ISCRI

Documentation technique

Cyril Vincent Conseil

Table des matières

[1 Introduction 4](#_Toc208954908)

[2 Stack technique 4](#_Toc208954909)

[3 GDELT 5](#_Toc208954910)

[4 Serveur Azure 5](#_Toc208954911)

[4.1 Description du serveur 5](#_Toc208954912)

[4.2 Connexion au serveur 6](#_Toc208954913)

[5 Github 9](#_Toc208954914)

[6 La base de données 9](#_Toc208954915)

[6.1 Connexion à la base en local 9](#_Toc208954916)

[6.2 Connexion à la base à distance via pgadmin 9](#_Toc208954917)

[6.3 MCD 11](#_Toc208954918)

[7 Application Python 13](#_Toc208954919)

[7.1 Architecture générale 13](#_Toc208954920)

[7.2 L’environnement virtuel 15](#_Toc208954921)

[7.3 Les fichiers de configuration 15](#_Toc208954922)

[7.4 Les tests unitaires 15](#_Toc208954923)

[7.5 SqlAlchemy 15](#_Toc208954924)

[7.6 Le workflow de scraping 16](#_Toc208954925)

[7.7 Le workflow de parsing 16](#_Toc208954926)

[8 Les métriques 17](#_Toc208954927)

[8.1 Les acteurs 17](#_Toc208954928)

[8.2 DailyRisk 17](#_Toc208954929)

[8.3 Risk 18](#_Toc208954930)

[8.4 ISCRI 18](#_Toc208954931)

[8.5 Le workflow de calcul 18](#_Toc208954932)

[9 Risk Service 19](#_Toc208954933)

[10 Jupyter 20](#_Toc208954934)

[11 CRONtab 20](#_Toc208954935)

[11.1 Les scripts 20](#_Toc208954936)

[11.2 CRON 20](#_Toc208954937)

[11.3 Logs 20](#_Toc208954938)

[12 Gestion des erreurs 21](#_Toc208954939)

[12.1 RiskService 21](#_Toc208954940)

[12.2 EventParser 21](#_Toc208954941)

[12.3 Scrapper 21](#_Toc208954942)

[13 Exécution « à la main » 22](#_Toc208954943)

[14 Documentation embarquée 22](#_Toc208954944)

[15 Bug lié à la panne de Juin et Juillet 2026 22](#_Toc208954945)

[16 Réponses au mail du 15/09/25 23](#_Toc208954946)

# Introduction

ISCRI est un projet développé par Cyril Vincent d’octobre 2024 à décembre 2024. La maitrise d’ouvrage est Rodolphe Desbordes.

Il s’agit d’importer des données GDELT vers une base de données

# Stack technique

Voici la stack technique :

* Base de données : PostgreSql 17
* Langage de programmation : Python 12
* Scripting : Bash
* OS : Linux Ubuntu

Voici l’architecture technique générale qui sera expliqué par la suit

Une image contenant texte, capture d’écran, diagramme, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.

# GDELT

Nos données se trouvent sur un serveur GDELT : [www.gdelt.com](http://www.gdelt.com)

L’explication fonctionnelle des données est dans les documents suivants :

* CAMEO.Manual.1.1b3.pdf
* GDELT-Data\_Format\_Codebook.pdf
* GDELT-Global\_Knowledge\_Graph\_Codebook.pdf

Les données se trouvent à l’URL suivante : <http://data.gdeltproject.org/events/index.html>

Elles sont mises à jour tous les jours.

Elles contiennent le fichier CSV à télécharger et sa signature MD5.

Il n’a a pas de services pour obtenir la liste des fichiers, un scraping est obligatoire.

Le site n’est pas sécurisé par SSL

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Il y a environ 4500 fichiers pour un poids de 0.4 To

# Serveur Azure

Le serveur Azure utilise l’OS Ubuntu et est géré par Skema.

