Altova RaptorXML Server 2024

Manuel de l'utilisateur et de référence

Altova RaptorXML Server 2024
Manuel de l'utilisateur et de référence

Published: 2024

Table des matières

1 Introduction

Altova RaptorXML Server (ci-après aussi dénommé RaptorXML en abrégé) est le processeur XML et XBRL* hyper-rapide de troisième génération d'Altova. Il a été optimisé pour s'adapter aux tout derniers standards et environnement de calcul parallèles. Conçu pour fonctionner sur plusieurs plateformes, le moteur profite de l'ubiquité actuelle des ordinateurs multicœurs pour fournir un traitement ultra-rapide des données XML et XBRL.

Note: Le traitement XBRL est uniquement disponible dans RaptorXML+XBRL Server, pas dans RaptorXML Server.

Cette documentation

Cette documentation est fournie avec l'application et est aussi disponible en ligne sur le site web d'Altova.
Cette documentation est organisée dans les fonctions suivantes :

Site web d'Altova : Serveur de validation XML, Valideur XML

Dernière mise à jour : 08.04.2024

2 À propos de RaptorXML Server

Editions et systèmes d'exploitation

Il existe deux éditions de RaptorXML, chacune d'entre elle étant applicable pour un ensemble d'exigences différentes. Ces éditions sont décrites dans la section Editions et Interfaces 11. RaptorXML est disponible pour Windows, Linux et macOS. Pour plus de détails concernant la prise en charge du système, voir la section Exigences de système 15.

Fonctions et spécifications prises en charge

RaptorXML apporte une validation XML, des transformations XSLT et des exécutions XQuery, chacune d'entre elle avec une gamme étendue d'options puissantes. Voir la section Fonctions 16 pour une liste des fonctions disponibles et des fonctions clé. La section Spécifications prises en charge 18 fournit une liste détaillée des spécifications auxquelles RaptorXML se conforme. Pour plus d'informations, consulter la Page RaptorXML dans le site web Altova.

2.1 Éditions et Interfaces

Editions

RaptorXML est disponible dans les éditions suivantes :

- *RaptorXML Server* est un moteur de traitement XML basé sur serveur pour la validation et le traitement de documents XML, Schéma XML, XML Signature, XSLT et XQuery.
- *RaptorXML+XBRL Server* propose toutes les fonctions de RaptorXML Server et une grande palette de fonctions de traitement XBRL.

Voir ici[18] une liste des spécifications prises en charge[18].

Interfaces

Une fois avoir installé RaptorXML, vous pouvez y accéder d'une ou de plusieurs manières :

- *Une interface par ligne de commande (CLI) :* disponible pour les installation Windows, Linux et macOS de RaptorXML
- *Une interface client HTTP REST :* utilise l'interface HTTP de RaptorXML
- *Une interface serveur COM/.NET (Windows)* : utilise l'interface (i) COM/.NET et (ii) HTTP de RaptorXML
- *Une interface serveur Java (*Windows, Linux, et macOS) : utilise l'interface (i) Java API et (ii) REST HTTP de RaptorXML
- *Une interface Altova XMLSpy :* RaptorXML peut être accédé depuis l'intérieur de l'interface d'utilisateur Altova XMLSpy
- *Une interface moteur Python* : utilise (i) une roue Python RaptorXML dans votre environnement Python et (ii) l'API Python de RaptorXML dans votre script Python. Cela permet d'utiliser la fonction RaptorXML à être utilisée dans les scripts Python ensemble avec des packs Python tiers
- *.NET engine interface (Windows):* utilise (i) une DLL de RaptorXML et (ii) l'API .NET de RaptorXML pour créer des applications .NET indépendantes qui utilisent les fonctions de RaptorXML

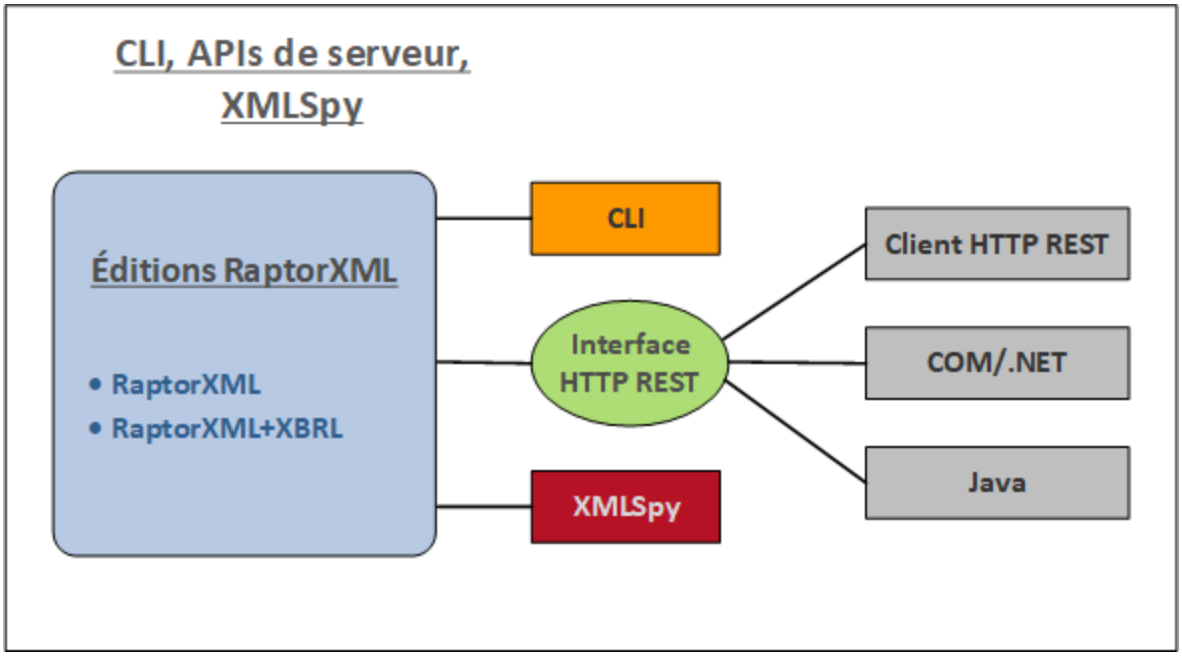
Ces sept interfaces peuvent être réparties en quatre groupes :

- Interface de ligne de commande (CLI)[57]
- API de serveur : HTTP, COM/.NET, Java[225]
- API de moteur : Python et .NET[342]
- Altova XMLSpy[13]

CLI, API de serveur et Altova XMLSpy

L'accès par le biais de la CLI, des APIs de serveur et Altova XMLSpy, peuvent être visualisés comme dans la figure ci-dessous.

RaptorXML Server définit une interface REST HTTP, qui est utilisée par des clients pour répartir les tâches de validation vers le serveur. Les clients peuvent soit accéder directement à l'interface REST HTTP soit utiliser les API Server de niveau supérieur COM/.NET et Java. Ces API offrent des classes COM/.NET et Java conviviales qui permettent de gérer la création et la répartition des requêtes REST HTTP. De plus, Altova XMLSpy peut être configuré pour exécuter des tâches de validation dans un RaptorXML Server à distance.



Interface de ligne de commande (CLI)

- RaptorXML est mis sous licence sur l'appareil sur lequel il est installé et cette instance est accédée par le biais de la ligne de commande
- Peut être installé sur Windows, Linux et macOS
- Fournit une Ligne de commande[57] pour la validation et le traitement de documents XML, Schéma XML, Signature XML, XQuery et XSLT
- Python 3.11.5 est regroupé dans RaptorXML et sera utilisé lorsqu'un script Python est invoqué avec l'option `--script`

Interface client HTTP REST

- RaptorXML est mis sous licence sur l'appareil sur lequel il est installé et cette instance est accédée par le biais d'une interface client HTTP REST[227]
- Les requêtes client sont effectuées dans un format JSON. Chaque requête est assignée à un répertoire de tâche sur le serveur, dans lequel les fichiers de sortie sont enregistrés. Les réponses Server au client comprennent toutes les informations pertinentes à propos du la tâche.
- Python 3.11.5 est regroupé dans RaptorXML et sera utilisé lorsqu'un script Python est invoqué avec l'option `--script`

COM/.NET interface

- Uniquement disponible sur Windows
- RaptorXML est automatiquement enregistré en tant qu'objet de serveur COM lorsqu'il est installé et il peut donc être invoqué depuis les applications et les langages de scripts qui prennent en charge la programmation d'appels COM
- RaptorXML est mis sous licence sur l'appareil sur lequel il est installé

- L'interface .NET est construit en tant que wrapper autour de l'interface COM
- L'API de serveur COM/.NET[268] de RaptorXML fournit des objets qui peuvent être utilisés dans des langages de scripting pour accéder aux fonctionnalités de RaptorXML
- Python 3.11.5 est regroupé dans RaptorXML et sera utilisé lorsqu'un script Python est invoqué avec l'option `--script`

Interface Java

- RaptorXML est mis sous licence sur l'appareil sur lequel il est installé et cette instance est accédée par le biais d'un programme Java
- La fonction RaptorXML est disponible dans l'API de serveur Java[274] comme des classes Java qui peuvent être utilisées dans des programmes Java.
- Python 3.11.5 est regroupé dans RaptorXML et sera utilisé lorsqu'un script Python est invoqué avec l'option `--script`

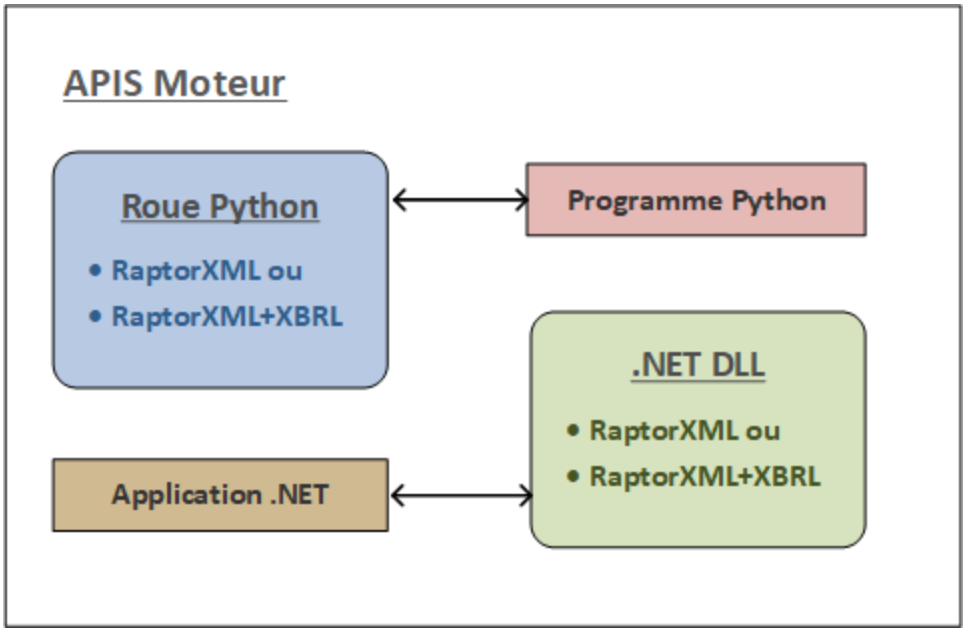
Altova XMLSpy

- Si vous avez installé et mis Altova XMLSpy sous licence et que XMLSpy peut accéder à RaptorXML Server sur un réseau, alors vous pouvez utiliser RaptorXML Server depuis la GUI XMLSpy pour valider les documents XML, ainsi qu'exécuter des transformations XSLT et XQuery.
- Vous pouvez valider le document actif ou tous les documents dans un dossier de projet XMLSpy.
- Les résultats de validation sont affichés dans la fenêtre de Messages de la GUI de XMLSpy.
- Dans XMLSpy, vous pouvez (i) valider des documents ou (ii) exécuter une transformation XSLT/XQuery en utilisant soit des moteurs de XMLSpy soit RaptorXML Server.
- Un des avantages de l'utilisation de Raptor est que vous pouvez configurer des validations individuelles par le biais d'une grande gamme d'options de validation. De plus, vous pouvez stocker un ensemble d'options Raptor en tant que "configuration" dans XMLSpy, puis choisir une de vos configurations définies pour une validation Raptor particulière. L'utilisation de Raptor est aussi avantageuse si des collections de données importantes doivent être validées.

API de moteur

Les API de moteur[342] sont différentes des APIs de Serveur en ce que RaptorXML est contenu dans la roue Python et dans la DLL .NET qui sont utilisées, respectivement, par les programmes Python et les applications .NET (*voir figure ci-dessous*). Ces programmes/applications doivent utiliser, respectivement, l'API Python[346] de Raptor et l'API .NET[356] de Raptor afin d'accéder aux fonctions RaptorXML.

Note : La fonction fournie par l'API Python[346] et l'API .NET[356] est considérablement plus importante que celle fournie soit par la CLI soit par les APIs de serveur ; Par exemple, la capacité de lire des documents et de manipuler des données.



Interface Python

- RaptorXML est disponible dans un package de roue Python qui peut être installé dans votre environnement Python 3.11.5
- Un programme Python qui utilise des objets depuis l'API Python[346] de RaptorXML. Cette API permet une plus grande fonctionnalité que celle disponible dans la CLI, et elle peut être combinée avec la fonctionnalité fournie par les bibliothèque tierces dans votre environnement Python
- Lorsque la fonction de %RXML%> est appelée par le biais de la roue Python de RaptorXML, un contrôle est effectué pour une licence RaptorXML avant que la commande soit exécutée

Interface .NET

- RaptorXML est disponible dans une DLL qui peut être intégrée dans une application qui prend en charge le Framework .NET. Voir la section .NET Framework API[356] pour plus d'informations concernant l'API.
- L'API .NET[356] de RaptorXML permet l'accès à RaptorXML. La fonctionnalité disponible est bien plus importante que celle disponible dans la CLI de RaptorXML.
- Lorsque la fonction de RaptorXML est appelée par le biais d'une application .NET, un contrôle est effectué pour une licence RaptorXML valide sur cet appareil.

