi arriver que le choix d'une certaine dimension restreigne le nombre de combinaisons possibles. L'architecture API de la Confédération autorise en principe toutes les combinaisons qui répondent aux exigences d'entreprise et qui peuvent être mises en œuvre avec un rapport coûts-bénéfice favorable. Au final, ce sont les développeurs API, et donc

⁴⁷ Par ex. la norme eCH-0051 pour l'échange de données dans le domaine de la police ou la norme eCH-0056 pour les profils d'application des géoservices.

le marché, qui décident du succès des API. Il est donc important de concevoir une API qui corresponde au mieux au cas d'espèce. Les chapitres suivants proposent des recommandations à ce sujet :

Le tableau 23 présente les conventions recommandées en matière d'échange de données⁴⁸. Ces recommandations sont des regroupements courants de combinaisons de conventions et doivent donc être considérées comme des bonnes pratiques. Les différentes caractéristiques fixées dans les conventions sont décrites au ch. 10.3. Lors de la conception d'une architecture de solution API, ces conventions doivent être conclues non seulement pour le canal de communication le plus évident allant du client au service Ressources en passant par le service *Gatekeeper*, mais aussi, par exemple, pour les services IAM ou le service *Logging*.

Protocole API	Type d'interface	MEP	Format de données	Type de message
HTTP	Ressources Méthode	requête-réponse	Aucune restriction	Requête Réponse
SOAP/WSDL	Ressources Méthode	Aucune restriction	XML	Requête Commande Réponse
gRPC	Ressources Méthode	Aucune restriction	Format binaire ⁴⁹	Aucune restriction
WebSockets	Ressources	Aucune restriction	Aucune restriction	Aucune restriction
MQTT	Ressources	Messagerie	Aucune restriction	Requête Réponse Évènement
AMQP	Ressources	Messagerie	Format binaire	Requête Réponse Évènement

Tableau 23 Bonnes pratiques concernant les conventions d'échange de données

Dans les chapitres suivants et au ch. 10.3, le terme client désigne d'une manière générale les objets d'information (objets de service) qui émettent des requêtes et reçoivent des réponses, et le terme serveur ceux qui reçoivent les requêtes et émettent des réponses. Ces termes sont utilisés à plusieurs reprises dans le modèle d'information.

8.3.2.1 RESTful API⁵⁰

Les RESTful API remplissent les conditions du style d'architecture REST. Elles ciblent les ressources. Ce type d'API est le plus courant sur Internet et comprend aussi la technologie HTML via HTTP. Toutes les combinaisons de conventions d'échange de données figurant dans le tableau 23 qui remplissent les conditions du style d'architecture REST peuvent être considérées comme des RESTful API. Il s'agit des six conditions du tableau 24, dont une est facultative.

L'architecture API de la Confédération recommande d'opter pour des RESTful API lorsqu'il s'agit de réaliser des services spécialisés axés sur les ressources à intégrer dans des services publics numériques.

Condition	Description
<i>Client/server architecture</i>	<ul style="list-style-type: none"> - Les RESTful API sont toujours intégrées dans une architecture client-serveur, où le client envoie une requête au serveur et reçoit une réponse de celui-ci. - Les rôles du client et du serveur sont clairement définis et statiques.
<i>Uniform Interface</i>	<ul style="list-style-type: none"> - Les RESTful API mettent en œuvre une interface uniforme entre le client et le serveur, ce qui permet de développer ces deux éléments séparément. - Les RESTful API permettent des opérations CRUD sur les ressources à l'instar, par exemple, des méthodes POST, GET, PUT, PATCH et DELETE du protocole HTTP.
<i>Stateless Operations</i>	<ul style="list-style-type: none"> - Des accès successifs de clients à serveurs sont toujours indépendants les uns des autres. Le serveur ne s'appuie pas sur des informations provenant de requêtes antérieures. - Le client est responsable du stockage et de la manipulation de toutes les informations liées à l'application qui concernent son côté de l'interface.
<i>Layered System of Servers</i>	<ul style="list-style-type: none"> - Cela doit revenir au même pour le client, qu'il parle au serveur ou à un intermédiaire tel qu'un service <i>Gatekeeper</i> (API gateway). - La communication du client au serveur peut passer par plusieurs intermédiaires.

⁴⁸ Partiellement inspiré de l'article sur les accès payants

⁴⁹ Données typifiées, les chaînes peuvent être des contenus formatés, par ex. XML ou JSON

⁵⁰ Roy Thomas Fielding, PhD « Architectural Styles and the Design of Network-based Software Architectures », 2000, https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Condition	Description
<i>Resource cache</i>	<ul style="list-style-type: none"> - Dans les communications entre le client et le serveur, des ressources peuvent être mises en cache dans le client ou dans l'intermédiaire afin d'augmenter la performance du système. - Les clients et les intermédiaires gèrent eux-mêmes à l'actualisation des ressources.
<i>Code on demand</i> (facultatif)	<ul style="list-style-type: none"> - Le serveur peut fournir au client un code de programme pour accéder à ses ressources. - Le client peut obtenir ce code et l'exécuter localement. - Cette condition est facultative.

Tableau 24 Conditions du style d'architecture REST

8.3.2.2 GraphQL

GraphQL est un langage de requête *open source* pour les API et un environnement d'exécution pour les requêtes sur des données existantes⁵¹. Il combine un langage de requête, un environnement d'exécution et un système de types original, qui offrent une structure d'API flexible.

Une architecture d'interface basée sur GraphQL met à disposition 1-N sources de données différentes au moyen d'1 API pour le client API, sans limitation du nombre de ressources interrogées. GraphQL permet ainsi de formuler une seule requête complexe et d'obtenir une réponse qui contient exactement les données demandées, sans retour de données surnuméraires (*over-fetching*) ou surnuméraires (*under-fetching*). Les requêtes se font dans la syntaxe propre à GraphQL, laquelle permet trois opérations : *query* (*Read*), *muation* (*Create, Update, Delete*), *subscription* (*publish-subscribe*). GraphQL utilise HTTP pour les opérations *query* et *muation*, et WebSockets pour les opérations *subscription*.

Par rapport aux REST API, GraphQL permet une plus grande complexité lors du développement des API.

Protocole API	Type d'interface	MEP	Format de données	Type de message
HTTP	Ressources	Requête-réponse	Syntaxe GraphQL Query, JSON	Requête Réponse
WebSockets	Ressources	Requête-réponse Messagerie	Syntaxe GraphQL Query, JSON	Requête Réponse Évènement

Tableau 25 Conventions d'échange de données pour GraphQL

8.3.2.3 Linked data

Les données liées (*linked data*) sont abordées dans l'architecture API de la Confédération, puisqu'elles sont déjà utilisées pour la publication de données dans l'environnement fédéral. On pense notamment au service de données liées des Archives fédérales (LINDAS)⁵² et à celui de swisstopo (Linked Data Service GeoDaten)⁵³. Il est recommandé en particulier de publier les métadonnées API les registres des API sur une plateforme de données liées (voir ch. 7.2.2.1), ce qui permet de consulter ces métadonnées en utilisant une API publique.

On entend par *linked data* des données structurées sur la base du RDF⁵⁴, qui sont publiées pour être utilisées dans des logiciels (communication M2M). Les données liées s'obtiennent au moyen de requêtes formulées dans le langage SPARQL à partir de serveurs SPARQL ou de points de sortie SPARQL. Les services *Linked data* sont très répandus sur Internet. Une seule requête SPARQL peut être adressée à plusieurs de ces services. Étant donné qu'elles sont basées sur le protocole API HTTP, les données liées sont couvertes par les conventions d'échange de données basé sur HTTP et orientées ressources (voir tableau 26). Les données liées sont gérées sous la forme de triplets RDF dans ce que l'on appelle des *triplestores RDF*. Les points de sortie SPARQL donnent accès à ces triplets RDF. De plus amples explications sur les *linked data* sont disponibles au ch. 10.4.

⁵¹ <https://graphql.org>

⁵² <https://lindas.admin.ch>

⁵³ <https://www.geo.admin.ch/fr/geo-services-proposes/geoservices/linkeddata.html>

⁵⁴ <https://www.w3.org/TR/rdf11-primer>

Protocole API	Type d'interface	MEP	Format de données	Type de message
HTTP	Ressources	Requête-réponse	JSON, XML, CSV	Requête Réponse

Tableau 26 Conventions d'échange de données pour les *linked data*

8.3.2.4 sedex – *secure data exchange*⁵⁵

sedex est un service de l'OFS. Il n'est pas seulement synonyme de conventions d'échange de données. C'est aussi une plateforme intermédiaire - comme celle de Swissdec - qui fonctionne sur la base de ces conventions. sedex permet à l'OFS d'intervenir en tant qu'intermédiaire, ou comme un facteur qui transmettrait des lettres recommandées par voie électronique. Le service peut être utilisé par tous les types de partenaires. Il peut notamment être utilisé par deux partenaires de communication externes, bien que tous les messages passent par la ou les zones de réseau de la Confédération. Les deux partenaires peuvent alors envoyer et recevoir des données.

Protocole API	Type d'interface	MEP	Format de données	Type de message
HTTP	Ressources	<i>Request-response</i>	XML	Requête Réponse
SOAP/WSDL	Ressources	Aucune restriction	XML	Requête Réponse

Tableau 27 Conventions d'échange de données pour sedex

La communication par sedex nécessite toujours une base légale.

sedex est axé sur les ressources et combine les protocoles API HTTP et SOAP/WSDL, le contenu XML pouvant contenir des documents de différents formats (csv, docx, jpg, pdf, xml, zip, etc.). Le tableau 27 montre où se situe sedex dans les conventions d'échange de données.

Les composants système de l'architecture de référence API sur les processus d'intégration IP3 et IP4 correspondent aux éléments suivants dans l'architecture sedex⁵⁶:

Client API	↔	Application + client sedex
Plateforme intermédiaire + API <i>gateway</i>	↔	Serveur sedex
Service spécialisé	↔	Application + client sedex

Il est à noter que le FP de la Confédération qui livre le client sedex fournit en même temps une partie du client API dont le partenaire est habituellement responsable.

8.3.3 Traitement des erreurs

L'architecture API de la Confédération recommande de concevoir le service *Gatekeeper* de manière à ce qu'il envoie dans tous les cas au client API des messages d'erreur précis en cas d'accès API mal formulés ou d'indisponibilité de services nécessaires au fonctionnement des services publics numériques. Les messages d'erreur peuvent également être émis sous la forme de codes d'erreur, qui sont expliqués dans la documentation API. Accompagner les codes d'erreur de paramètres d'appel permet d'optimiser les messages. Les messages d'erreur permettent au client API de réagir de manière appropriée et permettent au développeur API d'adapter le client API en conséquence.

8.4 Recommandation de conception *Identity & access management*

8.4.1 Introduction

Un IAM est une composante essentielle pour une cyberadministration intégrée et continue, en particulier lorsque les UA de l'administration fédérale souhaitent proposer des services publics numériques à leurs partenaires au moyen d'API. La capacité IAM entre en jeu dans l'architecture API de la Confédération lorsqu'une API exige l'enregistrement d'une identité, c'est-à-dire lorsque le client API réclame au moins la présentation d'une identité enregistrée dans le système IAM à laquelle le service spécialisé fait confiance.

Pour les API publiques avec enregistrement et les API partenaires, les identités doivent être authentifiées avant l'accès. Les API publiques avec enregistrement demandent en général un niveau d'authentification plus faible que les API partenaires, car tout le monde peut accéder aux premières, alors que l'accès aux dernières est limité à des

⁵⁵ <https://www.sedex.ch>

⁵⁶ <https://www.bfs.admin.ch/bfs/fr/home/registres/registre-personnes/sedex/downloads.assetdetail.8826827.html>

identités définies (DT). La qualité de l'authentification est définie en fonction du LoA (voir norme eCH-0170 Modèle de qualité pour l'authentification des sujets⁵⁷), que la capacité IAM doit faire respecter lorsque des identités accèdent aux ressources.

Les demandes d'accès des API partenaires sont toujours vérifiées et peuvent être acceptées ou refusées. Limiter l'accès aux API partenaires signifie donc que les rôles attribués aux identités définissent si l'identité est autorisée à accéder à une API.

Lorsqu'il est question d'IAM, il faut toujours respecter les directives sur la sécurité des réseaux (NCSC, Si003)⁵⁸, qui définissent dans les conditions requises pour qu'une identité puisse accéder aux zones de réseau.

L'ordonnance sur les systèmes de gestion des données d'identification et les services d'annuaires de la Confédération (OIAM)⁵⁹ est entrée en vigueur le 19 octobre 2016. Elle régit les compétences, le traitement et la publication de données personnelles ainsi que les exigences concernant la sécurité de l'information pour les systèmes de gestion des données d'identification, les services d'annuaires et la base centralisée des identités de la Confédération. Selon l'art. 3, un système IAM sert à gérer conjointement des données sur l'identité et les autorisations de personnes, de machines et de systèmes pour les mettre, sur demande, à la disposition des systèmes en aval et d'autres systèmes IAM. Dès lors, l'OIAM s'applique aux services publics numériques.

