arcOpole PRO Foncier
Impact PLU

Guide de l'administrateur

Version 3.3

19/05/2026

info@1spatial.com www.1spatial.com

A propos du Guide

Les informations contenues dans ce document pourront faire l'objet de modifications sans préavis de la part de 1Spatial.

Sauf mention contraire, les sociétés, les noms et les données utilisés dans les exemples sont fictifs.

Limitation de garantie et de responsabilité

1Spatial a réalisé tous les tests nécessaires et suffisants du Progiciel et a vérifié la conformité de son guide.

Le client reconnaît que dans l'état actuel de la technologie informatique, le fonctionnement du Progiciel est susceptible d'être interrompu ou affecté par des bogues. 1Spatial ne garantit pas que le Progiciel, le média d'installation, la clef ou la documentation livrés soient exempts d'erreurs, de bogues ou d'imperfections.

Ainsi, le client doit effectuer toutes les sauvegardes, prévoir et effectuer toutes les procédures de remplacement en vue d'éventuelles défaillances du Progiciel, prendre toutes les mesures appropriées pour se prémunir contre toute conséquence dommageable due à l'utilisation ou la non utilisation du Progiciel.

Il est expressément convenu que 1Spatial ne sera en aucun cas responsable des dommages directs ou indirects dus à l'utilisation du Progiciel.

L'utilisation du Progiciel est soumise à la signature par le client du contrat de droit d'utilisation des logiciels de 1Spatial.

Marques déposées

Ce progiciel est une marque déposée de 1Spatial.

Ce progiciel, développé par 1Spatial, est une marque déposée et est la propriété exclusive de 1Spatial. Toutes les autres marques citées appartiennent à leurs propriétaires respectifs.

TABLE DES MATIÈRES

1 PRÉAMBULE

La fonctionnalité Impact PLU permet d'obtenir automatiquement les éléments du PLU (zonages, prescriptions, servitudes, …) impactant une parcelle ou un dossier ADS, à partir de croisements géographiques préconfigurés.

L'Impact PLU permet de :

- Connaître immédiatement le zonage PLU d'une parcelle ;
- Identifier les prescriptions, informations et servitudes qui s'y appliquent ;
- Consulter les documents et règlements associés ;
- Utiliser des couches complémentaires (risques, données locales, …) si l'administrateur les a configurées ;
- Afficher clairement ces informations dans les widgets « Consultation Cadastre » « Consultation Cadastre Avancée » et « Consultation ADS ».

Ce document a pour but de détailler la procédure de paramétrage pour le calcul d'impact PLU d'une parcelle et d'un dossier ADS.

Ce guide s'adresse aux administrateurs SIG ou arcOpole PRO et décrit en détail :

- La logique d'implantation de l'Impact PLU ;
- Le paramétrage des fichiers d'impact ;
- Les règles à respecter et les erreurs courantes ;
- La gestion des couches et des requêtes SQL/vues associées aux impacts.

Cette documentation concerne arcOpole PRO Foncier 3.3.

2 PRINCIPES

La fonction Impact PLU permet de déterminer, pour une parcelle ou un dossier ADS, l'ensemble des informations réglementaires issues du PLU ou de couches complémentaires, grâce à des croisements géographiques paramétrables.

Elle répond à deux besoins majeurs :

- Identifier automatiquement les éléments du PLU impactant un objet ;
- Gérer plusieurs configurations d'impact, indépendantes les unes des autres (par exemple : PLU, servitudes, risques, …).

Le calcul de l'impact PLU concerne principalement la fiche d'information parcelle, la fiche d'information dossier, le renseignement d'urbanisme et la fonction « ImpactPLU » de l'API ADS qui délivre le résultat sous la forme d'un fichier XML.

Il est possible d'associer un impact différent à chacun des éléments indiqués ci-dessus.

2.1 ARCHITECTURE DES FICHIERS

Chaque impact correspond à un fichier distinct :

- Fichiers nommés obligatoirement : impact_xxx.json ;
- Tous les fichiers sont lus au démarrage du service « arcOpole PRO Serveur » ;
- Les erreurs sont ignorées automatiquement mais visibles dans les logs ;
- Les couches définies sont validées une par une lors du démarrage.

Chaque Impact comporte un identifiant (id) et un nom (name) uniques.

Les impacts sont définis dans le répertoire : **..\1Spatial\aOpServeur\config\foncier\impact**

2.2 STRUCTURE D'UN FICHIER IMPACT

Un fichier Impact est composé des éléments suivants :

```
{
    "id":1,
    "name":"Cadastre Impact",
    "layers": [ ]
    "minimumPercentagePLU":1.0
}
```

2.2.1 Propriétés principales

Propriété	Description
id	Identifiant unique de l'impact (entier).
name	Nom associé à l'impact.

