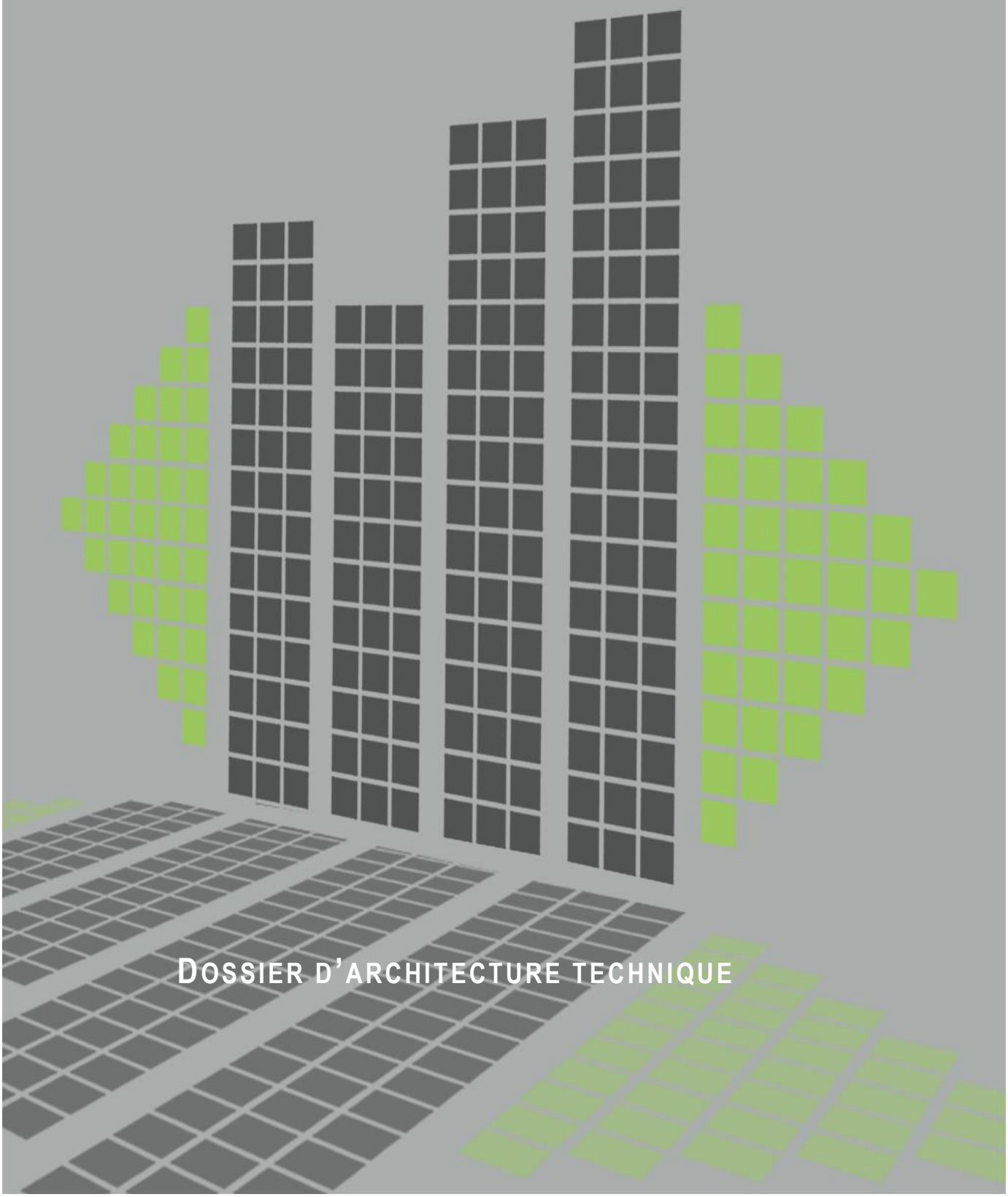
ASIT-VD

EXTRACT



DOSSIER D'ARCHITECTURE TECHNIQUE

arx iT

info@arxit.com ■ www.arxit.com

arx iT ■ 45 av. de la Praille ■ 1227 Carouge Genève ■ Tél. +41 (0)22.344.45.10

RÉFÉRENCE DU DOCUMENT

Référence : DC_ASITVD_Extract_DossierArchitectureTechnique_V1.12.docx

Version : 1.12

HISTORIQUE DU DOCUMENT

Version	Date	Contenu	Créé par
1.0	26/10/2016	Version initiale	JLO
1.1	15/12/2016	Modifications après revue ASIT-VD et planification itération 1	JLO
1.2	26/01/2017	Ajout des contraintes en base de données Modifications après planification itération 2 Modifications datamodel – Mots réservés Hibernate	JLO
1.3	16/02/2017	Modifications après planification itération 3	JLO
1.4	09/03/2017	Modifications après revue itération 3 et planification itération 4	JLO
1.5	06/04/2017	Modifications après revue itération 4 et planification itération 5	JLO
1.6	02/05/2017	Modifications après revue itération 5 et planification itération 6	JLO
1.7	18/05/2017	Modifications après revue itération 6 et planification itération 7	JLO
1.8	08/08/2017	Version finale	JLO
1.9	21/06/2018	Modifications pour Extract v1.1	JLO
1.10	21/06/2019	Modifications pour Extract v1.2	JLO
1.11	14/03/2022	Modifications pour Extract v2.0	JLO
1.12	28/07/2023	Authentification LDAP et double facteur	JLO

info@arxit.com ■ www.arxit.com

arx iT ■ 45 av. de la Praille ■ 1227 Carouge Genève ■ Tél. +41 (0)22.344.45.10

GUIDE DE LECTURE

info@arxit.com ■ www.arxit.com

arx iT ■ 45 av. de la Praille ■ 1227 Carouge Genève ■ Tél. +41 (0)22.344.45.10

1 PRÉAMBULE

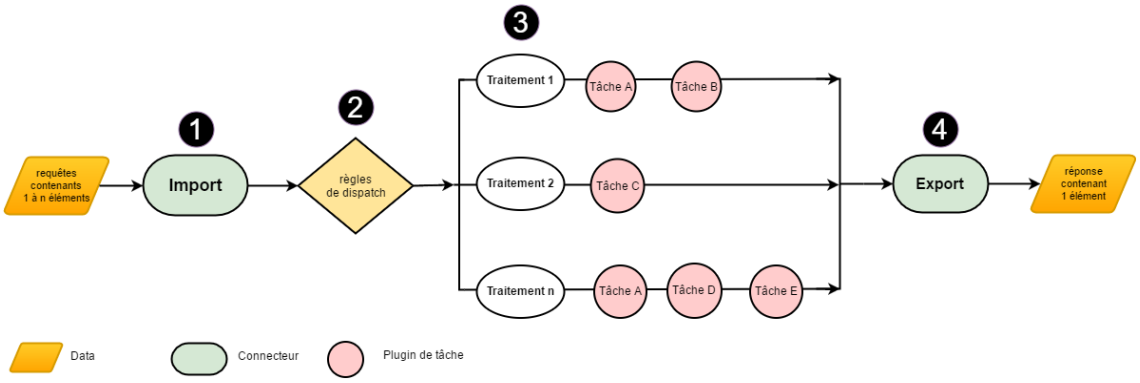
Le présent document est le dossier d'architecture technique du projet **Extract** de l'ASIT-VD. Il contient les éléments structurants du projet tels que l'architecture générale, les flux de données, les interfaces ou encore le modèle de données. Il s'adresse principalement :

- Au chef de projet informatique
- Aux architectes du projet
- A l'équipe de mise en production
- A l'équipe de maintenance

2 ARCHITECTURE GÉNÉRALE

2.1 Schéma logique de principe

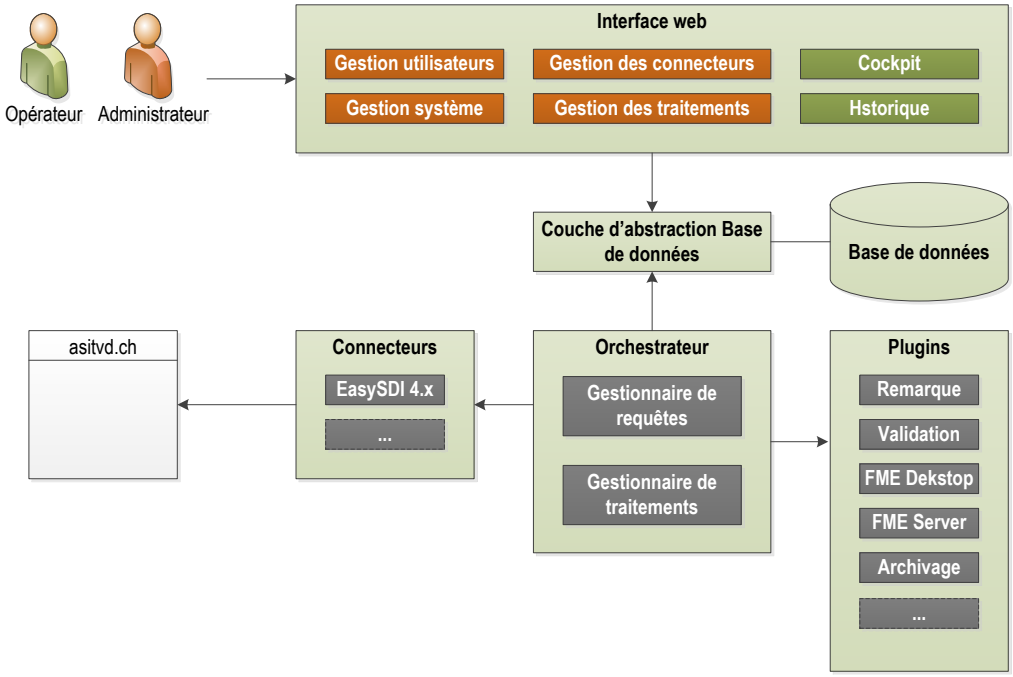
Le schéma logique ci-après présente le principe de processus paramétrable.