8.4.2 Accès à l'API par jeton sécurisé

Les API partenaires ne peuvent être utilisées que par les partenaires autorisés qui ont des relations avec l'administration fédérale. Elles requièrent l'enregistrement d'une identité. L'accès est ensuite possible au moyen d'un identifiant et d'un jeton sécurisé. Les API publiques peuvent également utiliser des jetons sécurisés, notamment lorsque seule l'identité doit être authentifiée.

Dans l'architecture API de la Confédération, il est recommandé de mettre en œuvre l'authentification et l'autorisation d'accès aux API publiques et partenaires au moyen de jetons sécurisés, car ils présentent certains avantages par rapport à l'utilisation du seul identifiant :

- Ils servent à transmettre les informations d'authentification et d'autorisation du client au service *Gatekeeper* et de ce dernier au service Ressources.
- Leur validité est en général limitée de sorte qu'on évite l'enregistrement permanent des identifiants dans le client ou sur le serveur.
- Leur utilisation évite d'avoir à gérer les sessions sur le serveur (pas de session permanente), car ils sont vérifiés par les services IAM (IAM fédérée, voir ch. 8.4.3).
- Les normes reconnues SAML⁶⁰ et OAuth/OIDC⁶¹, qui les définissent, peuvent être utilisées facilement avec HTTP.

Les jetons sécurisés présentent en outre les caractéristiques suivantes :

- Ils représentent une version dérivée de l'identifiant d'une identité connue.
- Ils contiennent la confirmation qu'une identité a été authentifiée.
- Ils peuvent porter des informations d'autorisation.

Les jetons sécurisés sont toujours émis pour un public cible spécifique, soit le client, soit le service *Gatekeeper*. Le service *Gatekeeper* ne doit pas transmettre le jeton sécurisé du client au service Ressources.

8.4.3 Architecture cadre IAM de la Confédération

Les services IAM jouent un rôle central dans le modèle d'information, car l'accès aux ressources est géré par un ensemble de tels services. Pour définir les services IAM, l'architecture API de la Confédération s'appuie sur la norme TNI relative à l'architecture cadre IAM de la Confédération⁶². En général, on distingue deux groupes de services IAM : les services IAM fournis lors de la définition (DT) et ceux fournis lors de l'exécution (RT). L'architecture cadre IAM de la Confédération subdivise ces services IAM en sous-services IAM, qui sont décrits dans l'environnement de référence.

⁵⁷ <https://www.ech.ch/index.php/fr/ech/ech-0170/2.0>

⁵⁸ <https://www.ncsc.admin.ch/ncsc/fr/home/dokumentation/sicherheitsvorgaben-bund/sicherheitsverfahren/grundschutz.html>

⁵⁹ [RS 172.010.59](#)

⁶⁰ https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

⁶¹ <https://openid.net/connect>

⁶² https://intranet.dti.bk.admin.ch/isb_kp/de/home/themen/iam-bund/dokumente-steuerung-iam-bund.html (en allemand uniquement)

L'architecture cadre IAM de la Confédération distingue deux grands types d'IAM :

- **IAM fédéré**
Les services IAM sont séparés du service *Gatekeeper* et peuvent être distribués par des sous-services IAM fédérés.
- **IAM non fédéré**
Le système IAM est intégré dans le service *Gatekeeper*, qui dispose de ses propres services IAM.

Afin de pouvoir utiliser les identités et rôles API de l'environnement fédéral pour plusieurs services publics numériques, il est recommandé d'utiliser un IAM fédéré. Le modèle d'information de la figure 13 recourt donc un IAM fédéré. Les IAM non fédérés sont autorisés, mais ils entraînent plus de travail pour les UA concernées.

L'architecture cadre IAM de la Confédération définit les principes, les règles et le cadre applicables à la conception de systèmes IAM qui doivent être pris en compte lors de la mise à disposition de solutions IAM fédérées dans l'administration fédérale. Elle décrit en particulier l'utilisation des jetons sécurisés.

En raison du traitement des jetons sécurisés défini dans l'architecture cadre IAM de la Confédération pour l'IAM fédéré, et en raison des avantages liés à ces jetons, il est recommandé d'appliquer la conception IAM aux protocoles API qui règlent le traitement des jetons sécurisés, ce qui est le cas lorsque HTTP est utilisé.

Lorsqu'un client présente un jeton sécurisé au service *Gatekeeper*, celui-ci pose les quatre questions représentées dans le tableau 28, valables dans tous les contextes IAM, auxquelles le modèle d'information de l'architecture API de la Confédération apporte les réponses figurant dans le même tableau. Les questions 3 et 4 ne sont posées que par les partenaires API.

	Question	Réponse du modèle d'information
1	Qui êtes-vous ?	- Les services IAM émettent une identité pour le partenaire API lors de la définition (DT).
2	Comment puis-je vous reconnaître ?	- Les services IAM émettent un identifiant pour le partenaire API lors de la définition (DT). - Les services IAM vérifient l'identifiant avant d'émettre un jeton sécurisé pour le partenaire API lors de l'exécution (RT). - Le service <i>Gatekeeper</i> vérifie le jeton sécurisé du client afin d'authentifier l'identité.
3	Vous êtes autorisé par « quelqu'un » à faire quoi ?	- Le fournisseur API valide les rôles API demandés pour l'identité lors de la définition (DT).
4	Comment faire respecter les limites de votre autorisation ?	- Le service <i>Gatekeeper</i> vérifie le jeton sécurisé du client afin d'autoriser l'accès. - Le service Ressources vérifie le jeton sécurisé du <i>gatekeeper</i> afin d'autoriser l'accès (il ne s'agit pas du même jeton sécurisé, voir ch. 8.4.4).

Tableau 28 Réponses du modèle d'information aux questions pertinentes pour l'IAM

L'architecture cadre IAM de la Confédération travaille avec les objets d'information pertinents pour l'architecture API de la Confédération: *fournisseur d'identités*, *fournisseur de fédération*, *Web-PEP* et *Web-PEP-on-behalf*. Le tableau 29 fournit la description et les correspondances des objets d'information de l'architecture API de la Confédération qui sont pertinents pour l'IAM. On garantit ainsi la cohérence de l'architecture API et de l'architecture cadre IAM de la Confédération.

Objets d'information Architecture cadre IAM de la Confédération	Description	Objets d'information Architecture API de la Confédération
Fournisseur d'identités	- Il émet et gère les identités, les identifiants et les jetons sécurisés	Services IAM (DT et RT)
Fournisseur de fédération	- Il authentifie les identités dans le cadre d'un IAM fédéré - Il consomme le jeton sécurisé du client - Il obtient un nouveau jeton sécurisé - Il donne accès à la zone réseau cible du service Ressources	Service <i>Gatekeeper</i>
<i>Web policy enforcement point</i> (Web-PEP)	- Il prend la décision (définitive) concernant l'accès à la ressource.	Service Ressources
<i>Web policy enforcement point on behalf</i> (Web-PEP-on-behalf)	- Il prend la décision (définitive) concernant l'accès à la ressource.	Service <i>Gatekeeper</i>

Tableau 29 Objets d'information de l'architecture cadre IAM de la Confédération

8.4.4 Formes d'architecture API pertinente pour l'IAM

L'architecture API de la Confédération distingue quatre grandes formes de conception d'architecture API pertinente pour l'IAM (représentées dans la figure 17) qui se distinguent par le type d'utilisation du client et le type de contrôle d'accès.

	Utilisation du client par un sujet (anonyme)	Utilisation client par un partenaire API
Contrôle d'accès auprès du service <i>Gatekeeper</i>	<u>Forme 1</u> Utilisation du client par un sujet (anonyme) avec contrôle d'accès auprès du service <i>Gatekeeper</i> (figure 18)	<u>Forme 2</u> Utilisation du client par un partenaire API avec contrôle d'accès auprès du service <i>Gatekeeper</i>
Contrôle d'accès auprès du service Ressources	<u>Forme 3</u> Utilisation du client par un sujet (anonyme) avec contrôle d'accès auprès du service Ressources	<u>Forme 4</u> Utilisation du client par un partenaire API avec contrôle d'accès auprès du service Ressources (figure 19)

Figure 17 Formes d'architecture API pertinente pour l'IAM

Deux des quatre formes sont représentées dans la figure 18 et la figure 19. Sur cette base, il est possible de se représenter les deux autres.

Le tableau 30 décrit les deux types d'utilisation de client. Soit le client est utilisé soit par un sujet (éventuellement anonyme) soit par le partenaire API lui-même. On part du principe qu'un client ne peut accéder à une API qu'avec une seule identité, ce qui signifie que l'API n'authentifie qu'une seule identité.

Type d'utilisation du client	Description
Utilisation du client par un sujet	<ul style="list-style-type: none"> - Le partenaire API est responsable du client. - Le partenaire API ne doit pas nécessairement connaître les sujets qui utilisent le client. - Les sujets peuvent aussi utiliser le client de manière anonyme. - L'identité représente le partenaire API et est donc une personne morale (par ex. une entreprise qui agit en tant qu'intermédiaire).
Utilisation du client par un partenaire API	<ul style="list-style-type: none"> - Le partenaire API est responsable du client. - Les sujets représentent le partenaire API (par ex. employés d'une entreprise). - Le sujet ne peut pas agir de manière anonyme. - L'identité représente à la fois le partenaire API et le sujet et est donc soit une personne physique soit une personne morale. - Si le sujet représente une personne morale, l'identité de la personne physique peut, si nécessaire, être ajoutée à l'identité de la personne morale en tant qu'attribut.

Tableau 30 Types d'utilisation du client

Le tableau 31 décrit, conformément à ce que prévoit l'architecture cadre IAM de la Confédération, les deux types de contrôle d'accès aux API passant par des jetons sécurisés. Ils se distinguent en cela que le contrôle de l'accès au service Ressources est assuré une fois par le service *Gatekeeper* et une fois par le service Ressources.

Type de contrôle d'accès	Étapes dans le cadre de l'exécution
Contrôle d'accès auprès du service <i>Gatekeeper</i> avec Web-PEP-on-behalf au moyen de jetons sécurisés	<ol style="list-style-type: none"> 1. Le client accède au service <i>Gatekeeper</i> au moyen d'un jeton sécurisé. 2. Le service <i>Gatekeeper</i> authentifie l'identité. 3. Le service <i>Gatekeeper</i> vérifie l'autorisation pour le compte du service Ressources et décide d'accorder ou non l'accès à ce service.
Contrôle d'accès auprès du service Ressources avec Web-PEP au moyen de deux jetons sécurisés différents	<ol style="list-style-type: none"> 1. Le client accède au service <i>Gatekeeper</i> au moyen d'un jeton sécurisé A. 2. Le service <i>Gatekeeper</i> authentifie l'identité. 3. Le <i>gatekeeper</i> obtient auprès des services IAM, au moyen d'un jeton sécurisé A qui fait office ici d'identifiant, un jeton sécurisé B avec une autorisation correspondant à celle du jeton A. 4. Le service <i>Gatekeeper</i> accède au service Ressources au moyen du jeton sécurisé B. 5. Le service Ressources authentifie l'identité du service <i>Gatekeeper</i>. 6. Le service Ressources vérifie l'autorisation et décide d'accorder ou non l'accès au service Ressources.

Tableau 31 Types de contrôle d'accès

Variantes :

- Dans le cas d'un IAM non fédéré, les services IAM (DT et RT) dans la figure 18 et la figure 19 ne sont pas séparés des services *Gatekeeper* et Ressources, mais en font partie.
- Les jetons sécurisés, comme l'identifiant, peuvent se voir attribuer une date d'expiration. On peut ainsi mettre fin à un enregistrement.
- Une personne physique peut se connecter à une application portail avec son identité aussi au moyen d'eIAM, qui est alors associé au jeton sécurisé que le client envoie au service *Gatekeeper*.
- En principe, les IAM fédérés affaiblissent le niveau de sécurité du service Ressources, car la responsabilité de la sécurité de l'information et de la protection des données est répartie entre plus d'acteurs que dans le cas des IAM non fédérés. C'est pourquoi il peut être utile de munir les clients de certificats clients, qui sont par exemple établis pour une version du logiciel client et qui permettent ainsi d'authentifier non seulement l'identité, mais aussi la source d'un accès à l'API.