Propriété	Description
	Utilisé pour le paramétrage des Widgets « Consultation Cadastre », « Consultation Cadastre Avancée » et « Consultation ADS ».
layers	Liste des couches associées à l'impact.
minimumPercentagePLU	Tolérance d'inclusion (%) utilisée pour le calcul de l'impact. S'applique à toutes les couches déclarées dans l'impact. En deçà de cette tolérance, les couches ne sont pas prises en compte. Propriété optionnelle avec une valeur par défaut égale à 1.0.

2.2.2 Définition des couches

Les couches peuvent provenir de différents schémas en base de données et n'ont pas besoin d'être présentes dans la carte Web exploitée par l'expérience pour être identifiées.

Chaque couche associée à l'impact comporte un ensemble de propriétés.

Propriété	Description
source	► standard : couches PLU CNIG issues du modèle de données FONCIER ; ► table : couche complémentaire ; ► view : couche complémentaire.
category	► Catégorie associée à la couche ; ► Une couche appartient à une seule catégorie . Cette propriété est obligatoire et la valeur doit être unique..
group	Libellé de groupement des couches à l'affichage dans la fiche d'informations d'une parcelle et d'un dossier et dans un Renseignement d'Urbanisme.
dimension	Géométrie de la couche : ► 0 (point) ; ► 1 (ligne) ; ► 2 (polygone). Propriété nécessaire uniquement pour les couches dont la source est de type « table » ou « view ». Propriété optionnelle avec une valeur par défaut égale à 2.

Propriété	Description
warning	Affichage de la couche accompagnée d'un pictogramme d'alerte dans la fiche d'information d'une parcelle et d'un dossier et dans un Renseignement d'Urbanisme lorsque la tolérance d'inclusion est inférieure à la valeur déclarée. ► True ; ► false. Propriété optionnelle avec une valeur par défaut égale à false.
minimumPercentage	Tolérance d'inclusion (%) dédiée à la couche. Propriété optionnelle. Si elle n'est pas déclarée ou renseignée, c'est la tolérance globale (**minimumPercentagePLU**) qui est prise en compte.
fieldmap	Pour chaque couche, plusieurs propriétés sont mises à disposition et exploitées au niveau : ► De la fiche d'information d'une parcelle et d'un dossier ; ► Du Renseignement d'Urbanisme ; ► De la fonction « ImpactPLU » de l'API ADS. Ces propriétés sont déjà définies par l'application pour les couches standards. Leur définition concerne uniquement les couches dont la source est de type « table » ou « view ». Ces propriétés sont utilisées de la manière suivante dans l'application : ► Fiche d'information parcelle et fiche d'information dossier : Propriété affichée : « **libellé** » ► Fiche d'informations parcelle PDF et RU : Propriétés affichées « **type** » et « **description** » ► Fiche d'informations dossier PDF : Propriétés affichées : « **type** » et « **description** » ► API ADS : Toutes les propriétés sont affichées. Détail des propriétés attendues : ► Propriété «**id**» : champ contenant l'identifiant unique des objets (obligatoire) ; ► Propriété «**libelle**» : champ contenant le nom court de l'objet ;

Propriété	Description
	► Propriété «**type**» : champ contenant le type de l'objet avec la nomenclature CNIG ; ► Propriété «**sous type**» : champ contenant le sous-type de l'objet avec la nomenclature CNIG ; ► Propriété «**commentaire**» : champs contenant le nom long de l'objet ; ► Propriété «**etiquette**» : champs contenant le nom détaillée l'objet ; ► Propriété « **description** » : champ contenant la description de l'objet ; ► Propriété «**reglement**» : champ contenant l'url du règlement associé à l'objet.
complements	Définition de propriétés complémentaires. Il est possible de définir des propriétés complémentaires pour toutes les couches quelque soit le type de source. Ces propriétés sont exploitées par : ► La fiche d'informations avancée d'une parcelle ; ► La fonction « ImpactPLU » de l'API ADS.
sqlquery	Requête SQL de récupération et de calcul des champs affectés à chaque propriété. Nécessaire uniquement pour les couches dont la source est de type **« table »**.
tablename	Vue associée à la couche. Nécessaire uniquement pour les couches dont la source est de type **« view »**.

2.3 NIVEAUX DE PARAMÉTRAGE

Deux niveaux de paramétrage sont disponibles :

► Paramétrage standard : s'applique uniquement sur les couches du modèle FONCIER ;

► Paramétrage avancé : s'applique aux couches en dehors du modèle FONCIER (couches complémentaires) ou à une couche du modèle FONCIER pour un traitement particulier.