1. L'import est réalisé par l'application pour chaque connecteur déclaré et actif
2. Pour chaque élément de requête, l'application applique les règles de dispatch qui définissent quel traitement doit être lancé
3. Au sein de chaque traitement, les tâches pré-paramétrées sont exécutées. Chaque traitement avance indépendamment. Les Tâches d'un Traitement sont potentiellement les suivantes : Ajout d'une remarque, Validation, Extraction, Archivage.
4. L'export est réalisé pour chaque Élément de requête par le Connecteur qui l'a importé

2.2 Schéma d'architecture logicielle

La solution proposée se base sur une architecture modulaire et évolutive. Les différentes briques fonctionnelles sont illustrées sur le schéma ci-dessous.



2.3 Description des flux

La solution n'est composée que d'un seul serveur, aucun flux particulier n'est donc à mettre en place si ce n'est l'accès http(s) vers ce serveur (port 80 ou 443)

2.4 Environnements mis en place

Un unique environnement de développement/intégration est mis en place dans l'infrastructure d'arx iT. Celui-ci est composé d'un simple serveur virtualisé accueillant :

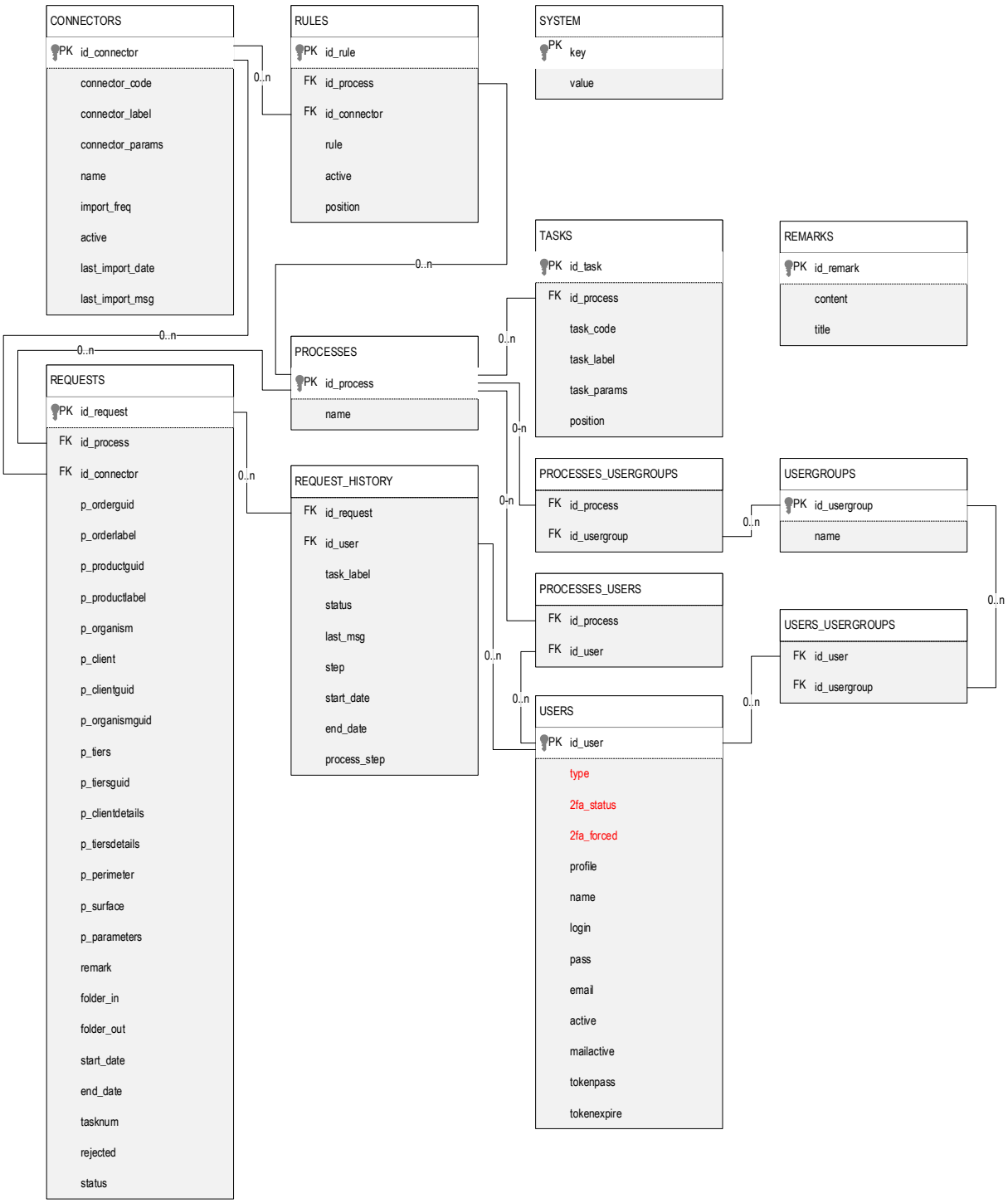
- Une base de données centralisée servant pour les développements et l'intégration de ceux-ci
- Une version de l'application intégrée et stabilisée à chaque fin d'itération

3 DESCRIPTION DES DONNÉES

3.1 Organisation de la base de données

Une unique base de données et un unique schéma sont créés, l'utilisateur propriétaire du schéma est utilisé pour toute connexion depuis la solution mise en place (lecture / écriture)

3.2 Modèle conceptuel de données



3.3 Description des tables

3.3.1 CONNECTORS

Table listant les connecteurs paramétrés dans le système. Il n'est pas possible de modifier/supprimer un connecteur si une requête liée est non terminée. La suppression d'un connecteur implique la suppression des règles associées et le passage à *null* de l'*id_connector* des requêtes liées dans la table REQUESTS

Attribut	Type	Description	Exemple
id_connector	int	**Clé primaire**	*1*
connector_code	varchar 50	Code identifiant le type de connecteur. Récupéré depuis le connecteur lui-même via une fonction spécifique. Le code doit être unique dans le catalogue de connecteurs	*easysdiv4*
connector_label	varchar 255	Label du connecteur Récupéré depuis le connecteur lui-même via une fonction spécifique	*EasySdi V4*
connector_params	varchar 4000	Paramètres propres au connecteur au format json La liste des paramètres et leur type sont récupérés depuis le connecteur lui-même via une fonction spécifique	*cf 4.1*
name	varchar 50	Nom donné à l'instance de connecteur	*asitvd_dev*
import_freq	int	Fréquence de requêtage du connecteur lorsqu'il est actif	*60*
active	boolean	Définit si le connecteur est actif ou non	*true*
last_import_msg	varchar 4000	Dernier message renvoyé par l'import du connecteur	*Success*
last_import_date	datetime	Date et heure du dernier import	*16/11/2016 21:15*

3.3.2 RULES

Table listant les règles appliquées à un connecteur donné et associant un traitement à chacune d'elles.

Attribut	Type	Description	Exemple
id_rule	int	**Clé primaire**	*1*
id_process	int	**Clé étrangère** assurant le lien avec la table PROCESSES	*1*
id_connector	int	**Clé étrangère** assurant le lien avec la table CONNECTORS	*1*
rule	varchar 4000	Définition de la règle selon la syntaxe propre au système	*prod_code == '11'*
active	boolean	Défini si la règle est active ou non	*true*
position	int	Ordonnancement des règles entre elles pour un même connecteur	*5*

3.3.3 PROCESSES

Table listant les traitements paramétrés dans le système. Il n'est pas possible de modifier (i.e. les tâches le composant) un traitement si une requête liée est ONGOING. Il n'est pas possible de supprimer un traitement si une requête liée est non terminée ou s'il est associé à une règle d'un connecteur.