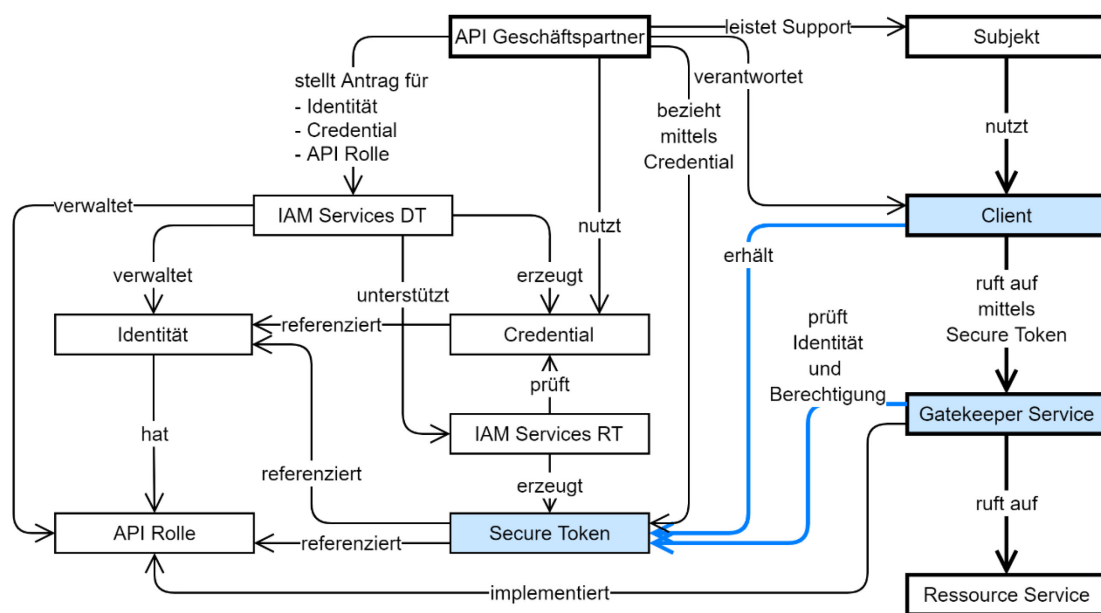


Figure 18 Modèle d'information pertinent pour l'IAM - forme 1

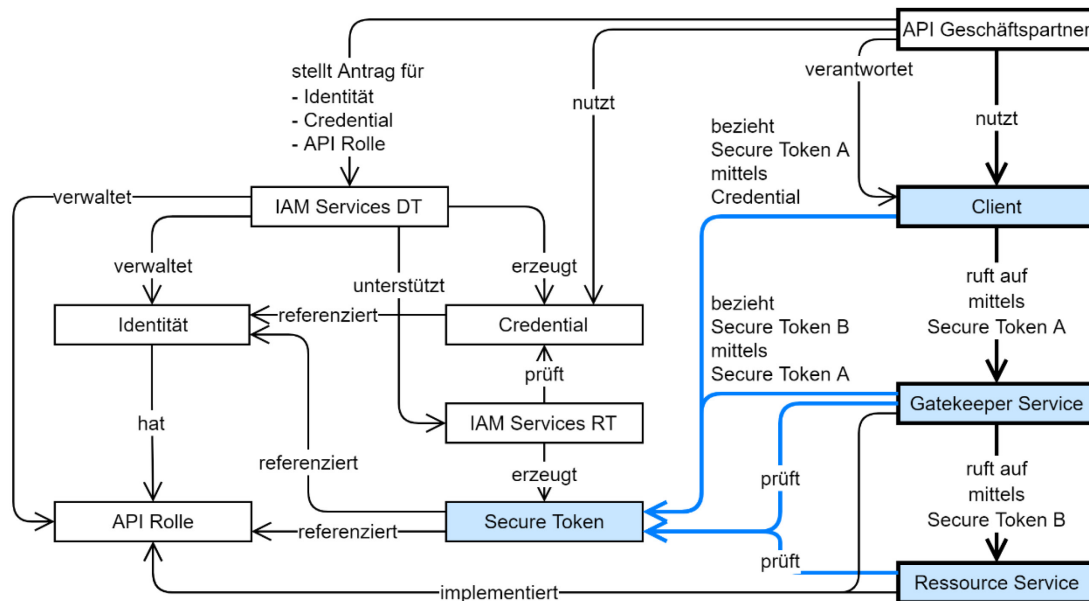


Figure 19 Modèle d'information pertinent pour l'IAM - forme 4

L'environnement fédéral présente déjà des architectures permettant de contrôler les accès aux services *Gatekeeper* et Ressources : l'Office fédéral de la douane et de la sécurité des frontières⁶³ et ⁶⁴le DFJP⁶⁵ utilisent ⁶⁶⁶⁷un service *Gatekeeper*. De plus, l'architecture de la solution du DFJP prévoit un niveau de contrôle d'accès supplémentaire dans le service Ressources. Dans les deux cas, le partenaire API est aussi l'utilisateur du client.

8.4.5 Single sign on

Un jeton sécurisé peut contenir des autorisations pour plusieurs services Ressources. Ces services Ressources sont des services spécialisés dans le cadre d'un même service public numérique. L'architecture API de la Confédération parle de *single sign on* (SSO, signature unique) lorsque, dans le cadre

- d'un service public numérique et
- d'une fenêtre de temps définie (*time-out*),

un client s'authentifie une fois lors de l'accès au premier service Ressources puis peut accéder à N services Ressources ($N > 1$) dans la fenêtre de temps définie. Dans le cadre d'un SSO, il n'est pas nécessaire de s'identifier à nouveau pour accéder aux autres services Ressources pendant la période définie, pour autant que l'identité satisfasse aux exigences LoA des N services Ressources. L'architecture API de la Confédération recommande de mettre en place un SSO dans les services IAM.

⁶³ Il s'agit d'une architecture de solution développée pour le programme DazIT, dans laquelle les services IAM sont séparés du service *Gatekeeper* et sont assurés par les composants système SAP MDG, Connex, PAMS et ePortal.

⁶⁴ Voir eIAM Définitions https://www.eiam.admin.ch/pages/fleiamglossary!pub_fr.html?c=fleiamglossary!pub&l=fr&ll=1 « Qu'est-ce que PAMS? Quel est le lien avec eIAM? »

⁶⁵ Il s'agit du service portail SSO DFJP, qui couvre à la fois le service *Gatekeeper* et les services IAM.

⁶⁶ <https://www.bk.admin.ch/bk/de/home/digitale-transformation-ikt-lenkung/e-services-bund/services/sso-ejpd.html>

⁶⁷ Voir eIAM Définitions https://www.eiam.admin.ch/pages/fleiamglossary!pub_fr.html?c=fleiamglossary!pub&l=fr&ll=1 « Qu'est-ce que le portail SSO DFJP et comment est-il lié à l'eIAM ? »

9 Architecture d'application

9.1 Système de l'architecture API de référence

9.1.1 Aperçu

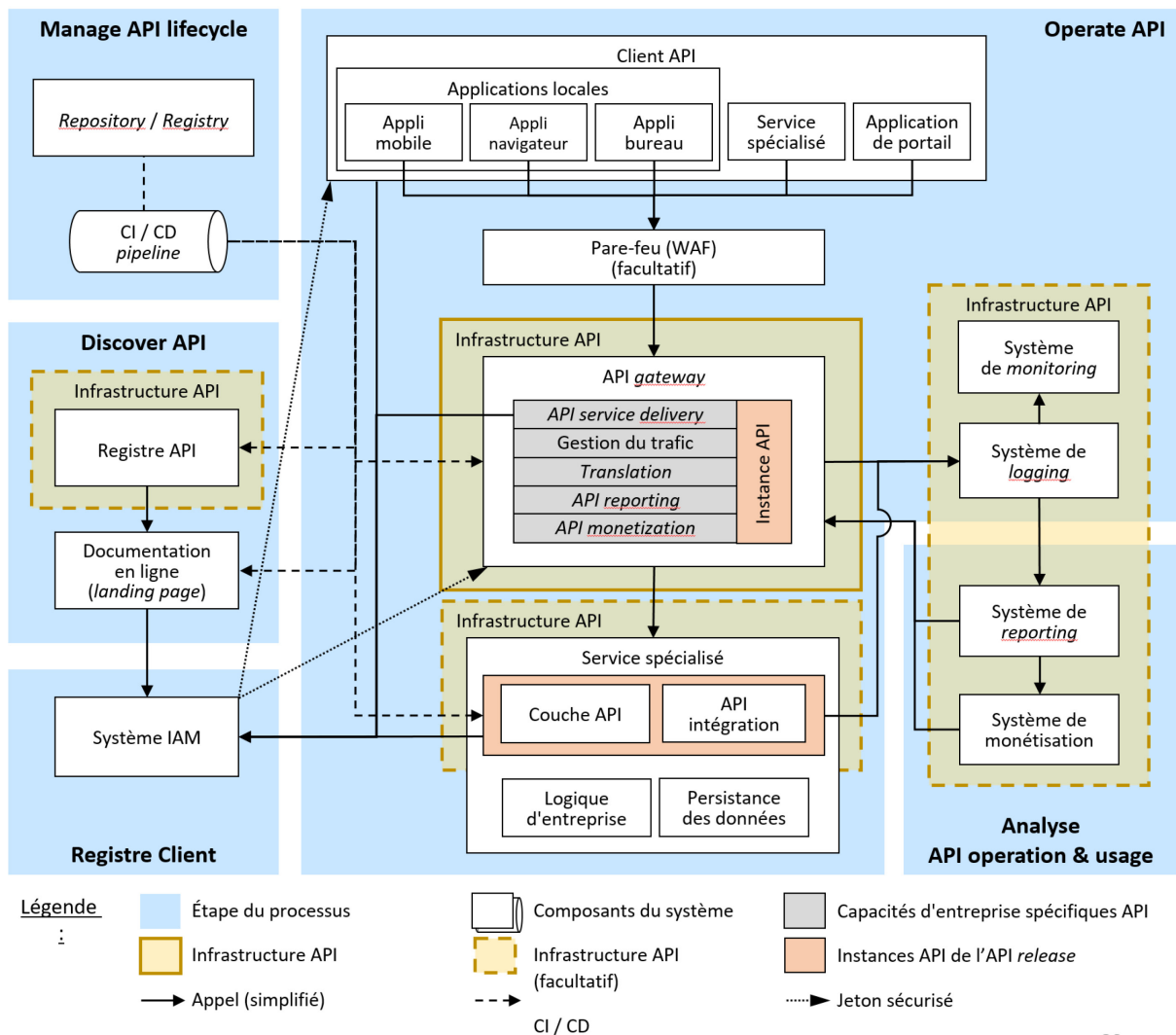


Figure 20 Système de l'architecture API de référence

L'architecture API de la Confédération comprend les composants système regroupés par étapes de processus présentés dans la figure 20. Lors du développement et du fonctionnement des API, elle fait la distinction entre les services et les composants de l'infrastructure API et les instances API, qui représentent les composants système spécifiques à l'API du service public numérique. Le développement et le fonctionnement des composants système de l'infrastructure API peuvent être pris en charge par une autre UO que celle chargée le développement et l'utilisation des instances API. En particulier, le développement et le fonctionnement des composants système de l'infrastructure API peuvent être centralisés, tandis que le développement et le fonctionnement des instances API doivent être décentralisés.

9.1.2 Infrastructure API

L'infrastructure API, généralement une plateforme technique, fournit les composants système et services génériques permettant le fonctionnement des API. Les produits disponibles dans la communauté *open source* et sur le marché offrent différentes fonctions. Selon l'utilisation envisagée, il convient de choisir une plateforme d'infrastructure API (donc un produit) couvrant les différents composants système requis. La figure 20 montre les composants optionnels d'une infrastructure API. Si le produit d'infrastructure API ne couvre pas un composant système requis pour l'utilisation envisagée, l'infrastructure informatique générique de l'UA doit alors offrir la fonction de

ce composant (par ex. un service *Logging*). L'infrastructure API peut dans ce cas être connectée à l'infrastructure informatique générique de l'UA.

Une infrastructure API peut être répartie sur plusieurs sites géographiques et proposer ainsi une instance API sur plusieurs sites géographiques à des adresses différentes de manière à mieux gérer le trafic. Une infrastructure API permet de gérer de manière centralisée toutes les instances API qui y sont exécutées et qui appartiennent à la même API.

9.1.3 Instances API

Les instances API sont utilisées à deux fins.

- Elles fonctionnent dans l'API *gateway*, le configurent en tant que partie de l'infrastructure API ou représentent des objets exécutables autonomes.
- Elles fonctionnent dans le service spécialisé soit sous la forme d'une couche API qui offre les fonctions d'un service spécialisé vis-à-vis de l'extérieur via des API, soit sous la forme d'un composant système pour l'intégration API qui se présente à son tour comme un client API et qui pilote d'autres API.

Les instances API concernent toujours l'aspect spécialisé d'un service public numérique, tandis que l'infrastructure API fournit l'environnement opérationnel. Le *repository* ou le *registry* fournit les API *builds* et les configurations API pour les deux types d'instances API. La logique opérationnelle et la persistance des données de l'application spécialisée ne font jamais partie de l'instance API. La mise en œuvre des aspects techniques relève toujours de la logique opérationnelle. Pour assurer une bonne mise à l'échelle, les instances API peuvent être multipliées et fonctionner sur plusieurs sites géographiques. Dans les environnements matures, la mise à l'échelle peut être gérée de manière dynamique par le système *Reporting*.

