DIRECTION TERRITORIALE CENTRE-EST
DEPARTEMENT APPUI TERRITORIAL ET ANIMATION DU COLLABORATIF

FICHE PROCESSUS
DOC LOGICIEL / TACHES
DATAC

Votre interlocuteur :

le 02 janvier 2024

gabriel.bregand@ign.fr ; nicolas.py@ign.fr ; datac-dtce@ign.fr

Ref. : <aaaammjj-CodeRef>
\\del1509n016\dep_backup\datac_technique\03-evaluation-donnees-externes_modèlesfichesprocessus\aaaammjj-fichesprocessus-datac-majec.docx

Objet

OSMOSECRACKER

Diffusion

Destinataires	Copies
DTCE / DATAC	

Plan / Synthèse :

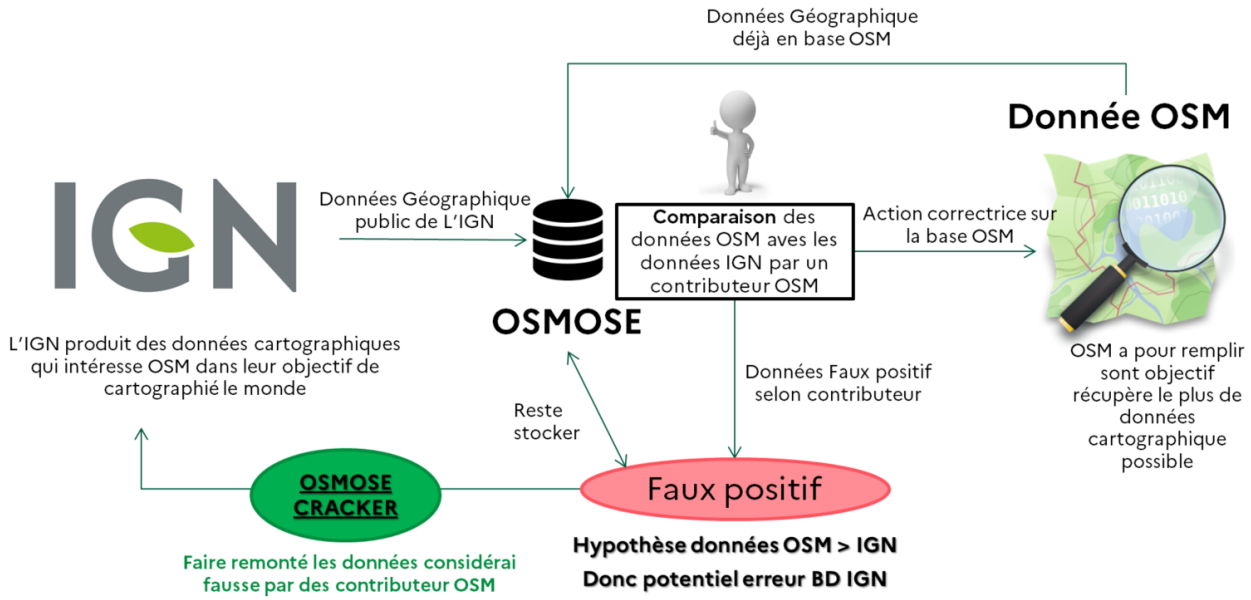
❑ OBJECTIF DU PROCESSUS

OSMOSE compare des données de référence (dont celles de l'IGN) à OSM (Open Street Map), créant ainsi des signalements pour les utilisateurs OSM lorsque des incohérences sont détectées.

OSM est un projet collaboratif mondial qui vise à créer une carte libre et éditable du monde. Les contributeurs d'OSM collectent et ajoutent des données géographiques, telles que les routes. Ces données sont mises à la disposition du public sous une licence quasiment libre, permettant à quiconque de les utiliser, de les modifier et de les partager.