Attribut	Type	Description	Exemple
id_process	int	**Clé primaire**	*1*
name	varchar 255	Intitulé du traitement	*Réseau de Gaz*

3.3.4 TASKS

Table listant les tâches (plugins) paramétrées pour un traitement donné.

Attribut	Type	Description	Exemple
id_task	int	**Clé primaire**	*1*
id_process	int	**Clé étrangère** assurant le lien avec la table PROCESSES	*1*
task_code	varchar 50	Code identifiant le type de plugin de tâche Récupéré depuis le plugin lui-même via une fonction spécifique. Le code doit être unique dans le catalogue de connecteurs	*FME2016*
task_label	varchar 255	Label du plugin de tâche Récupéré depuis le plugin lui-même via une fonction spécifique	*Extraction FME 2016*
task_params	varchar 4000	Paramètres propres au plugin de tâche au format json La liste des paramètres et leur type sont récupérés depuis le plugin lui-même via une fonction spécifique	*cf 0*
position	int	Ordonnancement des tâches entre elles pour un même traitement	*3*

3.3.5 PROCESSES_USERS

Table assurant l'affectation d'un ou plusieurs utilisateurs à un traitement donné.

Attribut	Type	Description	Exemple
id_process	int	**Clé étrangère** assurant le lien avec la table PROCESSES	*1*
id_user	int	**Clé étrangère** assurant le lien avec la table USERS	*1*

3.3.6 PROCESSES_USERGROUPS

Table assurant l'affectation d'un ou plusieurs groupes à un traitement donné.

Attribut	Type	Description	Exemple
id_process	int	**Clé étrangère** assurant le lien avec la table PROCESSES	*1*
id_usergroup	int	**Clé étrangère** assurant le lien avec la table USERGROUPS	*1*

3.3.7 USERS

Table des utilisateurs. Il n'est pas possible de supprimer un utilisateur si celui-ci est associé à un traitement. Lors de la suppression d'un utilisateur les éléments qui lui sont liés dans la table REQUEST_HISTORY se voient affectés d'un *id_user* à *null*, indiquant que l'utilisateur est dorénavant inconnu.

Attribut	Type	Description	Exemple
id_user	int	**Clé primaire**	*1*
user_type	string	Type d'utilisateur : LOCAL : utilisateur dont la base Extract est la source d'authentification LDAP : utilisateur dont un serveur LDAP est la source d'authentification	*LOCAL*
two_factor_status	string	Status du 2fa : ACTIVE / INACTIVE / STANDBY	*ACTIVE*
two_factor_forced	boolean	2fa imposé par l'administrateur ?	*true*
two_factor_token	varchar 100	Chaîne permettant la génération des codes d'authentification à deux facteurs (chiffrée)	*Xxxxx*
two_factor_standby_token	varchar 100	Chaîne permettant la génération des codes d'authentification à deux facteurs (chiffrée) en attente de validation par l'utilisateur	*Xxxxx*
profile	string	Attribut pouvant prendre 2 valeurs : ADMIN : Administrateur OPERATOR : Opérateur Note : les textes correspondant aux valeurs sont gérés dans l'application elle-même afin de garantir la gestion multilingue.	*ADMIN*
name	varchar 50	Nom complet de l'utilisateur	*Yves Blatti*
login	varchar 50	Login de l'utilisateur	*yblatti*
pass	varchar 60	Mot de passe de l'utilisateur	*****
email	varchar 50	Email de l'utilisateur	*yves.blatti@asitvd.ch*
active	boolean	Définit si l'utilisateur est actif ou non	*true*
mailactive	boolean	Définit si les notifications sont actives pour l'utilisateur	*true*
tokenpass	varchar 50	Token utilisé pour la récupération du mot de passe (à usage unique) Attribut réinitialisé à null lors de la connexion suivante de l'utilisateur (qqsoit le mot de passe utilisé)	*so37dd9sxwxdx3449ckl*
tokenexpire	datetime	Date/heure d'expiration du token	*2016.01.01 12 :00*

3.3.8 RECOVERY_CODES

Table des codes de récupération par utilisateur en cas de perte d'accès à l'appareil permettant l'authentification à deux facteurs. Lorsqu'un code est utilisé avec succès, il est supprimé de cette table.

Attribut	Type	Description	Exemple
id_code	int	**Clé primaire**	*1*
Id_user	int	**Clé étrangère** assurant le lien avec la table USERS	*1*
token	varchar 100	Code de récupération (hâché)	*Xxxxxx*

3.3.9 REMEMBER_ME_TOKENS

Table traçant les tokens délivrés pour ne pas demander l'authentification à deux facteurs pour un utilisateur sur une machine pendant un certain laps de temps.

Attribut	Type	Description	Exemple
id_code	int	**Clé primaire**	*1*
Id_user	int	**Clé étrangère** assurant le lien avec la table USERS	*1*
token	varchar 100	Chaîne aléatoire contenue dans le cookie d'autorisation	*Xxxxxx*
tokenexpire	timestamp	Date et heure d'expiration de l'autorisation	*154564566*

3.3.10 REQUESTS

Table listant les requêtes en cours et achevées.

Afin de gérer des connecteurs aux interfaces multiples, le connecteur lui-même doit fournir les éléments de requête au format standardisé pour inscription en base. Ce format est composé des attributs **p_xxx**

Attribut	Type	Description	Exemple
id_request	int	**Clé primaire**	*1*
id_process	int	**Clé étrangère** assurant le lien avec la table PROCESSES Ce paramètre reste à null si aucune règle ne match l'élément de requête	*1*
id_connector	int	**Clé étrangère** assurant le lien avec la table CONNECTORS Ce paramètre est à null si le connecteur associé a été supprimé	*1*
p_orderlabel	varchar 255	Paramètre standardisé de requête Nom de la commande	*123456 / 6*
p_orderguid	varchar 255	Paramètre standardisé de requête Identifiant de la commande, à renseigner également lors de l'envoi du résultat	*677587e2-057f-0144-ddc1-9feca766448f*
p_productguid	varchar 255	Paramètre standardisé de requête Identifiant du produit commandé, à renseigner également lors de l'envoi du résultat	*708e932b-81c3-2ce4-b907-ed07e61ac5f9*

p_productlabel	varchar 255	Paramètre standardisé de requête Intitulé du produit commandé	*Plan du réseau d'eau commune de Bex*
p_organism	varchar 255	Paramètre standardisé de requête Organisme destinataire de la commande	*Terrassements SA*
p_client	varchar 255	Paramètre standardisé de requête Client destinataire de la commande	*Terrassements SA*
p_clientguid	varchar 255	Paramètre standardisé de requête Identifiant du client	*708e932b-81c3-2ce4-b907-ed07e61ac5f9*
p_organismguid	varchar 255	Paramètre standardisé de requête Identifiant de l'organisme	*708e932b-81c3-2ce4-b907-ed07e61ac5f9*
p_tiers	varchar 255	Paramètre standardisé de requête Tiers lié à la commande	*Commune de Bex*
p_tiersguid	varchar 255	Paramètre standardisé de requête Guid du tiers lié à la commande	*708e932b-81c3-2ce4-b907-ed07e61ac5f9*
p_clientdetails	varchar 4000	Paramètre standardisé de requête Détails du client destinataire de la commande	*Rue du Simplon 37* *1880 Bex* *Tel :00.00.00.00.00* *Mail : xxx@yyy.com*
p_tiersdetails	varchar 4000	Paramètre standardisé de requête Détails du tiers lié à la commande	*Rue de la Gare 22* *1131 Tolochenaz* *Tel :00.00.00.00.00* *Mail : xxx@yyy.com*
p_perimeter	varchar 4000	Paramètre standardisé de requête Coordonnées du polygone d'extraction	*POLYGON((6.9378 46.1056,6.1245 …*
p_surface	float	Paramètre standardisé de requête Surface du polygone d'extraction	*123.4 (m2)*
p_parameters	varchar 4000	Paramètre standardisé de requête Liste de paramètres non standardisés passée telle quelle aux plugins de tâches	*{'format' :'pdf', … }*
p_external_url	varchar 255	URL permettant d'accéder aux détails de la commande sur le serveur d'origine	*https://viageo.ch/commandes/123456*
remark	varchar 4000	Eventuelle remarque liée à l'élément de requête Modifiable par les plugins de tâches	*Donnée fournie à titre confidentielle*
folder_in	varchar 255	Répertoire d'entrée de l'élément de requête. Défini par l'application à partir d'un chemin de base (paramétrable par l'administrateur). Il est transmis aux plugins de tâches avec les autres paramètres de l'élément de requête	*/677587e2-057f-0144-ddc1-9feca766448f/20161212180000/in*
folder_out	varchar 255	Répertoire de sortie de l'élément de requête. Défini par l'application à partir d'un chemin de base (paramétrable par l'administrateur). Il est transmis aux plugins de tâches avec les autres paramètres de l'élément de requête	*/677587e2-057f-0144-ddc1-9feca766448f/20161212180000/out*
start_date	datetime	Date et heure de création de la requête dans le système	*22.11.2016 09:00*
end_date	datetime	Date et heure de fin de traitement de la requête	*22.11.2016 09:15*