9.2 API gateway et message exchange pattern

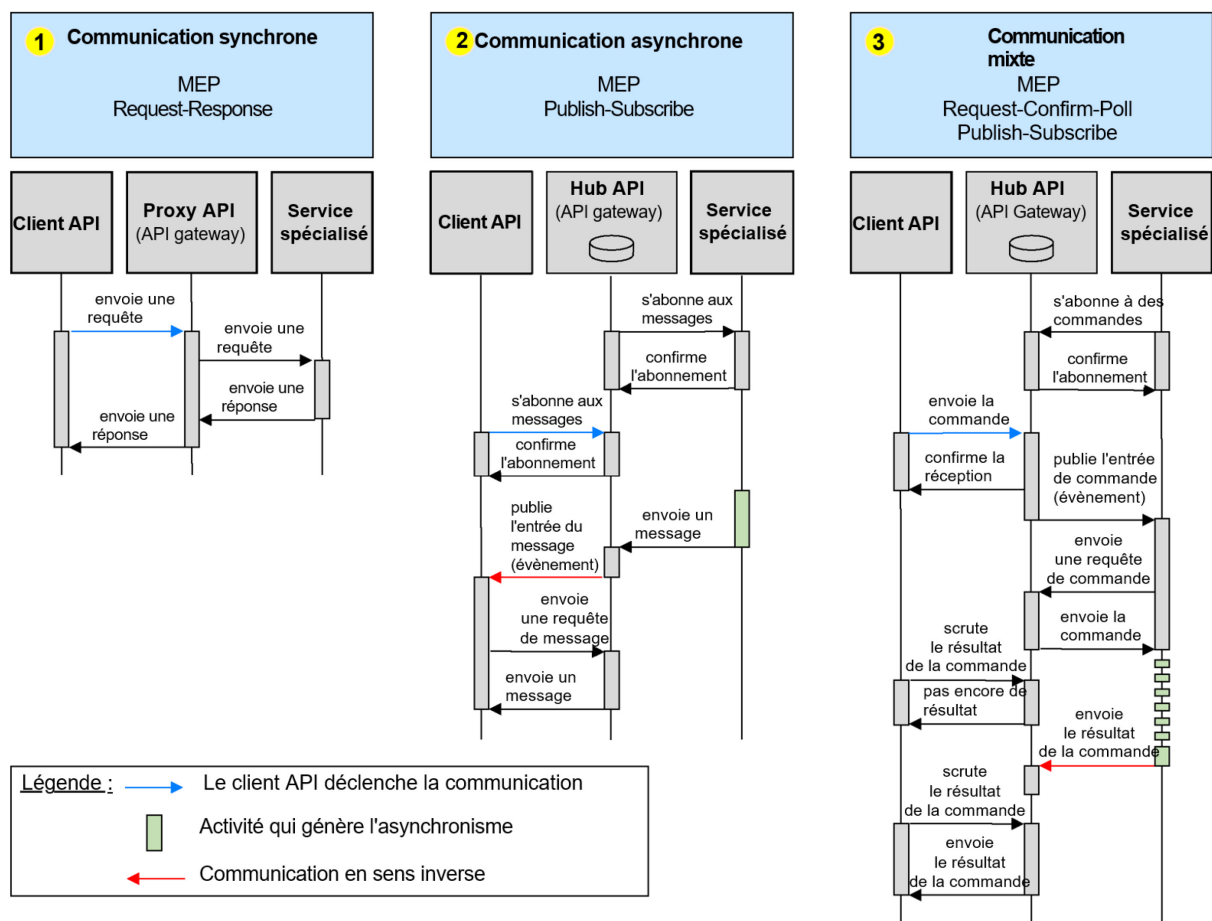


Figure 21 API gateway et message exchange pattern

L'*API gateway* peut avoir différentes formes. Il peut notamment être conçu comme un proxy API ou un hub API, la forme la plus simple étant un *reverse proxy*. Si le *gateway* est défini comme un proxy API, la communication passe par un MEP synchrone. S'il est défini comme un hub API, il faut utiliser un MEP asynchrone. En cas de communication asynchrone, les messages sont stockés temporairement dans le hub API, car ils ne sont pas échangés en temps réel entre le client API et le service spécialisé. C'est pourquoi, contrairement au proxy, le hub API dispose d'un cache (mémoire volatile) ou d'une base de données (mémoire non volatile).

La figure 21 montre à titre d'exemple, sous forme de diagrammes de séquence, trois cas d'application d'*API gateway* sous forme de proxy API ou de hub API avec communication synchrone, asynchrone ou mixte.

1. Proxy API avec communication synchrone en direction du client API et du service spécialisé
2. Hub API avec communication asynchrone en direction du client API et du service spécialisé
3. Hub API avec communication synchrone en direction du client API et communication asynchrone en direction du service spécialisé

Un *API gateway* sous forme de proxy ou de hub a des conséquences sur le MEP entre le client API et l'*API gateway* et sur celui entre l'*API gateway* et le service spécialisé. Les deux MEP n'ont pas à être identiques. Si l'un des deux MEP est asynchrone, il convient d'utiliser un hub API puisqu'il faut alors une mémoire tampon.

Une forme de MEP asynchrone est le MEP *publish-subscribe*, qui exige du client API qu'il puisse recevoir les événements générés par le serveur, ce qui requiert un protocole API bidirectionnel. Dans le cadre du MEP *publish-subscribe*, le client API s'abonne auprès du serveur à des messages qu'il récupère après avoir reçu un événement du serveur.

Lorsque le MEP *publish-subscribe* est utilisé en combinaison avec des API publiques, il n'y a pas de vérification des demandes d'enregistrement, de sorte que le fournisseur API n'a aucun contrôle sur le nombre de clients API qui doivent être avertis lorsque des messages sont prêts et qui doivent ensuite récupérer ces messages. Pour éviter un trafic excessif, il est possible d'utiliser la forme mixte, présentée dans la figure 21, qui autorise une communication asynchrone entre le hub API et le service spécialisé et prévoit une procédure de *polling* (synchrone) entre le client API et le hub API (vérification de la présence d'éventuelles données à transférer). Comme tous les clients API ne font pas leur *polling* à la même fréquence, les requêtes sont réparties dans le temps.

Afin de pouvoir contrôler le nombre de clients API, l'architecture API de la Confédération recommande de ne combiner MEP *publish-subscribe* entre le client API et le hub API qu'avec des partenaires API, étant donné que les demandes d'enregistrement de ces derniers sont vérifiées.

Bien que pour certains MEP asynchrones (voir ch. 10.3.3), le client API doit pouvoir recevoir des événements, c'est toujours lui qui déclenche la communication, tant pour les MEP synchrones que pour les MEP asynchrones (voir figure 21).

Il est recommandé de mettre en œuvre les exigences prévoyant un flux de données ou d'informations allant du service spécialisé vers le client API et, à cet effet, de faire en sorte que la communication soit lancée par le client API et non par le serveur. En effet, l'architecture API de la Confédération ne contient pas de directives sur les API proposées par des clients API agissant en tant que serveurs. Pour les flux de données à basse fréquence de ce type⁶⁸, où le développement et le fonctionnement d'une API ne sont pas rentables, il est possible de recourir à d'autres instruments, comme l'échange de courriel.

9.3 Description des composants système par le mappage des niveaux d'architecture

L'architecture d'entreprise et l'architecture des données traitées aux ch. 7 et 8 définissent les capacités et les objets d'information de l'architecture API de la Confédération. La description des composants système se fait par l'attribution des capacités et des objets d'information à ces composants. Cette attribution est illustrée dans le tableau 32 (à lire de gauche à droite). L'attribution d'une capacité à un composant système doit être comprise comme une mise en œuvre, par celui-ci, de la capacité ou de certaines de ses parties. L'attribution d'un objet d'information passif à un composant système veut dire que celui-ci consomme, émet ou traite celui-là.

⁶⁸ Par ex. lorsqu'il s'agit de rappeler chaque année à un partenaire API qu'il doit remplir telle ou telle formalité dans les délais.

Architecture d'application	Architecture de données		Architecture d'entreprise
Composants système	Objets de service	Objets d'information passifs	Capacité d'entreprise
<i>Repository / Registry</i>	- Service d'approvisionnement	<ul style="list-style-type: none"> - <i>API release</i> - <i>API version ID</i> - <i>API code</i> - <i>API build</i> - Configuration API - Documentation API - Métadonnées API 	<ul style="list-style-type: none"> - <i>Release planning</i> - <i>Transition</i> - <i>API versioning</i> - <i>Version control</i> - <i>API documentation management</i> - <i>API publication & revocation</i>
<i>CI/CD Pipeline</i>	- Service d'approvisionnement	<ul style="list-style-type: none"> - <i>API build</i> - Configuration API - Documentation API - Métadonnées API 	- <i>Transition</i>
Registre des API	- Service de catalogue	- Métadonnées API	<ul style="list-style-type: none"> - <i>API publication & revocation</i> - <i>API cataloging & searching</i>
Documentation en ligne	- Service de catalogue	- Documentation API	- <i>API documentation management</i>
Système IAM	<ul style="list-style-type: none"> - Services IAM DT - Service IAM RT 	<ul style="list-style-type: none"> - Identité - Rôle API - Identifiant - Jeton sécurisé 	- <i>Identity & access management</i>
Client API	- Client	<ul style="list-style-type: none"> - Ressource - Identifiant - Jeton sécurisé 	- <i>API Service Consumption</i>
WAF	- Service <i>Gatekeeper</i>	- Ressource	- <i>API service delivery</i>
<i>Gateway API</i>	- Service <i>Gatekeeper</i>	<ul style="list-style-type: none"> - <i>API build</i> - Configuration API - Ressource - Évènement - Jeton sécurisé 	<ul style="list-style-type: none"> - <i>API service delivery</i> - <i>Traffic management</i> - <i>Translation</i> - <i>Monitoring</i> - <i>Logging</i> - <i>API reporting</i> - <i>API monetization</i> - <i>Identity & access management</i>
Service spécialisé	- Service Ressources	<ul style="list-style-type: none"> - <i>API build</i> - Configuration API - Ressource - Évènement - Jeton sécurisé 	<ul style="list-style-type: none"> - <i>API service delivery</i> - <i>Monitoring</i> - <i>Logging</i> - <i>API reporting</i> - <i>API monetization</i> - <i>Identity & access management</i>
<i>Système Logging</i>	- Service <i>Logging</i>	- Évènement	- <i>Logging</i>
<i>Système Monitoring</i>	- Service <i>Monitoring</i>	- Évènement opérationnel	- <i>Monitoring</i>
<i>Système Reporting</i>	- Service <i>Reporting</i>	<ul style="list-style-type: none"> - Évènement - IPC - Rapport 	- <i>API reporting</i>
<i>Système Monetization</i>	- Service <i>Monetization</i>	<ul style="list-style-type: none"> - Modèle tarifaire - Contrôle de prestations - Compte 	- <i>API monetization</i>
n.a.	- Service Données de référence	<ul style="list-style-type: none"> - Objet Partenaire - Rôle partenaire 	n.a.

Tableau 32 Lien entre l'architecture d'application, l'architecture des données et l'architecture d'entreprise

Remarques sur le tableau 32:

- La capacité *Release planning* conduit à un *API release* dans le *repository*, mais va bien au-delà de la persistance de ce *release*. Les autres composants système et objets d'information nécessaires au *release planning* ne sont pas décrits dans l'architecture API de la Confédération, car ils ne sont pas spécifiques à l'API.
- La capacité *API design* n'est pas représentée, car elle a une influence sur tous les composants système et objets d'information.
- Les objets d'information Service des données de référence, Objet partenaire et Rôle partenaire sont des éléments architecturaux de l'infrastructure *Master Data Governance*, mise à disposition dans le cadre du programme SUPERB⁶⁹.
- Le composant système *CI/CD pipeline* est aussi souvent intégré dans le *repository*.

9.4 Recommandation de conception Intégration API

Les processus d'affaires nécessitent des services spécialisés. Lors de la fourniture de services publics numériques, les clients API accèdent aux services spécialisés par les API et intègrent de cette manière les fonctions des services spécialisés obtenus.

Si la fourniture d'un service numérique de la Confédération consiste en une succession de sous-services numériques publics, il est recommandé, conformément au principe *once only* (voir ch. 2.2.1), d'assurer la continuité des services numériques en intégrant le service spécialisé concerné à d'autres services spécialisés via des API.

On parle alors d'intégration d'API dans le service spécialisé de la Confédération (voir ch. 4.6). Un appel API lancé par le client API peut donc entraîner dans le service spécialisé le passage par une arborescence d'appels API ramifiée sur plusieurs niveaux. Si un tel appel d'API passe par les processus d'intégration IP3 ou IP4, l'utilisation d'un *API gateway* est indiquée. Dans le cas contraire, il est possible d'y renoncer. Pour des raisons de performance, de maintenance et de testage, la chaîne d'appels doit être aussi courte et simple que possible.

Si des données protégées et non anonymisées d'autres UA sont utilisées dans le cadre de l'intégration d'API dans le service spécialisé de la Confédération, les questions juridiques relatives à la protection des données doivent être clarifiées au préalable et un consentement doit être obtenu lors de la collecte des données. Il convient sinon d'indiquer dans la base légale concernée l'utilisation qui sera faite des données.

Mais il est aussi possible de mettre en place l'« intégration d'API dans le client API du partenaire ». C'est ce qui est fait lorsque le service public de la Confédération ne représente qu'un sous-service numérique du partenaire.

Lors de l'intégration de l'API, un composant système intermédiaire coordonne le passage par une arborescence d'appels API ramifiée sur plusieurs niveaux. Les différentes API partielles doivent toujours pouvoir être atteintes par le composant système intermédiaire via le réseau. Le composant système intermédiaire peut être un *middleware* d'intégration (par ex. WSO2⁷⁰), être basé sur un *data service framework* (par ex. une API GraphQL intégrant plusieurs sources de données) ou être une solution développée individuellement avec support API (liste non exhaustive). Les limites entre l'*API gateway*, l'intégration API et la couche API sont floues et dépendent des logiciels choisis.

