import os

import xml.etree.ElementTree as ET

def strip\_namespace\_from\_tag(tag):

"""Garde le namespace intact, mais enlève les préfixes de namespace."""

if '}' in tag:

return tag.split('}', 1)[1] # Garde le namespace mais retire le préfixe

return tag

def find\_element\_by\_xpath(root, xpath):

"""

Recherche un élément XML en tenant compte du namespace dans le chemin XPath.

Le namespace est identifié et retiré dans cette recherche.

"""

elements = xpath.split('/')

current\_element = root

for el in elements:

if el:

# Recherche l'élément en tenant compte du namespace

for child in current\_element:

# Strip namespace to match the tag correctly

if strip\_namespace\_from\_tag(child.tag) == strip\_namespace\_from\_tag(el):

current\_element = child

break

else:

current\_element = None

break

return current\_element

def remove\_namespace\_prefixes\_except\_root(elem):

"""Supprime uniquement les préfixes de namespace des sous-éléments, mais garde le namespace dans la racine."""

for child in elem.iter():

# Skip the root element

if child is elem:

continue

child.tag = strip\_namespace\_from\_tag(child.tag)

def update\_fields\_in\_folder(folder\_path, field\_paths, base\_value="StressTestEBS", start\_value=1, output\_folder=None, num\_duplicates=1):

"""

Parcourt tous les fichiers XML, modifie les champs, duplique les fichiers et sauvegarde avec des noms uniques.

Args:

folder\_path (str): Chemin du dossier contenant les fichiers XML.

field\_paths (list): Liste des chemins XPath des champs à modifier.

base\_value (str): Base de la nouvelle valeur (par défaut "StressTestEBS").

start\_value (int): Valeur de départ pour l'incrémentation.

output\_folder (str): Chemin vers le dossier de sortie. Si None, les fichiers sont enregistrés dans le dossier source.

num\_duplicates (int): Nombre de duplications souhaitées pour chaque fichier XML.

"""

if output\_folder is None:

output\_folder = folder\_path

if not os.path.exists(output\_folder):

os.makedirs(output\_folder)

# Liste tous les fichiers dans le dossier

for file\_name in os.listdir(folder\_path):

file\_path = os.path.join(folder\_path, file\_name)

# Vérifie si c'est un fichier XML

if os.path.isfile(file\_path) and file\_name.endswith('.xml'):

print(f"Traitement du fichier : {file\_name}")

try:

# Charge le fichier XML

tree = ET.parse(file\_path)

root = tree.getroot()

for duplicate\_num in range(1, num\_duplicates + 1):

# Incrémente à chaque duplication

count = start\_value + duplicate\_num - 1

# Modifie les champs spécifiés dans la liste field\_paths

for field\_path in field\_paths:

field\_elem = find\_element\_by\_xpath(root, field\_path)

if field\_elem is not None:

xpath\_as\_string = field\_path.replace('/', '\_').replace('.', '')

new\_value = f"{base\_value}{xpath\_as\_string}-{count:011d}"

field\_elem.text = new\_value

print(f"Champ modifié : {field\_path} -> {new\_value}")

# Modifie les champs $ConversationId et $MessageID par des valeurs uniques

conversation\_id\_elem = find\_element\_by\_xpath(root, ".//ConversationId")

if conversation\_id\_elem is not None:

new\_conversation\_id = f"$conversationId-{count:011d}"

conversation\_id\_elem.text = new\_conversation\_id

print(f"ConversationId modifié -> {new\_conversation\_id}")

message\_id\_elem = find\_element\_by\_xpath(root, ".//MessageID")

if message\_id\_elem is not None:

new\_message\_id = f"$messageId-{count:011d}"

message\_id\_elem.text = new\_message\_id

print(f"MessageID modifié -> {new\_message\_id}")

# Supprime les préfixes de namespaces des sous-éléments, mais garde le namespace de la racine intact

remove\_namespace\_prefixes\_except\_root(root)

# Crée le nouveau nom de fichier avec un suffixe unique pour chaque duplication

new\_file\_name = file\_name.replace('.xml', f'\_new{duplicate\_num}.xml')

new\_file\_path = os.path.join(output\_folder, new\_file\_name)

# Écrit dans le nouveau fichier en conservant le namespace

tree.write(new\_file\_path, encoding="utf-8", xml\_declaration=True)

print(f"Fichier modifié et dupliqué enregistré sous : {new\_file\_path}")

except ET.ParseError as e:

print(f"Erreur de parsing dans le fichier {file\_name}: {e}")

# Exemple d'utilisation :

folder\_path = '/chemin/vers/le/dossier' # Remplacez par le chemin de votre dossier contenant les fichiers XML

output\_folder = '/chemin/vers/dossier/sortie' # Dossier où enregistrer les fichiers modifiés

# Liste des chemins XPath des champs à modifier (sans namespace ici)

field\_paths = [

'.//Message/MirroredTrade/Trade/Instrument/SEDProduct/FlowPayoff/SwapReference/ExternalId',

'.//Message/MirroredTrade/Trade/Instrument/SEDProduct/FlowPayoff/SwapReference/OtherField',

]

# Appel de la fonction avec duplication de 5 fois chaque fichier XML

update\_fields\_in\_folder(folder\_path, field\_paths, base\_value="StressTestEBS", start\_value=1, output\_folder=output\_folder, num\_duplicates=5)