tasknum	int	Numéro de la tâche en cours dans le process	
rejected	boolean	*False* par défaut Flag indiquant que l'opérateur ou l'administrateur a souhaité stopper le traitement associé et rejeter ainsi la demande	*False*
status	string	Statut de la requête dans son cycle de vie. IMPORTED : Importée – La requête est initialisée dans le système suite à un import connecteur (ou au forçage du statut par l'administrateur) IMPORTFAIL : Echec import - La requête ne peut pas être traitée car son import via le connecteur a échoué ONGOING : En cours – Une règle a permis d'associer la requête à un traitement. L'orchestrateur peut faire avancer celui-ci STANDBY : En attente - Une action manuelle est nécessaire sur la tâche courante (e.g. plugin de validation) ERROR : En erreur - La tâche courante est en erreur (e.g. script FME introuvable) UNMATCHED : Non matchée – Aucune règle n'a permis d'associer un traitement à la requête TOEXPORT : A exporter – La dernière tâche d'un traitement s'est déroulée avec succès. La requête est prête pour l'exportation EXPORTFAIL : Echec export - La requête a été traitée mais son export via son connecteur d'origine a échoué FINISHED : Terminé – La requête a été traitée et l'export s'est correctement déroulé Note : les textes correspondant aux valeurs sont gérés dans l'application elle-même afin de garantir la gestion multilingue.	*IMPORTED*
last_reminder	datetime	Date et heure du dernier rappel dans le cas d'une requête en attente de validation	*22.11.2016 09:00*

3.3.11 REQUEST_HISTORY

Table listant l'historique de traitement pour chacune des requêtes initiées. Pour une requête donnée, il peut y avoir plus d'éléments dans cette table que de tâches puisque l'utilisateur peut revenir en arrière dans le traitement. Cette table doit rester totalement indépendante du process (et donc des tâches) attaché à la requête. Ceci afin de ne pas corrompre l'historique si ledit process est modifié a posteriori. Tout élément d'historique doit donc être dupliqué dans cette table (e.g. *task_label*).

Attribut	Type	Description	Exemple
Id_record	int	Clé primaire	*1*
id_request	int	**Clé étrangère** assurant le lien avec la table REQUESTS	*1*

id_user	int	**Clé étrangère** assurant le lien avec la table USERS 0 = system null = unknow (utilisateur supprimé) Note : les textes correspondant aux valeurs sont gérés dans l'application elle-même afin de garantir la gestion multilingue.	*1*
task_label	varchar 255	Label du plugin de la tâche correspondante ou *Exportation* pour l'étape d'exportation ou *Importation* pour l'étape d'importation	*Extraction FME 2016*
status	string	Statut de la tâche ONGOING : En cours - La tâche est en cours de traitement STANDBY : En attente - Une action manuelle est nécessaire (e.g. plugin de validation) ERROR : En erreur - La tâche est en erreur (e.g. script FME introuvable) ou l'import/export a échoué FINISHED : Terminé – La tâche/import/export s'est terminée avec succès Note : les textes correspondant aux valeurs sont gérés dans l'application elle-même afin de garantir la gestion multilingue.	*FINISHED*
last_msg	varchar 4000	Dernier message renvoyé par le plugin (message d'erreur ou message de succès) ou par le connecteur (étape importation/exportation)	
step	int	Incrémenté de 1 à chaque nouvelle entrée dans cette table (pour un même id_request)	*3*
start_date	datetime	Date et heure de début de traitement de la tâche	*22.11.2016 09:00*
end_date	datetime	Date et heure de fin de traitement de la tâche A null si le plugin de tâche est en cours d'exécution.	*22.11.2016 09:02*
process_step	int	Numéro de la tâche dans le process	*1*

3.3.12 SYSTEM

Table regroupant les paramètres système nécessaire au bon fonctionnement de la solution.

Attribut	Type	Description	Exemple
key	varchar 50	**Clé primaire**	*smtp_port*
value	varchar 65000	Valeur de l'attribut système	*25*

Ci-dessous la liste des clés contenues dans cette table, celles-ci sont non modifiables.

key	value (exemple ou domaines)
freq_scheduler_sec	*1*
dashboard_interval	*10*
smtp_server	*mail.laboite.ch*
smtp_port	*25*
smtp_from_name	*EX3K*
smtp_from_mail	*ex3k.noreply@laboite.ch*

smtp_pass	****
smtp_user	*admin*
smtp_ssl	*0*
base_path	*D:\extract\orders*
display_temp_folder	*false*
mails_enable	*true*
op_mode	*ON / RANGES / OFF*
op_timeranges	*[* *{dayfrom :1, dayto :4, timefrom :7, timeto :18},* *{dayfrom :5, dayto :5, timefrom :7, timeto :16},* *{…}* *]*
ldap_on	*true*
ldap_servers	*ldap.example.com, ldap2.example.com*
ldap_encryption_type	*LDAPS / STARTTLS*
ldap_base_dn	*cn=admin,dc=example,dc=com*
ldap_admins_group	*CN=YourGroup, OU=Users,DC=YourDomain,DC=COM*
ldap_operators_group	*CN=YourGroup, OU=Users,DC=YourDomain,DC=COM*
ldap_synchro_on	*true*
ldap_user	*Service_account_extract*
ldap_password	*xxxxx*
ldap_synchro_freq	*24*
ldap_last_synchro	*22.11.2016 09:00*
standby_reminder_days	*3*
validation_focus_properties	REMARK, PROJECTION, FORMAT

3.3.13 REMARKS

Table regroupant les messages prédéfinis à utiliser comme remarques lors de l'étape de validation d'un traitement

Attribut	Type	Description	Exemple
Id_remark	int	Clé primaire	*1*
content	varchar 65000	Texte de la remarque	*Madame, Monsieur,* *Votre commande n'a malheuresement pas pu être honorée car elle ne correspond pas au périmètre des données…*
title	varchar 255	Titre de la remarque (pour affichage dans la liste)	*Périmètre non valide*

3.3.14 USERGROUPS

Table listant les groupes d'utilisateurs pouvant être affectés à un traitement

Attribut	Type	Description	Exemple
Id_usergroup	int	Clé primaire	*1*
name	varchar 50	Nom du groupe d'utilisateurs	*Responsables cadastre*

3.3.15 USERS_USERGROUPS

Table assurant l'apparrtenance des utilisateurs à des groupes d'utilisateurs

Attribut	Type	Description	Exemple
Id_usergroup	int	Clé étrangère assurant le lien avec la table USERGROUPS	*1*
Id_user	int	Clé étrangère assurant le lien avec la table USERS	*1*

3.4 Suppression des éléments en base

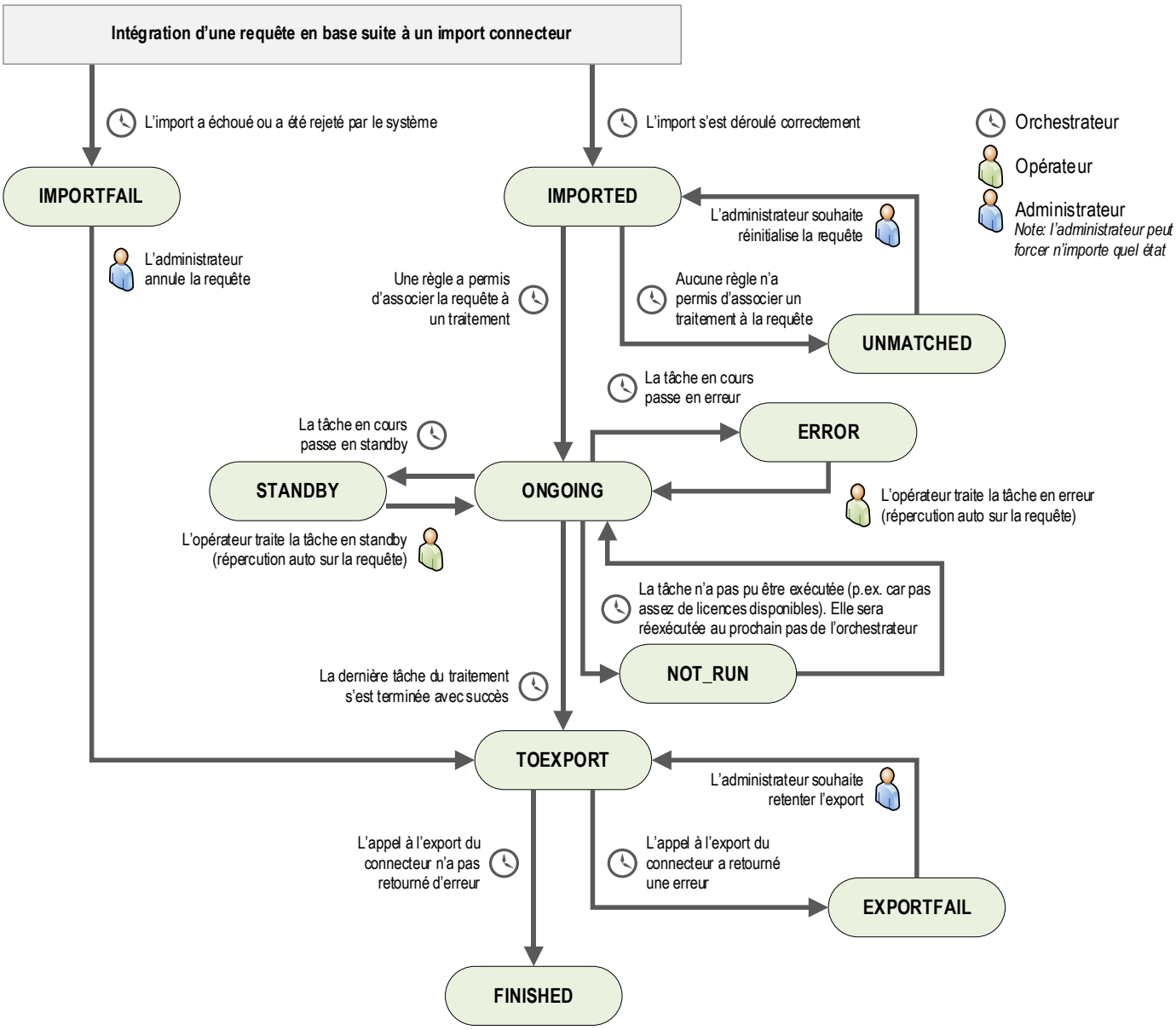
Le tableau ci-dessous indique les contraintes et triggers établis directement en base de données pour la gestion des suppressions en cascade.