2.2 Exigences de système

RaptorXML Server est pris en charge sur les systèmes d'exploitation suivants :

- *Windows*

  Windows 10, Windows 11

- *Windows Server*

  Windows Server 2016 ou plus récent

Exigences de système (Linux)

- <% REDHAT%>
- <% CENTOS%>
- <% DEBIAN%>
- <% UBUNTU%>
- AlmaLinux 9.0
- Rocky Linux 9.0

Prérequis

- Réaliser l'installation soit en tant qu'utilisateur **root**, soit en tant qu'utilisateur avec des privilèges **sudo**.
- La version précédente de RaptorXML Server doit être désinstallée avant d'installer une nouvelle.
- Si vous avez l'intention d'utiliser la fonctionnalité des graphiques d'Altova, vous devez installer au moins une police sur votre système pour vous assurer que les graphiques seront rendus correctement. Pour recenser les polices installées, utilisez par exemple la commande **`fc-list`** de Fontconfig library.
- Les bibliothèques suivantes sont requises comme prérequis pour installer et exécuter l'application. Si les packages ci-dessous ne sont pas déjà disponibles sur l'appareil Linux, exécutez la commande `yum` (ou `apt-get`, si applicable) pour les installer.

CentOS, RedHat	Debian	Ubuntu
krb5-libs	libgssapi-krb5-2	libgssapi-krb5-2

- *macOS*

  macOS 10.15 ou plus récent

RaptorXML est disponible aussi bien pour les appareils 32-bit et 64-bit. En particulier, il s'agit de cœurs basés sur des ensembles d'instruction x86 et amd64 (x86-64) : Intel Core i5, i7, XEON E5. Pour utiliser RaptorXML par le biais d'une interface COM, les utilisateurs doivent avoir des privilèges pour utiliser l'interface COM, c'est-à-dire d'enregistrer l'application et d'exécuter les applications pertinentes et/ou les scripts.

2.3 Fonctions

RaptorXML fournit la fonction recensée ci-dessous. La plupart des fonctions peut être utilisée par le biais de la commande de ligne et l'interface COM. Une différence majeure est que l'utilisation de l'interface COM sur Windows permet aux documents d'être créés depuis des strings de texte par le biais de l'application ou du code de scripting (au lieu de référencer XML, DTD, XML Schéma, XSLT, ou des fichiers XQuery).

Validation XML

- Valide le document XML fourni par rapport à des DTD internes ou externes ou des Schémas XML
- Vérifie la bonne formation de XML, DTD, schéma XML, XSLT, et les documents XQuery

Transformations XSLT

- Transforme XML à l'aide du document XSLT 1.0, 2.0 ou 3.0 fourni
- Les documents XML et XSLT peuvent être fournis en tant qu'un fichier (par le biais d'une URL) ou, dans le cas de l'utilisation COM, en tant que string de texte
- La sortie est retournée en tant que fichier (sous un emplacement nommé) ou, dans le cas de l'utilisation COM, en tant que string de texte
- Les paramètres XSLT peuvent être fournis par le biais de la ligne de commande et par le biais de l'interface COM
- Les fonctions d'extension Altova, ainsi que des fonctions d'extension XBRL, Java et .NET permettent un traitement spécialisé. Cela permet, par exemple la création de fonctions comme des graphiques et des codes-barres dans les documents de sortie

Exécution XQuery

- Exécute des documents XQuery 1.0 et 3.0
- Les documents XQuery et XML peuvent être fournis en tant qu'un fichier (par le biais d'une URL) ou, dans le cas d'une utilisation COM, en tant que string de texte
- La sortie est retournée en tant que fichier (sous un emplacement nommé) ou, dans le cas de l'utilisation COM, en tant que string de texte
- Des variables XQuery externes peuvent être fournies par le biais de la ligne de commande et par le biais de l'interface COM
- Les options de sérialisation incluent : encodage de sortie, méthode de sortie (c'est-à-dire si la sortie est XML, XHTML, HTML, ou texte), omission de la déclaration XML et indentation

Validation/Conversion JSON et Avro

- Validation de schéma JSON et de documents de schéma Avro
- Validation d'instances JSON par rapport aux schémas JSON et des schémas Avro
- Validation de binaires Avro
- Conversion de binaires Avro en schéma Avro et données Avro en format JSON
- Conversion de données Avro JSON en binaire Avro

Fonctions hyper-performantes

- Optimisations de code de la performance ultra élevée
  - Implémentations de sets d'instructions natives
  - Version u32-bit ou 64-bit
- Empreinte mémoire très basse
  - Représentation in-memory extrêmement compacte de l'ensemble d'information XML
  - Validation d'instance de streaming
- Capacités multiplateforme
- Code hautement évolutif pour l'informatique multi-CPU/multi-cœur/parallèle
- Chargement, validation et traitement parallèle sur conception

Fonctions de développeur

- Capacités de rapport d'erreur supérieur
- Mode serveur Windows et mode daemon Unix (par le biais des options de ligne de commande)
- Interpréteur Python 3.x pour scripting est inclus
- Fonction RaptorXML dans un pack Python permet l'importation de la fonction en tant que bibliothèque Python
- API Framework .NET permet l'accès au modèle de données XML sous-jacent
- Plateforme Windows sur API COM
- API Java partout
- Fonctions d'extension XPath Java, .NET, etc.
- Sérialisation de streaming
- Serveur HTTP intégré avec API validation REST

Pour plus information, voir la section Spécifications prises en charge[18] sur le site web d'Altova.

2.4 Spécifications prises en charge

RaptorXML prend en charge la spécifications recensées ci-dessous.

Recommandations W3C

Site web : World Wide Web Consortium (W3C)

- Extensible Markup Language (XML) 1.0 (cinquième édition)
- Extensible Markup Language (XML) 1.1 (deuxième édition)
- Espaces de noms dans XML 1.0 (troisième édition)
- Espaces de noms dans XML 1.1 (deuxième édition)
- Ensemble d'information XML (deuxième édition)
- Base XML (deuxième édition)
- Inclusions XML (XInclude) Version 1.0 (deuxième édition)
- XML Linking Language (XLink) Version 1.0
- Schéma XML Partie 1: Structures Deuxième édition
- Schéma XML Partie 2: Datatypes Deuxième édition
- Langage de définition de Schéma XML W3C (XSD) 1.1 Partie 1: Structures
- Langage de définition de Schéma XML W3C (XSD) 1.1 Partie 2: Types de données
- XPointer Framework
- XPointer xmlns() Scheme
- XPointer element() Scheme
- Langage de chemin XML (XPath) Version 1.0
- Transformations XSL (XSLT) Version 1.0
- Langage de chemin XML (XPath) 2.0 (deuxième édition)
- Transformations XSL (XSLT) Version 2.0
- XQuery 1.0 XQuery 1.0: un langage de requête XML (deuxième édition)
- Fonctions XQuery 1.0 et XPath 2.0 et Opérateurs (deuxième édition)
- Sérialisation XSLT 2.0 et XQuery 1.0 (deuxième édition)
- Langage de chemin XML (XPath) 3.0
- Langage de chemin XML (XPath) 3.1
- XQuery 3.0: Un langage XML Query
- XQuery Update Facility 1.0
- Fonctions XPath et XQuery et Opérateurs 3.0
- Sérialisation XSLT et XQuery 3.0

W3C Ébauche de travail & Recommandations des candidats

Site web : World Wide Web Consortium (W3C)

- Transformations XSL (XSLT) Version 3.0 (sous-ensemble)
- XQuery 3.1: Un langage XML Query
- Fonctions XPath et XQuery et Opérateurs 3.1
- XQuery Update Facility 3.0
- Sérialisation XSLT et XQuery 3.1

Standards OASIS

Site web : Standards OASIS

- Catalogues XML V 1.1 - Standard OASIS V1.1

Standards JSON/Avro

Sites web : Schéma JSON et Apache Avro

- JSON Schema Draft 4
- JSON Schema Draft 6
- JSON Schema Draft 7
- JSON Schema Draft 2019-09
- JSON Schema Draft 2020-12
- Apache Avro™ 1.8.1

2.5 Changements notables

Ci-dessous, vous trouverez des changements dans chaque version qui pourrait solliciter votre attention.

v2024

Dans la ligne de commande, l'option **`--network-timeout`** prend une valeur en millisecondes à partir de ce release (à la place de secondes comme c'était le cas dans les releases précédents). Cette option peut être définie pour un nombre de commandes et, dans la description de la commande, se trouver sous *Options communes*. Pour un exemple, voir la commande valxml-withxsd (xsi)[63].

3 Installation et licence

Cette section décrit l'installation, la gestion de licence et d'autres procédures de configuration. Elle est organisée en sections comme suit :

3.1 Configuration sur Windows

Cette section décrit l'installation[22] et la licence[28] de RaptorXML Server sur les systèmes Windows.

Exigences de système (Windows)

Notez les exigences système suivantes :

- Windows 10, Windows 11
- Windows Server 2016 ou plus récent

Prérequis

Notez les exigences préalables suivantes :

- Réaliser une installation en tant qu'utilisateur de privilèges administratifs.
- À partir de la version 2021, une version 32-bit de RaptorXML Server ne peut pas être installée sur la version 64-bit, ou une version 64-bit sur une version 32-bit. Vous devez soit (i) supprimer la version plus ancienne avant d'installer la nouvelle version ou (ii) mettre à niveau vers la version plus nouvelle qui est la même version bit que votre ancienne installation.

3.1.1 Installation sur Windows

RaptorXML Server est disponible pour l'installation sur des systèmes Windows. La procédure élargie d'installation et de configuration est décrite ci-dessous. Pour des informations détaillées sur des parties spécifiques de la procédure d'installation, voir les sections respectives.

Installation de RaptorXML Server

RaptorXML Server peut être installé sur les systèmes Windows comme suit :

- En tant que produit de serveur autonome séparé. Pour installer RaptorXML Server, téléchargez et exécutez le programme d'installation RaptorXML Server. Suivez les instructions sur écran.
- Pour installer RaptorXML Server comme partie du package FlowForce Server , téléchargez et exécutez le programme d'installation FlowForce Server. Suivez les installations sur écran veillez à cocher l'option pour installer RaptorXML Server.

Les programmes d'installation pour RaptorXML Server et FlowForce Server sont disponibles au Altova Download Center (http://www.altova.com/download.html). Vous pouvez sélectionner votre langue d'installation depuis la zone inférieure gauche de l'assistant. Notez que cette sélection définit également la langue par défaut de RaptorXML Server. Vous pouvez changer la langue plus tard depuis la ligne de commande.

Installer LicenseServer
Pour que RaptorXML Server fonctionne, il faut qu'il ait une enregistré licence par le biais d'un Altova LicenseServer sur votre réseau. Lorsque vous installez RaptorXML Server ou FlowForce Server sur les systèmes Windows, vous pouvez installer LicenseServer ensemble avec RaptorXML Server ou FlowForce Server. Pour les détails, voir Installer LicenseServer[27].

Après l'installation, l'exécutable RaptorXML Server sera situé par défaut sous le chemin suivant :

```
<ProgramFilesFolder>\Altova\RaptorXMLServer2024\bin\RaptorXML.exe
```

Tous les enregistrements nécessaires pour utiliser RaptorXML Servervia une interface COM, comme une interface Java, et dans l'environnement .NET seront effectués par le programme d'installation.Ceci inclut l'enregistrement du programme d'exécution RaptorXML Server comme objet de serveur COM et l'ajout du fichier `Altova.RaptorXML.dll` à la bibliothèque de référence .NET.

Installer sur Windows Server Core

Windows Server Core n'a pas de GUI et doit être installé via la ligne de commande. Voir la section Installer sur Windows Server Core 23 pour l'installation.

Désinstaller RaptorXML Server

Désinstaller RaptorXML Server comme suit :

1. Cliquez de la touche droite sur le bouton Windows **Démarrer** et sélectionnez **Paramètres**.
2. Ouvrez le panneau de configuration (commencez à écrire « Panneau de configuration » et cliquez sur l'entrée suggérée).
3. Sous *Programmes*, cliquez **Désinstaller un programme**.
4. Dans le panneau de configuration, sélectionnez RaptorXML Server et cliquez sur **Désinstaller**.

Licence d'évaluation

Pendant le processus d'installation, vous recevrez l'option de demander une licence d'évaluation de 30 jours pour RaptorXML Server. Après avoir soumis la demande, une licence d'évaluation sera envoyée à l'adresse e-mail que vous avez enregistré.

3.1.2 Installer sur Windows Server Core

Windows Server Core est une installation Windows minimale qui n'utilise qu'un certain nombre de fonctions de la GUI. Vous pouvez installer RaptorXML Server sur un appareil Windows Server Core comme suit :

1. Téléchargez le programme d'installation RaptorXML Server exécutable depuis le site web d'Altova. Ce fichier est nommé **`RaptorXMLServer<version>.exe`**. Assurez-vous que l'exécutable corresponde à votre plate-forme de serveur (32-bit ou 64-bit).
2. Sur un appareil standard Windows (pas l'appareil Windows Server Core), exécutez la commande **`RaptorXMLServer<version>.exe /u`**. Ceci déballe le fichier **`.msi`** vers le même dossier que celui du programme d'installation.
3. Copiez le fichier décompressé **`.msi`** vers l'appareil Windows Server Core.
4. Si vous mettez à jour une version antérieure de RaptorXML Server, fermez RaptorXML Server avant de réaliser la prochaine étape.
5. Utilisez le fichier **`.msi`** pour l'installation en exécutant la commande **`msiexec /i RaptorXMLServer.msi`**. Ceci lance l'installation de Windows Server Core.