Pour les API partielles qui nécessitent un enregistrement dans le cadre de l'intégration de l'API, il faut passer, pour chaque étape d'intégration au moment de la définition, par l'étape de processus *Register Client*.

Si une succession d'appels API comprend plusieurs opérations d'écriture, il convient d'accorder une attention particulière à la sécurité des transactions. L'architecture API de la Confédération distingue les deux types de processus représentés dans la figure 22 : opérations d'écriture en parallèle et opérations d'écriture en cascade. Afin la sécurité des transactions soit garantie de manière aussi peu complexe que possible, il est recommandé d'éviter les opérations d'écriture en parallèle.

⁶⁹ <https://www.efd.admin.ch/efd/fr/home/numerisation/superb.html>

⁷⁰ <https://wso2.com>

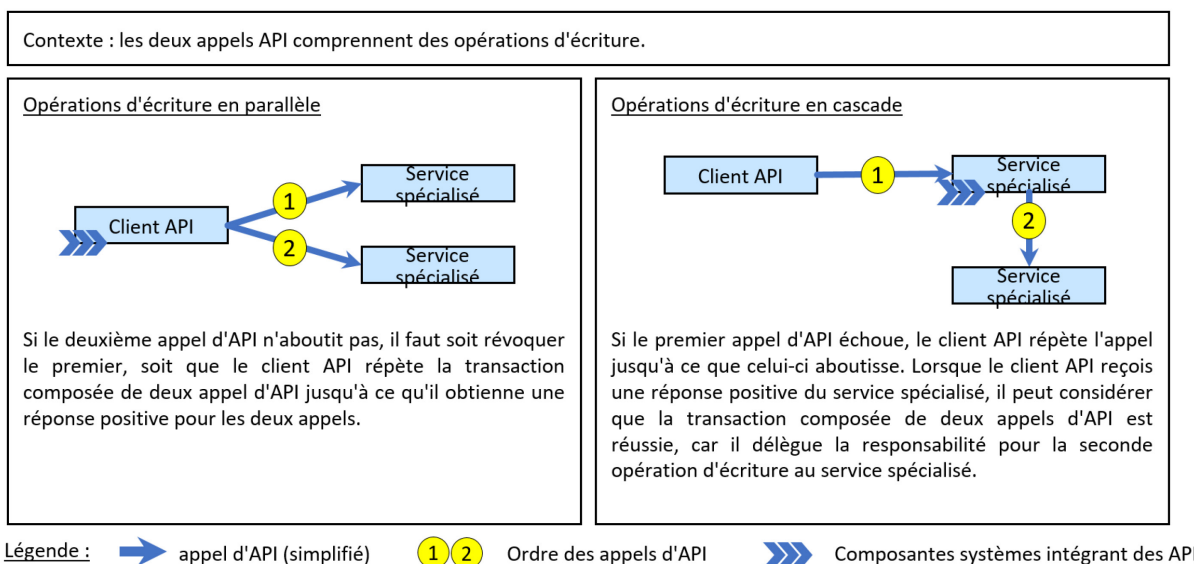


Figure 22 Opérations d'écriture en parallèle et en cascade

9.5 Recommandation de conception *API versioning*

9.5.1 Versionnage des types d'objets livrés et des normes spécifiques aux API

Le versionnage des API soulève la question de savoir quels types d'objets livrés et quelles normes spécifiques aux API doivent y être soumis au sein de l'architecture d'application. L'architecture API de la Confédération distingue à cet effet les trois types d'objets livrés et normes spécifiques aux API représentés dans le tableau 33.

Types d'objets livrés / normes	Description	Marche à suivre recommandée lors de la conception d'architectures de solution API
API release	Il représente les aspects techniques (spécificités) d'un service public numérique (voir définition au ch. 10.2.1).	<ul style="list-style-type: none">- Le versionnage des aspects techniques est consigné dans le versionnage de l'API Release.- Il est géré uniquement par le fournisseur API.
Infrastructure API	Il s'agit des plateformes d'infrastructure API et les <i>frameworks</i> API (voir définition au ch. 9.1). On peut citer par exemple les produits d'infrastructure API tels que WSO2 API Manager 4.0.0 ou RabbitMQ 3.8.18.	<ul style="list-style-type: none">- Le versionnage de l'infrastructure API est géré par la communauté <i>open source</i> ou par les fournisseurs commerciaux d'infrastructures API.- Les développeurs API du côté des fournisseurs API choisissent la version du produit qui répond le mieux aux exigences spécialisées et techniques.
Protocoles API	Ce sont des normes faisant l'objet de versions (voir ch. 10.3.5), comme HTTP/2.0 ou WSDL 2.0.	<ul style="list-style-type: none">- Le versionnage de protocoles API établis est contrôlé par des comités internationaux ou des grands groupes Internet.- Les développeurs API du côté des fournisseurs API choisissent la version de protocole qui répond le mieux aux exigences spécialisées et techniques ou reprennent la version proposée par l'infrastructure API.

Tableau 33 Types d'objets livrés et normes spécifiques aux API

Ce sont donc les versions API et leur versionnage sur le plan spécialisé qui sont traités ci-après dans la recommandation de conception.

9.5.2 Versionnage des *API releases*

L'architecture API de la Confédération recommande d'utiliser la norme Semantic Versioning 2.0.0 (SemVer 2.0.0)⁷¹ pour le versionnage des API releases. Cette norme très répandue est basée sur un numéro de version à trois chiffres, ce qui permet des extensions optionnelles :

⁷¹ <https://semver.org>

Numéro de version des *API releases* dans le contrôle de version : MAJOR.MINOR.PATCH(-Extension)
Exemples : 1.4.1, 2.0.0-rc1

Le tableau 34 explique la signification des quatre positions du numéro de version.

Position	Description
MAJOR	<ul style="list-style-type: none">- Version principale, nombre- Incrémenté chaque fois que la rétrocompatibilité est rompue
MINOR	<ul style="list-style-type: none">- Version secondaire, nombre- Incrémenté lorsque des extensions sont effectuées- La rétrocompatibilité est maintenue
PATCH	<ul style="list-style-type: none">- Version du patch, nombre- Incrémenté lorsque des corrections de défauts sont livrées- La rétrocompatibilité est maintenue
Extension	<ul style="list-style-type: none">- Suffixe alphanumérique facultatif- Sert à désigner les versions alpha/bêta et les versions admissibles (<i>release candidates</i>)- Peut éventuellement correspondre au numéro de <i>build</i>

Tableau 34 Composants du numéro de version selon SemVer 2.0.0

Si des versions de l'API sont publiées et deviennent ainsi des *API releases*, le code API dans le *repository* est doté d'une étiquette (*tagging*) dans le cadre du contrôle des versions (voir capacité *Version control*), qui correspond au numéro de version selon SemVer 2.0.0. Un label représente l'*API version ID* (voir figure 15). Il est utilisé dans la documentation en ligne pour référencer l'*API release*. Le *tagging* peut aussi être utilisé pour identifier les niveaux de développement internes du fournisseur API. Dans ces cas, les *API version ID* ne sont pas visibles pour l'extérieur, car les niveaux de développement internes ne sont pas publiés.

9.5.3 Degrés de liberté

Le fournisseur API est libre de publier et de représenter techniquement les positions MINOR, PATCH et Extension dans la documentation en ligne. (voir ch. 9.5.4) Cette liberté a été prévue, car, lors du fonctionnement d'un seul *API release*, l'incrémentation de la version secondaire et de la version patch n'entrave jamais la rétrocompatibilité et le partenaire API ne peut que profiter des extensions et des corrections de défauts. C'est pourquoi il est recommandé d'utiliser le modèle suivant pour la publication des numéros de version :

Numéros de version publiés des *API releases* : (v)MAJOR(.MINOR.PATCH-Extension)
Exemples : v2, 1.2, v1.3.3

Le fournisseur API qui gère plusieurs *API releases* en parallèle est tenu de publier des positions supplémentaires dans le numéro de version. Il peut toutefois renoncer à la publication de la version patch et de l'extension.

Il est permis de ne pas publier ni de représenter techniquement les versions d'API si l'on peut d'emblée supposer que l'évolution d'une API n'apporte que des extensions et des corrections de défauts qui ne nécessitent pas d'adaptation du client API.

9.5.4 Représentation technique du numéro de version

Une représentation technique peut par exemple être l'indication du numéro de version dans l'URL ou l'utilisation d'un en-tête HTTP. Indépendamment du nombre d'*API releases* exploités en parallèle et du nombre de positions du numéro de version, les numéros de version publiés dans la documentation en ligne sont représentés dans les *API releases* publiés de manière à ce qu'ils soient clairement identifiables par le client API à chaque requête API, que ce soit dans l'URL, dans un en-tête HTTP ou, pour les API non basées sur HTTP, à un autre endroit de la couche *Applications* du modèle OSI (couche 7).

Le tableau 35 donne un aperçu de l'endroit où le numéro de version peut se situer techniquement dans l'architecture de l'application, compte tenu des protocoles API mentionnés au ch. 10.3.5.

Protocole API	Emplacement du numéro de version
HTTP (modèle RESTful API)	<ul style="list-style-type: none"> - L'architecture API de la Confédération décrit cinq options : <ol style="list-style-type: none"> 1. <i>URL Namespace</i> : <code>https://api.v2.admin.ch/services</code> 2. <i>URL Resource Level</i> : <code>https://api.v2.admin.ch/services</code> 3. <i>URL Query Parameter</i> : <code>https://api.admin.ch/services?version=2</code> 4. <i>HTTP Headers, Custom Headers</i> <ul style="list-style-type: none"> - <code>Accept-version</code> : <code>v2</code> 5. <i>HTTP Headers, Content Negotiation</i> <ul style="list-style-type: none"> - <code>Accept</code> : <code>application/myapi.v2+json</code> - <code>Accept</code> : <code>application/myapi+json;version=2</code> - Le choix de l'option a une influence sur la complexité technique du client API. Si le numéro de version ne figure pas, par exemple, dans l'URL, il doit être précisé dans le client API. - Les options 4 et 5 s'utilisent de la même manière.
<i>Linked data</i> (cas particulier de HTTP)	<ul style="list-style-type: none"> - SPARQL permet d'interroger la structure des données sous-jacentes, y compris la sémantique. Il n'est dès lors pas nécessaire de tenir à jour un numéro de version. Les développeurs API en font usage et créent des requêtes SPARQL adaptées au modèle de données. - Si l'on souhaite faire une distinction entre les versions API, celles-ci doivent être représentées dans le modèle de données et interrogées au moyen de requêtes SPARQL. - Le fournisseur de données se prononce sur la stratégie de versionnage.
Messagerie (WebSockets, AMQP, MQTT)	<ul style="list-style-type: none"> - Le format de message comporte un champ pour le numéro de version. - Les exigences étant variées, il n'y a pas d'autres recommandations concernant le versionnage des messages.
gRPC	<ul style="list-style-type: none"> - Versionnage de la spécification d'interface protobuf (IDL) - Utilisation du <i>Package ID</i> du fichier protobuf
SOAP/WSDL	<ul style="list-style-type: none"> - Versionnage de la spécification d'interface WSDL-/XSD (IDL) - Les exigences étant variées, il n'y a pas d'autres recommandations concernant le versionnage des fichiers WSDL-/XSD.

Tableau 35 Emplacement technique du numéro de version

9.5.5 Migration forcée

L'utilisation recommandée de la norme de versionnage SemVer 2.0 permet de rompre la rétrocompatibilité lors du passage à l'*API release* suivant (voir ch. 6 Principe architectural AP7). Si la publication d'une nouvelle version principale est prévue et qu'une version principale de l'API doit être mise hors service pour des raisons d'efficacité des coûts, il convient d'accorder au partenaire API un délai de transition adapté pour qu'il puisse migrer son client API sur une version principale de l'API en service.

Pendant ce délai de transition, en cas de migration forcée, le fournisseur API utilise temporairement plusieurs *API releases*. La planification des *API releases* est superposée à la documentation en ligne ou publiée en tant que partie de la documentation d'API (*release*).

9.5.6 *API gateway* et service spécialisé

Dans l'architecture API de la Confédération, il est prévu que pour un service public numérique, la communication entre le client API et le service spécialisé passe systématiquement par un *API gateway*. L'*API gateway* est composé de l'infrastructure API et d'une instance API. Dans l'instance API du *gateway* et dans celle du service spécialisé, l'*API release* est représenté dans les deux composantes système que sont la couche API et l'intégration API.

Il est recommandé de toujours indiquer la version de l'*API release* et donc le versionnage technique dans la documentation en ligne, car l'*API release* qui contient les composantes système évoquées a une incidence sur l'installation du client API. Il faut veiller à ce que le passage à la version supérieure de l'infrastructure API n'ait, si possible, pas d'impact sur les *API releases*.

Il est également recommandé d'attribuer ensemble les responsabilités techniques pour le développement et l'utilisation de l'infrastructure API et des instances API. La responsabilité de l'infrastructure peut être séparée de la responsabilité technique des divers services spécialisés et des instances API correspondantes, dans la mesure où cela permet d'éliminer des obstacles organisationnels et d'améliorer la mise à disposition des API.