Note : ces actions / suppression n'ont donc pas à être gérées au niveau de l'interface web.

Elément source	Elément cible	Suppression / action en cascade
CONNECTORS	RULES	Suppression des éléments dans la table liée RULES Note : ce cas de figure n'est cependant possible que pour les connecteurs sans requêtes liées non terminées. En effet, l'interface ne doit pas permettre de supprimer un connecteur si une requête en cours ou en attente lui est associée
CONNECTORS	REQUESTS	Passage à *null* du *id_connector* de la table REQUESTS Note : ce cas de figure n'est cependant possible que pour les connecteurs sans requêtes liées non terminées. En effet, l'interface ne doit pas permettre de supprimer un connecteur si une requête en cours ou en attente lui est associée
REQUESTS	REQUEST_HISTORY	Suppression des éléments dans la table liée REQUEST_HISTORY.
PROCESSES	REQUESTS	Passage à *null* du *id_process* de la table REQUESTS Note : ce cas de figure n'est cependant possible que pour les requêtes terminées. En effet, l'interface ne doit pas permettre de supprimer un traitement si une requête en cours ou en attente lui est associée.
PROCESSES	RULES	Suppression des éléments dans la table liée RULES Note : ce cas de figure ne devrait cependant jamais arriver puisque l'interface ne doit pas permettre de supprimer un traitement si une règle lui est associée
PROCESSES	TASKS	Suppression des éléments dans la table liée TASKS
PROCESSES	PROCESSES_USERS	Suppression des éléments dans la table liée PROCESSES_USERS
PROCESSES	PROCESSES_USERGROUPS	Suppression des éléments dans la table liée PROCESSES_USERGROUPS
USERS	REQUEST_HISTORY	Passage à *null* du *id_user* de la table REQUEST_HISTORY
USERS	PROCESSES_USERS	Suppression des éléments dans la table liée PROCESSES_USERS
USERS	USER_USERGROUPS	Suppression des éléments dans la table liée USER_USERGROUPS
USERS	RECOVERY_CODES	Suppression des éléments dans la table liée RECOVERY_CODES
USERS	REMEMBER_ME_TOKENS	Suppression des éléments dans la table liée REMEMBER_ME _TOKENS
USERGROUPS	PROCESSES_USERGROUPS	Suppression des éléments dans la table liée PROCESSES_USERGROUPS
USERGROUPS	USER_USERGROUPS	Suppression des éléments dans la table liée USER_USERGROUPS

3.5 Cycle de vie des requêtes

Les requêtes suivent le cycle de vie illustré sur le schéma ci-dessous. Les transitions font apparaitre l'évènement système ou utilisateur à leur origine.



4 DESCRIPTION DES INTERFACES ET SYSTÈMES TIERS

4.1 Catalogue de connecteur

4.1.1 Introduction

Le catalogue de connecteur regroupe l'ensemble des connecteurs disponibles pour le système. Ils doivent tous implémenter les fonctions d'interface suivantes :

- *String getCode* : retourne le code du connecteur, qui doit être unique dans le catalogue
- *String getLabel* : retourne l'intitulé du connecteur
- *String getDescription* : retourne un texte formaté décrivant le connecteur
- *String getHelp* : retourne un texte formaté permettant de guider l'utilisateur dans la saisie des paramètres
- *String getParams* : retourne les paramètres propres au connecteur, au format json. Chaque paramètre est défini par un code unique, un label, un type, son caractère obligatoire ou non et d'éventuels autres balises propres au type. Les types disponibles pour les connecteurs sont les suivants :
  - **text** : chaine de caractères à renseigner via une zone de texte. La longueur max de la chaîne est précisée dans un attribut supplémentaire.
  - **pass** : chaine de caractères à renseigner via une zone de texte masquée. La longueur max de la chaîne est précisée dans un attribut supplémentaire.
- *Object new(params)* : crée une instance du connecteur en passant les paramètres définis par l'utilisateur. Retourne l'objet créé.
- *Object importCommands()* : lance l'import de commande. Retourne un objet composé d'un code erreur, de la liste des éléments de requêtes au format normalisé et le cas échéant un message associé.
- *Object exportResults(request)* : lance l'export du résultat d'une commande. L'élément de requête (contenant notamment le répertoire de sortie, l'attribut remarque, le status, …) est passé en paramètre (objet complet issu de la table REQUESTS). Retourne un code erreur et le cas échéant un message associé.

Les chapitres suivants listent le retour de ces fonctions pour chacun des connecteurs disponibles dans la première version de la solution A noter que les plugins doivent implémenter le multilingue dans le retour des fonctions. Les descriptions ci-dessous correspondent à la version française.

4.1.2 Connecteur Easy SDI v4

Fonction	Retour
getCode	easysdiv4
getLabel	EasySdi V4
getDescription	Connecteur pour la solution EasySdi V4
getHelp	*Se référer directement au connecteur*
getParams	[{'code' : 'url', 'label' : 'URL du service', 'type' : 'text', 'req' : 'true', 'maxlength' : 255}, {'code' : 'login', 'label' : 'Login distant', 'type' : 'text', 'req' : 'true', 'maxlength' : 50}, {'code' : 'pass', 'label' : 'Mot de passe', 'type' : 'pass', 'req' : 'true', 'maxlength' : 50}] *Ci-dessous un exemple de paramétrage correspondant (attribut connector_params de la table CONNECTOR) :* *{* *'url' : 'http://www.asitvd.ch/extractpoint/rest',* *'login' : 'fournisseur12445',* *'pass' : 'motdepasse'* *}*

4.2 Catalogue de plugin de tâches

4.2.1 Introduction

Le catalogue de tâches regroupe l'ensemble des plugins de tâches disponibles pour le système. Ils doivent tous implémenter les fonctions d'interface suivantes :

- *String getCode* : retourne le code du plugin, qui doit être unique dans le catalogue
- *String getLabel* : retourne l'intitulé du plugin de tâche
- *String getDescription* : retourne un texte formaté décrivant le plugin
- *String getHelp* : retourne un texte formaté permettant de guider l'utilisateur dans la saisie des paramètres
- *String getPictoClass* : retourne la classe à utiliser pour le pictogramme du plugin
- *String getParams* : retourne les paramètres propres au plugin de tâche, au format json. Chaque paramètre est défini par un code unique, un label, un type, son caractère obligatoire ou non et d'éventuels autres balises propres au type. Les types disponibles pour les plugins de tâches sont les suivants :
  - **text** : chaine de caractères à renseigner via une zone de texte. La longueur max de la chaîne est précisée dans un attribut supplémentaire.
  - **multitext** : chaine de caractères à renseigner via une zone de texte multiligne. La longueur max de la chaîne est précisée dans un attribut supplémentaire.

    Les paramètres de ce type supportent les chaînes dynamiques selon les mêmes mots clés que le type *text*.

