**Nouveau metadata\_delta\_finder**

* 1 custom transformer (METADATA\_DELTA\_FINDER\_GEN.fwx) et 1 script python appelé par le pythonCaller\_7 (METADATA\_DELTA\_FINDER\_GEN.py)
* Le custom transformer (CT) METADATA\_DELTA\_FINDER\_NG.fwx comparait les valeurs de certains attributs (id,record\_last\_modified,resources{}.url,title,notes, resources{}.name) entre les attribut dans le pycsw ou local vs les nouvelles valeurs des api des provinces; afin d’évaluer si les métadonnées étaient inchangées, modifiées, insérées ou supprimées. Ce transformer se basait sur le fait que si le record\_last\_modified des provinces était modifié les métadonnées étaient modifiées. Par contre, pour certaines provinces cet attribut était un timestamp lorsqu’on appelait l’api; et pour d’autres il n’y avait pas d’attributs lié aux modifications de métadonnées (on prenait donc les attributs liés aux modifications des données).
* On a donc décidé de modifier ce CT afin de comparer tous les attributs obligatoires du HNAP (pour le géo) et CKAN (pour le non-géo) sauf le ‘record\_last\_modified’.
* 1 fichier yaml par province (1 pour le géo et 1 pour le non-géo) contient la liste des attributs de référence (pycsw ou JSON/XML locaux) et ceux de l’api de la province (nom des attributs des workspaces des provinces avant le CT metadata\_delta\_finder); selon :

attr\_1:

attr\_name: title

attr\_ref: MD\_Metadata.identificationInfo.MD\_DataIdentification.citation.CI\_Citation.title.CharacterString

attr\_new: title

ou : attr\_1 n’est que l’attribut 1 (seul l’attribut ‘id’ doit se nomme id puisqu’on l’utilise dans le script python, pour les autres on peut les appeler n’importe comment); attr\_name est le nom qu’on veut donner à l’attribut (on peut mettre n’importe quel nom, ce n’est que pour les identifier en cas de différence ou d’erreurs); et attr\_ref et attr\_new correspondent aux noms des attributs de référence vs des provinces.

*Pour l’instant, le yaml doit être inséré comme paramètre du CT; par contre on devrait améliorer pour qu’il puisse lire directement le fichier yaml dans les lookuptable de chaque province.*

* ~~Si un attribut n’a que des valeurs nulles; il ne faut pas l’inclure puisqu’il n’apparait pas dans la liste des noms d’attributs (voir pythonCaller\_2 et pythonCaller\_3 du CT). Ces listes sont utilisées pour voir si les attributs new et ref nommés dans le yaml (attr\_ref et attr\_new) existent ou non. Sinon le CT plante et log l’erreur. Si certaines valeurs sont nulles parmi toutes les métadonnées ce n’est pas un problème (elles vont juste être changées pour ‘NULL’ dans le script python).~~

*~~Une amélioration au CT serait de trouver comment faire pour pouvoir inclure les attributs n’ayant que des valeurs nulles.~~*

* Les attributs et les listes peuvent être traitées par ce CT; par contre pour les listes tous les éléments de la liste vont être triés puis joints pour en faire 1 attribut.
* ~~Si le nom des attributs contient des chiffres, il faut les renommer pour enlever ces nombre (voir le pythonCaller\_2 et le pythonCaller\_3). Par contre, si on veut prendre uniquement l’index 0 d’une liste, comme on ne peut pas mettre le ‘0’ on peut juste mettre la liste si les autres éléments sont nuls; le CT ne va traiter que l’élément non-null (mais attention s’il y a des valeurs dans les autres index; mieux vaut renommer l’attribut).~~ *~~Une autre amélioration serait de trouver comment inclure des noms d’attributs contenant des chiffres dans les fichiers yaml au lieu de devoir les renommer~~*~~.~~
* ~~exemple d’erreur et pistes de solution :~~

*~~ERROR: attribute: MD\_Metadata.contact.CI\_ResponsibleParty.role.CI\_RoleCode.codeListValue (yaml\_name: contact\_role) does not exists. Revise attribute name in the yaml file~~*

~~Soit l’attribut MD\_Metadata.contact.CI\_ResponsibleParty.role.CI\_RoleCode.codeListValue n’existe pas (pas le bon nom d’attribut), soit toutes ces valeurs sont vides. Parfois il existe pour certaines métadonnées mais pas d’autres, alors il faut renommer le bon attribut (les attributs ref ont souvent ce genre de problèmes puisqu’il ajoute des {} lorsqu’il y a plusieurs index mais pas lorsqu’il n’y en a qu’un seul. Il faut donc ajouter {0} lorsqu’il n’y a qu’une seule valeur dans la ‘liste’ : voir attributeRenamer\_3 dans le CT).~~