Note : lors de la mise à jour vers une version principale, vous pouvez garder vos paramètres RaptorXML Server en utilisant les propriétés énumérées dans les sous-sections de cette section : (i) Propriétés du serveur Web 25, (ii) Propriétés du serveur SSL Web 25, and (iii) Propriétés de Service 26.

Important : Garder le fichier MSI !

Veuillez noter les points suivants :

- Gardez le fichier **`l'extraction.msi`** à un endroit sûr. Vous en aurez besoin plus tard pour désinstaller, réparer ou modifier votre installation.
- Si vous voulez renommer le fichier MSI, faites-le avant d'installer RaptorXML Server.
- Le nom du fichier MSI est stocké dans le registre. Vous pouvez mettre à jour son nom ici si le nom du fichier a changé.

Inscrire RaptorXML Server avec LicenseServer

Si vous installez RaptorXML Server pour la première fois ou si vous mettez à jour vers une **version majeure**, vous allez devoir enregistrer RaptorXML Server avec Altova LicenseServer sur votre réseau. Si vous êtes en train de mettre à jour vers une version non majeure de RaptorXML Server, alors l'enregistrement précédent de LicenseServer sera connu par l'installation et vous n'aurez pas besoin d'enregistrer RaptorXML Server avec LicenseServer. Toutefois, si vous voulez changer le LicenseServer qui est utilisé par RaptorXML Server à tout moment, vous allez devoir enregistrer RaptorXML Server avec le nouveau LicenseServer.

Pour enregistrer RaptorXML Server avec Altova LicenseServer pendant l'installation, exécutez la commande d'installation avec la propriété **`REGISTER_WITH-LICENSE_SERVER`**, telle que recensée ci-dessous, fournissant le nom ou l'adresse de la machine de LicenseServer en tant que valeur de la propriété, par exemple :

```
msiexec /i RaptorXMLServer.msi REGISTER_WITH_LICENSE_SERVER="localhost"
```

Pour enregistrer RaptorXML Server avec un Altova LicenseServer après l'installation, exécutez la commande suivante :

```
msiexec /r RaptorXMLServer.msi REGISTER_WITH_LICENSE_SERVER="<MyLicenseServer-Machine-
   Address>"
```

Commandes utiles

Vous trouverez ci-dessous un ensemble de commandes utiles dans le contexte de l'installation.

Pour tester la valeur retour de votre installation, exécutez un script semblable à celui ci-dessous. Le code retour sera dans la variable d'environnement `%errorlevel%` . Un code retour `0` indique un succès.

```
start /wait msiexec /i RaptorXMLServer.msi /q
echo %errorlevel%
```

Pour une installation silencieuse avec un code retour et un log de la procédure d'installation :

```
start /wait msiexec /i RaptorXMLServer.msi /q /L*v! <pathToInstallLogFile>
```

Pour modifier l'installation :

```
msiexec /m RaptorXMLServer.msi
```

Pour réparer l'installation :

```
msiexec /r RaptorXMLServer.msi
```

Pour désinstaller RaptorXML Server :

```
msiexec /x RaptorXMLServer.msi
```

Pour la désinstallation de RaptorXML Server en silence et faire rapport du résultat détaillé dans un fichier log :

```
start /wait msiexec /x RaptorXMLServer.msi /q /L*v! <pathToUninstallLogFile>
```

Pour installer RaptorXML Server en utilisant un autre langage (les codes de langage disponibles sont : allemand=`de`; espagnol=`es`; français=`fr`) :

```
msiexec /i RaptorXMLServer.msi INSTALLER_LANGUAGE=<languageCode>
```

Note : sur Windows Server Core, la fonctionnalité des graphiques de RaptorXML Server ne sera pas disponible.

3.1.2.1 Propriétés du serveur Web

Vous pouvez configurer le serveur web RaptorXML Server en utilisant les propriétés ci-dessous. Pour définir une propriété, exécutez une commande d'installation avec le paramètre de propriété ajouté, comme suit :

```
msiexec /i RaptorXMLServer.msi RXML_WebServer_Host=127.0.0.1
```

Liste des propriétés

Les propriétés du serveur web RaptorXML Server :

`RXML_WebServer_Host=<IP4 Address>`
Utilisez `127.0.0.1` si vous voulez accéder au serveur web depuis cet appareil uniquement. Utilisez `0.0.0.0` pour rendre le serveur web globalement accessible.

`RXML_WebServer_Port=<Port Number>`
Spécifie le port qui est utilisé pour accéder au serveur web.

`RXML_WebServer_Enabled=<0 or 1>`
Sélectionnez `1` pour activer l'écoute à la définition du port actuel. Sélectionnez `0` pour désactiver l'écoute à ce port.

3.1.2.2 Propriétés du serveur Web-SSL

Vous pouvez configurer le serveur SSL web RaptorXML Server en utilisant les propriétés ci-dessous. Pour définir une propriété, exécutez une commande d'installation avec le paramètre de propriété ajouté, comme suit :

```
msiexec /i RaptorXMLServer.msi RXML_SSLWebServer_Host=127.0.0.1
```

Liste des propriétés

Pour configurer les serveur SSL Web de RaptorXML Server, utilisez les propriétés suivantes :

`RXML_SSLWebServer_Host=<IP4 Address>`
Utilisez `127.0.0.1` si vous voulez accéder au serveur SSL web (pour une transmission chiffrée) depuis cet appareil uniquement. Utilisez `0.0.0.0` pour rendre le serveur SSL web globalement accessible.

RXML_SSLWebServer_Port=<Port Number>
Spécifie le port qui est utilisé pour accéder au serveur SSL web (pour une transmission chiffrée).

RXML_SSLWebServer_Enabled=<0 or 1>
Sélectionnez **1** pour activer l'écoute à la définition du port actuel. Sélectionnez **0** pour désactiver l'écoute à ce port.

RXML_SSLWebServer_Certificate=<Path-to-certificate-file>
Chemin complet vers un certificat SSL, inséré en guillemets doubles.

RXML_SSLWebServer_PrivateKey=<Path-to-private-key-file>
Chemin complet vers un fichier clé privé, inséré en guillemets doubles.

3.1.2.3 Propriétés de service

Vous pouvez configurer le service RaptorXML Server en utilisant les propriétés ci-dessous. Pour définir une propriété, exécutez une commande d'installation avec le paramètre de propriété ajouté, comme suit :

```
msiexec /i RaptorXMLServer.msi RXML_Service_DisplayName=RaptorXMLServer
```

Liste des propriétés

Pour configurer les services de RaptorXML Server, utilisez les propriétés suivantes :

RXML_Service_DisplayName=<Serveice Display Name>
Nom qui sera affiché pour ce service. Insérer le nom en guillemets doubles.

RXML_Service_StartType=<Startup Type>
Spécifie comment le service a démarré pendant une start-up de système. Les valeurs peuvent être comme suit : **auto | auto-delayed | demand | disabled**.

RXML_Service_Username=<UserName>
Spécifie l'utilisateur de connexion pour ce service. Utilisez l'un des : **LocalSystem | NT Authority\LocalService | NT Authority\NetworkService | <any user with relevant rights>**.

RXML_Service_Password=<Password>
Le mot de passe de l'utilisateur de démarrage du service en texte brut. (Conseil : utilisez l'interface utilisateur du programme d'installation pour éviter de saisir les mots de passe en texte brut.) Aucun mot de passe n'est requis si le nom utilisateur est l'un des : **LocalSystem | NT Authority\LocalService | NT Authority\NetworkService**.

3.1.3 Installer LicenseServer (Windows)

Pour que RaptorXML Server fonctionne, il faut qu'il ait une licence par le biais d'un Altova LicenseServer sur votre réseau. Lorsque vous installez RaptorXML Server ou FlowForce Server sur les systèmes Windows, vous pouvez installer LicenseServer ensemble avec RaptorXML Server ou FlowForce Server. Si un LicenseServer est déjà installé sur votre réseau, vous ne devez pas installer un autre — sauf si une plus nouvelle version de LicenseServer est requise. (*Voir le point suivant, versions de LicenseServer.*)

Pendant la procédure d'installation de RaptorXML Server ou de FlowForce Server, activez ou désactivez l'option pour installer LicenseServer, si besoin. Veuillez noter les points suivants :

- Si vous n'avez pas encore installé LicenseServer, laissez les paramètres par défaut tels quels. L'assistant installera la dernière version sur l'ordinateur sur lequel vous exécutez l'assistant.
- Si vous n'avez pas encore installé LicenseServer et souhaitez installer Altova LicenseServer sur un autre ordinateur, décochez la case *Installer Altova LicenseServer sur l'appareil* et choisissez **Plus tard**. Dans ce cas, vous devrez installer LicenseServer séparément et inscrire RaptorXML Server après.
- Si LicenseServer a déjà été installé sur votre ordinateur mais qu'il s'agit d'une version inférieure à celle indiquée par l'assistant d'installation, laissez les paramètres par défaut (pour mettre à jour à la version plus récente) tels quels. Dans ce cas, l'assistant d'installation mettre automatiquement votre version de LicenseServer à jour. L'information d'inscription et de licence existante sera reportée à la nouvelle version de LicenseServer.
- Si LicenseServer a été installé sur votre ordinateur ou sur votre réseau et a la même version que celle indiquée par l'assistant, suivez les étapes suivantes :
  - Décochez la case *Installer Altova LicenseServer sur l'appareil*.
  - Sous *Inscrire ce produit avec* , choisissez le serveur de licence avec lequel vous voulez vous inscrire RaptorXML Server. De manière alternative, choisissez **Plus tard**. Veuillez noter que vous pouvez toujours sélectionner **Plus tard** si vous voulez ignorer les associations de LicenseServer et poursuivre l'installation de RaptorXML Server.

Pour plus d'information sur l'inscription et la licence RaptorXML Server avec Altova LicenseServer, voir la section Licence RaptorXML Server 28.

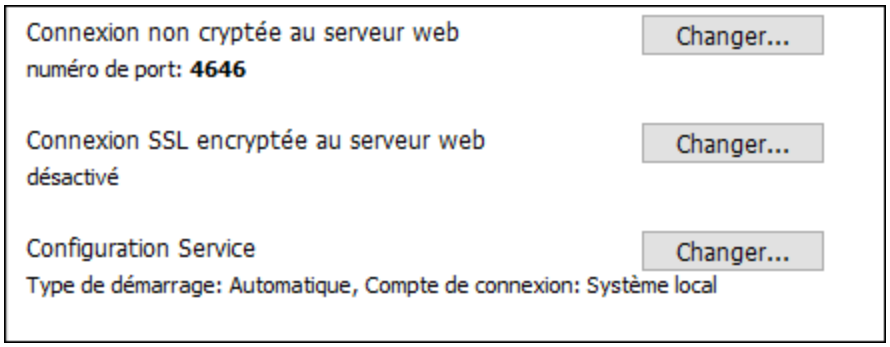
Versions de LicenseServer

- Les produits de Altova doivent être dotés d'une licence soit () avec une version de LicenseServer qui correspond à la version installée RaptorXML Server, soit pour une version ultérieure de LicenseServer.
- La version LicenseServer qui correspond à la version actuelle de RaptorXML Server est ***3.14***.
- Sur Windows, vous pouvez installer la version correspondante de LicenseServer comme faisant partie de l'installation de RaptorXML Server ou installez LicenseServer séparément. Sur Linux et macOS, vous devez installer LicenseServer séparément.
- Avant que la nouvelle version de LicenseServer ne soit installée, chaque version plus ancienne doit être désinstallée.
- Lors de la désinstallation de LicenseServer, toute l'information liée à l'inscription et à la licence contenue dans la version plus ancienne de LicenseServer sera enregistrée dans une base de données sur votre appareil de serveur. Ces données seront importées automatiquement dans la version plus nouvelle lorsque celle-ci est installée.
- Les versionsLicenseServer sont rétro-compatibles. Elles fonctionneront avec des versions plus anciennes de RaptorXML Server.
- La dernière version de LicenseServer disponible sur le site web d'Altova. Cette version fonctionnera avec toute version actuelle ou ancienne de RaptorXML Server.

- Le numéro de version de la licence LicenseServer actuellement installée est indiqué au bas de la page de configuration LicenseServer (tous les onglets).

3.1.4 Configuration service et réseau (Windows)

Pendant l'installation de RaptorXML Server, vous pouvez configurer des paramètres pour accéder à RaptorXML Server par le biais du réseau et pour exécuter RaptorXML Server en tant que service Windows (*voir la capture d'écran ci-dessous*).



Les paramètres recensés ci-dessous sont disponibles. Laissez les paramètres par défaut tels quels s'ils sont acceptables ou si vous avez un doute. Si vous souhaitez modifier un paramètre, sélectionnez son bouton **Changer** (*voir la capture d'écran ci-dessus*).

- Le port à utiliser la communication chiffrée avec RaptorXML Server.
- Si des connexions sécurisées (chiffrées SSL) vers RaptorXML Server sont autorisées. Si oui, sur quel port. Par défaut, les connexions sécurisées sont désactivées. Pour plus d'information, voir la section sur la configuration du chiffrement SSL 237.
- Paramètres de service Windows. Ceux-ci contiennent :
  - La manière dont RaptorXML Server devrait démarrer en tant que service Windows : automatique, sur demande, retarder automatique ou désactiver.
  - Le compte utilisateur à utiliser par RaptorXML Server pour le service Windows : *Système local, Service local, Service réseau,* ou *Autre utilisateur*. Si vous utilisez *Autre utilisateur*, vous pouvez définir le nom d'utilisateur et le mot de passe pour cet utilisateur, similairement à la manière dont c'est fait dans la console Windows Services management. Notez que l'utilisateur sélectionné doit avoir un accès de lecture/d'écriture pour accéder à `C:\ProgramData\Altova`. Autrement, l'installation ou le démarrage pourrait échouer.