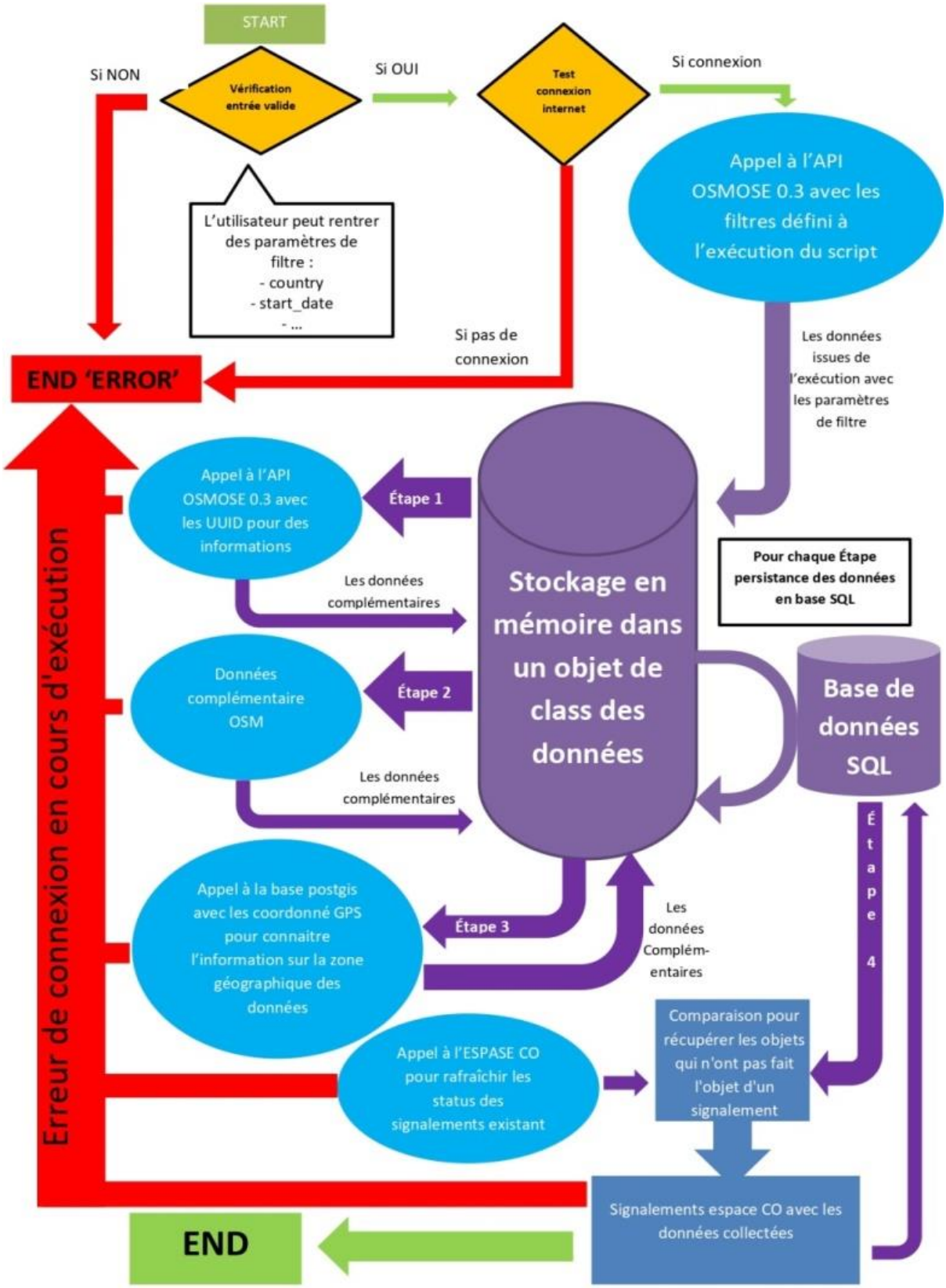
Osmose est un outil de validation de données géographiques créé par la communauté OpenStreetMap (OSM). L'objectif principal d'Osmose est de détecter automatiquement les erreurs potentielles ou les incohérences dans les données cartographiques d'OpenStreetMap grâce à une comparaison avec d'autres bases de données externes à OSM.

L'objectif principal du programme **OsmoseCracker** est la création de signalements IGN en cas de retour d'informations jugées mauvaises par un contributeur OSM par rapport à une donnée OSM. En théorie, cela impliquerait que la donnée OSM soit « supérieure » à la donnée de l'IGN, ce qui suggérerait un problème dans notre base de données.



❑ PRINCIPES DU PROCESSUS

Le programme OsmoseCracker est développé en Python et effectue une requête web (API) vers la base de données d'OSMOSE pour récupérer les données d'intérêt, qui sont ensuite stockées dans une base de données compagnon du programme OsmoseCracker au format SQLite. Ensuite, un ensemble de données complémentaires est ajouté, par exemple par interrogation de la BDuni. L'ensemble de ces données nous permet de créer un signalement clair à destination de la MAJEC.