Seul un accès SSH est disponible avec une clé \*.pem ou \*.ppk

Cette clé est à demander à l’IT Skema

## Description du serveur

Voici la description du serveur fournie par l’IT Skema :

* Standard D4ds v5 = 4 CPU / 16 RAM / 1T DISK
* Compte local = azureuseriscri
* Port ouvert = SSH / HTTPS / HTTP / TCP 8840-8849
* IP Publique = 51.144.115.12
* L'url publique = iscri.skema.edu

## Connexion au serveur

La manière la plus simple de se connecter au serveur est d’utiliser Putty

Voici la configuration utilisée :

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Le contenu généré par l’IA peut être incorrect.

Mettre la clé \*.ppk ici

Une image contenant texte, capture d’écran, affichage, Police

Le contenu généré par l’IA peut être incorrect.

Pour transformer une clé \*.pem en \*.ppk voici un tutoriel : <https://puttygen.com/convert-pem-to-ppk>

On obtiens alors :

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

L’utilisation du logiciel WinSCP est un plus mais n’est pas obligatoire : <https://winscp.net/eng/download.php>

Voici les paramètres pour WinSCP :

Une image contenant texte, capture d’écran, Police, nombre

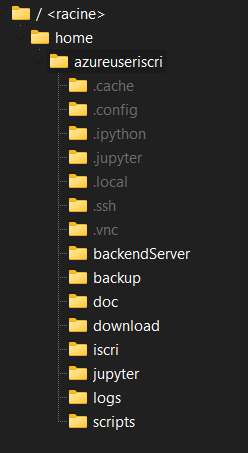
Le contenu généré par l’IA peut être incorrect.

Puis dans avancé

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Le contenu généré par l’IA peut être incorrect.

Voici l’arborescence du projet :



Description des répertoires :

* Backup : LE backup de la bd
* Doc : contient toutes les documentations
* Download : les fichier CSV téléchargés
* Iscri : Le projet
* Jupyter : Le POC jupyter
* Logs : les logs
* Scripts : les scripts bash utilisé par la CRONtab

# Github

Le projet est disponible sur Github : <https://github.com/cyrilvincent/iscri>

Cette documentation est dans le répertoire doc

Seul la clé SSH n’est pas disponible dans le github

# La base de données

La base de données est PostgreSql 17

Elle a été installée à partir de cette documentation : <https://documentation.ubuntu.com/server/how-to/databases/install-postgresql/>

Le login est : postgres

Le mot de passe est : sa

Nom de la base : iscri

Le mot de passe n’est pas sécurisé car la bd n’est pas accessible en dehors du serveur hormis en passant par SSH qui possède un certificat sécurisé.

## Connexion à la base en local

Pour se connecter en local utiliser la commande : psql --host 127.0.0.1 --username postgres --password --dbname iscri

Pour restaurer une base : pg\_restore -d iscri iscri.backup -U postgres -h 127.0.0.1 -c

Pour redémarrer la base : sudo systemctl restart postgresql.service

## Connexion à la base à distance via pgadmin

Voici les paramètres pour accéder à la base via pgadmin :

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Puis

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Puis après la connexion le mot de passe sa

Une image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.

Nous obtenons ensuite la base de données et ses 6 tables

Une image contenant texte, Police, capture d’écran, nombre

Le contenu généré par l’IA peut être incorrect.

## MCD

Voici le MCD de la base

Une image contenant texte, capture d’écran, nombre, Parallèle

Le contenu généré par l’IA peut être incorrect.

Explications des tables :

* Table file : liste des fichiers CSV et leur workflow de traitement dans la base pour pouvoir reprendre en cas de panne. Un file possède plusieurs event et un event possède 1 seul file
* Table event : liste des events GDELT, 1 ligne = 1 ligne d'un CSV
* Table url : 1 url peut être liée à 1 seul event, séparation de la table url car elle est très volumineuse, n'était pas chargé au départ. Les vieux event n'ont pas d'URL
* Table daily\_risk : métriques journalières, c'est de la redondance mais cette table est nécessaire car les calculs sont très lents. Généré à partir de la table event
* Table iscri : métriques mensuelles dont l'iscri, calculé à partir de daily\_risk. Il n'y a pas de FK entre iscri et daily\_risk car c'est une relation many\_to\_many et ça aurait complexifié le traitement pour rien
* Table country : liste de tous les pays et autres (acteurs)

La table event est très volumineuse, elle représente 95% du volume total.