Vous pouvez modifier les paramètres après l'installation. Pour modifier la configuration de service Windows, ouvrez la console Windows Services management (en tapant `Services.msc` dans une fenêtre de ligne de commande) et modifiez le service requis depuis cet endroit.

3.1.5 Licence RaptorXML Server (Windows)

Afin d'utiliser RaptorXML Server, il doit être doté d'une licence de Altova LicenseServer. La gestion des licences est une procédure à deux étapes :

1. **Inscrire RaptorXML Server** avec LicenseServer. L'inscription est réalisée depuis RaptorXML Server.
2. **Attribuer une licence** à RaptorXML Server depuis LicenseServer. Télécharger la dernière version de LicenseServer depuis le site web d'Altova et installez-la sur votre appareil local ou un appareil sur votre réseau.

Ces deux étapes sont décrites dans cette section. Pour toute information détaillée, voir le manuel utilisateur LicenseServer sur le site web d'Altova.

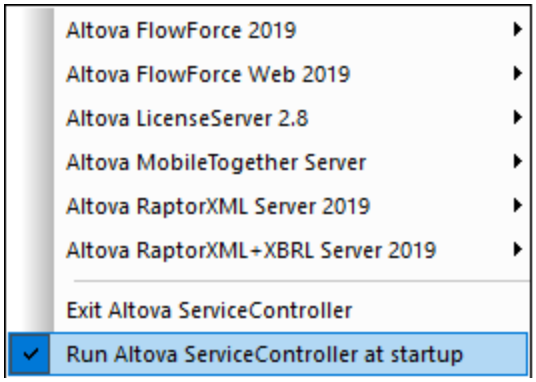
3.1.5.1 Démarrer LicenseServer, RaptorXML Server

Altova LicenseServer (désigné LicenseServer) et RaptorXML Server sont tous les deux démarrés depuis Altova ServiceController.

Altova ServiceController

Altova ServiceController (ServiceController en abrégé) est une application permettant de lancer, d'arrêter et de configurer confortablement les services Altova **ms**. ServiceController est installé avec l'Altova LicenseServer et avec les produits de serveur Altova installés en tant que services (DiffDog Server, FlowForce Server, Mobile Together Server et RaptorXML(+XBRL) Server). ServiceController peut être accédé par le biais de la barre d'état système (*voir la capture d'écran ci-dessous*).

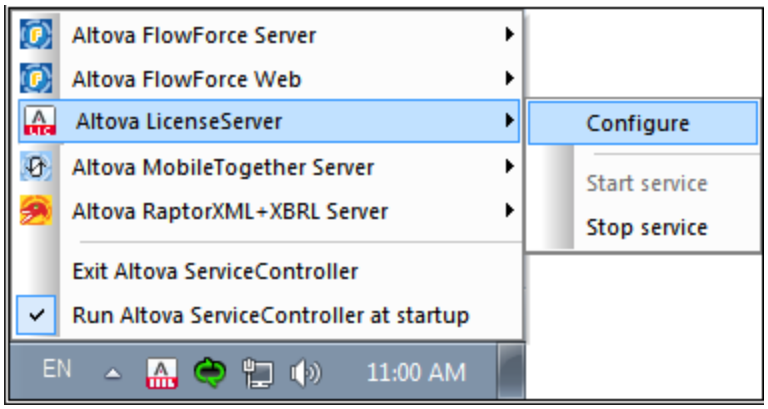
Pour spécifier que ServiceController démarre automatiquement lors de la connexion au système, cliquer sur l'icône **ServiceController** dans la zone de notification pour afficher le menu **ServiceController** (*voir la capture d'écran ci-dessous*) et basculez sur la commande **Run Altova ServiceController at Startup**. (Cette commande est activée par défaut.) Pour quitter ServiceController, cliquer sur l'icône **ServiceController** dans la zone de notification et, dans le menu qui apparaît (*voir la capture d'écran ci-dessous*), cliquez sur **Exit Altova ServiceController**.



Démarrer LicenseServer

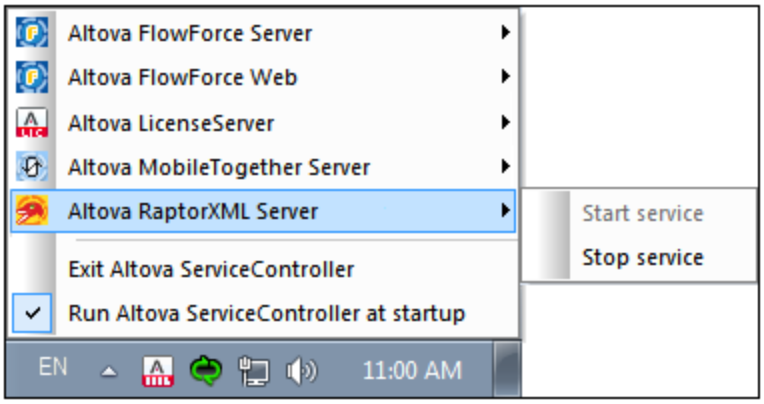
Pour démarrer LicenseServer, cliquez sur l'icône **ServiceController** dans la zone de modification, passez au-dessus de **Altova LicenseServer** dans le menu qui s'ouvre (*voir la capture d'écran ci-dessous*),puis sélectionnez **Start Service** depuis le sous-menu de LicenseServer. Si LicenseServer est déjà en cours

d'exécution, l'option *Démarrer le service* sera désactivée. Vous pouvez également arrêter le service par le biais de ServiceController.



Démarrer RaptorXML Server

Pour démarrer RaptorXML Server, cliquez sr l'icône **ServiceController** dans la zone de modification, passez au-dessus d'**Altova RaptorXML Server** dans le menu quii s'ouvre (*voir la capture d'écran ci-dessous*), puis sélectionnez **Start Service** depuis le sous-menu RaptorXML Server. Si RaptorXML Server est déjà en cours d'exécution, l'option *Démarrer le service* sera désactivée. Vous pouvez également arrêter le service par le biais de ServiceController.



Note : If RaptorXML Server a une licence pour exécuter uniquement des exécutions thread unique (généralement parce que votre appareil est multicœur, mais votre licence est single-core), alors vous pouvez utiliser uniquement une instance de RaptorXML Serverà la fois : soit en tant que service, soit depuis la ligne de commande. Ceci est dû au fait que la licence single-core sera attribuée automatiquement à la première instance qui est lancée et actuellement exécutée. La deuxième instance ne peut pas être lancée avant la fin de l'exécution de la première instance.

- Si vous souhaitez utiliser RaptorXML Server depuis la ligne de commande, mais que le service est en cours d'exécution, vous devez arrêter le service avant d'utiliser la ligne de commande.
- Si vous souhaitez lancer RaptorXML Server en tant que service, assurez-vous qu'aucune action de ligne de commande ne soit en cours d'exécution. Autrement, vous ne pourrez pas démarrer le service.

3.1.5.2 Enregistrer RaptorXML Server

Afin de pouvoir détenir une licence RaptorXML Server depuis Altova LicenseServer, RaptorXML Server doit être enregistré avec LicenseServer
Pour enregistrer RaptorXML Server depuis l'interface de la ligne de commande, utilisez la commande `licenseserver`et donnez l'adresse de l'appareil LicenseServer (*voir ci-dessous*).

```
RaptorXML licenseserver [options] ServerName-Or-IP-Address
```

Par exemple, si `localhost` est le nom du serveur sur lequel LicenseServer est installé , utilisez la commande suivante :

```
RaptorXML licenseserver localhost
```

Si RaptorXML Server a été installée faisant partie d'une installation de FlowForce Server, enregistrer FlowForce Server avec LicenseServer enregistrera aussi automatiquement RaptorXML Server. Procédez comme suit : (i) démarrez Altova FlowForce Web par le biais de ServiceController (*voir les points précédents*); (ii) saisissez votre mot de passe pour accéder à la page Setup; (iii) sélectionnez le nom ou l'adresse du LicenseServer et cliquez sur **Enregistrer avec LicenseServer**. Pour plus d'informations, voir Enregistrer FlowForce Server.

Après avoir réussi l'enregistrement, allez à l'onglet de gestion des clients de la page de configuration de LicenseServer pour attribuer une licence à RaptorXML Server.

Pour plus d'informations sur l'enregistrement des produits Altova avec LicenseServer, voir le manuel utilisateur de LicenseServer.

3.1.5.3 Attribuer des licences RaptorXML Server

Après avoir réussi l'inscription de RaptorXML Server, elle sera recensée dans l'onglet de gestion des clients de la page de configuration de LicenseServer. Allez-y et attribuez une licence à RaptorXML Server.

L'obtention de la licence des produits de serveur Altova, est basée sur le nombre de cœurs de processeurs disponibles sur la machine du produit. Par exemple, un processeur double cœur a deux cœurs, un processeur quadricœur a quatre cœurs, un processeur hexacœurs a six cœurs, etc. Le nombre de cœurs pour lesquels une licence a été délivrée pour un produit doit être supérieur ou égal au nombre de cœurs disponibles sur cette machine de serveur, que ce serveur soit une machine physique ou virtuelle. Par exemple, si un serveur a huit cœurs (un processeur octacœur), vous devrez acheter au moins une licence octacœur. Vous pouvez aussi additionner les licences pour obtenir le nombre souhaité de cœurs. Ainsi, deux licences de quadricœurs peuvent être utilisées pour un serveur octacœur au lieu d'acheter une licence octacœur.

Si vous utilisez un serveur d'ordinateur avec un grand nombre de cœurs CPU, mais ne disposez que d'un faible volume à traiter, vous pouvez aussi créer une machine virtuelle qui disposera d'un plus petit nombre de cœurs et acheter une licence pour ce nombre de cœurs. Il va de soi que la vitesse de traitement d'un tel déploiement sera moins rapide que si tous les cœurs disponibles sur le serveur étaient utilisés.

Note : chaque licence de produit de serveur Altova peut être utilisée pour une seule machine client à la fois, même si la licence a une capacité de licence qui n'est pas utilisée (l'appareil client est l'appareil sur lequel le produit de serveur Altova est installé). Par exemple, si une licence de 10-cœurs est utilisée pour une machine client qui détient 6 cœurs CPU, les 4 cœurs restants de la capacité de licence ne

pourront pas être utilisés simultanément pour une autre machine client.

Exécution thread unique
Si un produit de serveur Altova permet une exécution single-thread, une option pour une *exécution single-thread* sera disponible. Dans ces cas, la licence produit du serveur Altova pour uniquement un cœur est disponible dans le pool des licences, un appareil avec des cœurs multiples peut être assigné à cette licence à one-core. Dans un tel cas, l'appareil exécutera ce produit en single-core. Le traitement sera donc plus lent car le multi-threading (qui est possible sur de multiples cœurs) ne sera pas disponible. Le produit sera exécuté en mode single thread sur cet appareil.

Pour assigner une licence single-core à un appareil multiple-core dans LicenseServer, sélectionnez la case à cocher *Limit to single thread execution* pour ce produit.

Estimation des exigences core
Il existe de nombreux facteurs externes divers qui influent sur les volumes de données et les temps de traitement que votre serveur arrive à gérer (par exemple : le matériel, la charge actuelle sur le CPU, et l'attribution de la mémoire d'autres applications exécutées sur le serveur). Afin de mesurer la performance aussi précisément que possible, testez les applications dans votre environnement avec les volumes de données et les conditions qui établissent aussi fidèlement que possible des situations professionnelles réelles.

3.2 Configuration sur Linux

Cette section décrit l'installation 33 et la licence 36 de RaptorXML Server sur les systèmes Linux (Debian, Ubuntu, CentOS, RedHat).

Exigences de système (Linux)

- <% REDHAT%>
- <% CENTOS%>
- <% DEBIAN%>
- <% UBUNTU%>
- AlmaLinux 9.0
- Rocky Linux 9.0

Prérequis

- Réaliser l'installation soit en tant qu'utilisateur **root**, soit en tant qu'utilisateur avec des privilèges **sudo**.
- La version précédente de RaptorXML Server doit être désinstallée avant d'installer une nouvelle.
- Si vous avez l'intention d'utiliser la fonctionnalité des graphiques d'Altova, vous devez installer au moins une police sur votre système pour vous assurer que les graphiques seront rendus correctement. Pour recenser les polices installées, utilisez par exemple la commande **`fc-list`** de Fontconfig library.
- Les bibliothèques suivantes sont requises comme prérequis pour installer et exécuter l'application. Si les packages ci-dessous ne sont pas déjà disponibles sur l'appareil Linux, exécutez la commande `yum` (ou `apt-get`, si applicable) pour les installer.

CentOS, RedHat	Debian	Ubuntu
krb5-libs	libgssapi-krb5-2	libgssapi-krb5-2

3.2.1 Installer sur Linux

RaptorXML Server est disponible pour l'installation sur les systèmes Linux. Réaliser installation soit en tant qu'utilisateur `root`, soit en tant qu'utilisateur avec des privilèges `sudo`.

Intégration de FlowForce Server et d'autres produits de serveur d'Altova

Si vous installez RaptorXML Server ensemble avec FlowForce Server, il est recommandé d'installer d'abord FlowForce Server. Si vous installez RaptorXML Server avant FlowForce Server, il faut, après avoir installé RaptorXML Server et FlowForce Server, exécuter la commande suivante :

```
cp /opt/Altova/RaptorXMLServer2024/etc/*.tool /opt/Altova/FlowForceServer2024/tools
```

Cette commande copie le fichier **`.tool`** depuis le répertoire **`/etc`** de RaptorXML Server vers le répertoire FlowForce Server **`/tools`**. Le fichier **`.tool`** est requis par FlowForce Server. Il contient le chemin vers l'RaptorXML Server exécutable. Vous n'avez pas besoin d'exécuter cette commande si vous installez FlowForce Server avant d'installer RaptorXML Server.