3 PARAMÉTRAGES

3.1 DÉFINITION

La fonction « Impact PLU » prend en charge les couches « standard » et les couches « complémentaires ».

Une couche « standard » est une couche issue directement du modèle FONCIER du PLU. Elle correspond aux classes officielles du modèle CNIG (ZU, SCC, AP, AL, AS, IL, IP, IS, ISC, ILC, IPC, PS, PL, PP).

Une couche « complémentaire » est une couche personnalisée que l'on souhaite intégrer en plus des couches standards pour le calcul d'impact PLU. Elle correspond à une couche provenant d'une table ou d'une vue dans la base de données.

3.2 COUCHE STANDARD

Une couche « standard » correspond à une couche PLU du modèle FONCIER, reconnue automatiquement et qui ne nécessite pas de configuration SQL.

Pour ce type de couches, l'application propose un paramétrage par défaut.

Caractéristiques d'une couche « standard » :

- La propriété **source** est égale à « standard » ;
- La propriété **category** doit être l'une des catégories PLU officielles ;
- Elle n'utilise pas les propriétés :
  - **sqlquery,**
  - **tablename,**
  - **fieldmap,**
  - **dimension.**
- Elle repose entièrement sur les couches PLU fournies par le modèle FONCIER.

Les propriétés **source**, **category** et **group** sont obligatoires.

Les propriétés **warning**, **minimumPercentage** et **complements** sont facultatives.

Cependant, il est possible de configurer une couche PLU comme une couche complémentaire dans le cas où la correspondance des champs ne donne pas satisfaction.

Exemple de paramétrage sur la couche des zonages et les couches des prescriptions :

```
{
    "source": "standard",
    "category": "ZU",
    "group":"Zonages"
},
```

```
{
    "source": "standard",
    "category": "PS",
    "group":"Prescriptions",
    "warning": true,
    "minimumPercentage": 1.0
},
{
    "source": "standard",
    "category": "PL",
    "group":"Prescriptions",
    "warning": true,
    "minimumPercentage": 1.0
},
{
    "source": "standard",
    "category": "PP",
    "group":"Prescriptions",
    "warning": true,
    "minimumPercentage": 1.0
},
```

3.2.1 Mapping

Le tableau ci-dessous présente le mappage standard entre chaque propriété par défaut et le fichier XML résultat de la demande d'impact PLU côté API ADS avec le modèle de données FONCIER.

Propriété	Balise	Zonage	Prescriptions	Information	Secteur	Assiette
id	ID	ID_ZONE	ID_PRESS ID_PRESL ID_PRESP	ID_INFOS ID_INFOL ID_INFOP	ID_SECT	ID_OBJ
label	TYPE	ZONAGES	PRESCRIPTIONS	INFORMATIONS	SECTEURS	ASSIETTES
libelle	NOM	LIBELLE	Concaténation des champs LIBELLE et TXT. Le champ TXT est mis entre parenthèse.	Concaténation des champs LIBELLE et TXT. Le champ TXT est mis entre parenthèse.	LIBELLE	1ere sous chaîne du champ NOMASS divisé par « _ ».
commentaire	LIBELLE	LIBELONG	LABEL	LABEL	Label du sous type TYPESECT dans la table des nomenclatures	Concaténation des champs TYPESUP et TYPEASS séparés par « : ».
reglement	REGLEMENT	NOMFIC	NOMFIC	NOMFIC	NOMFIC	-
type	CODE	TYPEZONE	TYPEPSC	TYPEINF	TYPESECT	TYPESUP
soustype	SOUS_CODE	-	S TYPEPSC	STYPEINF	-	TYPEASS
etiquette	ETIQUETTE	LIBELLE	TXT	TXT	LIBELLE	TYPESUP
description	DESCRIPTION	LIBELLE	LIBELLE	LIBELLE	LIBELONG	TYPEASS

4 COUCHE COMPLÉMENTAIRE

Une couche « complémentaire » est une couche personnalisée provenant d'une table ou d'une vue, nécessitant une configuration SQL ou une vue dédiée.

Caractéristiques d'une couche « complémentaire » :

- La propriété **source** est égale à « table » ou « view » ;
- Des propriétés supplémentaires doivent être définies :
  - **sqlquery** dans le cas d'une source de type « table »,
  - **tablename** dans le cas d'une source de type « view »,
  - éventuellement **fieldmap**,
  - **dimension** (0 = point, 1 = ligne, 2 = surface – valeur par défaut = 2).

La propriété **fieldmap** peut être occultée dans le cas où la requête SQL utilisée définit les alias correspondants aux éléments déclarés dans **fieldmap**.

Les propriétés **source**, **group**, **dimension** et **sqlquery** ou **tablename** sont obligatoires.