- **pass** : chaine de caractères à renseigner via une zone de texte offusquée. La longueur max de la chaîne est précisée dans un attribut supplémentaire.
- **list** : liste de choix à renseigner via une liste déroulante. Les options disponibles sont précisées dans un attribut supplémentaire (valeurs séparées par des |)
- **boolean** : booléen à renseigner via une case à cocher
- **email** : adresse mail ou liste d'adresses mail séparées par un ; et/ou une ,
- **list_msgs (pour nouveau plugin de validation) :** liste de choix multiple remplie par le core Extract avec les messages de validation définis au niveau de l'instance Extract
- **numeric (pour plugin FME Desktop)** : Valeur numérique. Min, max et step sont précisés dans 3 attributs supplémentaires.

- *Object new(params)* : crée une instance du plugin en passant les paramètres définis par l'utilisateur. Retourne l'objet créé.
- *Object execute(request)* : exécute le plugin de tâche selon pour l'élément de requête passé en paramètres. Retourne un objet du type :
  - *State* : état parmi *success, error, standby*
  - *Code :* le cas échéant, code de retour
  - *Message* : message associé à l'état (stocké dans *last_msg* pour info). Il peut s'agir d'un message brut d'un outil sous-jacent du plugin (e.g. FME), d'un message informatif expliquant pourquoi le plugin est en standby, …
  - *Request* : élément de requête passé en entrée et éventuellement modifié par le plugin

Les chapitres suivants listent le retour de ces fonctions pour chacun des plugins disponibles dans la première version de la solution. A noter que les plugins doivent implémenter le multilingue dans le retour des fonctions (sauf bien sûr pour les messages d'erreur/succès des outils sous-jacents). Les descriptions ci-dessous correspondent à la version française.

4.2.2 Plugin Archivage fichiers

Fonction	Retour
getCode	ARCHIVE
getLabel	Archivage fichiers
getDescription	Copie des fichiers dans un répertoire local ou réseau
getHelp	*Se référer directement au connecteur*
getPictoClass	*Se référer directement au connecteur*
getParams	[{'code' : 'path', 'label' : 'Chemin d'archivage', 'type' : 'text', 'req' : 'true', 'maxlength' : 255}, {'code' : 'login', 'label' : 'Login', 'type' : 'text', 'req' : 'false', 'maxlength' : 255}, {'code' : 'pass', 'label' : 'Mot de passe', 'type' : 'pass', 'req' : 'false', 'maxlength' : 255},] *Ci-dessous un exemple de paramétrage correspondant (attribut task_params de la table TASK) :* *{* *'path' : '/var/extraction/{no_commande}-{date}/{code-produit}/'* *}*

Note : Ce plugin supporte les chaînes dynamiques dans son paramétrage selon les mots clés suivants :

- {orderLabel}
- {orderGuid}
- {productGuid}
- {productLabel}
- {startDate}
- {organism}
- {client}

4.2.3 Plugin Extraction FME 2016

Fonction	Retour
getCode	FME2016
getLabel	Extraction FME 2016
getDescription	Workbench ou script FME Desktop 2016
getHelp	*Se référer directement au connecteur*
getPictoClass	*Se référer directement au connecteur*
getParams	[{'code' : 'path', 'label' : 'Chemin du fichier workbench', 'type' : 'text', 'req' : 'true', 'maxlength' : 255}, {'code' : 'pathFME', 'label' : 'Chemin du programme FME', 'type' : 'text', 'req' : 'true', 'maxlength' : 255}, {'code' : 'instances', 'label' : 'Nombre de fme.exe lancés par ce workspace', 'type' : 'numeric', 'req' : 'true', 'min' : 1, 'step' : 1, 'max : 8}] *Ci-dessous un exemple de paramétrage correspondant (attribut task_params de la table TASK) :* *{* *'path' : '/var/FME/exports_scripts/gaz_prod.fmw',* *'pathFME': '/usr/share/fme-destop-2016/fme',* *'instances': 3* *}*

Attention, pour le bon fonctionnement du plugin FME 2016, les scripts utilisés doivent exposer les paramètres suivants :

- **opt_responseformat** : paramètre FME, géré de façon interne par le plugin car celui-ci attend un retour en json (html par défaut)
- **Perimeter** : périmètre au format WKT, en WGS84, fourni par le connecteur d'import. Par exemple :

  POLYGON((6.295363612819898 46.32659377501397,6.288481575413728 46.65130021757865,7.055293317619941 46.656470518789625,7.057591178865101 46.33173342736276,6.295363612819898 46.32659377501397))

- **Parameters** : paramètres dynamiques fournis au format json par le connecteur d'import. Par exemple :

  {"FORMAT":"GDB", "SELECTION" : "PASS_THROUGH", "PROJECTION":"SWITZERLAND"}
- **Product** : GUID du produit à exporter, fourni par le connecteur d'import. Par exemple :

  a049fecb-30d9-9124-ed41-068b566a0855

4.2.4 Plugin Extraction FME Server

Fonction	Retour
getCode	FMESERVER
getLabel	Extraction FME Server
getDescription	Job FME Server API 1.2
getHelp	*Se référer directement au connecteur*
getPictoClass	*Se référer directement au connecteur*
getParams	[{'code' : 'url', 'label' : 'Url du service', 'type' : 'text', 'req' : 'true', 'maxlength' : 255}, {'code' : 'login', 'label' : 'Login distant', 'type' : 'text', 'req' : '**false**', 'maxlength' : 50}, {'code' : 'pass', 'label' : 'Mot de passe', 'type' : 'pass', 'req' : '**false**', 'maxlength' : 50}] *Ci-dessous un exemple de paramétrage correspondant (attribut task_params de la table TASK) :* *{* *'url' : 'http://localhost/FME/exports_scripts/gaz_prod'* *'login' : 'admin'* *'pass' : 'password01'* *}*

Attention, pour le bon fonctionnement du plugin FME Server, les scripts utilisés doivent exposer les paramètres suivants :

- **opt_responseformat** : paramètre FME, géré de façon interne par le plugin car celui-ci attend un retour en json (html par défaut)
- **Perimeter** : périmètre au format WKT, en WGS84, fourni par le connecteur d'import. Par exemple :

  POLYGON((6.295363612819898 46.32659377501397,6.288481575413728 46.65130021757865,7.055293317619941 46.656470518789625,7.057591178865101 46.33173342736276,6.295363612819898 46.32659377501397))
- **Parameters** : paramètres dynamiques fournis au format json par le connecteur d'import. Par exemple :

  {"FORMAT":"GDB", "SELECTION" : "PASS_THROUGH", "PROJECTION":"SWITZERLAND"}
- **Product** : GUID du produit à exporter, fourni par le connecteur d'import. Par exemple :

  a049fecb-30d9-9124-ed41-068b566a0855

4.2.5 Plugin Remarque fixe

Fonction	Retour
getCode	REMARK
getLabel	Remarque fixe
getDescription	Ajoute une remarque pré-définie à l'élément de requête
getHelp	*Se référer directement au connecteur*
getPictoClass	*Se référer directement au connecteur*
getParams	[{'code' : 'remark', 'label' : 'Remarque', 'type' : 'multitext', 'req' : 'true', 'maxlength' : 5000}, {'code' : 'overwrite', 'label' : 'Ecraser la remarque existante', 'type' : 'boolean'}] *Ci-dessous un exemple de paramétrage correspondant (attribut task_params de la table TASK) :* *{* *'remark' : 'Les positions exactes des conduits doivent être vérifiées par sondage.\nVoir nos conditions sur http://laboite.ch/exploitation',* *'overwrite' : true* *}*

4.2.6 Plugin Validation opérateur

Fonction	Retour
getCode	VALIDATION
getLabel	Validation opérateur
getDescription	Demande à un opérateur de valider l'élément
getHelp	*Se référer directement au connecteur*
getPictoClass	*Se référer directement au connecteur*
getParams	[{'code': 'valid_msgs', 'label': 'Modèles de remarque pour la validation', 'type' : 'list_msgs', 'req' : false}, {'code': 'reject_msgs', 'label': 'Modèles de remarque pour l'annulation', 'type' : 'list_msgs', 'req' : false},] *Ci-dessous un exemple de paramétrage correspondant (attribut task_params de la table TASK) :* *{* *'valid_msgs : '312,313,314',* *'reject_msgs' : '313,315'* *}* *Note : la validation doit être faite par un des opérateurs assignés au process. Il n'est donc pas nécessaire de spécifier un opérateur au niveau du plugin lui-même*

4.2.7 Plugin Annulation

Fonction	Retour
getCode	CANCELATION
getLabel	Annulation
getDescription	Annule le traitement et définit la remarque pour le client
getHelp	*Se référer directement au connecteur*
getPictoClass	*Se référer directement au connecteur*
getParams	[{'code' : 'reason', 'label' : 'Raison de l'annulation', 'type' : 'multitext', 'req' : 'true', 'maxlength' : 5000}] *Ci-dessous un exemple de paramétrage correspondant (attribut task_params de la table TASK) :* *{* *'remark' : 'Désolé, nous n'avons pas de donnée sur cette zone'* *}*