Désinstaller RaptorXML Server

Avant d'installer RaptorXML Server, vous devez désinstaller toute version plus ancienne.

Pour vérifier quels produits de serveur Altova sont installés :

```
[Debian, Ubuntu] :   dpkg --list | grep Altova
[CentOS, RedHat] :   rpm -qa | grep server
```

Pour désinstaller une version plus ancienne de RaptorXML Server:

```
[Debian, Ubuntu] :   sudo dpkg --remove raptorxmlserver
[CentOS, RedHat] :   sudo rpm -e raptorxmlserver
```

Sur les systèmes Debian et Ubuntu, il se pourrait que RaptorXML Server apparaisse encore dans la liste des produits installés après son installation. Dans ce cas, exécutez la commande **purge** pour effacer RaptorXML Server de la liste. Vous pouvez aussi utiliser la commande **purge** *au lieu* de la commande **remove** recensée ci-dessus.

```
[Debian, Ubuntu] :   sudo dpkg --purge raptorxmlserver
```

Télécharger le package Linux RaptorXML Server

Les packages d'installation pour RaptorXML Server pour les systèmes Linux suivants sont disponibles sur le site web d'Altova.

Distribution	Extension de package
Debian	.deb
Ubuntu	.deb
CentOS	.rpm
RedHat	.rpm

Après avoir téléchargé le package Linux, copiez-le dans n'importe quel répertoire sur le système Linux. Puisque vous allez devoir mettre sous licence RaptorXML Server with an Altova LicenseServer, vous téléchargerez LicenseServer depuis le site web d'Altova en même temps que vous téléchargez RaptorXML Server, plutôt que de le télécharger plus tard.

Installer RaptorXML Server

Dans une fenêtre de terminal, basculez vers le répertoire où vous copiez le package Linux. Par exemple, si vous le copiez dans un répertoire d'utilisateur appelé `MyAltova` qui est situé dans le répertoire `/home/User`, basculez vers ce répertoire comme suit :

```
cd /home/User/MyAltova
```

Installer RaptorXML Server utilisant la commande pertinente :

```
[Debian] :   sudo dpkg --install raptorxml-2024-debian.deb
```

```
[Ubuntu] :   sudo dpkg --install raptorxml-2024-ubuntu.deb
[CentOS] :   sudo rpm -ivh raptorxml-2024-1.x86_64.rpm
[RedHat] :   sudo rpm -ivh raptorxml-2024-1.x86_64.rpm
```

Vous devrez éventuellement ajuster le nom du package ci-dessus pour qu'il corresponde au release ou à la version service pack actuels.

Le package RaptorXML Server sera installé dans le dossier suivant :

```
/opt/Altova/RaptorXMLServer2024
```

3.2.2 Installer LicenseServer (Linux)

Pour que RaptorXML Server fonctionne, il faut qu'il ait une licence par le biais d'un Altova LicenseServer sur votre réseau. Téléchargez LicenseServer depuis le site web d'Altova et copiez le package dans n'importe quel répertoire. Installez-le comme vous avez installé RaptorXML Server (*voir page précédente* 33).

```
[Debian] :   sudo dpkg --install licenseserver-3.14-debian.deb
[Ubuntu] :   sudo dpkg --install licenseserver-3.14-ubuntu.deb
[CentOS] :   sudo rpm -ivh licenseserver-3.14-1.x86_64.rpm
[RedHat] :   sudo rpm -ivh licenseserver-3.14-1.x86_64.rpm
```

Le pack de LicenseServer sera installé dans le chemin suivant :

```
/opt/Altova/LicenseServer
```

Pour plus d'information sur l'inscription et la licence RaptorXML Server avec Altova LicenseServer, voir la section Licence RaptorXML Server 36. Voir également la documentation LicenseServer pour des informations plus détaillées.

Versions de LicenseServer

- Les produits de Altova doivent être dotés d'une licence soit () avec une version de LicenseServer qui correspond à la version installée RaptorXML Server, soit pour une version ultérieure de LicenseServer.
- La version LicenseServer qui correspond à la version actuelle de RaptorXML Server est ***3.14***.
- Sur Windows, vous pouvez installer la version correspondante de LicenseServer comme faisant partie de l'installation de RaptorXML Server ou installez LicenseServer séparément. Sur Linux et macOS, vous devez installer LicenseServer séparément.
- Avant que la nouvelle version de LicenseServer ne soit installée, chaque version plus ancienne doit être désinstallée.
- Lors de la désinstallation de LicenseServer, toute l'information liée à l'inscription et à la licence contenue dans la version plus ancienne de LicenseServer sera enregistrée dans une base de données sur votre appareil de serveur. Ces données seront importées automatiquement dans la version plus nouvelle lorsque celle-ci est installée.
- Les versionsLicenseServer sont rétro-compatibles. Elles fonctionneront avec des versions plus anciennes de RaptorXML Server.
- La dernière version de LicenseServer disponible sur le site web d'Altova. Cette version fonctionnera avec toute version actuelle ou ancienne de RaptorXML Server.
- Le numéro de version de la licence LicenseServer actuellement installée est indiqué au bas de la page de configuration LicenseServer (tous les onglets).

3.2.3 Licence RaptorXML Server (Linux)

Afin d'utiliser RaptorXML Server, il doit être doté d'une licence de Altova LicenseServer. La gestion des licences est une procédure à deux étapes :

1. **Inscrire RaptorXML Server** avec LicenseServer. L'inscription est réalisée depuis RaptorXML Server.
2. **Attribuer une licence** à RaptorXML Server depuis LicenseServer. Télécharger la dernière version de LicenseServer depuis le site web d'Altova et installez-la sur votre appareil local ou un appareil sur votre réseau.

Ces deux étapes sont décrites dans cette section. Pour toute information détaillée, voir le manuel utilisateur LicenseServer sur le site web d'Altova.

3.2.3.1 Démarrer LicenseServer, RaptorXML Server

Démarrer Altova LicenseServer et RaptorXML Server soit comme utilisateur `root` ou comme utilisateur avec des privilèges `sudo`.

Démarrer LicenseServer

Pour s'enregistrer et gérer la licence RaptorXML Server avec LicenseServer, celui-ci doit être exécuté en tant que daemon sur le réseau. Démarrez LicenseServer en tant que daemon avec la commande suivante :

```
sudo systemctl start licenseserver
```

Si à un moment ou un autre, vous êtes amenés à devoir arrêter LicenseServer, remplacez **`start`** par **`arrêter`** dans les commandes ci-dessus. Par exemple :

```
sudo systemctl stop licenseserver
```

Démarrer RaptorXML Server

Démarrez RaptorXML Server en tant que daemon avec la commande suivante :

```
sudo systemctl start raptorxmlserver
```

Si à un moment ou un autre, vous êtes amenés à devoir arrêter RaptorXML Server, remplacez **`démarrer`** par **`arrêter`** dans les commandes ci-dessus. Par exemple :

```
sudo systemctl stop raptorxmlserver
```

Vérifier le statut de daemons

Pour vérifier si un daemon est exécuté, exécutez la commande suivante, remplaçant **`< servicename >`** avec le nom de daemon que vous voulez vérifier :

```
sudo service < servicename > status
```

3.2.3.2 Enregistrer RaptorXML Server

Afin de pouvoir détenir une licence RaptorXML Server depuis Altova LicenseServer, RaptorXML Server doit être enregistré avec LicenseServer
Pour enregistrer RaptorXML Server, utilisez la commande `licenseserver` :

```
sudo /opt/Altova/RaptorXMLServer2024/bin/raptorxml licenseserver [options] ServerName-Or-IP-Address
```

Par exemple, si `localhost` est le nom du serveur sur lequel LicenseServer est installé :

```
sudo /opt/Altova/RaptorXMLServer2024/bin/raptorxml licenseserver localhost
```

Dans la commande ci-dessus, `localhost` est le nom du serveur sur lequel LicenseServer est installé. Notez également que l'emplacement de l'RaptorXML Server exécutable est :

```
/opt/Altova/RaptorXMLServer2024/bin/
```

Après avoir réussi l'enregistrement, allez à l'onglet de gestion des clients de la page de configuration de LicenseServer pour attribuer une licence à RaptorXML Server.

Pour plus d'informations sur l'enregistrement des produits Altova avec LicenseServer, voir le manuel utilisateur de LicenseServer.

3.2.3.3 Licence RaptorXML Server

Après avoir réussi l'inscription de RaptorXML Server, elle sera recensée dans l'onglet de gestion des clients de la page de configuration de LicenseServer. Allez-y et attribuez une licence à RaptorXML Server.

L'obtention de la licence des produits de serveur Altova, est basée sur le nombre de cœurs de processeurs disponibles sur la machine du produit. Par exemple, un processeur double cœur a deux cœurs, un processeur quadricœur a quatre cœurs, un processeur hexacœurs a six cœurs, etc. Le nombre de cœurs pour lesquels une licence a été délivrée pour un produit doit être supérieur ou égal au nombre de cœurs disponibles sur cette machine de serveur, que ce serveur soit une machine physique ou virtuelle. Par exemple, si un serveur a huit cœurs (un processeur octacœur), vous devrez acheter au moins une licence octacœur. Vous pouvez aussi additionner les licences pour obtenir le nombre souhaité de cœurs. Ainsi, deux licences de quadricœurs peuvent être utilisées pour un serveur octacœur au lieu d'acheter une licence octacœur.

Si vous utilisez un serveur d'ordinateur avec un grand nombre de cœurs CPU, mais ne disposez que d'un faible volume à traiter, vous pouvez aussi créer une machine virtuelle qui disposera d'un plus petit nombre de cœurs et acheter une licence pour ce nombre de cœurs. Il va de soi que la vitesse de traitement d'un tel déploiement sera moins rapide que si tous les cœurs disponibles sur le serveur étaient utilisés.

Note : chaque licence de produit de serveur Altova peut être utilisée pour une seule machine client à la fois, même si la licence a une capacité de licence qui n'est pas utilisée (l'appareil client est l'appareil sur lequel le produit de serveur Altova est installé). Par exemple, si une licence de 10-cœurs est utilisée pour une machine client qui détient 6 cœurs CPU, les 4 cœurs restants de la capacité de licence ne

pourront pas être utilisés simultanément pour une autre machine client.

Exécution thread unique
Si un produit de serveur Altova permet une exécution single-thread, une option pour une *exécution single-thread* sera disponible. Dans ces cas, la licence produit du serveur Altova pour uniquement un cœur est disponible dans le pool des licences, un appareil avec des cœurs multiples peut être assigné à cette licence à one-core. Dans un tel cas, l'appareil exécutera ce produit en single-core. Le traitement sera donc plus lent car le multi-threading (qui est possible sur de multiples cœurs) ne sera pas disponible. Le produit sera exécuté en mode single thread sur cet appareil.

Pour assigner une licence single-core à un appareil multiple-core dans LicenseServer, sélectionnez la case à cocher *Limit to single thread execution* pour ce produit.

Estimation des exigences core
Il existe de nombreux facteurs externes divers qui influent sur les volumes de données et les temps de traitement que votre serveur arrive à gérer (par exemple : le matériel, la charge actuelle sur le CPU, et l'attribution de la mémoire d'autres applications exécutées sur le serveur). Afin de mesurer la performance aussi précisément que possible, testez les applications dans votre environnement avec les volumes de données et les conditions qui établissent aussi fidèlement que possible des situations professionnelles réelles.

3.3 Configuration sur macOS

Cette section décrit l'installation[39] et la licence[41] de RaptorXML Server sur les systèmes macOS.

Exigences de système (macOS)

Notez les exigences système suivantes :

- macOS 12 ou plus récent

Prérequis

Notez les exigences préalables suivantes :

- Assurez-vous d'avoir installé Altova LicenseServer et qu'il est exécuté.
- Réaliser l'installation soit en tant qu'utilisateur `root`, soit en tant qu'utilisateur avec des privilèges `sudo`.
- La version précédente de RaptorXML Server doit être désinstallée avant d'installer une nouvelle.
- Si vous avez l'intention d'utiliser la fonctionnalité des graphiques d'Altova, vous devez installer au moins une police sur votre système pour vous assurer que les graphiques seront rendus correctement. Pour recenser les polices installées, utilisez par exemple la commande **`fc-list`** de Fontconfig library.
- L'appareil macOS doit être configuré de telle façon que son nom se résout en adresse IP. Ceci signifie que vous devez être en mesure d'effectuer un ping avec succès du nom de l'hôte depuis le terminal en utilisant la commande **`ping <hostname>`**.

3.3.1 Installer sur macOS

Ce chapitre décrit l'installation et la configuration de RaptorXML Server sur les systèmes macOS.

Intégration dans FlowForce

Si vous installez RaptorXML Server ensemble avec FlowForce Server, il est recommandé d'installer d'abord FlowForce Server. Si vous installez RaptorXML Server avant FlowForce Server, il faut, après avoir installé, exécuter la commande suivante :

```
cp /usr/local/Altova/RaptorXMLServer2024/etc/*.tool /usr/local/Altova/FlowForceServer2024/tools
```

Cette commande copie le fichier **`.tool`** depuis le répertoire **`/etc`** de RaptorXML Server vers le répertoire FlowForce Server **`/tools`**. Le fichier **`.tool`** est requis par FlowForce Server. Il contient le chemin vers l'RaptorXML Server exécutable. Vous n'avez pas besoin d'exécuter cette commande si vous installez FlowForce Server avant d'installer RaptorXML Server.