Les propriétés **category**, **warning**, **minimumPercentage** et **complements** sont facultatives.

4.1 SOURCE « TABLE »

Une source **« table »** est définie avec une **requête SQL complète** fournie directement dans la configuration.

- Elle nécessite une propriété **sqlquery** ;
- La requête doit inclure :
  - La géométrie sous le nom **shape**,
  - Tous les champs nécessaires pour définir la propriété **fieldmap**,
- La commande **DISTINCT** est strictement interdite.

Avantages :
✓ Configuration centralisée, ✓ Pas besoin d'intervention DBA.
Inconvénients :
✗ Requêtes longues ou complexes dans les fichiers JSON de déclaration d'un impact, ✗ Moins réutilisable.

4.1.1 Exemples de paramétrage

Paramétrage impact avec la propriété fieldmap

```
{
        "source": "table",
```

```
        "category": "ZI_FAIBLE",
        "group": "Zones inondables",
        "dimension": 2,
        "fieldmap": {
                "id": "objectid",
                "commentaire": "cours_deau",
                "libelle": "scenario",
                "type": "typ_inond",
                "soustype": "typ_inond",
                "reglement":"",
                "description": "cours_deau",
                "etiquette": "scenario"
        },
        "sqlquery":"select objectid, scenario, cours_deau, typ_inond, id_s_inond, shape
                from ${pluSchema}.zi_rfaible"
}
```

! **sqlquery** : la variable ${pluSchema}. est définie dans le fichier **foncier.properties**. il est également possible d'utiliser une couche stockée dans un autre schéma. Dans ce cas, le nom de la couche doit être préfixé par le nom du schéma (tmp.zi_rfaible) et l'utilisateur de base de données déclaré dans le fichier doit avoir le droit d'accès à la couche concernée.

! **Reglement** : l'absence de valeur permet de ne pas activer de lien dans la fiche d'informations d'une parcelle ou d'un dossier.

Paramétrage impact sans la propriété fieldmap

```
{
        "source": "table",
        "group": "Zones inondables",
        "dimension": 2,
        "sqlquery":"select objectid as id, cours_deau as commentaire, scenario as libelle, typ_inond as type, typ_inond as soustype, null as reglement, cours_deau as description, scenario as etiquette, shape
                from ${pluSchema}.zi_rfaible"
}
```

! **sqlquery** : ajout d'un alias sur tous les champs déclarés comme éléments présents dans **fieldmap**,

! valeur nulle permet de ne pas activer de lien dans la fiche d'informations d'une parcelle ou d'un dossier.

4.2 Source « View »

Une source « view » fait référence à une vue SQL déjà existante dans la base.

- Elle utilise la propriété **tablename** ;
- La vue doit être créée en amont dans le SGBD ;
- La géométrie doit être présente sous le nom **shape** ;
- Les champs complémentaires doivent être définis dans la vue (alias obligatoires si nécessaires) ;
- La commande **DISTINCT** est strictement interdite dans la définition de la vue.

Avantages :
✓ Requête SQL propre et maintenable,
✓ Réutilisable par plusieurs impacts ou applications.
Inconvénients :
✗ Demande une intervention en base,
✗ Nécessite des droits de création de vues.

4.2.1 Exemples de paramétrage

Création de la vue

```
CREATE VIEW tpm.view_zi_rfort AS
SELECT  zi_rfort.objectid AS id,
        zi_rfort.typ_inond AS type,
        zi_rfort.typ_inond AS soustype,
        zi_rfort.scenario AS libelle,
        NULL::text AS reglement,
        zi_rfort.cours_deau AS commentaire,
        zi_rfort.cours_deau AS description,
        zi_rfort.scenario AS etiquette,
        zi_rfort.shape
FROM tpm.zi_rfort;
```

Paramétrage impact

```
{
        "source": "view",
        "category": "ZI_FORT",
        "group": "Zones inondables",
        "dimension": 2,
        "tablename": "${pluSchema}.view_zi_rfort"
}
```

4.3 DIFFÉRENCES « TABLE » / « VIEW »

Aspect	**Source « table »**	**Source « view »**
Définition	Directement via une requête SQL dans le fichier impact.	Repose sur une vue SQL existante.
Propriété utilisée	sqlquery	tablename
Gestion SQL	Dans le fichier impact.	Dans la base de données.
Champs complémentaires	Dans la requête.	Dans la vue.
Géométrie shape	Doit être présente dans la requête.	Doit être présente dans la vue.
DISTINCT	Interdit.	Interdit.

5 CHAMPS COMPLÉMENTAIRE

Un champ complémentaire est un attribut supplémentaire que l'on associe à une couche déclarée dans l'impact pour enrichir l'information renvoyée lors du calcul.