10 Annexe

10.1 Planification basée sur les capacités selon TOGAF

Les capacités peuvent être planifiées de manière ciblée avec le plan de planification qui leur est consacré et mesurées avec des indicateurs clés de performance (ICP). Elles peuvent être décrites dans quatre dimensions (voir figure 23) : collaborateurs, infrastructure, processus et informations. Les étapes du développement des capacités sont traitées comme des incréments de capacités attribués à l’une des quatre dimensions. Les éléments architecturaux définis aux trois niveaux d’architecture peuvent être utilisés pour définir des incréments de capacités et développer ainsi pas à pas la capacité *API management*. Le tableau 36 montre des exemples d’incréments utilisés pour le développement de la capacité *API Monetization*.

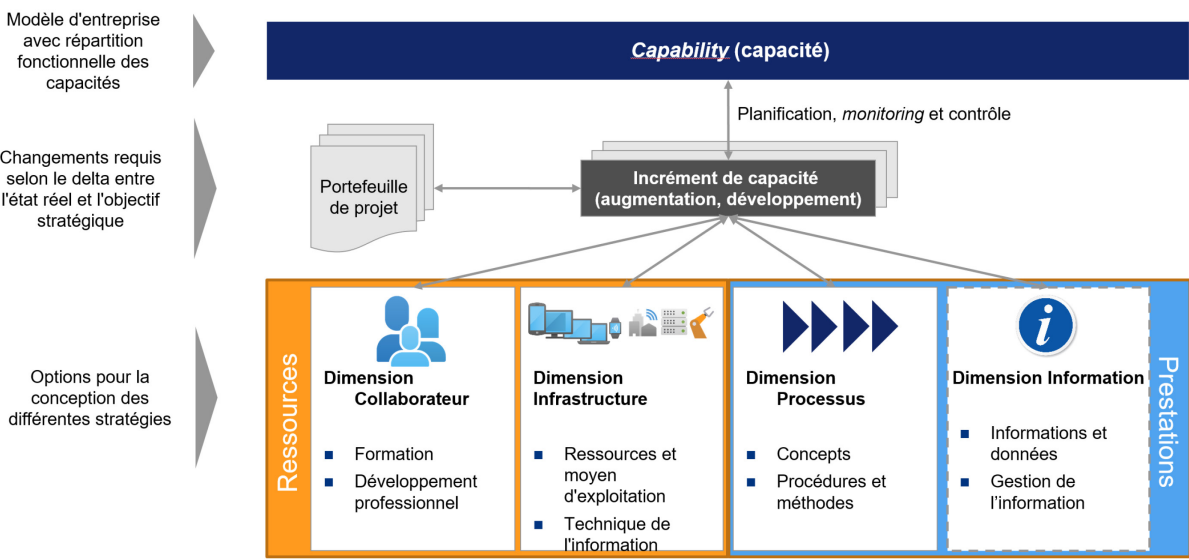


Figure 23 Incréments et dimensions des capacités

Dimensions de capacités		Exemples d'incréments de capacités
Ressources	Collaborateurs	Développer la compétence de monétisation, attirer des collaborateurs informatiques d'un certain profil
	Infrastructure	Évaluation d'un produit de monétisation
Prestations	Processus	Définition d'un processus de planification des recettes provenant de l'exploitation de l'API
	Informations	Développement de modèles de données pour les modèles tarifaires

Tableau 36 Exemples d'incréments de capacités pour la capacité *API Monetization*

10.2 Description des objets d'information

10.2.1 *Manage API lifecycle*

Objet d'information	Description
Fournisseur API	Il s'agit de l'UA qui propose le service public numérique. Pour les API avec enregistrement d'une identité, le fournisseur API représente les intérêts de la ressource dans le système IAM. C'est pourquoi le partenaire API du fournisseur API est également la partie de confiance (<i>relying party</i>) ⁷² .
Fournisseur FP	Il peut déléguer le développement ou le fonctionnement du service public numérique à un FP.
Partenaire API	Il s'agit du partenaire qui utilise le service public numérique. Ce peut être soit une personne, soit une entreprise, soit une autre UA de la Confédération.
Partenaire FP	Il peut lui aussi déléguer le développement ou le fonctionnement du client à un FP.
<i>API release</i>	Une version API est une version exécutable ⁷³ d'une API. Un <i>API release</i> est une version de l'API publiée par le service de catalogue et mise à disposition des partenaires API qui peuvent l'utiliser sous forme d'instance API exécutée. Un <i>API release</i> comprend en général les objets de données <i>API code</i> et/ou configuration API ainsi que la documentation API.
<i>API version ID</i>	Il s'agit de la version d'un <i>API release</i> . Chaque <i>API release</i> a une <i>API version ID</i> . Celle-ci suit un plan de versionnage (voir ch. 9.5).
<i>API code</i>	Il représente le code exécutable de l'API. Il peut être développé par le FP ou acquis gratuitement ou contre paiement. Il sert de base pour générer l' <i>API build</i> . Il n'est pas obligatoire pour les applications qui fournissent une <i>API build</i> comme produit d'infrastructure API.
Configuration API	Elle permet de paramétrer le service <i>Gatekeeper</i> et le service Ressources. Les paramètres peuvent notamment être un nombre d'instances API défini, une dotation en mémoire ou des quotas d'accès API. Le fournisseur FP met à disposition la configuration API du service <i>Gatekeeper</i> lors du développement (CT), le partenaire API la choisit lors la définition (DT) (le choix implique l'enregistrement d'une identité) et le service <i>Gatekeeper</i> l'utilise lors de l'exécution (RT). La configuration API du service Ressources dépend en général de celle du service <i>Gatekeeper</i> . Dans les environnements d'exploitation matures, les configurations API peuvent être contrôlées par le service <i>Reporting</i> et/ou le service <i>Monetization</i> .
Documentation API	La documentation API comprend toutes les informations techniques nécessaires pour pouvoir décider d'utiliser l'API et pour s'y adresser correctement à partir d'un client API. Elle contient en particulier la spécification API, les <i>API release notes</i> et les SLO, qui ne sont pas représentés en tant qu'objets d'information par souci de simplification.
Documentation spécialisée	Elle décrit les services publics numériques d'un point de vue technique. Elle renvoie également aux bases légales et contient un plan directeur avec des informations sur le cycle de vie API, plan qui annonce le cas échéant la révocation possible de l'API. Elle contient aussi la convention d'utilisation (voir capacité <i>Legal contract management</i>). Elle constitue la base de la documentation API, qui s'y réfère pour décrire les aspects techniques couverts par un <i>API release</i> .
<i>API build</i>	Il s'agit de progiciels livrables. Ils constituent la base de l'infrastructure API et, potentiellement, des instances API (voir ch. 9.1). Ils peuvent être gratuits ou payants ou encore être générés à partir de l' <i>API code</i> et leur persistance est assurée dans le <i>repository</i> ou dans un <i>registry</i> . ⁷⁴ Si une couche API ou une composante système est livrée en vue de son intégration API dans un service spécialisé (voir figure 20), ce dernier disposera aussi d'une instance API.

Tableau 37 Objets d'information de l'étape de processus *Manage API lifecycle*

⁷² https://intranet.dti.bk.admin.ch/isb_kp/de/home/themen/iam-bund/dokumente-steuerung-iam-bund.html

⁷³ Pour simplifier, la version API n'est pas modélisée dans le modèle d'information. Les versions API ne sont utilisées que par les FP à des fins internes.

⁷⁴ Un *API build* est par exemple un conteneur chargé dans un *container registry*.

10.2.2 Discover APIs

Objet d'information	Description
Métadonnées API	Elles constituent une description structurée du profil de l'API, qui peut être traitée par une machine, un peu comme une entrée de catalogue. Elle sert à publier en ligne dans des registres d'API des informations sur l'existence et les caractéristiques des API dans l'environnement fédéral et sur les UA qui les proposent. Elles renvoient à la documentation API sur la <i>landing page</i> .
Service de catalogue	Il s'agit de l'élément clé du registre des API. Il gère les métadonnées API publiées par les UA et propose une fonction de recherche.

Tableau 38 Objets d'information de l'étape de processus *Discover API*

10.2.3 Register client

Objet d'information	Description
Objet du partenaire	Une UA propose des services publics numériques à ses partenaires API, qui peuvent être des personnes morales ou des personnes physiques. Si l'identification des partenaires API est nécessaire pour obtenir un service public numérique, ceux-ci apparaissent dans le modèle d'information. Ils sont alors visibles dans des objets de données dédiés de la gestion des données de référence.
Rôle de partenaire	Si un objet partenaire existe dans la gestion des données de référence, il est possible de lui attribuer les rôles de partenaire liés à l'obtention de services publics (pas uniquement numériques). Pour obtenir des services publics numériques, il faut que les rôles correspondants soient attribués au partenaire API.
Service Données de référence	Il s'agit de l'objet de service qui gère les objets de partenaire et les rôles de partenaire.
Identité	L'identité est la représentation du sujet dans le monde numérique. Elle peut représenter une personne physique ou un utilisateur technique. Elle est créée lors de la définition (DT). Elle possède des attributs et des rôles API, peut porter un identifiant et être soumise à un niveau de confiance (LoA). Il faut définir un LoA pour les partenaires API et faire en sorte que l'IAM veille à son respect lors d'un accès ; c'est-à-dire que l'identifiant doit remplir les conditions du LoA. Les comptes des services spécialisés sont attribués aux identités. Si le partenaire API doit s'identifier pour obtenir un service public, il faut un objet partenaire pour pouvoir créer l'identité.
Rôle API	Le rôle est la contrepartie applicative d'un droit modélisé sur la base de rôles. D'un point de vue applicatif, les rôles peuvent être organisés de manière hiérarchique et contenir des règles d'attribution (automatique) à une identité. Si le partenaire API doit s'identifier pour obtenir un service public, il faut un rôle de partenaire (droit mentionné) pour créer le rôle API. Les rôles API sont définis et mis en œuvre lors du développement (CT) et attribués à une identité lors de la définition (DT).
Services IAM DT/RT	L'objet de service <i>IAM Services</i> représente deux groupes définis de services IAM : <ul style="list-style-type: none"> - Services IAM fournis lors de la définition (par ex. service d'identifiant) → <i>IAM Services Definition Time</i> (DT), comme partie de l'étape de processus <i>Register client</i> - Services IAM fournis lors de l'exécution (par ex. service d'authentification). → <i>IAM Service Run Time</i> (RT), comme partie de l'étape de processus <i>Operate API</i> Les services IAM CT gèrent notamment les identités, les rôles API et les attributions de rôles API aux identités. Les services IAM RT génèrent entre autres le jeton sécurisé. Les services IAM DT permettent à une personne physique de se connecter (par les services standard eIAM ou le portail SSO) et de relier un client ou son identité à une API représentée dans les rôles API.
Identifiant	L'identifiant est la représentation applicative d'une preuve d'identité, qui est attribuée exactement à une identité.

Tableau 39 Objets d'information de l'étape de processus *Register client*

10.2.4 Operate API

Objet d'information	Description
Sujet	Il s'agit d'une personne physique qui utilise la ressource. Il utilise la ressource directement en utilisant un client ou indirectement par l'intermédiaire d'un portail de services publics.
Ressource	Elle représente les données et les prestations pour lesquelles le service public numérique a été conçu.

Objet d'information	Description
Client	Le client est la chose et/ou la fraction de logiciel qui accède au service Ressources en passant par le service <i>Gatekeeper</i> . Les applications mobiles, les applications de navigateur ou de bureau sont des exemples types de clients API. Un client peut également être intégré dans une application de portail qui offre des services publics. L'objet de service Client est axé sur l'accès API et masque les aspects du guidage de l'utilisateur.
Service <i>Gatekeeper</i>	Il s'agit du service intermédiaire entre le client et le service Ressources, qui protège les ressources contre les dangers du monde extérieur (p. ex. accès non autorisé, surcharge) et assure un fonctionnement conforme au niveau de service. Il est paramétré sur la base de la configuration de l'API. Le WAF peut être associé au service <i>Gatekeeper</i> mais il est facultatif, car l' <i>API gateway</i> est conçu pour couvrir la fonction de pare-feu d'application web.
Service Ressources	Il s'agit du service <i>backend</i> , par exemple un service spécialisé auquel le service <i>Gatekeeper</i> accède. Il met à disposition toutes les ressources liées au service public numérique, même s'il doit faire appel à d'autres services de ressources pour cela. Il est paramétré sur la base de la configuration de l'API.
API build	Voir ch. 10.2.1
Configuration API	Voir ch. 10.2.1
Service d'approvisionnement	Il s'agit de l'objet de service qui met à disposition les <i>API builds</i> et les configurations API pour l'approvisionnement (livraison). Il fournit l'infrastructure API et les instances API pour le service <i>Gatekeeper</i> et le service Ressources. La livraison est déclenchée dans le cadre de la publication de l'API.
Services IAM DT/RT	Voir ch. 10.2.3
Jeton sécurisé	Il s'agit de l'élément central d'une fédération, car il contient toutes les informations relatives à l'identité elle-même, à son authentification et aux informations sur les rôles API. Le jeton sécurisé peut aussi être considéré comme une sorte de preuve d'identité ou d'identifiant, qui est acceptée et évaluée par un consommateur de jetons sécurisés. Il n'est délivré qu'après une authentification réussie de l'identité et transmis par voie cryptographique sécurisée.
Évènement	Un évènement est un évènement qui doit être enregistré.
Évènement opérationnel	C'est un évènement qui sert de base à la mesure des IPC opérationnels ou à la journalisation d'autres informations opérationnelles. Les évènements pertinents pour le SLA sont des évènements opérationnels.
Évènement technique	C'est un évènement qui sert de base à la mesure des IPC techniques et à la détermination de la valeur commerciale.
Service <i>Logging</i>	Il s'agit de l'objet de service qui journalise tous les évènements et les met à la disposition de du FP qui fournit l'API.
Service <i>Monitoring</i>	Il s'agit de l'objet de service qui détecte les évènements pertinents pour le SLA et les signale au FP qui fournit l'API en passant par les canaux de notification appropriés.