Voici une étude de volumétrie et de son contenu au 23/10/2024 :

Bd = 345Go

Nombre d’event : 816 879 081

event.is\_root\_event is true : 512 051 501 (63%)

event.quad\_class = 3 : 104 220 502 (20% of 63%)

event.quad\_class = 4 : 116 555 662 (23% of 63%)

event.actor1\_type = ‘GOV’ or event.actor2\_type = ‘GOV’ = 67 976 587 (13% of 63%)

Puis avec un code is not null = 17 132 444 (3.3% of 63%)

Q4+GOV+Code = 1 623 738 (8.3% of 3.3%)

Q3+GOV+Code = 1 850 132 (10.8% of 3.3%)

Q3+Q4 = 19.1%

## Conception

La base de données a été conçues avec les bonnes pratiques suivantes :

* Présence de Primary Key
* Présence de Foreign Key
* Présence d’index

Conformément aux bonnes pratiques de relations entre les tables, quand une relation n’existe pas elle n’est pas créée.

Par exemple pour une date donnée il n’y a pas de RISK entre FRA et ZBW alors la relation n’existe pas.

Fonctionnellement cela correspond à une valeur à zéro.

Il est également nécessaire de ne pas mettre les relations avec une valeur à zéro pour optimiser la taille de la base qui augmenterai de 500 Go.

Pour exporter les données avec les valeurs à zéro, une jointure externe est alors nécessaire :

with d as (

select year, month from iscri

group by year, month

)

select coalesce(iscri.id,0) as id, d.year as year, d.month as month, actor1.iso3 as actor1\_code, actor2.iso3 as actor2\_code, coalesce(iscri.iscri3,0) as iscri3, coalesce(iscri.iscri4,0) as iscri4, coalesce(iscri.iscri,0) as iscri

from country as actor1

join country as actor2 on true

join d on true --d.year=2025 --and d.month=6

left outer join iscri on iscri.actor1\_code=actor1.iso3

and iscri.actor2\_code=actor2.iso3

and iscri.year=d.year and iscri.month=d.month

# Application Python

L’application est programmée en Python 3.12 et est disponible dans le répertoire iscri.

Le code Python est en orienté objet.

## Architecture générale

Voici l’architecture générale

Une image contenant texte, capture d’écran, diagramme, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.

Scrapper.py va vérifier sur GDELT si un nouveau fichier est disponible, puis va le télécharger, vérifier sa signature et le dézipper.

Ensuite event\_parser.py va parser le fichier CSV dans la table event.

Puis risk\_service.py va effectuer les calculs journaliers dans la table daily et les calculs mensuels dans la table iscri.

Voici le diagramme d’activité du projet

Une image contenant texte, capture d’écran, affichage, ligne

Le contenu généré par l’IA peut être incorrect.

Il y a 3 workflows :

* Scrapper va télécharger les fichiers
* Event\_parser va parser les fichiers
* Risk\_service va effectuer les calculs

Ceci est résumé dans ce diagramme technique

Une image contenant texte, capture d’écran, diagramme, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.

## L’environnement virtuel

Les programmes python sont dans le répertoire ~/iscri.

Le projet utilise un environnement virtuel Python

L’environnement virtuel est situé dans ~/iscri/.venv

Pour activer l’environnement virtuel il faut taper la commande : source .venv/bin/activate

## Les fichiers de configuration

Le fichier config.py contient la configuration générale de l’application

Le fichier requirements.txt contient la liste des packages à installer via la commande pip install -f requirements.txt

## Les tests unitaires

De nombreux tests unitaires sont disponibles dans tests.py.