Ces champs permettent d'ajouter :

- Des valeurs textuelles ;
- Des valeurs calculées ;
- Des attributs issus d'une table ou d'une vue ;
- Ou même des attributs encodés en JSON (uniquement pour les couches standard).

Il existe deux catégories de champs complémentaires :

- Issus d'une source table/vue ;
- Issu d'un champ JSON (couches standard).

Les champs complémentaires sont définis par la propriété **complements** qui contient la liste des champs complémentaires associés à la couche.

Propriété	Description
source	► table : attribut supplémentaire dans la table ; ► json : attribut complémentaire CNIG – modèle FONCIER (champ JSON « ATTCOMP »).
name	Alias de l'expression associée au champ dans la requête SQL contenue dans la propriété **sqlquery**.
label	Nom du champ affiché dans la fiche d'information d'une parcelle ou d'un dossier.
type	Type du champ : ► Date ; ► String ; ► Double ; ► Integer.
expression	► table : nom du champ supplémentaire ou expression SQL calculée associée au champ ; ► json : nom de l'attribut issu du champ JSON. Le préfixe « LIB_ », s'il existe, doit être enlevé dans cette déclaration.

! **name** : Balise associée au champ complémentaire dans le fichier XML produit par la fonction « ImpactPLU » de l'API ADS.

5.1 SOURCE « STANDARD »

Paramétrage champ complémentaire

```
"complements":[
        {
                "source": "table",
                "name": "DATE_VALIDATION",
                "label": "Date de validation",
                "type": "Date",
                "expression": "datvalid"
        }
]
```

Le champ complémentaire est un champ de la couche PLU.

Ce cas ne nécessite pas de requête SQL.

Limite :

- ► Impossible d'ajouter des champs venant d'autres tables (sauf si l'on bascule en mode table/view) ;
- ► Pas de SQL.

Paramétrage champ JSON

```
"complements":[
        {
                "source": "json",
                "name": "DESTINATION",
                "label": "Destination",
                "type": "String",
                "expression": "DESTDOMI"
        }
]
```

Le système va lire dans le champ ATTCOMP de la coche PLU un objet JSON contenant les champs complémentaires.

5.2 SOURCE « TABLE »

Le champ complémentaire doit être directement ajouté dans le requête SQL contenue dans la propriété **sqlquery**.

Définition du champ complémentaire avec alias

<expression> AS <nom_du_champ>

Paramétrage champ complémentaire

```
"sqlquery":"select objectid as id, …, datentree AS DATE_MAJ, shape from ${pluSchema}.zi_rmoyen",
"complements":[
        {
                "source": "table",
                "name": "DATE_MAJ",
                "label": "Date de mise à jour",
                "type": "Date",
                "expression": " datentree "
        }
]
```

5.3 SOURCE « VIEW »

Le champ complémentaire doit être défini dans la vue.

Création de la vue

```
CREATE VIEW tpm.view_zi_rfort AS
SELECT  zi_rfort.objectid AS id,
        …,
        zi_rfort.datentree,,
        zi_rfort.shape
FROM tpm.zi_rfort;
```

Paramétrage champ complémentaire

```
"sqlquery":"select objectid as id, …, datentree AS DATE_MAJ, shape from ${pluSchema}.zi_rmoyen",
"complements":[
        {
                "source": "table",
                "name": "DATE_MAJ",
                "label": "Date de mise à jour",
                "type": "Date",
                "expression": " datentree "
        }
]
```

5.4 RÉCAPITULATIF

Source	Où définir les champs complémentaires ?	Contraintes
standard	Dans propriété **complements** ou via JSON (attcomp - modèle FONCIER)	Pas de SQL.
table	Dans la **SQL** (obligatoire), + descriptif dans la propriété **complements**.	shape obligatoire, DISTINCT interdit.
view	Dans la **SQL** (obligatoire), + descriptif dans la propriété **complements**.	shape obligatoire, DISTINCT interdit.

6 GUIDE DE RECETTE

Ce guide a pour but de vérifier :

- La bonne prise en compte des *fichiers d'impact* (impact_xxx.json) ;
- Le fonctionnement des couches standard et complémentaires (table/view) ;
- La conformité du calcul : croisement spatial, propriétés renvoyées, tolérances ;
- La bonne restitution dans :
  - Fiche parcelle,
  - Fiche dossier ADS,
  - Renseignement d'urbanisme (PDF),
  - API ADS (fonction ImpactPLU),
- Les règles de paramétrage : **fieldmap**, **sqlquery**, **tablename**, **complements**, etc.