**À faire pour terminer :**

* Les yaml pour le géo des provinces sont fait (ils sont dans les lookuptable de chaque province); mais devraient tout de même être validés avant de modifier tous les workspaces.
* Les yaml pour le non-géo ont été créés à partir des fichiers JSON\_Config (et parfois une validation avec les attributs des workspaces mais pas dans la majorité des cas). Ils devront donc tous être validés et c’est possible que plusieurs attributs soient mal nommés.
  + Pour les attributs new contact\_name (email); comme on ne peut inclure qu’un contact pour CKAN la plupart des provinces ont pris le premier élément/index de la liste (contact\_email{0}). Comme pour l’instant on ne peut pas inclure de chiffre dans les noms des attributs, je n’ai pas inclus les attributs contact\_name dans les yaml de ces provinces. Pour les ajouter, il va falloir soit trouver comment pouvoir inclure des chiffres, soit renommer ces attributs dans les workspaces.
  + Pour les attributs resources{}.language, lorsqu’il n’y a qu’une valeur par défaut l’attribut par défaut ‘souvent : tbs\_resource\_lang’ est utilisé dans les JSON\_Config. Je n’ai donc pas inclus cet attribut dans ce cas. Afin de pouvoir comparer avec les listes des JSON locaux, il faudrait créer une liste dans les workspaces (ou simplement ne pas inclure cet attribut comme ce sont des valeurs par défaut; mais si éventuellement ça change il faudrait modifier les yaml pour inclure cet attribut).
* Pour valider que tout est correct : regarder la liste des attributs différents (soit dans le translation log ou l’attribut delta\_attr{} après le pythonCaller\_7 du CT (AttributeExposer\_6)

1 problème potentiel qui ne mènerait pas à une erreur dans le script python mais qui fait qu’un attribut serait différent entre ref et new pour toutes les métadonnées est l’ajout d’un ‘list attribute’ sans index. Par exemple pour NL dans le Default\_NL.csv il y avait ‘resources{}.protocol,HTTPS’, sans index.

* ~~Dans le CT, les PythonCaller\_2 et PythonCaller\_3 génèrent une liste du nom de tous les attributs (afin de s’assurer que les noms des attributs ref et new existent). Cette étape pourrait être effectuée directement dans le script python. Par contre il faut s’assurer que tous les features passent puisque les attributs ayant une valeur nulle ne sont pas inclues dans la liste de feature.getAllAttributeNames(). Pour ce faire, les sections ‘#apply create\_hash\_key() for ref/new attributes’ (\_order == 4 et \_order == 5) devraient être effectuées dans le close(self). De plus, il serait préférable de ne pas avoir à enlever les chiffres du nom de tous les attributs, juste trouver comment enlever les index des listes de la fonction feature.getAllAttributeNames().~~
* ~~Dans le script python (METADATA\_DELTA\_FINDER\_GEN.py) à la ligne 70 : identifie les coordonnées par ‘bound\_l’ : trouver une autre manière d’identifier les coordonnées (peuvent être appelées différemment d’une province à l’autre)~~
* ~~Dans le script python (METADATA\_DELTA\_FINDER\_GEN.py) à la ligne 73 : renommer attr\_update2 et attr\_update3 pour que ce soit plus compréhensible~~
* Finalement, modifier les workspaces de chaque province pour remplacer le METADATA\_DELTA\_FINDER\_NG.fwx par le nouveau METADATA\_DELTA\_FINDER\_GEN.fwx. ajouter un paramètre (published parameters) pour validate\_pourcent\_insert et mettre à yes par défaut.

**Particularité du HNAP et de la structure XML.**

Gestion des dates (création/publication)

MD\_Metadata.identificationInfo.MD\_DataIdentification.citation.CI\_Citation.date{} contient à la fois la date de publication et la date de création.

Le CT trie les éléments de la liste en fonction CI\_Date.dateType.CI\_DateTypeCode pour toujours avoir le même order

index 0 --> création

index 1 --> publication

Gestion des étendus temporels et spatiaux

MD\_Metadata.identificationInfo.MD\_DataIdentification.extent{} contient à la fois l'étendu temporelle de collection de données ainsi que l'étendu geospatiale.

Le CT trie les éléments de la liste pour faire passer l'étendu temporel en premier

Index 0 --> Étendu temporel

Index 1 --> Étendu spatial

Mots clés libres et ceux du Thésaurus

MD\_Metadata.identificationInfo.MD\_DataIdentification.descriptiveKeywords{} contient les mots clés liés <<freetext>> et les mots clés liés au thésaurus.

Le CT trie la liste pour faire passer les mots clés libres avant ceux du theraurus.

Index 0 --> Free Text

Index 1 --> Thesaurus

Éléments répétables

Dans le HNAP, plusieurs sections sont répétables. En fonction du contenu des métadonnées on peut retrouver de 1 à N fois certaines. Malheureusement, lorsque le XML est lu par FME, si la section répétable n’est là qu’une seule fois, cette dernière n’est pas considérée comme une liste ( attributs avec {}). Le CT effectue un renommage dynamique des attributs pour qu’ils soient considérés comme des liste de un item pour ces sections du HNAP.

MD\_Metadata.contact.

MD\_Metadata.referenceSystemInfo.

MD\_Metadata.identificationInfo.MD\_DataIdentification.citation.CI\_Citation.citedResponsibleParty.

MD\_Metadata.identificationInfo.MD\_DataIdentification.resourceMaintenance.

MD\_Metadata.identificationInfo.MD\_DataIdentification.descriptiveKeywords{0}.MD\_Keywords.keyword. (FreeText)

MD\_Metadata.identificationInfo.MD\_DataIdentification.topicCategory.

MD\_Metadata.distributionInfo.MD\_Distribution.transferOptions.

MD\_Metadata.distributionInfo.MD\_Distribution.distributor.MD\_Distributor.distributorContact.CI\_ResponsibleParty.

MD\_Metadata.distributionInfo.MD\_Distribution.distributionFormat.