Il est conseillé d’exécuter les tests unitaires à chaque mise en production.

Le fichier test.py affiche les versions des principaux packages

## SqlAlchemy

Les relations entre Python et la BD sont effectuées via un l’ORM SqlAlchemy.

Le paramétrage de SqlAlchemy est effectué dans db\_context.py.

La méthode create\_engine est une méthode lente et peut avoir un UnitOfWork long.

La méthode create\_session va créer la session SqlAlchemy et devrait avoir un UnitOfWork court.

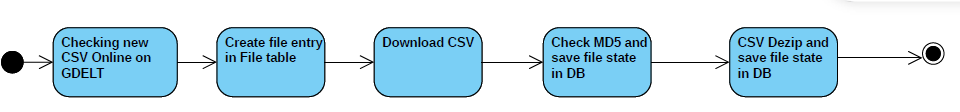
Le mapping objet – relationnel est effectué dans sqlentities.py

## Le workflow de scraping

Ce workflow va télécharger les fichiers CSV sur le serveur dans le répertoire data

Le code est dans scrapper.py

Diagramme d’activité du scraping



La méthode scrap va effectuer les tâches suivantes :

* Tester que gdelt.com est actif
* Elle va regarder la liste des fichiers qui ne sont pas présent dans la table file de la bd
* Si un nouveau fichier est présent en ligne, une ligne est ajoutée dans la table file avec les colonnes name, date, online\_date et md5 renseignée
* La méthode download va télécharger le fichier
* La méthode check\_md5 va vérifier la signature du fichier, si la signature est bonne la colonne download\_date est renseignée
* Le méthode dezip va dézipper le fichier et renseigner la colonne dezip\_date
* Le workflow est terminé

## Le workflow de parsing

Ce workflow va parser les fichiers CSV dans la bd via le programme event\_parser.py

Diagramme d’activité du parsing

Une image contenant texte, ligne, Police, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Le programme a été largement testé unitairement.

A chaque colonne du CSV correspond une colonne de la table event

La méthode load de la classe de base BaseParser dans base\_parser.py va effectuer les tâches suivantes :

* Requêter dans la bd la table file avec download\_date != null et import\_end\_date == null
* Elle va charger le ficher et mapper chaque ligne via la méthode map
* Si le fichier est entièrement chargé la colonne import\_end\_date est renseigné, le workflow est alors terminé
* Si un parsing s’interrompt en cours de chargement import\_start\_date != null et import\_end\_date == null, le chargement pourra reprendre

Le programme event\_parser\_batch.py va appeler event\_parser pour tous les events non parsés

# Les métriques

Plusieurs métriques sont calculées par l’application.

Les calculs des risques et des iscri sont expliqué en détail dans le fichier Presentation\_IFRI\_SKEMA\_v2.pdf.

Le détail du calcul sera néanmoins rappelé dans ce document.

Les calculs des risques étant très lents, de nombreux pré-calculs sont effectués.

## Les acteurs

Un acteur est un pays ou une entité.

Chaque event possède 2 acteurs : actor1 et actor2

C’est une paire d’acteur

Chaque acteur à un code à 3 chiffres dans actor1\_code et actor2\_code et un type dans actor1\_type\_code et actor2\_type\_code.

Voici un exemple :

* Actor1\_code = FRA
* Actor2\_code = CAN
* Actor1\_type\_code = GOV
* Signifie un event du gouvernement Français vers le Canada

Seuls les événements dont un des acteurs est de type GOV sont utilisés

## DailyRisk

Il existe 3 risques journaliers :

* DailyRisk3 : les risques pour event.quad\_class == 3
* DailyRisk4 : les risques pour event.quad\_class == 4
* DailyRisk = DailyRisk3 + DailyRisk4

Pour une paire d’acteurs gouvernementaux et pour une journée, le DailyRisk3 se calcul comme suit :

Une image contenant Police, Graphique, texte, symbole

Le contenu généré par l’IA peut être incorrect.