6.1 PRÉ REQUIS AVANT RECETTE

6.1.1 Vérifier l'infrastructure

- arcOpole PRO Serveur configuré et redémarré après modification ;
- Accès au répertoire :..\1Spatial\aOpServeur\config\foncier\impact\ () ;
- Accès BD + droits sur table/view PLU et couches complémentaires ;
- Schémas PLU (${pluSchema}) correctement renseignés dans le fichier foncier.properties.

6.1.2 Données attendues

- PLU conforme au modèle CNIG / FONCIER ;
- Données complémentaires (table ou vue) avec :
  - Un champ géométrique **shape**,
  - Absence de **DISTINCT** dans les requêtes ().

6.2 PLAN DE TESTS – STRUCTURE DU FICHIER IMPACT

6.2.1 Objectif

S'assurer que chaque fichier **impact_xxx.json** est valide.

6.2.2 Test 1 – Présence des propriétés obligatoires

6.2.2.1 Cas d'usage

Charger un fichier **impact_PLU.json**.

6.2.2.2 Attendus

- **id** et **name** présents et uniques ;
- Propriété **layers** contenant une liste ;
- Tolérance globale optionnelle : **minimumPercentagePLU**.

6.2.2.3 Critères de succès

- Le service démarre sans erreur dans les logs ;
- Les couches sont listées dans les widgets.

6.3 PLAN DE TESTS – COUCHE STANDARD

6.3.1 Objectif

Valider la détection automatique des couches CNIG.

6.3.2 Test 2 – Détection automatique modèles PLU

6.3.2.1 Cas d'usage

Configurer un impact standard PLU avec zonage + prescriptions.

6.3.2.2 Attendus

L'application lit correctement :

- TYPEZONE ;
- LIBELLE ;
- NOMFIC ;
- etc. (mapping fourni dans le guide,).

Le résultat apparaît dans :

- Fiche parcelle ;
- Fiche ADS ;
- RU (PDF).

6.3.2.3 Critères de succès

- Tous les champs mappés apparaissent correctement ;
- Aucun SQL n'est nécessaire.

6.3.3 Test 3 – Tolérance d'inclusion

6.3.3.1 Cas d'usage

- Définir minimumPercentage = 30% sur une prescription ;

- Tester une parcelle intersectée à 25% puis 35%.

6.3.3.2 Attendus

- 25 % → la couche n'apparaît pas ;
- 35 % → la couche apparaît ;
- Si warning=true : pictogramme ⚠ en cas d'inclusion < valeur attendue ;
- **Plan de tests – Couche *complémentaire* (source = table).**

6.4 PLAN DE TESTS – COUCHE COMPLÉMENTAIRE (SOURCE = TABLE)

6.4.1 Objectif

Vérifier la prise en compte des couches définies par SQL.

6.4.2 Test 4 – Lecture SQL avec fieldmap

6.4.2.1 Cas d'usage

Utiliser un paramétrage comme celui-ci (exemple du guide) :

- Champs explicités dans **fieldmap** ;
- SQL simple sans alias ().

6.4.2.2 Attendus

- Tous les champs mappés (id, libelle, type, etc.) sont lus ;
- Les géométries sont croisées correctement ;
- Le résultat s'affiche dans les fiches.

6.4.2.3 Cas d'erreurs possibles

- Oubli du champ shape ;
- Utilisation de DISTINCT : doit échouer.

6.4.3 Test 5 – Lecture SQL SANS fieldmap

6.4.3.1 Cas d'usage

SQL contenant des alias identiques aux propriétés attendues.

6.4.3.2 Attendus

- Aucun champ manquant ;
- Identique au test 4.

6.5 Plan de tests – Couche complémentaire (source = view)

6.5.1 Objectif

Vérifier l'utilisation d'une vue SQL existante.

6.5.2 Test 6 – Vue simple

6.5.2.1 Cas d'usage

- Créer la vue view_xxxx conforme au guide ;
- Déclarer "source": "view" + "tablename": "${pluSchema}.view_xxxx" (),

6.5.2.2 Attendus

- L'impact lit la vue sans erreur ;
- Les alias définis dans la vue sont repris ;
- Le résultat est identique à une source table.

6.5.2.3 Erreurs à détecter

- Vue contenant DISTINCT : doit être rejetée ;
- Vue sans champ shape.

6.6 Plan de tests – Champs complémentaires

6.6.1 Objectif

Valider les 3 sources possibles de compléments.

6.6.2 Test 7 – Champ complémentaire standard

6.6.2.1 Cas d'usage

Ajouter un champ de table PLU : datvalid.

6.6.2.2 Attendus

- Le champ apparaît dans la fiche ;
- Fonctionne SANS SQL.

6.6.3 Test 8 – Champ JSON (ATTCOMP)

6.6.3.1 Cas d'usage

Utiliser un complément source "json" (exemple DESTDOMI).