4.2.8 Plugin Notification mail

Fonction	Retour
getCode	NOTIFICATION
getLabel	Notification email
getDescription	Envoie une notification email personnalisable
getHelp	*Se référer directement au connecteur*
getPictoClass	*Se référer directement au connecteur*
getParams	[{'code' : 'emails', 'label' : 'Emails de destinataires, séparés par des points-virgules', 'type' : 'email', 'req' : 'true', 'maxlength' : 255}, {'code' : 'subject', 'label' : 'Objet du message', 'type' : 'text', 'req' : 'true', 'maxlength' : 255}, {'code' : 'body', 'label' : 'Contenu du message', 'type' : 'multitext', 'req' : 'true', 'maxlength' : 5000}]

Note : Ce plugin supporte les chaînes dynamiques dans son paramétrage selon les mots clés suivants :

- {orderLabel}
- {orderGuid}
- {productGuid}
- {productLabel}
- {startDate}
- {organism}
- {client}

4.2.9 Plugin QGIS Server - Atlas

Fonction	Retour
getCode	QGISSERVERATLAS
getLabel	Impression Atlas QGIS Server
getDescription	Imprime les pages d'un Atlas QGIS Server touchées par le périmètre, en PDF
getHelp	*Se référer directement au connecteur*
getPictoClass	*Se référer directement au connecteur*
getParams	[{'code' : 'url', 'label' : 'Url de base QGIS Server, 'type' : 'text', 'req' : 'true', 'maxlength' : 255}, {'code' : 'template', 'label' : 'Template à utiliser pour le layout', 'type' : 'text', 'req' : 'true', 'maxlength' : 255}, {'code' : 'path', 'label' : 'Chemin du projet QGIS', 'type' : 'text', 'req' : 'false', 'maxlength' : 255}, {'code' : 'login', 'label' : 'Login distant', 'type' : 'text', 'req' : 'false', 'maxlength' : 50}, {'code' : 'pass', 'label' : 'Mot de passe', 'type' : 'pass', 'req' : 'false', 'maxlength' : 50}, {'code' : 'layers', 'label' : 'Layers à utiliser (séparation par des virgules)', 'type' : 'text', 'req' : 'false', 'maxlength' : 255}, {'code' : 'crs', 'label' : 'CRS (Code EPSG)', 'type' : 'text', 'req' : 'false', 'maxlength' : 50}] *Ci-dessous un exemple de paramétrage correspondant (attribut task_params de la table TASK) :* *{* *'url' : 'https://monserveur.lan/qgis-server',* *'template' : 'myplan',* *'path' : '/etc/qgis-server/maps/world/world.qgs',* *'login' : 'admin',* *'pass' : 'password01',* *'layers' : 'places,airports,water',* *'crs' : 'EPSG:2056',* *}*

5 PRÉCISIONS TECHNIQUES

5.1 Gestion des règles

La solution met en œuvre un langage syntaxique visant à interpréter les commandes entrantes pour les associer à des traitements spécifiques. L'utilisateur peut ainsi définir des règles composées d'un ou plusieurs critères, séparé par un opérateur booléen :

critère1 ***and/or*** *critère2* ***and/or*** *critere3* ***and/or*** *… **and/or** critèreN*

Chaque critère suit la syntaxe suivante :

<*propriété*> <*opérateur*> <*valeur*>

Par exemple : *productguid == "708e932b-81c3-2ce4-b907-ed07e61ac5f9"*

5.1.1 Propriétés

Les propriétés utilisables sont listées dans le tableau ci-dessous. Pour chacune d'elle, il est indiqué le type d'opérateur possible (attributaire / géographique) et le type de valeur à renseigner (texte / numérique / géométrie).

Propriété	Champ BD	Description / Remarque	Type opérateur	Type valeur
orderlabel	p_orderlabel	p_orderlabel de l'élément de requête	Attributaire	Texte
orderguid	p_orderguid	p_orderguid de l'élément de requête	Attributaire	Texte
productguid	p_product_guid	p_productguid de l'élément de requête	Attributaire	Texte
productlabel	p_productlabel	p_productlabel de l'élément de requête	Attributaire	Texte
organism	p_organism	p_organism de l'élément de requête	Attributaire	Texte
client	p_client	p_client de l'élément de requête	Attributaire	Texte
tiers	p_tiers	p_tiers de l'élément de requête	Attributaire	Texte
tiersguid	p_tiersguid	p_tiersguid de l'élément de requête	Attributaire	Texte
perimeter	p_perimeter	p_perimeter de l'élément de requête, c'est-à-dire le polygone d'export au format WKT	Géographique	Géométrie
surface	p_surface	p_surface de l'élément de requête	Attributaire	Numérique
parameters.xxx	p_parameters.xxx	Propriété dynamique de l'élément de requête. Le xxx est à remplacer par la propriété voulue (e.g. *format*, *projection*, …) selon le connecteur d'import	Attributaire	Selon le type de la propriété dynamique : texte ou numérique

5.1.2 Opérateurs

Deux types d'opérateurs sont possibles : attributaire et géographique, selon la propriété utilisée.

Le tableau ci-dessous contient la liste des opérateurs attributaires utilisables.

Opérateurs att.	Description / Remarque
==	La propriété est égale à la valeur
!=	La propriété est différente de la valeur
>	La propriété est supérieure à la valeur
<	La propriété est inférieure à la valeur
>=	La propriété est supérieure ou égale à la valeur
<=	La propriété est inférieure ou égale à la valeur
IN	La propriété est présente dans une liste de valeurs (spécifiées entre parenthèses, séparées par des virgules)
NOT IN	La propriété n'est pas présente dans une liste de valeurs (spécifiées entre parenthèses, séparées par des virgules)

Le tableau ci-dessous contient la liste des opérateurs géométriques utilisables.

Opérateurs géo.	Description / Remarque
intersects	La géométrie contenue dans la propriété intersecte la géométrie passée en valeur
contains	La géométrie contient la géométrie passée en valeur
disjoint	La géométrie est totalement disjointe de la géométrie passée en valeur
equals	La géométrie est égale à la géométrie passée en valeur
within	La géométrie est incluse dans la géométrie passée en valeur

5.1.3 Valeurs

Selon la propriété et l'opérateur, les valeurs peuvent être de trois types.

Type valeur	Description	Exemple
Texte	Texte à indiquer entre guillemets	"PDF"
Numérique	Nombre (entier ou décimal)	1234
Géométrie	Géométrie passée au format WKT non entourés de guillemets	POLYGON((6.82 46.39,6.92 46.39,6.92 46.19,6.92 46.19,6.82 46.39)) LINESTRING(6.82 46.39, 6.92 46.39, 6.92 46.19) POINT(6.82 46.39)

5.1.4 Exemples

Quelques exemples types de règles :

- **orderlabel** == "92445" *AND* **perimeter** *intersects* POLYGON((6.82 46.39,6.92 46.39,6.92 46.19,6.92 46.19,6.82 46.39))
- **orderlabel** == "92445" *AND* **parameters.format** == "PDF"

5.2 Authentification

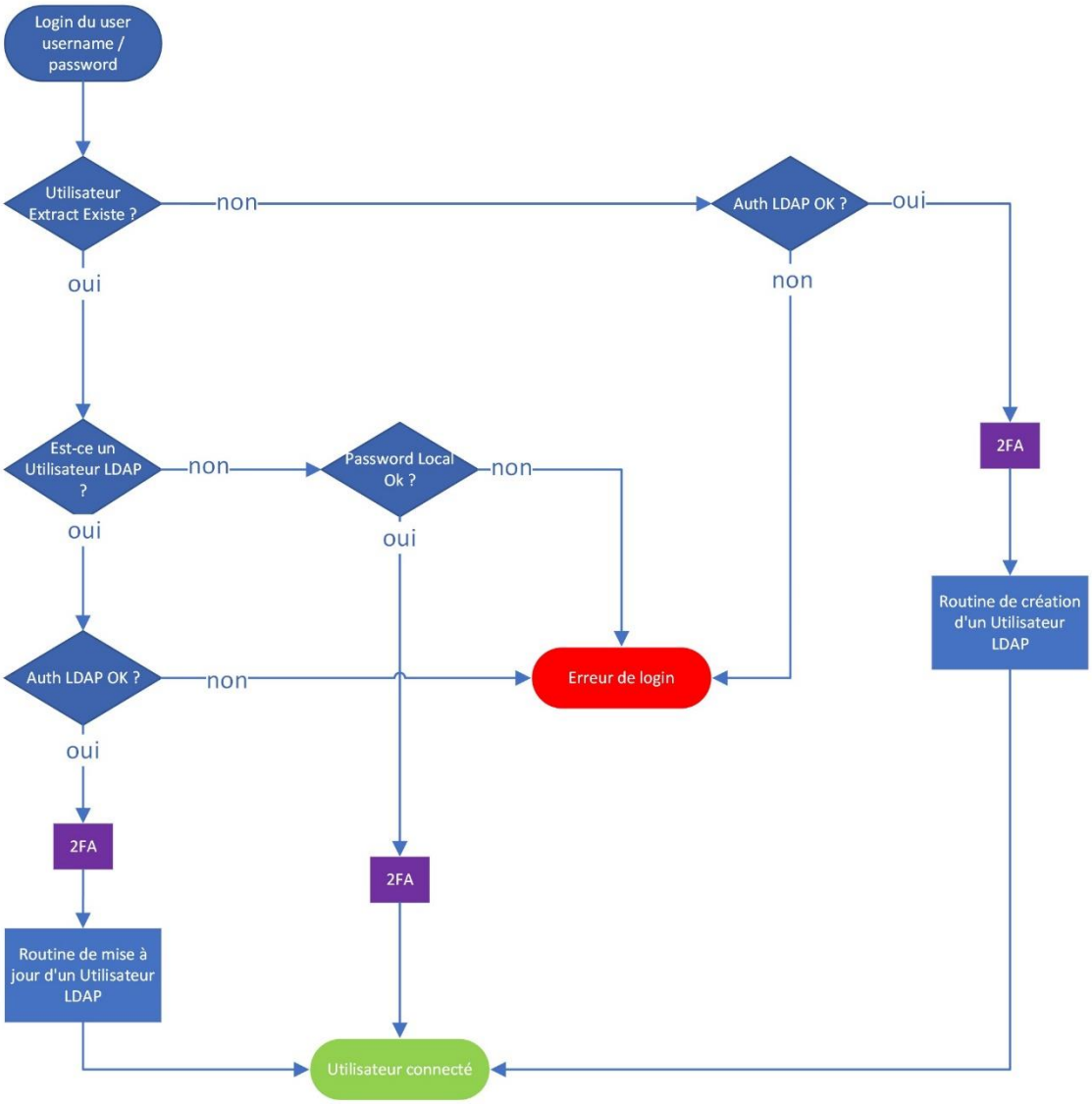
Un utilisateur peut être de deux types :