Désinstaller RaptorXML Server

Avant de désinstaller RaptorXML Server, arrêtez le service avec la commande suivante :

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

Pour vérifier si le service a été arrêté, ouvrez le moniteur d'activités dans Finder et assurez-vous que RaptorXML Server n'est pas dans la liste. Dans le dossier Applications, cliquez avec la touche de droite sur l'icône RaptorXML Server et choisissez **Déplacer vers la Corbeille**. L'application sera déplacée dans la corbeille. Vous allez toutefois devoir déplacer l'application du dossier `usr`. Pour ce faire, utilisez la commande suivante :

```
sudo rm -rf /usr/local/Altova/RaptorXMLServer2024/
```

Si vous devez désinstaller une ancienne version de Altova LicenseServer, vous devez d'abord arrêter son exécution en tant que service. Pour ce faire, utilisez la commande suivante :

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

Pour vérifier si le service a été arrêté, ouvrez le moniteur d'activités et assurez-vous que LicenseServer n'est pas dans la liste. Puis, procédez à la désinstallation de la même manière que décrit ci-dessus pour RaptorXML Server.

Installer RaptorXML Server

Pour installer RaptorXML Server, suivez les étapes suivantes :

1. Téléchargez le fichier de l'image de disque (**`.dmg`**) de RaptorXML Server depuis le site Altova website (http://www.altova.com/download.html).
2. Cliquez pour ouvrir l'image de disque téléchargée (**`.dmg`**). Ceci fait que le programme d'installation RaptorXML Server apparaît en tant que nouveau lecteur virtuel sur votre ordinateur.
3. Sur le nouveau lecteur virtuel, double-cliquez sur le package d'installation (**`.pkg`**).
4. Suivez les étapes successives de l'assistant d'installation. Celles-ci sont explicites et incluent un étape lors de laquelle vous devez accepter le contrat de licence avant de pouvoir continuer. *Voir aussi* *Licensing RaptorXML Server* 41.
5. Pour éjecter le lecteur après l'installation, cliquez de la touche droite sur le lecteur et sélectionnez **Éjecter**.

Le package RaptorXML Server sera installé dans le dossier :

`/usr/local/Altova/RaptorXMLServer2024` (application binaries)
`/var/Altova/RaptorXMLServer` (data files : database and logs)

Le server daemonRaptorXML Server démarre automatiquement après l'installation et redémarre l'appareil. Vous pouvez toujours démarrer RaptorXML Server en tant que daemon avec les commandes suivantes :

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

3.3.2 Installer LicenseServer (macOS)

Altova LicenseServer peut être téléchargé depuis le site web d'Altova (http://www.altova.com/download.html). Exécutez l'installation tel que décrit ici 39.

Le package de LicenseServer sera installé dans le suivantes dossier :

`/usr/local/Altova/LicenseServer`

Pour toute information relative à l'enregistrement de RaptorXML Server avec Altova LicenseServer et à la licence, voir Licence sur macOS 41.

Versions de LicenseServer

- Les produits de Altova doivent être dotés d'une licence soit () avec une version de LicenseServer qui correspond à la version installée RaptorXML Server, soit pour une version ultérieure de LicenseServer.
- La version LicenseServer qui correspond à la version actuelle de RaptorXML Server est ***3.14***.
- Sur Windows, vous pouvez installer la version correspondante de LicenseServer comme faisant partie de l'installation de RaptorXML Server ou installez LicenseServer séparément. Sur Linux et macOS, vous devez installer LicenseServer séparément.
- Avant que la nouvelle version de LicenseServer ne soit installée, chaque version plus ancienne doit être désinstallée.
- Lors de la désinstallation de LicenseServer, toute l'information liée à l'inscription et à la licence contenue dans la version plus ancienne de LicenseServer sera enregistrée dans une base de données sur votre appareil de serveur. Ces données seront importées automatiquement dans la version plus nouvelle lorsque celle-ci est installée.
- Les versionsLicenseServer sont rétro-compatibles. Elles fonctionneront avec des versions plus anciennes de RaptorXML Server.
- La dernière version de LicenseServer disponible sur le site web d'Altova. Cette version fonctionnera avec toute version actuelle ou ancienne de RaptorXML Server.
- Le numéro de version de la licence LicenseServer actuellement installée est indiqué au bas de la page de configuration LicenseServer (tous les onglets).

3.3.3 Licence RaptorXML Server (macOS)

Afin d'utiliser RaptorXML Server, il doit être doté d'une licence de Altova LicenseServer. La gestion des licences est une procédure à deux étapes :

1. **Inscrire RaptorXML Server** avec LicenseServer. L'inscription est réalisée depuis RaptorXML Server.
2. **Attribuer une licence** à RaptorXML Server depuis LicenseServer. Télécharger la dernière version de LicenseServer depuis le site web d'Altova et installez-la sur votre appareil local ou un appareil sur votre réseau.

Ces deux étapes sont décrites dans cette section. Pour toute information détaillée, voir le manuel utilisateur LicenseServer sur le site web d'Altova.

3.3.3.1 Démarrer LicenseServer, RaptorXML Server

Démarrer Altova LicenseServer et RaptorXML Server soit comme utilisateur `root` ou comme utilisateur avec des privilèges `sudo`.

Démarrer LicenseServer

Pour s'enregistrer et gérer la licence RaptorXML Server correctement avec LicenseServer, celui-ci doit être exécuté en tant que daemon. Lancez LicenseServer en tant que daemon avec la commande suivante :

```
sudo launchctl load /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

Si à un moment ou un autre, vous êtes amenés à devoir arrêter LicenseServer, remplacez **charger** par **unload** dans les commandes ci-dessus.

Démarrer RaptorXML Server

Le server daemonRaptorXML Server démarre automatiquement après l'installation et redémarre l'appareil. Vous pouvez démarrer RaptorXML Server en tant que daemon avec les commandes suivantes :

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer.plist
```

Si à un moment donné ou un autre, vous devez arrêter RaptorXML Server, utilisez la commande suivante :

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer.plist
```

3.3.3.2 Enregistrer RaptorXML Server

Afin de pouvoir détenir une licence RaptorXML Server depuis Altova LicenseServer, RaptorXML Server doit être enregistré avec LicenseServer
Pour enregistrer RaptorXML Server depuis l'interface de ligne de commande, utilisez la commande `licenseserver` :

```
sudo /usr/local/Altova/RaptorXMLServer2024/bin/RaptorXML licenseserver [options]
ServerName-Or-IP-Address
```

Par exemple, si `localhost` est le nom du serveur sur lequel LicenseServer est installé :

```
sudo /usr/local/Altova/RaptorXMLServer2024/bin/RaptorXML licenseserver localhost
```

Dans la commande ci-dessus, `localhost` est le nom du serveur sur lequel LicenseServer est installé. Notez également que l'emplacement de l'RaptorXML Server exécutable est :

```
/usr/local/Altova/RaptorXMLServer2024/bin/
```

Après avoir réussi l'enregistrement, allez à l'onglet de gestion des clients de la page de configuration de LicenseServer pour attribuer une licence à RaptorXML Server.

Pour plus d'informations sur l'enregistrement des produits Altova avec LicenseServer, voir le manuel utilisateur de LicenseServer.

3.3.3.3 License RaptorXML Server

Après avoir réussi l'inscription de RaptorXML Server, elle sera recensée dans l'onglet de gestion des clients de la page de configuration de LicenseServer. Allez-y et attribuez une licence à RaptorXML Server.

L'obtention de la licence des produits de serveur Altova, est basée sur le nombre de cœurs de processeurs disponibles sur la machine du produit. Par exemple, un processeur double cœur a deux cœurs, un processeur quadricœur a quatre cœurs, un processeur hexacœurs a six cœurs, etc. Le nombre de cœurs pour lesquels

une licence a été délivrée pour un produit doit être supérieur ou égal au nombre de cœurs disponibles sur cette machine de serveur, que ce serveur soit une machine physique ou virtuelle. Par exemple, si un serveur a huit cœurs (un processeur octacœur), vous devrez acheter au moins une licence octacœur. Vous pouvez aussi additionner les licences pour obtenir le nombre souhaité de cœurs. Ainsi, deux licences de quadricœurs peuvent être utilisées pour un serveur octacœur au lieu d'acheter une licence octacœur.

Si vous utilisez un serveur d'ordinateur avec un grand nombre de cœurs CPU, mais ne disposez que d'un faible volume à traiter, vous pouvez aussi créer une machine virtuelle qui disposera d'un plus petit nombre de cœurs et acheter une licence pour ce nombre de cœurs. Il va de soi que la vitesse de traitement d'un tel déploiement sera moins rapide que si tous les cœurs disponibles sur le serveur étaient utilisés.

Note : chaque licence de produit de serveur Altova peut être utilisée pour une seule machine client à la fois, même si la licence a une capacité de licence qui n'est pas utilisée (l'appareil client est l'appareil sur lequel le produit de serveur Altova est installé). Par exemple, si une licence de 10-cœurs est utilisée pour une machine client qui détient 6 cœurs CPU, les 4 cœurs restants de la capacité de licence ne pourront pas être utilisés simultanément pour une autre machine client.

Exécution thread unique
Si un produit de serveur Altova permet une exécution single-thread, une option pour une *exécution single-thread* sera disponible. Dans ces cas, la licence produit du serveur Altova pour uniquement un cœur est disponible dans le pool des licences, un appareil avec des cœurs multiples peut être assigné à cette licence à one-core. Dans un tel cas, l'appareil exécutera ce produit en single-core. Le traitement sera donc plus lent car le multi-threading (qui est possible sur de multiples cœurs) ne sera pas disponible. Le produit sera exécuté en mode single thread sur cet appareil.

Pour assigner une licence single-core à un appareil multiple-core dans LicenseServer, sélectionnez la case à cocher *Limit to single thread execution* pour ce produit.

Estimation des exigences core
Il existe de nombreux facteurs externes divers qui influent sur les volumes de données et les temps de traitement que votre serveur arrive à gérer (par exemple : le matériel, la charge actuelle sur le CPU, et l'attribution de la mémoire d'autres applications exécutées sur le serveur). Afin de mesurer la performance aussi précisément que possible, testez les applications dans votre environnement avec les volumes de données et les conditions qui établissent aussi fidèlement que possible des situations professionnelles réelles.

3.4 Mise à jour RaptorXML Server

La manière la plus simple de reporter une licence depuis la version précédente de RaptorXML Server vers la version plus récente est par le biais du processus d'installation. Les étapes clés au cours de l'installation sont :

1. Enregistrer la nouvelle version de RaptorXML Server avec le serveur de licence qui détient la licence utilisée par l'ancienne version de RaptorXML Server.
2. Acceptez le contrat de licence de RaptorXML Server. (Si vous n'acceptez pas le contrat, la nouvelle version ne sera pas installée.)

Note : Si vous n'inscrivez pas RaptorXML Server avec LicenseServer au cours du processus d'installation, vous pouvez faire ceci plus tard et ensuite compléter le processus de licence.

3.5 Migrer RaptorXML Server vers un nouvel appareil

Si vous voulez migrer RaptorXML Server depuis un appareil vers un autre (y compris sur des plateformes prises en charge), suivez les directives ci-dessous.

Migrer RaptorXML Server vers un nouvel appareil consiste à réattribuer la licence depuis l'ancien appareil vers le nouveau. Pour ce faire, procédez comme suit :

1. Installez RaptorXML Server sur votre nouvel appareil. Si l'installation a déjà été réalisée en tant que partie de l'installation de FlowForce Server, ignorez cette étape.
2. Sur le nouvel appareil, enregistrez RaptorXML Server avec Altova LicenseServer.
3. Sur l'ancien appareil, assurez-vous qu'aucun client n'utilise le serveur.
4. Ouvrir la page d'administration de Altova LicenseServer. Désactivez la licence de l'ancien appareil RaptorXML Server et réattribuez-la au nouvel appareil.

Note : Migrez le fichier de configuration du serveur afin de garder vos paramètres de configuration précédents.

Note : Si vous êtes en train d'utiliser des catalogues XML sur les anciens appareils, migrez-les sur votre nouvel appareil.

4 Procédures générales

RaptorXML propose des options spéciales qui prennent en compte les Catalogues XML 47 et les ressources globales Altova 54, tous deux augmentent la portabilité et la modularité. Vous pouvez donc mettre à profit l'utilisation de ces fonctions dans votre environnement.

Cette section décrit les rubriques suivantes :

- Comment utiliser les Catalogues XML 47.
- Comment travailler avec les ressources globales Altova 54.
- Les problèmes de sécurité 56 liés aux procédures de RaptorXML et comment les gérer.

4.1 Catalogues XML

Le mécanisme de catalogue XML permet aux fichiers d'être extraits de dossiers locaux, donc augmentant la vitesse de traitement général, ainsi qu'améliorant la portabilité des documents—puisqu'uniquement les URI de fichiers catalogue doivent être modifiés. Voir la section Le fonctionnement des catalogues 47 pour tout détail.

Les produits XML d'Altova utilisent un mécanisme de catalogue pour accéder rapidement à et charger des fichiers communément utilisés, tels que les DTD et Schémas XML. Ce mécanisme de catalogue peut être personnalisé et étendu par l'utilisateur et est décrit dans la section Structure du catalogue dans RaptorXML Server 49 et Personnaliser vos catalogues 50. La section Variables pour les emplacements de Système 52 rassemble les variables Windows pour les emplacements de systèmes communs. Ces variables peuvent être utilisées dans les fichiers de catalogue pour situer des dossiers utilisés communément.

Cette section est organisée dans les sous-sections suivantes :

- Comment fonctionnent les catalogues 47
- Structure du catalogue RaptorXML Server 49
- Personnaliser vos catalogues 50
- Variables pour les emplacements de Système 52

Pour plus d'informations sur les catalogues, voir la spécification de catalogues XML.

Installer des schémas par le biais du Gestionnaire de schéma

Le Gestionnaire de schéma 357 vous permet d'installer rapidement et en toute facilité d'importants schémas et de configurer des fichiers catalogue pour accéder correctement aux schémas installés. Voir la section Gestionnaire de schéma 357 pour plus d'informations.

Si un document est validé par rapport à un schéma qui n'est pas installé, mais disponible par le biais du Gestionnaire de schéma 357, alors l'installation par le biais du Gestionnaire de schéma sera déclenchée automatiquement. Toutefois, si le pack de schéma à installer par le biais du Gestionnaire de schéma contient des mappages d'espace de noms, alors il n'y aura pas d'installation automatique ; dans ce cas, vous devez lancer Gestionnaire de schéma, sélectionner le/s pack/s que vous voulez installer et exécuter l'installation. Si, après l'installation, RaptorXML Server n'est pas capable de localiser correctement un composant de schéma, alors redémarrez RaptorXML Server et réessayez.

4.1.1 Le fonctionnement des catalogues

Les catalogues peuvent être utilisés pour rediriger les Schémas DTD et XML. Alors que le concept derrière les mécanismes dans les deux cas est le même, les détails sont différents et expliqués ci-dessous.

DTD

Les catalogues sont communément utilisés pour rediriger un appel vers un DTD ou un URI local. Pour ce faire, des identifiants publics ou système sont mappés dans le fichier catalogue vers l'URI local requis. Donc, si la déclaration `DOCTYPE` dans un fichier XML est lue, son identifiant public ou système localise la ressource locale requise par le biais du mappage du fichier catalogue.

Pour les schémas populaires, l'identifiant `PUBLIC` est normalement prédéfini, requérant uniquement que l'URI

dans le fichier catalogue mappe l'identifiant **PUBLIC** à la copie locale correcte. Lorsque le document XML est parsé, l'identifiant **PUBLIC** qui le compose est lu. Si cet identifiant est trouvé dans un fichier catalogue, l'URL correspondant dans le catalogue fichier sera consulté et le schéma sera lu depuis cet emplacement. Donc, par exemple, si le fichier SVG est ouvert dans RaptorXML Server :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="20" height="20" xml:space="preserve">
  <g style="fill:red; stroke:#000000">
     <rect x="0" y="0" width="15" height="15"/>
     <rect x="5" y="5" width="15" height="15"/>
  </g>
</svg>
```