START
Si NON
Vérification entrée valide
Si OUI
Test connexion internet
Si connexion
Appel à l'API OSMOSE 0.3 avec les filtres défini à l'exécution du script
L'utilisateur peut rentrer des paramètres de filtre :
- country
- start_date
- ...
Si pas de connexion
END 'ERROR'
Les données issues de l'exécution avec les paramètres de filtre
Appel à l'API OSMOSE 0.3 avec les UUID pour des informations
Étape 1
Les données complémentaires
Stockage en mémoire dans un objet de class des données
Pour chaque Étape persistance des données en base SQL
Données complémentaire OSM
Étape 2
Les données complémentaires
Base de données SQL
Appel à la base postgis avec les coordonné GPS pour connaitre l'information sur la zone géographique des données
Étape 3
Les données Complém-entaires
Étape 4
Comparaison pour récupérer les objets qui n'ont pas fait l'objet d'un signalement
Appel à l'ESPASE CO pour rafraîchir les status des signalements existant
Signalements espace CO avec les données collectées
Erreur de connexion en cours d'exécution
END

❑ INFRASTRUCTURE LOGICIEL DE REFERENCE ET GRANDES LIGNES DU DEVELOPPEMENT

Le programme OsmoseCracker est développé en Python. Les données collectées par celui-ci sont automatiquement stockées dans une base de données SQLite.

Bibliothèques python utilisées par OsmoseCracker :

Nom Bibliothèque	Utilité
__future__	Fournit des fonctionnalités Python futures dans les versions actuelles.
argparse	Permet de traiter les arguments de ligne de commande de manière plus poussée que sys.
dataclass_csv	Extension de dataclasses pour faciliter la sérialisation/désérialisation CSV.
dataclasses	Simplifie la création de classes de conteneurs de données.
datetime	Fournit des classes pour la manipulation de dates et d'heures.
json	Permet de travailler avec le format JSON pour la sérialisation/désérialisation des données.
logging	Gestion/Création des Log
math	Contient des fonctions mathématiques standard.
os	Facilite l'interaction avec le système d'exploitation pour récupérer des paramètres.
pathlib	Aide à manipuler les chemins de fichiers.
psycopg2	Adaptateur de base de données PostgreSQL pour Python.
psycopg2.extras	Contient des extensions utiles pour psycopg2. Permettant de renvoyer des objets de type dict.
sqlite3	Module d'accès à la base de données SQLite.
sys	Donne accès à certaines fonctionnalités spécifiques du système, comme les arguments de ligne de commande.
requests	Facilite les requêtes API.
requests.auth	Contient des méthodes d'authentification pour les requêtes API.
typing	Force certaine variable d'être d'un certain type.
urllib3	Fait des requêtes API (pour OsmoseCracker GET)
urllib.parse	Complémentaire à urllib3 pour les requêtes API
uuid	Génère et manipule des UUID.

Dossiers composant OsmoseCracker :

Doc	La documentation détaillée d'OsmoseCracker.
Log	Les log du programme.
Script	Le programme.

Le programme est lui composé :

Script		
Base de la structure du programme	__pycache__	Est le répertoire créé par l'interpréteur Python lors de l'importation d'un module.
	osmosecracker.py	Est le squelette principal du programme. Il effectue des tests de connexion aux serveurs d'OSMOSE et de BDuni, coordonne les différentes parties du