Cstdy est le nombre d’event avec un event.quad\_class == 3

Estdy est le nombre d’event total

Le tout divisé par le nombre de jour du mois

Idem pour Dailyrisk4

## Risk

Pour une paire d’acteurs données et un mois donné, Risk est la somme des DailyRisk.

Cela revient à dire que Risk est la moyenne des Cstdy / Estdy

Voici la définition de Risk :

Une image contenant Police, symbole, texte, ligne

Le contenu généré par l’IA peut être incorrect.

## ISCRI

Voici la définition de l’ISCRI

Une image contenant texte, Police, blanc, ligne

Le contenu généré par l’IA peut être incorrect.

Pour une paire d’acteurs données et un mois m donné ISCRIm = Riskm + 0.9 \* ISCRIm-1

Si Riskm n’existe pas alors Riskm = 0

Si ISCRIm n’existe pas alors ISCRIm = 0

## Le workflow de calcul

Les calculs des risques étant très lents, de nombreux calculs sont effectués dans ce workflow.

Ce workflow va effectuer les calculs dans risk\_service.py

Diagramme d’activité du calcul

Une image contenant texte, ligne

Le contenu généré par l’IA peut être incorrect.

Le programme a été largement testé unitairement.

Le programme va effectuer les tâches suivantes :

* Il va requêter la bd sur les file.import\_end\_date != null pour connaitre tous les fichiers présents dans la table event
* Va calculer les daily\_risk pour chaque journée et les stocker dans la table daily
* Va calculer les risk pour chaque mois et chaque paire d’acteurs dans la table monthly
* Va calculer les iscri pour chaque mois et chaque paire d’acteurs dans la table monthly

# Risk Service

Le programme risk\_service va calculer toutes les métriques vues précédemment.

Il possède les arguments suivants :

* Python risk\_service.py --daily : calcul tous les daily risks
* Python risk\_service.py --monthly : calcul tous les monthly risks
* Python risk\_service.py –iscri : calcul tous les ISCRI non encore calculés

Risk\_service est docuenté en docstring : <https://github.com/cyrilvincent/iscri/blob/main/risk_service.py>

Fonctionnement de la classe RiskService :

* Is\_all\_files\_prensets\_by\_year\_month permet de érifier si un mois est complet, c’est cette méthode qui gère la variable not\_in\_html qui gère les mois où des fichiers sont absents
* Compute\_daily : calcul tous les RISKd d’un jour donné et sauvegarde le tout dans une seule transaction, il est donc impossible d’avoir une journée partiellement calculée
* Compute\_dailies : Appel compute\_daily d’une date de départ (par défaut le 1er janvier de l’année précédente) à une date d’arrivée (par défaut le jour même)
* Compute\_monthly : Calcul tous les RISKm d’un mois donné, uniquement si le mois est complet. Sauvegarde le tout dans une seule transaction, il est donc impossible d’avoir unemois partiellement calculé.
* Compute\_monthlies : Appel compute\_monthly d’une date de départ (par défaut le mois de janvier de l’année précédente) à une date d’arrivée (par défaut le mois en cours)
* Compute\_iscri : Calcul un ISCRIm par rapport à ISCRIm-1 et RISKm
* Compute\_iscri\_monthly : Calcul tous les ISCRI d’un mois donné, uniquement si le mois est complet
* Compute\_iscri\_monthlies : Appel compute\_iscri\_monthly d’une date de départ (par défaut le mois de janvier de l’année précédente) à une date d’arrivée (par défaut le mois en cours)
* Les autress méthodes sont des utilitaires

Workflow de RiskService appelé dans le CRON avec les 3 options –dailly –monthly --iscri

* Compute\_dailies => Compute\_daily
* Puis Compute\_monthlies => Compute\_monthly
* Puis Compute\_iscri\_monthlies => Compute\_iscri\_monthly