L'identifiant **PUBLIC** de ce fichier SVG file est recherché dans le catalogue. Disons que le fichier catalogue contient l'entrée suivante :

```
catalog>
  ...
    <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

Dans ce cas, il y a une correspondance pour l'identifiant **PUBLIC**. En conséquence, le lookup pour SVG DTD est redirigé vers l'URL **schemas/svg/svg11.dtd** (qui est associé au fichier catalogue). Il s'agit d'un fichier local qui sera utilisé en tant que DTD pour le fichier SVG. S'il n'y a pas de mappage pour l'ID **Public** dans le catalogue, l'URL dans le document XML sera utilisé (dans l'exemple du fichier SVG ci-dessus, l'URL Internet est la suivante : **http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd**).

Schémas XML

Dans RaptorXML Server, vous pouvez utiliser des catalogues avec des **Schémas XML**. Dans le fichier d'instance XML, la référence au schéma apparaîtra dans l'attribut **xsi:schemaLocation** de l'élément de premier niveau du document XML. Par exemple,

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

La valeur de l'attribut **xsi:schemaLocation** a deux parties : une partie d'espace de noms (vert ci-dessus) et une partie URI (en surbrillance). La partie d'espace de noms est utilisée dans le catalogue pour effectuer le mappage vers la ressource alternative. Par exemple, la saisie catalogue suivante redirige la référence du schéma ci-dessus vers un schéma à un emplacement alternatif.

```
<uri name="http://www.xmlspy.com/schemas/orgchart" uri="C:\MySchemas\OrgChart.xsd"/>
```

Normalement, la partie URI de la valeur de l'attribut **xsi:schemaLocation** est le chemin vers l'emplacement actuel du schéma. Toutefois, si le schéma est référencé par le biais du catalogue, la partie URI doit pointer vers le schéma XML actuel mais doit exister pour que la validité lexicale de l'attribut **xsi:schemaLocation** soit maintenu. Une valeur `foo`, par exemple, suffirait à la partie URI de la valeur de l'attribut pour qu'elle soit valide.

4.1.2 Structure du catalogue dans RaptorXML Server

Lorsque RaptorXML Server est lancé, il charge un fichier désigné **RootCatalog.xml** (*la structure est affichée dans la liste ci-dessous*), qui contient une liste des fichiers catalogue qui seront consultés. Vous pouvez modifier ce fichier et saisir autant de fichiers catalogue que vous souhaitez consulter, chacun est référencé dans un élément **nextCatalog**. Ces fichiers catalogue sont consultés et les URI contenus sont résolus conformément à leurs mappages.

Liste du RootCatalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
   xmlns:spy="http://www.altova.com/catalog_ext"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/CustomCatalog.xml"/>
  <!-- Inclut tous les catalogues sous les schémas communs du premier répertoire dans
l'arborescence -->
  <nextCatalog spy:recurseFrom="%CommonSchemasFolder%" catalog="catalog.xml"
spy:depth="1"/>
  <nextCatalog spy:recurseFrom="%ApplicationWritableDataFolder%/pkgs/.cache"
catalog="remapping.xml" spy:depth="0"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
</catalog>
```

La liste des références ci-dessus renvoie à un catalogue personnalisé (désigné **CustomCatalog.xml**) et à un ensemble de catalogues qui localisent des schémas communément utilisés (tels que les Schémas W3C XML et le schéma SVG).

- **CustomCatalog.xml** est situé dans le sous-dossier RaptorXML Server du dossier d'application **etc**. Vous devez le créer depuis un modèle de fichier appelé **CustomCatalog_template.xml**. CustomCatalog.xml est fichier squelette dans lequel vous pouvez créer vos propres mappages. Vous pouvez ajouter des mappages au **CustomCatalog.xml** pour chaque schéma dont vous avez besoin qui n'est pas adressé par les fichiers catalogue dans le dossier des Schémas communs. Pour ce faire, utilisez les éléments pris en charge par le mécanisme de catalogue OASIS (*voir prochaine section*).
- Le dossier des Schémas communs (localisé par la variable **%CommonSchemasFolder%**) contient un ensemble de schémas communément utilisés. À l'intérieur de chaque dossier de schéma, il y a un fichier **catalog.xml** qui mappe des identifiants publics et/ou de système aux URI qui dirigent vers des copies enregistrées localement des schémas respectifs.
- **CoreCatalog.xml** est situé dans le dossier d'application in the RaptorXML Server et est utilisé pour localiser les schémas et les feuilles de style utilisés par des processus spécifiques de RaptorXML Server, tels que StyleVision Power Stylesheets qui sont des feuilles de style utilisées pour générer le mode Authentique des documents XML de Altova.

Veuillez noter les points suivants :

- Lors d'une nouvelle installation de la même version majeure (mêmes versions ou différentes versions mineures), le modèle de fichier sera remplacé par un nouveau modèle de fichier, mais **CustomCatalog.xml** ne sera pas touché.
- Toutefois, si vous installez une nouvelle version majeure au-dessus de la version majeure précédente, alors le dossier de la version majeure précédente sera supprimé - ensemble avec son **CustomCatalog.xml**. Donc, si vous voulez continuer à utiliser **CustomCatalog.xml**, assurez-vous

d'enregistrer **CustomCatalog.xml** depuis le dossier de la version majeure à un endroit sûr. Une fois que la nouvelle version majeure a été installée, vous pouvez copier **CustomCatalog.xml** que vous avez enregistré dans le dossier **etc** de la nouvelle version majeure et l'éditer là tel que requis.

Variables d'emplacement
Les variables utilisées dans **RootCatalog.xml** (*liste ci-dessus*) ont les valeurs suivantes :

%PersonalFolder%	Dossier personnel de l'utilisateur actuel, par exemple C:\Users\<name>\Documents
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2024\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova

Emplacement des fichiers catalogue et des schémas
Veuillez noter l'emplacement des différents fichiers catalogue.

- **RootCatalog.xml**, **CustomCatalog.xml**, **CustomCatalog_template.xml**, et **CoreCatalog.xml** sont dans le dossier d'application RaptorXML Server.
- Les fichiers **catalog.xml** sont chacun dans un dossier de schéma spécifique, ces dossiers de schéma étant dans le dossier des Schémas communs.

4.1.3 Personnaliser vos catalogues

Lorsque vous créez des entrées dans **CustomCatalog.xml** (ou tout autre fichier catalogue qui doit être lu par RaptorXML Server), utilisez uniquement les éléments suivants de la spécification de catalogue OASIS. Chacun des éléments ci-dessous est répertorié avec une explication de leurs valeurs attribut. Pour plus d'informations sur les catalogues, voir la spécification des catalogues XML. Notez que chaque élément peut prendre l'attribut **xml:base**, qui est utilisé pour spécifier la base URI de cet élément.

- `<public publicId="PublicID of Resource" uri="URL of local file"/>`
- `<system systemId="SystemID of Resource" uri="URL of local file"/>`
- `<uri name="filename" uri="URL of file identified by filename"/>`
- `<rewriteURI uriStartString="StartString of URI to rewrite" rewritePrefix="String to replace StartString"/>`
- `<rewriteSystem systemIdStartString="StartString of SystemID" rewritePrefix="Replacement string to locate resource locally"/>`

Veuillez noter les points suivants :

- Dans le cas où il n'y a pas d'identifiant public, comme pour toutes les feuilles de style, l'identifiant système peut être directement mappé avec un URL par le biais de l'élément **système**.
- Un URI peut être mappé avec un autre URI en utilisant l'élément `uri`.
- Les éléments `rewriteURI` et `rewriteSystem` permettent la réécriture respectivement de la partie initiale d'un URI ou d'un identifiant système. Ceci permet de lancer un chemin de fichier à remplacer et, par conséquent, permet de cibler un autre répertoire. Pour plus d'informations sur les éléments, voir la spécification des catalogues XML.

À partir de la version 2014, RaptorXML Server adhère étroitement à la spécification [spécification des catalogues XML (OASIS Standard V1.1, 7 octobre 2005)]. Cette spécification sépare strictement les look-up d'identifiants externes (ceux avec une ID Publique ou une ID Système) des look-up URI (les URI qui sont pas des ID Publiques ou des ID Système). Les URI d'espace de noms doivent donc être considérés comme étant des URI simples —et pas des ID Publiques ou des ID Système—et doivent être utilisés en tant que look-up URI plutôt que des look-up d'identifiants externes. Dans les versions RaptorXML Server antérieures à la version 2014, les URI d'espace de noms ont été traduits par les mappages **<public>**. À partir de la version 2014, les mappages **<uri>** doivent être utilisés.

Avant v2014 :
```
<public publicID="http://www.MyMapping.com/ref"
    uri="file:///C:/MyDocs/Catalog/test.xsd"/>
```
à partir de V-2014 :
```
<uri name="http://www.MyMapping.com/ref"
    uri="file:///C:/MyDocs/Catalog/test.xsd"/>
```

Comment RaptorXML Server trouve un schéma référencé

Un schéma est référencé dans un document XML par le biais de l'attribut **xsi:scemaLocation** (*voir ci-dessous*). La valeur de l'attribut **xsi:schemaLocation** a deux parties : une partie d'espace de noms (vert) et une partie URI (en surbrillance).

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

L'ensemble d'étapes à suivre pour trouver un schéma référencé dépend des options de validation **--schemalocation-hints** et **--schema-mapping**. Veuillez trouver ci-dessous les procédures pour chaque valeur des deux options :

- **--schemalocation-hints=load-by-schemalocation | load-by-namespace | load-combining-both | ignore**
  Spécifie le comportement des attributs **xsi:schemaLocation** et **xsi:noNamespaceSchemaLocation** : pour déterminer s'il est nécessaire de charger un document de schéma et si oui, quelle information devrait être utilisée pour le trouver ; (par défaut est le suivant : **load-by-schemalocation**).
  - **load-by-schemalocation**
    1. Si la partie URI de **xsi:schemaLocation** est mappée dans un catalogue, téléchargez l'URI qui en résulte
    2. Télécharger l'URI directement
  - **load-by-namespace**
    1. Si la partie espace de noms de **xsi:schemaLocation** est mappée dans un catalogue, téléchargez l'URI qui en résulte.
    2. Rien télécharger.
  - **load-combining-both**
    1. Si la partie URI de **xsi:schemaLocation** est mappée dans un catalogue, téléchargez l'URI qui en résulte.
    2. Si la partie espace de noms de **xsi:schemaLocation** est mappée dans un catalogue, téléchargez l'URI qui en résulte.
    3. Télécharger la partie URI directement.

- **--schema-mapping=prefer-schemalocation | prefer-namespace**
  Si l'emplacement et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, alors cette option spécifie lequel des deux doit être utilisé de préférence pendant la consultation du catalogue ; (par défaut c'est **prefer-schemalocation**). Cette option est utilisée

pour cahnger l'ordre des deux premières étapes dans le variant **load-combining-both** ci-dessus.

Spécifications de schéma XML

L'information de spécification de schéma XML est prédéfinie dans RaptorXML Server et la validité des documents de schéma XML (`.xsd`) est comparée à l'information interne. Pour cela, dans un document de schéma XML, il ne devrait pas y avoir de références faites à n'importe quel schéma qui définit la spécification de schéma XML.

Le fichier **catalog.xml** dans le dossier `%AltovaCommonSchemasFolder%\Schemas\schema` contient des références aux DTD qui implémentent des spécifications de schéma XML antérieures. Vous ne devriez pas valider vos documents de schéma XML par rapport à ces schémas. Les fichiers référencés sont inclus uniquement pour donner à RaptorXML Server des informations sur les assistants de saisie à des fins d'édition si vous vouliez créer des documents conformément à ces recommandations antérieures.