Tableau 40 Objets d'information de l'étape de processus *Operate API*

10.2.5 Analyse API operation & usage

Objet d'information	Description
Indicateur clé de performance (ICP)	Il s'agit d'un indicateur permettant de mesurer la performance d'un service public numérique. Les évènements, techniques ou opérationnels, fournissent les données permettant de mesurer les ICP.
Service <i>Reporting</i>	Il fournit les prestations suivantes : <ul style="list-style-type: none"> - Il permet d'effectuer des évaluations et de générer des rapports. - Il peut se servir d'une intelligence artificielle pour les évaluations. - Il peut également contrôler le service <i>Gatekeeper</i> et le service Ressources au moyen de la configuration API.
Rapport	Il s'agit d'un rapport sur des évènements. Il en existe de nombreux types. L'architecture API de la Confédération en retient trois. D'autres sont autorisées selon les cas.
Contrôle de prestations	Il s'agit d'un rapport spécialisé qui indique la consommation d'un service public numérique sur une période donnée. Le compte y fait référence.
Rapport sur la valeur commerciale	Il s'agit d'un rapport spécialisé qui indique la valeur commerciale d'un service public numérique. Il renvoie à un ou plusieurs modèles tarifaires.
Rapport ICP	Il s'agit d'un rapport spécialisé qui montre l'évolution des ICP sur une période donnée.
Modèle tarifaire	Il sert à facturer les services publics numériques selon une certaine méthode afin de générer des recettes.
Compte	Le compte est établi sur la base du contrôle de prestations en appliquant un modèle tarifaire.
Service <i>Monetization</i>	Il fournit les prestations suivantes :

Objet d'information	Description
	<ul style="list-style-type: none"> - Il reçoit le contrôle des prestations, y applique un modèle tarifaire et génère ainsi le compte. - Il sert à établir des plans utilisation. - Il peut contrôler le service <i>Gatekeeper</i> et le service Ressources au moyen de la configuration API.

Tableau 41 Objets d'information de l'étape de processus *Analyse API operation & usage*

10.3 Conventions d'échange de données

10.3.1 Types d'interface

Le type de service spécialisé influence le type d'interface à utiliser lors du développement d'une API. Les services spécialisés avec base de données connectée sont surtout utilisés pour effectuer des requêtes de données ou de documents, tandis que les services spécialisés sans base de données connectée le sont surtout pour appeler des fonctions ou effectuer des opérations. Les types d'interfaces peuvent donc être répartis entre les deux types représentés dans le tableau 42.

Type d'interface	Description
Axé sur les ressources	Le service spécialisé sert à fournir des ressources qui peuvent être consultées ou modifiées via l'API. Les clients communiquent avec le serveur en échangeant des ressources dans un format de données défini (voir ch. 10.3.2). Les ressources peuvent aussi être des documents (csv, docx, jpg, pdf, xml, zip, etc.). Les API axées sur les ressources utilisent le style d'architecture REST (voir ch. 8.3.2.1). Les REST API peuvent transmettre n'importe quel format de données et communiquent toujours via HTTP. Les REST API sont très répandues et se prêtent très bien à une utilisation dans les API publiques.
Axé sur les méthodes	Le service spécialisé vise à fournir des opérations (méthodes) sur les ressources que les clients peuvent appeler. Parmi les API axées sur les méthodes les plus courantes, on trouve les services web WS-* qui utilisent SOAP/WSDL (voir ch. 10.3.5). Pour les API axées sur les méthodes, XML est le format de données le plus répandu.

Tableau 42 Types d'interface

10.3.2 Formats de données

Le tableau 43 présente une vue d'ensemble des formats de données utilisés lors des échanges de données via les API. Les formats de données se distinguent entre autres par l'*overhead*, le typage, le style et la lisibilité. Si les formats de données sont choisis selon les bonnes pratiques du tableau 23, l'API sera mieux accueillie par les développeurs de clients API.

Format de données	Description
JSON ⁷⁵	JavaScript Object Notation (JSON) est un format de données textuel léger ⁷⁶ dérivé de JavaScript. Il prend en charge les paires clé/valeur (objets), les listes (tableaux) et les imbrications. Il est plus utilisé que XML, car il entraîne moins d' <i>overhead</i> .
XML ⁷⁷	<i>Extensible markup language</i> est un format de données textuel qui fonctionne comme HTML avec des balises, des attributs et des imbrications. Les balises, les attributs et la structure peuvent être spécifiés dans une définition de schéma XML (XSD). La définition de schémas permet aussi de valider les contenus XML et rend le traitement du XML plutôt complexe.
CSV	<i>Comma separated values</i> est un format de données textuel fonctionnant en lignes sur lesquelles les valeurs sont séparées par des virgules (ou par des points-virgules ou des deux-points). Il ne permet de représenter que des structures en tableau.
Format binaire	Les formats binaires sont utilisés dans les cas nécessitant une performance élevée. Exemples : les codecs audio/vidéo pour le <i>streaming</i> ou les <i>protocol buffers</i> (protobuf) lors de l'utilisation du protocole API gRPC.

Tableau 43 Formats de données

JSON, XML et CSV sont des formats de données indépendants du langage de programmation et peuvent être combinés aussi bien avec les interfaces axées sur les ressources qu'avec celles axées sur les méthodes. Les REST

⁷⁵ <https://www.json.org>

⁷⁶ « Léger » fait référence au « poids » que représente la transmission au moyen d'un format de données ou d'un protocole. Plus l'*overhead* associé à une syntaxe de format de données ou à une syntaxe de protocole API est faible, plus ce format de données ou ce protocole API est léger.

⁷⁷ <https://www.w3.org/TR/xml>

API (axées sur les ressources) fonctionnent généralement avec JSON. Le XML vient en deuxième position. Les REST API les plus courantes proposent donc souvent les deux formats.

10.3.3 Message exchange patterns (MEP)

Un MEP décrit le déroulement et l'ordre des interactions entre le client et le serveur. Les MEP et les interfaces peuvent être combinés avec beaucoup de liberté. En revanche, les MEP sont étroitement liés aux protocoles API. Le tableau 44 donne un aperçu des différents MEP.

MEP	Sync/async	Description
<i>Request-response</i>	synchrone	Ce MEP représente la forme la plus simple de communication entre client et serveur et définit le comportement standard de HTTP. Le client envoie une requête au serveur et ne poursuit son travail qu'après avoir reçu la réponse. C'est pourquoi ce MEP est qualifié de synchrone. Le MEP est utilisé lorsque le serveur répond rapidement.
<i>Request-confirm-poll</i>	asynchrone	Dans ce MEP asynchrone, la réponse à la requête consiste uniquement en une confirmation de son acceptation. La confirmation contient une <i>request ID</i> . Le serveur place la requête dans une file d'attente et ne la traite que lorsque des capacités sont disponibles. Une fois la confirmation reçue, le client interroge (<i>polling</i>) le serveur à intervalle défini avec la <i>request ID</i> . L'interrogation n'aboutit que lorsque le serveur a terminé le traitement de la requête et qu'il transmet la réponse au client dans une <i>polling-response</i> . Ce MEP est utilisé lorsque le traitement dans le serveur prend un certain temps et que le client doit rester responsable du transfert du résultat.
<i>Request-confirm-callback</i>	asynchrone	Ce MEP asynchrone s'inspire du MEP <i>request-confirm-poll</i> , mais donne la responsabilité du transfert du résultat au serveur. La requête du client contient une adresse de rappel que le serveur pourra appeler une fois le traitement terminé. Le client n'a pas besoin d'interroger le serveur. La communication entre le client et le serveur est bidirectionnelle. Ce MEP est particulièrement pratique lorsque pour améliorer les performances, le serveur utilise plusieurs instances pour traiter des requêtes et qu'on ne sait donc pas quelle instance transmettra le résultat.
<i>Point-to-point</i> (messaging)	asynchrone	Ce MEP consiste en un simple transfert de messages asynchrone du client vers le serveur via une file d'attente. Le client émet et le serveur reçoit. Le premier connaît le second. Même si la file d'attente contient des messages pour différents destinataires, chaque message n'est destiné qu'à un seul serveur. Les événements concernant les entrées de messages dans la file d'attente sont toujours délivrés. Les messages sont stockés dans la file d'attente jusqu'à ce que le destinataire soit prêt à les récupérer.
<i>Publish-subscribe</i>	asynchrone	Ce MEP permet une communication axée sur les événements entre le client et le serveur en mode bidirectionnel. Le serveur permet à plusieurs clients, lors de la définition (DT), de s'abonner à des événements sur des thèmes spécifiques (sujet). Lors de l'exécution (RT), le serveur informe les clients via des événements dès que de nouvelles données sont disponibles pour un sujet. Le serveur n'a pas besoin de connaître ses clients. Selon le contenu informatif de l'événement qu'il reçoit, le client réagit de l'une des trois manières suivantes ⁷⁸ : <ul style="list-style-type: none"> - L'événement contient une ID qui permet de récupérer un certain enregistrement auprès du serveur. - L'événement contient des ID ou des attributs qui définissent la manière dont le client doit poursuivre son traitement. Il se peut que des données supplémentaires doivent être obtenues auprès du serveur. - Il n'est pas nécessaire d'obtenir d'autres données du serveur, car l'événement contient toutes les données pertinentes pour le client (par ex. événements de journalisation).
<i>Streaming</i>	asynchrone	Le <i>streaming</i> est un type de MEP <i>publish-subscribe</i> , dans lequel l'événement contient toutes les données pertinentes pour le client : le client se connecte au serveur avec des informations sur le sujet. La connexion avec le serveur est ensuite maintenue. Dès qu'elles sont disponibles sur le serveur, les nouvelles données sont transmises au client par cette connexion.

Tabelle 44 Message exchange patterns

⁷⁸ <https://martinfowler.com/articles/201701-event-driven.html>

Bien que pour les MEP *request-confirm-callback* et *publish-subscribe* la communication soit bidirectionnelle, c'est toujours le client qui initie la communication et le serveur qui y répond. Pour le MEP *request-confirm-callback*, l'impulsion est donnée par la requête, pour le MEP *publish-subscribe* par l'abonnement.

10.3.4 Types de messages

Les messages sont les unités de base de la communication entre le client et le serveur. Ils peuvent être par exemple des ID, des chaînes de caractères, des objets, des commandes, des événements, etc. Les messages peuvent être de nature très variée et n'ont pas de but conceptuel précis. Les messages sont donc génériques. L'architecture API de la Confédération fonctionne avec les types de messages définis dans le tableau 45.

Type de message	Description	Caractéristiques
Requête	Une requête est un message qui permet de demander des données ou des informations. Il s'agit d'une communication 1:1 entre l'émetteur et le récepteur.	<ul style="list-style-type: none"> - Une requête est adressée à un seul destinataire. L'émetteur connaît le récepteur. - Une requête est une interrogation, c'est-à-dire que l'émetteur demande du récepteur ce dont il a besoin. - L'émetteur attend dans tous les cas une réponse : soit le résultat de la requête, soit un message indiquant que la requête ne peut pas être traitée. - La gestion des erreurs est de la responsabilité de l'émetteur. Celui-ci surveille le traitement du côté du destinataire.
Commande	Une commande est un message sous la forme d'une communication 1:1 entre un émetteur et un récepteur.	<ul style="list-style-type: none"> - Une commande est adressée à un seul destinataire. L'émetteur connaît le récepteur. - Une commande est un ordre, c'est-à-dire que l'émetteur dit au récepteur ce qu'il doit faire. - L'émetteur attend 0 - N réponses. <ul style="list-style-type: none"> - 0 = <i>fire & forget</i> - 1 = normal - N = lorsque le destinataire traite un processus et génère plusieurs réponses - La gestion des erreurs est de la responsabilité de l'émetteur. Il surveille le traitement du côté du destinataire (sauf dans le cas de <i>fire & forget</i>). - Le récepteur peut rejeter la commande s'il ne peut pas la traiter.
Réponse	Une réponse est un message qui répond à une requête ou à une commande.	<ul style="list-style-type: none"> - Une réponse est un message qui répond à une seule requête ou une seule commande. - Chaque requête et chaque commande doit donner lieu à une réponse (sauf dans le cas de <i>fire & forget</i>).
Évènement	Un évènement est un message qui informe d'un changement. Il n'est pas important pour l'auteur d'un évènement de savoir qui prend connaissance de l'évènement et y réagit.	<ul style="list-style-type: none"> - Un évènement informe d'un changement. - Les évènements utilisent toujours le MEP <i>publish-subscribe</i>. - L'auteur n'a pas besoin de savoir s'il y a des abonnés. - L'abonné décide s'il réagit et comment. - Il n'y a pas de réponse. Il n'est pas important pour l'auteur de savoir si quelqu'un réagit. - La gestion des erreurs est de la responsabilité de l'abonné. - Comme les évènements sont des faits, les abonnés ne peuvent pas les rejeter.