# Jupyter

Un POC de dataviz a été fait en Jupyter

Le code est dans le jupyter\_service.py et les fichiers \*.ipynb

N’est plus utilisé

# CRONtab

## Les scripts

Les scripts bash qui automatisent les CRON sont stockés dans ~/scripts

Daily.sh représente toutes les actions à effectuer tous les jours

Ce script lance risk\_service.py avec différentes options

Monthly.sh représente toutes les actions à effectuer tous les mois

Ce script lance le backup de la base de données

## CRON

L’appel des scripts bash est automatisé par une CRONTab

Les lanceurs de programmes sont scriptés en bash dans le répertoire scripts.

Les scripts exécutés journalièrement sont dans daily.sh

Les scripts exécutés mensuellement sont dans monthly.sh

Le lancement des scripts est automatisé dans la crontab

Pour lister la crontab, utiliser crontab -l

Pour modifier la crontab, utiliser crontab e

Une image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.

Daily.sh est exécuté tous les jours à 8h

Monthly.sh est exécuté tous les 4 du mois à 1h

## Logs

L’intégralité des logs python sont mis dans les fichiers logs du répertoire ~/logs.

En cas de bug il est utile de vérifier les traces des scripts dans daily.log et monthly.log

Daily.log contient tous les logs liés à risk\_service.py

Les anciens logs sont mis dans les fichiers \*.log.1, \*.log.2, \*.log.3, …

# Gestion des erreurs

Différents messages sont affichés et logués par les différents programmes.

Lors des exécutions des scripts une gestion des erreurs est effectuée.

Certaines sont bloquantes.

Ces messages d’erreurs sont affiches dans les logs vus précédemment.

## RiskService

Lors du calcul des RISKd, le message suivant apparait : Compute daily risk 2025-09-16.

Un jour ne peut être calculé qu’une seule fois, si la date n’apparait pas cela veut dire que le RISKd a déjà été calculé. Comme le calcul d’une journée est dans une seule transaction il est impossible d’avoir une journée partiellement calculée.

Lors du calcul des RISKm le message suivant apprait : Compute monthly risks

Si un mois est complet, le message suivant apparait : Compute risk month 2025-09.

Si un mois a déjà été calculé le message suivant apparait : Risks is already computed for month

2025-09.

Si un mois est incomplet, le message suivant apparait : Month 2025-09 is not complete.

Lors des calculs des ISCRIs le message suivant apprait : Compute iscris

Si un ISCRIm est calculable, c’est-à-dire que les ISCRIm-1 existent, le message suivant apparait : Compute iscri month 2025-09.

Si un ISCRIm est déjà calculé le message suivant apprait : scri is already computed for 2025-09

Si un mois n’est pas complet les ISCRIm et ISCRIm+i ne peuvent plus être calculés et le message suivant apprait : Stop at 2025-09 et le script s’arrête avec le code 0.

## EventParser

Si le fichier CSV n’est pas conforme, le message suivant apparait : ERROR Event row 1234 suivi de l’exception Python et de la ligne en erreur, le programme s’arrête avec le code 1.

## Scrapper

Si le site GDELT est UP, le message suivant apparait : Open <http://data.gdeltproject.org/events/index.html> OK

Sinon le message suivant apparait : WARNING URL Error: avec l’exception Python et le programme s’arrête avec le code 1.

Si le download du fichier est en échec le message suivant apparait : WARNING download Error.

Si la signature di fichier n’est pas conforme le message suivant apparait : WARNING bas md5. Le fichier n’est alors pas considéré comme downloader.

Si un fichier n’apparait pas dans le HTML de GDELT alors que sa date est passée, alors le message suivant apparait : 20250916.export.CSV.zip not in HTML.

# Exécution « à la main »

Il est possible d’exécuter les différents scripts en ligne de commande sans passer par le CRON.

Les programmes python sont dans le répertoire ~/iscri.