6.6.3.2 Attendus

- Le système lit l'objet JSON ;
- Le champ apparaît dans :
  - fiche avancée parcelle,
  - API ADS → XML.

6.6.4 Test 9 – Champ complémentaire table/view

6.6.4.1 Cas d'usage

- Alias dans SQL : datentree AS DATE_MAJ ;
- Déclaration dans **complements**.

6.6.4.2 Attendus

- Champ lisible ;
- Valeur conforme en PDF + API.

6.7 PLAN DE TESTS – FONCTIONNEMENT MÉTIER : CALCUL D'IMPACT

6.7.1 Test 10 – Parcelle avec zone + prescription + infos

6.7.1.1 Attendus

- Le zonage principal remonte ;
- Les prescriptions apparaissent dans le groupe défini ;
- Les liens URL (règlement) fonctionnent.

6.7.2 Test 11 – Parcelle sans impact

6.7.2.1 Attendus

- Message "Aucun impact PLU" ;
- Aucun plantage widget.

6.7.3 Test 12 – Dossier ADS

6.7.3.1 Attendus

- Impact identique à celui de la parcelle correspondante ;
- API ADS restitue bien l'XML complet.

6.8 PLAN DE TESTS – API ADS : FONCTION IMPACTPLU

6.8.1 Test 13 – Appel API simple

6.8.1.1 Attendus

- Réponse XML valide ;
- Champs :
  - ID
  - TYPE
  - DESCRIPTION
  - COMPLEMENTAIRES
  - etc. (mapping CNIG,)

6.8.2 Test 14 – Vérification des compléments XML

Concerne les champs **name** des **complements**.

6.8.2.1 Attendus

- Chaque champ complémentaire apparaît dans le XML ;
- Noms = alias déclarés.

6.9 VÉRIFICATION

Élément	**Vérification**	**Statut**
Fichier impact présent	impact_xxx.json dans le bon dossier.	
id et **name** uniques	Vérifier absence de doublons.	
Couches « standard »	**source**=standard + **category** CNIG, pas de SQL.	
Couches « table »	sqlquery présent + champ shape, **DISTINCT** interdit	
Couches « view »	La vue existe + tablename correct	
Aucun DISTINCT	Vérifier SQL et vues.	
fieldmap correct	Champs aliasés ou définis.	
Tolérance d'inclusion	minimumPercentage ou minimumPercentagePLU.	
Champs complémentaires	Configurés selon json, table, ou view.	
URL règlement	Champ **reglement** valide.	

Redémarrage serveur	arcOpole PRO Serveur redémarré.	

7 GUIDE D'ANALYSE DES ERREURS

Ce guide s'articule en 4 parties :

- Erreurs liées à la structure des fichiers d'impact ;
- Erreurs liées aux couches (standard, table, view) ;
- Erreurs liées aux champs complémentaires ;
- Erreurs observées côté application (fiches, RU, API).

Chaque erreur est accompagnée :

- De sa cause probable ;
- De sa méthode de diagnostic ;
- Et de la solution recommandée.

7.1 ERREURS LIÉES AUX FICHIERS IMPACT_XXX.JSON

! Erreur : Le service arcOpole PRO Serveur démarre avec des avertissements dans les logs
Documentation : les fichiers d'impact sont lus au démarrage et les erreurs *sont ignorées mais visibles dans les logs*.
Causes probables
► JSON invalide (syntaxe → virgule manquante, accolade oubliée) ; ► Propriété obligatoire manquante (**id**, **name**, **layers**) ; ► Duplication d'un **id** ou **name**.
Diagnostic
► Vérifier les logs de démarrage :...\1Spatial\aOpServeur\logs*, ► Tester la validité JSON via un parseur en ligne.
Solution
► Corriger la syntaxe JSON ; ► Vérifier que chaque impact a un **id** et **name** uniques.
! Erreur : Aucune couche ne remonte dans les widgets
Causes probables
► Erreur de structure dans la propriété **layers ;** ► Catégorie manquante pour une couche standard (obligatoire) ; ► Mauvais chemin de dépôt des fichiers impact.
Diagnostic

► Vérifier présence du fichier dans :..\config\foncier\impact\, ► Redémarrer le serveur et relire les logs.
Solution
► Ajouter la propriété manquante ; ► Vérifier l'intégrité de la liste des couches.

7.2 ERREURS LIÉES AUX COUCHES STANDARD / TABLE / VIEW

7.2.1 Couches standard

! Erreur : Zonages ou prescriptions non détectés
Causes probables
► Catégorie (category) manquante → obligatoire ; ► Données PLU non conformes au modèle FONCIER/CNIG ; ► Champ attendu absent (TYPEZONE, LIBELLE, etc.).
Diagnostic
► Comparer la table PLU avec le mapping standard (tableau du guide) ; ► Tester une parcelle supposée impactée.
Solution
► Ajouter la propriété **category ;** ► Mettre à jour ou corriger les données CNIG.