Tabelle 45 Types de messages

10.3.5 Protocoles API

Les MEP sont étroitement liés aux protocoles API. Comme les autres aspects des conventions d'échange de données, le MEP doit être choisi en fonction du public cible. Les protocoles API les plus répandus sont énumérés dans le tableau 46.

Protocoles API	Description
HTTP	HTTP est le protocole standard d'Internet. La version la plus récente est HTTP/2, qui a été publiée en 2015 et est désormais prise en charge par les navigateurs les plus répandus. HTTP est un protocole de la couche Application, mais il est parfois utilisé par d'autres protocoles, par exemple gRPC ou SOAP/WSDL, comme protocole de la couche Transport. Il est limité au MEP <i>request-response</i> .
WebSockets ⁷⁹	WebSockets est un protocole de la couche Application de l' <i>Internet Engineering Task Force</i> (IETF) qui permet une communication en continu en duplex intégral via TCP et qui est supporté par les navigateurs les plus répandus. Il s'agit d'un protocole dans lequel, contrairement à HTTP, les deux parties sont sur un pied d'égalité. Les WebSockets permettent une communication bidirectionnelle continue entre le client et le serveur avec une connexion toujours ouverte. Ils conviennent donc pour la transmission d'événements du serveur au client.
MQTT ⁸⁰	MQTT est une norme de l'OASIS pour les applications IdO. Il est conçu comme un protocole de la couche Transport fondé sur le MEP <i>publish-subscribe</i> , fonctionne avec une bande passante réseau minimale et est donc considéré comme très léger.
AMQP ⁸¹	AMQP est un standard OASIS fondé sur les MEP <i>point-to-point</i> et <i>publish-subscribe</i> (messagerie) pour l'échange de messages commerciaux.
gRPC ⁸²	gRPC est un cadre <i>open source</i> moderne et puissant pour les <i>remote procedure calls</i> (RPC), qui peut être exécuté dans n'importe quel environnement. Il peut connecter des services de manière efficace et offre un soutien pour le <i>load balancing</i> , le <i>tracing</i> , le <i>health checking</i> et l'authentification. gRPC est fondé sur HTTP/2 et utilise l'IDL protobuf.
SOAP/WSDL ⁸³	SOAP est un protocole du W3C conçu pour l'échange d'informations structurées dans un environnement décentralisé. Il utilise les technologies XML pour définir un cadre de messagerie extensible qui permet d'échanger des messages via une multitude de protocoles sous-jacents. Il utilise XML comme format de données et généralement HTTP comme protocole de la couche Transport pour permettre aux clients sur serveurs d'exécuter les méthodes des services web SOAP. Les méthodes sont spécifiées au moyen du WSDL en combinaison avec une définition de schéma XML (XSD) comme IDL.

Tabelle 46 Protocoles API

10.4 Resource description framework / linked data⁸⁴

On entend par *linked data* des données structurées sur la base du RDF publiées pour être utilisées dans des logiciels (communication M2M). Le RDF est un cadre de représentation des informations sur les ressources. Les ressources peuvent être de nature très variées : documents, personnes, objets physiques, concepts abstraits, etc. Le RDF est conçu pour les situations dans lesquelles les informations sur le web doivent non seulement affichées pour des personnes, mais également traitées par des applications logicielles.

Les *linked data* sont utiles pour relier des objets de données en réseau, pour suivre des connexions, pour agréger des données et pour définir (sémantique) les objets de données de manière conventionnelle. Le RDF permet de faire des déclarations sur les ressources. Le format des déclarations est simple. Il a toujours la même structure :

< sujet > < prédicat > < objet >

Cette structure est appelée triplet RDF. Une déclaration (ou assertion) RDF exprime une relation entre deux ressources. Le sujet et l'objet représentent les deux ressources en relation. Le prédicat précise la nature de la relation. La relation est formulée de manière directive (du sujet à l'objet) et est appelée propriété RDF. La signification du sujet, du prédicat et de l'objet est définie dans des vocabulaires RDF recourant aux IRI.

Un IRI identifie une ressource. Les URL, utilisées par les personnes comme adresses web, sont une forme d'IRI. Pour la définition de la sémantique, on trouve d'autres formes d'IRI qui sont des identificateurs de ressources ne traitant pas de la localisation ou de l'accès à celles-ci. Le terme IRI est une généralisation de l'URI, qui permet d'utiliser des caractères non ASCII dans la chaîne de caractères IRI.

Les données liées s'obtiennent au moyen de requêtes formulées dans le langage SPARQL à partir de serveurs SPARQL ou de points de sortie SPARQL. Les services *Linked data* sont très répandus sur Internet. Une seule requête SPARQL peut être adressée à plusieurs de ces services. Les données liées sont gérées sous la forme de triplets RDF dans ce que l'on appelle des *triplestores* RDF. Les points de sortie SPARQL donnent accès à ces triplets RDF.

⁷⁹ <https://datatracker.ietf.org/doc/html/rfc6455>

⁸⁰ <https://mqtt.org>

⁸¹ <https://www.amqp.org>

⁸² <https://grpc.io>

⁸³ <https://www.w3.org/TR/soap12-part1>

⁸⁴ <https://www.w3.org/TR/rdf11-primer>

10.5 Table des abréviations

Abréviation	Signification
ABB	Conseil de l'architecture de la Confédération
AMQP	<i>Advanced message queuing protocol</i>
API	<i>Application programming interface</i> , terme courant pour désigner une interface électronique
BDAT	<i>Business, data, application, technology</i> (niveaux d'architecture TOGAF)
CI/CD	<i>Continuous INTEGRATION / Continuous DEPLOYMENT</i>
CPSV-AP	<i>Core public service vocabulary application profile</i>
CRUD	<i>Create, read, update, delete</i>
DCAT-AP	<i>Data catalogue application profile</i>
FP	Fournisseur de prestations
G2B	De gouvernement à entreprise (<i>gouvernement to business</i>). Il s'agit des interactions entre des administrations publiques et des personnes morales, aussi appelées A2B.
G2C	De gouvernement à citoyen. Il s'agit des interactions entre des administrations publiques et des personnes physiques, aussi appelées A2C.
G2G	De gouvernement à gouvernement. Il s'agit des interactions entre des administrations publiques, aussi appelées A2A (A = administration).
gRPC	<i>gRPC remote procedure calls</i>
H2M	De l'homme à la machine
HTML	<i>HyperText markup language</i>
HTTP	<i>HyperText transfer protocol</i>
IAM	<i>Identity & access management</i>
IDL	<i>Interface definition language</i>
IdO	Internet des objets
IRI	<i>International resource identifier</i>
ISA	Solution d'interopérabilité et cadres communs pour les administrations publiques, les entreprises et les citoyens
JSON	<i>JavaScript Object Notation</i>
LB	<i>Load balancer</i>
LoA	<i>Level of assurance</i>
M2M	de machine à machine
MDG	<i>Master data government</i>
MEP	<i>Message exchange pattern</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
NCSC	Centre national pour la cybersécurité
OAS	<i>OpenAPI specification</i>
OASIS	<i>Organization for the advancement of structured information standards</i>
OGD	<i>Open government data</i>
OTNI	Ordonnance du 25 novembre 2020 sur la transformation numérique et l'informatique (RS 172.010.58)
PAMS	<i>Polymorphic access management system</i>
RC	Recommandation de conception
RDF	<i>Resource description framework</i>
REST	<i>Representational state transfer</i>
SLA	<i>Service level agreement</i>
SLO	<i>Service level objective</i>
SOA	<i>Service oriented architecture</i>
SOAP	<i>Simple object access protocol</i>
SPARQL	<i>SPARQL protocol and RDF query language</i>
TNI	Secteur Transformation numérique et gouvernance de l'informatique, rattaché à la Chancellerie fédérale
TOGAF	<i>The open group architecture framework</i>
UA	Unité administrative
UML	<i>Unified modeling language</i>
URI	<i>Uniform resource identifier</i>

Abréviation	Signification
URL	<i>Uniform resource locator</i>
W3C	<i>World wide web consortium</i>
WAF	<i>Web application firewall</i>
WSDL	<i>Web service description language</i>
XML	<i>Extensible markup language</i>
XSD	<i>XML schema definition</i>

Tableau 47 Abréviations

10.6 Table des figures

Figure 1 Position de l'architecture API de la Confédération	1
Figure 2 Aperçu des initiatives de la stratégie numérique de la Confédération	4
Figure 3 Processus d'intégration	8
Figure 4 Deux expressions des processus d'intégration IP3 et IP4, avec représentation du chemin inverse	9
Figure 5 Cas d'intégration API	11
Figure 6 Position de l'architecture API de la Confédération	13
Figure 7 Structure de l'architecture API de la Confédération et domaines architecturaux selon la norme TOGAF	13
Figure 8 Groupes de principes architecturaux dans l'architecture API de la Confédération	15
Figure 9 Étapes de processus en vue du regroupement des capacités	19
Figure 10 Carte des capacités	21
Figure 11 Rôles spécifiques aux API	31
Figure 12 Gouvernance API selon le modèle de gouvernance informatique	33
Figure 13 Modèle d'information -partie 1	35
Figure 14 Modèle d'information -partie 2	36
Figure 15 Modèle d'information avec objets de données de l'étape de processus <i>Manage API lifecycle</i>	37
Figure 16 Modèle d'information avec objets de données de l'étape de processus <i>Analyse API operation & usage</i>	37
Figure 17 Formes d'architecture API pertinente pour l'IAM	44
Figure 18 Modèle d'information pertinent pour l'IAM - forme 1	45
Figure 19 Modèle d'information pertinent pour l'IAM - forme 4	46
Figure 20 Système de l'architecture API de référence	47
Figure 21 <i>API gateway</i> et <i>message exchange pattern</i>	48
Figure 22 Opérations d'écriture en parallèle et en cascade	52
Figure 23 Incréments et dimensions des capacités	55

10.7 Table des tableaux

Tableau 1 Dimensions de la vision	1
Tableau 2 Dimensions de la vision	5
Tableau 3 Notions	6
Tableau 4 Définition des partenaires	6
Tableau 5 Définition des types d'API	7
Tableau 6 Processus d'intégration	9
Tableau 7 Cas d'intégration API	12
Tableau 8 Principes de l'architecture API de la Confédération	18
Tableau 9 Étapes du processus et notions de temps	20
Tableau 10 Capacités de l'étape de processus <i>Manage API lifecycle</i>	23
Tableau 11 Capacités de l'étape de processus <i>Discover API</i>	23
Tableau 12 Capacités de l'étape de processus <i>Register client</i>	24

Tableau 13 Capacités de l'étape de processus <i>Operate API</i>	25
Tableau 14 Capacités de l'étape de processus <i>Analyse API operation & usage</i>	25
Tableau 15 Points de vue sur les métadonnées API	27
Tableau 16 Normes de métadonnées API	27
Tableau 17 Modèle de monétisation de l'API direct et indirect	29
Tableau 18 Modèles tarifaires pour la monétisation de l'API	30
Tableau 19 Rôles du fournisseur	32
Tableau 20 Rôles du partenaire	32
Tableau 21 Types d'objets d'information	34
Tableau 22 Objets d'information par type	34
Tableau 23 Bonnes pratiques concernant les conventions d'échange de données	39
Tableau 24 Conditions du style d'architecture REST	40
Tableau 25 Conventions d'échange de données pour GraphQL	40
Tableau 26 Conventions d'échange de données pour les <i>linked data</i>	41
Tableau 27 Conventions d'échange de données pour sedex	41
Tableau 28 Réponses du modèle d'information aux questions pertinentes pour l'IAM	43
Tableau 29 Objets d'information de l'architecture cadre IAM de la Confédération	44
Tableau 30 Types d'utilisation du client	44
Tableau 31 Types de contrôle d'accès	45
Tableau 32 Lien entre l'architecture d'application, l'architecture des données et l'architecture d'entreprise	50
Tableau 33 Types d'objets livrés et normes spécifiques aux API	52
Tableau 34 Composants du numéro de version selon SemVer 2.0.0	53
Tableau 35 Emplacement technique du numéro de version	54
Tableau 36 Exemples d'incréments de capacités pour la capacité <i>API Monetization</i>	55
Tableau 37 Objets d'information de l'étape de processus <i>Manage API lifecycle</i>	56
Tableau 38 Objets d'information de l'étape de processus <i>Discover API</i>	57
Tableau 39 Objets d'information de l'étape de processus <i>Register client</i>	57
Tableau 40 Objets d'information de l'étape de processus <i>Operate API</i>	58
Tableau 41 Objets d'information de l'étape de processus <i>Analyse API operation & usage</i>	59
Tableau 42 Types d'interface	59
Tableau 43 Formats de données	59
Tabelle 44 <i>Message exchange patterns</i>	60
Tabelle 45 Types de messages	61
Tabelle 46 Protocoles API	62
Tableau 47 Abréviations	64