Le projet utilise un environnement virtuel Python

L’environnement virtuel est situé dans ~/iscri/.venv

Pour activer l’environnement virtuel il faut taper la commande : source .venv/bin/activate

Ensuite il est possible d’exécuter les programmes python manuellement :

* Python scrapper.py : pour scraper les fichiers comme indiqué plus haut
* Python event\_parser.py data/20250916.export.CSV : parse le fichier event dans la base de données
* Python event\_parser.batch.py : upload tous les fichiers event qui n’ont pas encore été importés dans la base de données
* Python risk\_service.py --daily : calcul tous les daily risks
* Python risk\_service.py --monthly : calcul tous les monthly risks
* Python risk\_service.py –iscri : calcul tous les ISCRI non encore calculés

# Documentation embarquée

Toutes les classes et méthodes pythons sont documentées en docstring :

* <https://github.com/cyrilvincent/iscri/blob/main/risk_service.py>
* <https://github.com/cyrilvincent/iscri/blob/main/scrapper.py>
* <https://github.com/cyrilvincent/iscri/blob/main/event_parser.py>
* <https://github.com/cyrilvincent/iscri/blob/main/event_parser.py>
* <https://github.com/cyrilvincent/iscri/blob/main/sqlentities.py>

# Bug lié à la panne de Juin et Juillet 2026

Le site GDELT a été en panne en Juin et Juillet 2026.

Grâce à la méthode RiskService. is\_all\_files\_presents\_by\_year\_month les ISCRI et les RISKm n’ont pas été calculés en Juin et Juillet 2026.

En revanche les RISKd ont été calculés uniquement les jours où les fichiers étaient présents.

Le 4 Septembre 2025, RiskService. is\_all\_files\_presents\_by\_year\_month a renvoyé True pour le mois 2025-08 ce qui était effectivement le cas, le RISKm d’Aout a été calculé. Comme les RISKm et les ISCRI de Juin et Juillet n’ont pas été calculés, ils sont considérés comme égale à zéro. Il n’y a pas de bug du côté des scripts Python, le bug provient surement du display qui a remplacé les valeurs manquantes par zéro.

Pour corriger le bug :

* Dans RiskService.compute\_iscri\_monthlies le code Une image contenant texte, Police, capture d’écran, ligne

  Le contenu généré par l’IA peut être incorrect. permet de bloquer le calcul de l’iscri
* Dans RiskService. is\_all\_files\_presents\_by\_year\_month la gestion des mois de Juin et Juillet a été ajouté manuellement Une image contenant texte, Police, capture d’écran, ligne

  Le contenu généré par l’IA peut être incorrect.
* J’ai effacé les risks de Juin et Juillet : delete from iscri where year=2025 and month>=6
* J’ai relancé les calculs mensuels : python risk\_service.py --monthly –iscri

# Réponses au mail du 15/09/25

À documenter plus en détail:

• Modules / services:

Fonction de chaque composant : Voir la docstring

Entrées / sorties attendues : Cf chapitre 13 et la docstring

Cas d’erreurs possibles et traitements associés : cf chapitre 12

Dépendances, paramètres ajustables : Il n’y a aucune dépendance Python entre scrapper, event\_parser et risk\_service, seul leur ordre d’exécution import, ce qui est expliqué dans le workflow : scraper puis event\_parser puis risk\_service

• Tâches planifiées / Cron:

Scripts déclenchés, fréquence d’exécution : Cf chapitre 11

Comportement attendu en cas d’échec : Le code a été amélioré dans ce sens, voir les cas d’erreurs

Procédures de contrôle et de relance : Entièrement automatisé dans le workflow, sauf pour not\_in\_html qui est fait manuellement.

• Calculs / Base de données

Calcul et ingestion de Daily, Monthly et ISCRI dans la base de données : Pour les calculs voire le chapitre 8 et le nouveau chapitre 9, pour l’ingestion voire le chapitre 7