7.2.2 Couches table

! Erreur : Aucune donnée ne remonte pour la couche table
Causes probables
► Champ **shape** absent de la requête SQL (obligatoire) ; ► Utilisation interdite de **DISTINCT ;** ► Alias manquants dans la SQL si **fieldmap** n'est pas utilisé.
Diagnostic
► Exécuter la requête SQL dans PgAdmin/SQL Developer ; ► Vérifier les alias et le champ shape.
Solution

► Ajouter le champ géométrique **shape ;** ► Retirer **DISTINCT ;** ► Revoir les alias SQL.
! Erreur : Crash silencieux / aucune erreur visible
Causes probables
► Lié à des erreurs SQL dans la propriété **sqlquery.**
Diagnostic
► Checker logs serveur : erreur du type *invalid geometry* ou *column does not exist ;* ► Exécuter la requête manuellement.
Solution
► Vérifier cohérence des noms de champs ; ► Vérifier existence du schéma ${pluSchema} ou usage d'un autre schéma.

7.2.3 Couches view

! Erreur : La vue est déclarée mais aucune donnée ne remonte
Causes probables
► Champ **shape** absent de la vue (obligatoire) ; ► Utilisation prohibée de **DISTINCT** dans la définition de la vue ; ► Droits insuffisants pour lire la vue.
Diagnostic
► Ouvrir la vue : SELECT * FROM <vue>, ► Vérifier qu'elle contient **shape**.
Solution
► Recréer la vue avec tous les alias nécessaires ; ► Supprimer tout **DISTINCT ;** ► Vérifier les permissions de l'utilisateur BD.

7.3 ERREURS LIÉES AUX CHAMPS COMPLÉMENTAIRES

! Erreur : Les champs complémentaires n'apparaissent pas dans la fiche avancée
Causes probables

► Propriété **complements** mal structurée ; ► Alias SQL manquant dans une source table/view ; ► Mauvaise déclaration source=json (champ non présent dans ATTCOMP).
Diagnostic
► Vérifier que l'alias SQL correspond exactement à la propriété **name ;** ► Vérifier la structure JSON du champ ATTCOMP pour les couches standard.
Solution
► Harmoniser la propriété **name** ↔ alias SQL ; ► Corriger la clé dans ATTCOMP.
! Erreur : Champs visibles dans l'API mais pas dans les PDF RU
Causes probables
► Certaines propriétés ne sont utilisées que pour l'API (ex. toutes les propriétés **fieldmap**) ; ► PDF ne prend que type + description pour couches standard.
Solution
► Vérifier que les bons champs sont mappés dans description ; ► Ajouter un champ complémentaire si nécessaire.

7.4 Erreurs observées dans l'application (widget, RU, API)

! Erreur : Le widget affiche un pictogramme ⚠ pour une couche
Causes probables
► La tolérance d'inclusion n'est pas atteinte.
Solution
► Ajuster la propriété **minimumPercentage ;** ► Désactiver l'avertissement : "warning": false.
! Erreur : Dans le RU PDF, certaines couches n'apparaissent pas
Causes probables
► Le mappage PDF n'utilise pas toutes les propriétés (ex. libelle, commentaire, etc.) ; ► Champs absents ou mal renseignés dans la propriété **fieldmap**.
Solution

► Vérifier le tableau de mapping standard du guide ; ► Corriger la propriété **fieldmap**.
! Erreur : L'API ADS retourne un XML incomplet
Causes probables
► Propriétés manquantes dans **fieldmap** (couches table/view) ; ► Champs non aliasés dans SQL ; ► Nom de propriété interdit ou mal orthographié.
Diagnostic
► Appeler l'API /ImpactPLU?id=<parcelle> et comparer le XML aux champs attendus.
Solution
► Compléter et harmoniser la propriété **fieldmap** et SQL.

7.5 PROBLÈMES FRÉQUENTS ET SOLUTIONS RAPIDES (SYNTHÈSE)

Élément	Vérification	Statut
Aucune couche ne remonte.	JSON invalide, propriété **category** manquante.	Corriger syntaxe JSON + ajouter le champ **category**.
Couche table vide.	Champ **shape** absent, DISTINCT.	Ajouter le champ **shape**, supprimer DISTINCT.
Vue non lue.	Champ **shape** manquant dans la vue.	Recréer la vue.
Champs complémentaires absents.	alias incorrect.	Corriger alias SQL.
PDF incomplet.	mauvais mappage.	Vérifier mapping standard.