		programme, fait l'interface entre les appels serveur et la classe de stockage d'informations, puis agit en tant qu'intermédiaire entre cette classe et la base de données SQLite. Enfin, il gère l'écriture des données collectées en format CSV si cela est demandé dans les paramètres.
	osmosecracker_config.py	Il rassemble tous les paramètres du programme, incluant les définitions en français des objets, des classes d'erreurs initialement prévues, ainsi que les variables d'environnement.
	osmosecracker_exceptions.py	Simplifie la compréhension des erreurs en définissant des types d'erreurs spécifiques.
	osmosecracker_workflow.py	Mesure le temps écoulé entre chaque étape du programme, permettant de suivre son exécution et de stocker ces informations pour une analyse ultérieure.
Récolte de données Externe à l'IGN	osmosecracker_query_osm.py	Permet de faire les appels à l'API d'OSM pour avoir des données complémentaires sur l'objet OSMOSE (en développement).
	osmosecracker_query_osmose.py	Effectue des appels à l'API d'OSMOSE pour récupérer des objets OSMOSE, puis extrait des informations complémentaires à partir des UUID collectés.
Récolte de données Interne à l'IGN	osmosecracker_espacecollaboratifign.py	Émet des signalements sur l'espace collaboratif IGN en cas de création de nouveaux signalements par OsmoseCracker. Il interroge également la base de données de l'espace collaboratif pour mettre à jour le statut des signalements dans notre base de données SQLite.
	osmosecracker_query_bduni.py	Collecte des données de la BDUni en utilisant les coordonnées et la classe d'objet spécifiée. Les données recueillies comprennent le collecteur lié à la zone géographique, la reprojection des coordonnées, la commune, des informations sur l'objet, ainsi que la vérification de l'appartenance du point à une ZICAD afin d'éviter d'émettre un signalement si nécessaire.
Gestion du stockage en mémoire des données	osmosecracker_issue.py	Crée une classe abstraite python des objets Osmose instanciés selon la requête à l'API. Ces objets sont ensuite enrichis et peuvent être persisté en base si les données correspondent aux filtres renseignés au lancement du programme.
Gestion de la persistance des données en base SQLite	osmosecracker_database.sqlite	La base de données SQLite d'OsmoseCracker, comprenant 2 tables, une avec les informations sur les signalements (les données OSMOSE, BDuni, et autres) dans la table « osmoseissue ». la deuxième table contient les informations liées au lancement du programme dans la table « workflowerexecution ».
	osmosecracker_database_management.py	Gère le stockage des données dans la base de données SQLite. Il assure également la mise à jour des informations sur le statut des signalements dans l'espace collaboratif et identifie automatiquement les lacunes de données pour les compléter.

Fonctionnement :

Dans un premier temps, OsmoseCracker récupère les faux positifs OSM, ces derniers sont stockés dans la base de données Osmose nous pouvons interroger à l'aide d'une API nommée « API OSMOSE 0.3 ». Cette API permet le filtrage des données avant leur collecte. Nous appliquons donc des filtres pour sélectionner les faux positifs liés à une donnée IGN dans les régions souhaitées. Les données extraites sont, pour l'instant, uniquement de type tronçon de route. Le programme les

superpose aux données de la BDUNI pour identifier l'objet tronçon de route concerné. Ensuite, le programme récupère toutes les données liées à cet objet et les ordonne pour créer un signalement Espace CO.

Il est important de noter certaines informations utiles pour la compression du programme.

Dans la vérification des paramètres rentrés par l'utilisateur, initialement nous vérifions les numéros de département INSEE grâce à la table BDUnis « département » cependant St-Barthélémy (977) et St-Martin (978) ne sont pas dans la table département, car ils sont ne sont a pu utiliser la table collectivite_territoriale, mais avec le jumelage du 68 et du 67 il ne nous était pas possible de remplir certaines demandes nous avons donc gardé la table « département », à laquelle nous rajoutons manuellement le 977 et le 978.

Pour les requete BDUnis il est important de prendre en compte que OSM n'utilise pas la même projection que (OSM (WGS-84) ; l'IGN (Lambert-93)), ce qui signifie que nous devons effectuons une reprojection. Cela se fait grâce à une formule SQL contenue dans la fonction « bduni_get_reprojected_point » contenue dans le fichier « osmosecracker_query_bduni.py ».

Les données des faux positifs d'OSMOSE n'ont pas les mêmes informations entre une requête API basée sur l'item + class erreur qu'une requête basée sur l'UUID du faux positif, nous avons donc dû faire une fonction complémentaire nommée « extracte_osmose_uuid » contenue dans « osmosecracker_query_osmose.py »

4

Avant la création de signalement, il est important de vérifier que le faux positif ne se situe pas dans une ZICAL. Nous avons donc pris en compte la ZICAL dans la fonction « is_in_zicad » contenue dans le fichier « osmosecracker_query_bduni.py », rendant l'outil moins fiable.

De plus, OsmoseCracker suit le statut des signalements effectués à chaque lancement, il vérifie si un signalement créé par lui n'a pas été traité. Si tel est le cas, il actualise le statut du signalement dans sa base de données.

Toutes les données collectées sont donc stockées dans une base de données compagnon locale pour permettre un suivi d'OsmoseCracker.

De plus, le programme est très linéaire un objet de type class python « workflow » permet de suivre l'exécution du programme et documente les potentielles erreurs que le programme a pu rencontrer. Toutes les informations récoltées par l'objet workflow son reportés dans la table du fichier SQLite « workflowerexecution ». Cela facilite une reprise sur incident.