- LOCAL : utilisateur dont la base Extract est la source d'authentification
- LDAP : utilisateur dont un serveur LDAP est la source d'authentification

Pour pouvoir se connecter, un utilisateur LDAP doit faire parti d'un groupe Extract Admin ou Extract Operator dans l'annuaire LDAP.

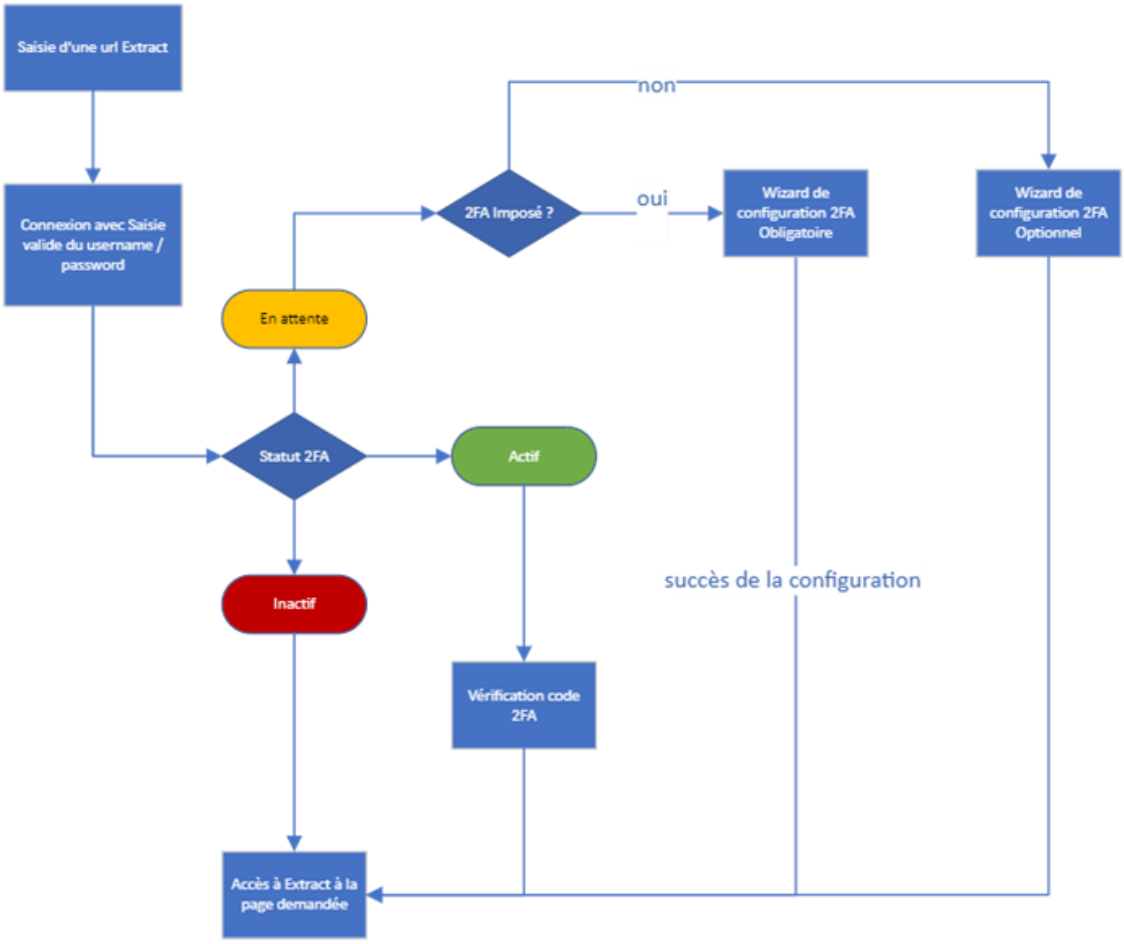
Si un utilisateur se connecte à Extract avec LDAP et qu'il n'est pas présent dans la base locale, il est créé à la volée.

A chaque connexion d'un utilisateur LDAP, ses informations sont mises à jour (nom, mail, rôle).



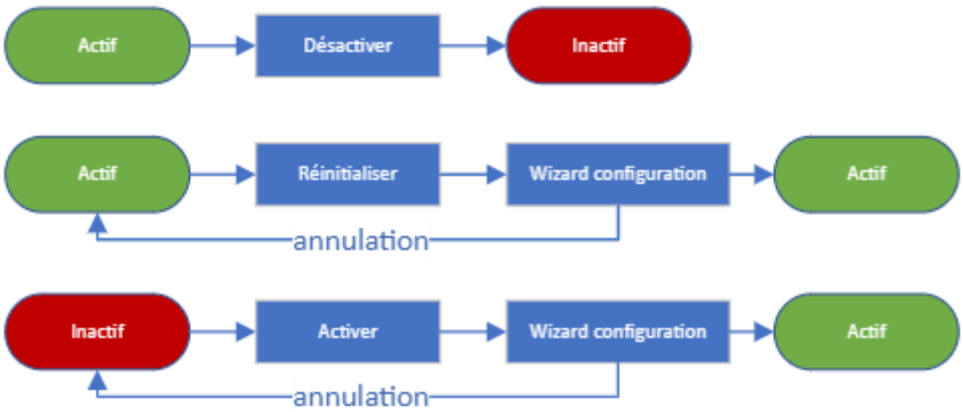
5.3 Fonctionnement 2FA Google Authenticator

5.3.1 Flux de connexion spécifique 2FA

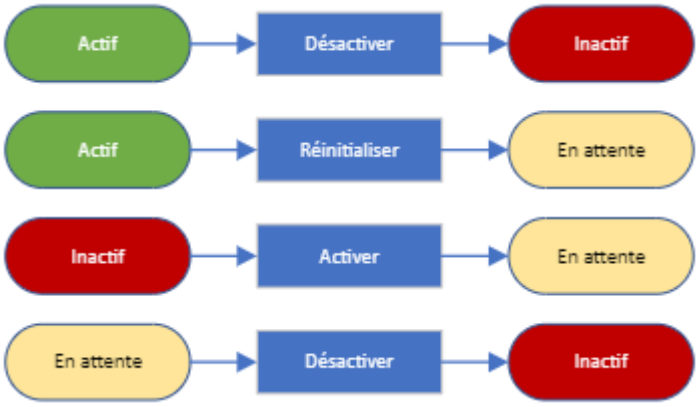


5.3.2 Modification des paramètres 2FA

Modification de son propre compte



Modification d'un utilisateur par un administrateur



5.4 Synchronisation LDAP

Il est possible de lancer une synchronisation LDAP (si l'authentification LDAP est activée). Cette synchronisation effectue les opérations suivantes :

- Création des Utilisateurs LDAP (comptes existants dans l'annuaire LDAP, pas encore dans la base Extract)
- Mise à jour des propriétés des Utilisateurs LDAP existants déjà dans la base Extract (nom complet, email, rôle)
- Désactivation des Utilisateurs LDAP de la base Extract si ils sont désactivés ou supprimés de l'annuaire LDAP