❑ PARAMETRES D'EXECUTION

OsmoseCracker peut se lancer avec une multitude de paramètres. Vous pouvez tous les retrouver dans la documentation utilisateur d'OsmoseCracker. Voici quelques paramètres à retenir pour l'utilisation d'OsmoseCracker :

- La plage temporelle sur laquelle nous voulons récupérer des informations OSMOSE :
  - Pour le commencement « -sd {YYYY-MM-DD} ». Par défaut, la date d'aujourd'hui - 31 jours.
  - Pour la date de fin « -ed {YYYY-MM-DD} ». Par défaut, la date d'aujourd'hui.

- Si des signalements de l'espace CO sont émis ou non et de quel type « -ts {…} » ["skip", "test", "submit"]. Par défaut, «"skip"» pour ne pas émettre de signalements sans le faire exprès, "test" crée lui des signalements tests et "submit" crée lui des vrais signalements.

- Le code INSEE des départements (au sens service de l'État) pour lesquels les signalements Osmose doivent être téléchargés « - fdep {…} » il faut rentrer un tableau de str, il n'y a pas de paramètres par défaut . Exemple : -fdep "68" "74" "977"

❑ MISE EN PRODUCTION

Pour la mise en production d'OsmoseCracker nous avons utilisé l'outil AG.IRE (outil interne développé par Jean-Francois Burillier) qui permet le lancement automatique de notre programme pour cela nous avons mis les fichier permettant l'exécution du programme dans les fichier d'AG.IRE (Fichier), puis dans AG.IRE dans l'onglet « Requête & Scripts » nous avons créé une requête qui vérifie l'import des bibliothèques python nommé « Verif-Import-OsmoseCracker ». Cela permet d'être sûr que toutes les bibliothèques permettant le bon fonctionnement d'OsmoseCracker sont dans AG.IRE. Pour finir, dans « Config d'enchaînement de requêtes », nous avons créé un scénario qui exécute le programme avec un certain nombre de paramètres. Le lancement du programme OsmoseCracker est exécuté avec les paramètres suivants :

- La création de nouveaux signalements (-ts submit)
- Et la liste des départements mis en production (-freg "84" "27")

Ceux qui donnent : J:\DEP_ECHANGE\DATAC2023_OsmoseCracker\Script\osmosecracker.py -ts submit -freg "84" "27"

Le programme se lance automatiquement, cela se paramètre dans les « Batchs AG.IRE ». Nous avons paramétré le batch pour qu'il se lance tous les jours à 22h22 et grâce au paramètre « Adresse Mail » chaque fin d'exécution programme est suivie par l'envoi de mails qui informent du bon déroulement ou non du programme, en PJ de ce mail nous retrouvons les Log du programme qui permet de connaître le nombre de signalements créés par exemple.

❑ ANALYSE DES DONNEES

Pour analyser les données, vous devez utiliser la base SQLite qu'AG.IRE aliment tous les jours ou une copie disponible ici.
Toutes les informations sur la structure des tables « osmoseissue » et « workflowerexecution » sont disponible dans la documentation utilisateur dans l'annexe « structure des tables »

❑ ARCHIVE LOGICIEL

\\del1509n016\NASShare2\DEP_ECHANGE\DATAC2023_OsmoseCracker

Veuillez prendre en compte la dernière archive, à savoir au jour de la rédaction de cette documentation (le 18/01/2024) l'archive OSMcraker_Backup2024017.7z (OSMcraker_Backup{aaaammjj}.7z)

❑ DESCRIPTION DES TACHES A EFFECTUER PAR LE DATAC

Le programme OSMOSECRACKER est automatisé, donc aucune intervention du DATAC n'est nécessaire. Les interventions possibles pourraient survenir lors de discussions avec OSM/OSMOSE concernant d'autres données IGN collectées qui font l'objet de signalements, ou en cas de modification de leur API.

Ces interventions ne représentent que des mises à jour mineures, car le programme est facile à modifier et l'ajout d'un nouveau type d'objet serait simple.

❑ AUTRE …

Des mises à jour sont envisagées, telles que la collecte de données OSM complémentaires aux données OSMOSE déjà collectées lorsqu'un signalement Osmose identifie un objet OSM.

Pour plus d'information consulter la documentation utilisateur d'OsmoseCracker.