4.1.4 Variables pour les emplacements de Système

Les variables d'environnement shell peuvent être utilisées dans l'élément `nextCatalog` pour spécifier le chemin menant aux différents emplacements de système (*voir RootCatalog.xml liste ci-dessus*). Les variables d'environnement shell suivantes sont prises en charge :

`%PersonalFolder%`	Chemin complet vers le dossier Personnel de l'utilisateur actuel, par exemple `C:\Users\<name>\Documents`
`%CommonSchemasFolder%`	`C:\ProgramData\Altova\Common2024\Schemas`
`%ApplicationWritableDataFolder%`	`C:\ProgramData\Altova`
`%AltovaCommonFolder%`	C:\Program Files\Altova\Common2024
`%DesktopFolder%`	Chemin complet vers le dossier Bureau de l'utilisateur actuel.
`%ProgramMenuFolder%`	Chemin complet vers le dossier Menu Programme pour l'utilisateur actuel.
`%StartMenuFolder%`	Chemin complet vers le dossier Démarrage pour l'utilisateur actuel.
`%StartUpFolder%`	Chemin complet vers le dossier Démarrage pour l'utilisateur actuel.
`%TemplateFolder%`	Chemin complet vers le dossier Modèle pour l'utilisateur actuel.
`%AdminToolsFolder%`	Chemin complet vers le répertoire de système de fichier qui stocke des outils administratifs pour l'utilisateur actuel.
`%AppDataFolder%`	Chemin complet vers le dossier Données d'Application pour l'utilisateur actuel.
`%CommonAppDataFolder%`	Chemin complet vers le répertoire de fichier contenant les données d'application pour tous les utilisateurs.
`%FavoritesFolder%`	Chemin complet du dossier Favoris pour l'utilisateur actuel.
`%PersonalFolder%`	Chemin complet vers le dossier Personnel pour l'utilisateur actuel.

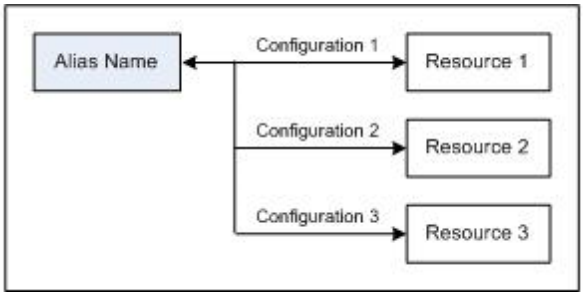
%SendToFolder%	Chemin complet vers le dossier EnvoyerÀ pour l'utilisateur actuel.
%FontsFolder%	Chemin complet vers le dossier Polices Système.
%ProgramFilesFolder%	Chemin complet vers le dossier Fichiers Programme pour l'utilisateur actuel.
%CommonFilesFolder%	Chemin complet vers le dossier Fichiers Communs pour l'utilisateur actuel.
%WindowsFolder%	Chemin complet vers le dossier Windows pour l'utilisateur actuel.
%SystemFolder%	Chemin complet vers le dossier Système pour l'utilisateur actuel.
%LocalAppDataFolder%	Chemin complet vers le répertoire de fichier système qui sert en tant qu'archivage de données pour les applications locales (nonroaming).
%MyPicturesFolder%	Chemin complet vers le dossier MesPhotos.

4.2 Ressources globales

Cette section :

À propos des Ressources globales

Un fichier de ressource globale Altova mappe un alias vers des ressources multiples par le biais de configurations différentes, comme indiqué dans le diagramme ci-dessous. Un alias peut donc être changé pour accéder à une ressource différence en changeant sa configuration.



Les ressources globales sont définies dans des produits Altova, comme Altova XMLSpy, et sont enregistrées dans un fichier XML de ressources globales. RaptorXML est en mesure d'utiliser les ressources globales en tant qu'entrées. Pour ce faire, il nécessite le nom et l'emplacement des fichiers de ressources globales et l'alias et la configuration à utiliser.

L'avantage d'utiliser des ressources globales est que la ressource peut être modifiée simplement en changeant le nom de la configuration. Lorsque vous utilisez RaptorXML, cela signifie qu'en fournissant une valeur différente de l'option `--globalresourcesconfig | --gc`, une ressource différente peut être utilisée. (*Voir l'exemple ci-dessous.*)

Utiliser les Ressources globales avec RaptorXML

Pour spécifier une ressource globale en tant qu'une entrée pour une commande RaptorXML, les paramètres suivants sont exigés :

- Le fichier XML de ressources globales (spécifié dans la CLI avec l'option `--globalresourcesfile | --gr`)
- La configuration exigée (spécifiée dans la CLI avec l'option `--globalresourcesconfig | --gc`)
- L'alias. Cela peut être spécifié directement dans la CLI où un nom de fichier est requis, ou il peut se trouver à un emplacement dans un fichier XML où RaptorXML cherche le nom d'un fichier (comme dans un attribut `xsi:schemaLocation`).

Par exemple, si vous souhaitez transformer `input.xml` avec `transform.xslt` en `output.html`, cela s'effectuera généralement sur la CLI avec la commande suivante qui utilise les noms de fichier :

```
raptorxml xslt --input=input.xml --output=output.html transform.xslt
```

Si, néanmoins, vous détenez une définition de ressource globale qui correspond à l'alias `MyInput` à la ressource de fichier `FirstInput.xml` comme une configuration appelée `FirstConfig`, alors vous pourriez utiliser l'alias `MyInput` sur la CLI comme suit :

```
raptorxml xslt --input=altova://file_resource/MyInput --gr=C:\MyGlobalResources.xml --
gc=FirstConfig --output=Output.html transform.xslt
```

Maintenant, si vous avez une autre ressource de fichier, comme par exemple `SecondInput.xml`, qui correspond à l'alias `MyInput` par le biais d'une configuration appelée `SecondConfig`, alors cette ressource peut être utilisée en ne changeant que l'option `--gc` de la commande précédente :

```
raptorxml xslt --input=altova://file_resource/MyInput --gr=C:\MyGlobalResources.xml --
gc=SecondConfig --output=Output.html transform.xslt
```

Note : Dans l'exemple ci-dessus, une ressource de fichier a été utilisée ; une ressource de fichier doit être préfixée par **`altova://file_resource/`**. Vous pouvez aussi utiliser des ressources globales qui sont des dossiers. Pour identifier une ressource de dossier, utiliser : **`altova://folder_resource/AliasName`**. Veuillez noter que, sur la CLI, vous pouvez aussi utiliser des ressources de dossier en tant que partie d'un chemin de fichier. Par exemple : **`altova://folder_resource/AliasName/input.xml`**.

4.3 Problèmes de sécurité

Cette section :

Certaines fonctions d'interface de RaptorXML Server posent des problèmes de sécurité. Ceux-ci sont décrits ci-dessous avec leurs solutions.

Problèmes de sécurité liés à l'interface REST HTTP

L'interface REST HTTP, par défaut, permet aux documents de résultat d'être rédigés sur tout emplacement spécifié par le client (qui est accessible avec le protocole HTTP). C'est pourquoi il est important de considérer ces aspects de sécurité lors de la configuration de RaptorXML Server.

Si vous estimez que la sécurité ait pu être compromise ou que l'interface ait pu être utilisée à mauvais escient, le serveur peut être configuré pour écrire les documents de résultat dans un répertoire de sortie réservé sur le serveur lui-même. Cela est spécifié en configurant l'option `server.unrestricted-filesystem-access` 233 du fichier de configuration du serveur sur `false`. Lorsque l'accès est limité de cette manière, le client peut télécharger les documents de résultat depuis le répertoire de sortie réservé avec des requêtes `GET`. En alternative, un administrateur peut copier/charger les fichiers de document de résultat depuis le serveur vers l'emplacement cible.

Rendre les scripts Python sûrs

Lorsqu'un script Python est spécifié dans une commande via HTTP pour RaptorXML Server, le script fonctionnera uniquement s'il est situé dans le répertoire de confiance 233. Le script est exécuté depuis le répertoire de confiance. La spécification d'un script Python provenant d'un autre répertoire quelconque résultera en une erreur. Le répertoire de confiance est spécifié dans le paramètre `server.script-root-dir` 232 du fichier de configuration de serveur 231, et un répertoire de confiance **doit** être spécifié si vous souhaitez utiliser des scripts Python. Veuillez vous assurer que tous les scripts Python à utiliser soient sauvegardés dans ce répertoire.

Bien que toutes les sorties générées par le serveur pour les requêtes de tâche HTTP soient écrites dans le répertoire de sortie de tâche 233 (qui est un sous-répertoire de `output-root-directory` 233), cette limitation ne s'applique pas aux scripts Python, qui peuvent écrire dans n'importe quel emplacement. L'administrateur de serveur doit revoir les scripts Python dans le répertoire de confiance 233 en ce qui concerne des problèmes de vulnérabilité potentiels.

5 Interface de ligne de commande (CLI)

L'exécutable RaptorXML Server propose une fonction d'application qui peut être appelée depuis l'interface de commande de ligne (CLI). Le chemin vers l'exécutable est :

Linux	`/opt/Altova/RaptorXMLServer2024/bin/`**`raptorxml`**
Mac	`/usr/local/Altova/RaptorXMLServer2024/bin/`**`raptorxml`**
Windows	`<ProgramFilesFolder>\Altova\RaptorXMLServer2024\bin\`**`RaptorXML.exe`**

Utilisation

La syntaxe de ligne de commande est :

```
raptorxml --h | --help | --version | <command> [options] [arguments]
```

- `--help` (short form `--h`) affiche le texte d'aide d'une commande donnée. Si aucune commande n'est nommée, alors toutes les commandes du programme d'exécution sont recensées, chacune avec une brève description de la commande.
- `--version` affiche le numéro de version de RaptorXML Server.
- `<command>` est la commande à exécuter. Les commandes sont décrites dans les sous-sections de cette section *(voir liste ci-dessous)*.
- `[options]` sont les options d'une commande ; ils sont listés et décrits avec leurs commandes respectives.
- `[arguments]` sont les arguments d'une commande ; ils sont listés et décrits avec leurs commandes respectives.

▼ Casse et barres obliques sur la ligne de commande

`RaptorXML(` et **`RaptorXMLServer`** pour des commandes d'administration) *sur Windows*
`raptorxml(` et **`raptorxmlserver`** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Commandes CLI

Les commandes CLI disponibles sont décrites dans cette section et organisées par fonctions.

5.1 Commande de validation XML, DTD, XSD

Les commandes de validation XML peuvent être utilisées pour valider les types de document suivantes :

- valxml-withdtd[59] : Valide un document XML par rapport à un DTD.
- valxml-withxsd[63] : Valide un document d'instance XML par rapport au Schéma XML.
- valdtd[70] : Valide un document document DTD
- valxsd[74] : Valide un document (XSD)un document de Schema XML W3C .

5.1.1 valxml-withdtd (xml)

La commande **`valxml-withdtd | xml`** valide un ou plusieurs documents d'instance XML par rapport à un DTD.

```
raptorxml valxml-withdtd | xml [options] InputFile
```

- L'argument *`InputFile`* est le document XML à valider. Si une référence à un DTD existe dans le document XML, l'option `--dtd` n'est pas requise.
- Pour valider de multiples documents, soit : (i) lister les fichiers à valider sur la CLI, chaque fichier étant séparé de l'autre par un espace ; ou (ii) lister les fichiers à valider dans un fichier de texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *`InputFile`* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).

Exemples

Exemples de la commande **`valxml-withdtd`** :

- **`raptorxml`** `valxml-withdtd --dtd=c:\MyDTD.dtd c:\Test.xml`
- **`raptorxml`** `xml c:\Test.xml`
- **`raptorxml`** `xml --verbose=true c:\Test.xml`
- **`raptorxml`** `xml --listfile=true c:\FileList.txt`

▼ Casing and slashes on the command line

`RaptorXML` (and **`RaptorXMLServer`** for administration commands) *on Windows*
`raptorxml` (and **`raptorxmlserver`** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **`"My File"`**. Note, however, that a backslash followed by a double-quotation mark (for example, **`"C:\My directory\"`**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **`\"`** stands for the double-quotation mark character. If you want to

escape this sequence of characters, use a preceding backslash, like this: `\\"`. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: `"C:\My Directory\ \"`.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Validation et traitement
  - dtd

    `--dtd =` *`FILE`*
    Spécifie le document DTD externe à utiliser pour la validation. Si une référence à un DTD externe est présente dans le document XM, alors l'option CLI contourne la référence externe.

  - listfile

    `--listfile =` `true|false`
    Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - namespaces

    `--namespaces =` `true|false`
    Permet le traitement sensible aux espaces de noms. Cela est utile pour contrôler l'instance XML pour des erreurs dues à des espaces de noms incorrectes. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - recurse

    `--recurse =` `true|false`
    Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une

valeur.

- streaming

  --streaming = `true|false`
  Permet la validation de streaming. La valeur par défaut est `true`. Dans le mode streaming, les données stockées dans la mémoire sont minimisées et le traitement est plus rapide. L'inconvénient est que l'information que peut être requise par conséquent, par exemple, un modèle de données du document d'instance XML —ne sera pas disponible. Dans des situations où cela est significatif, le mode de streaming devra être éteint (en donnant à `--streaming` une valeur de `false`). En utilisant l'option `--script` avec la commande `valxml-withxsd`, désactiver le streaming. Veuillez noter que l'option `--streaming` est ignorée si`--parallel-assessment` est défini sur `true`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Catalogues et ressources globales
  - catalog

    --catalog = *FILE*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    --user-catalog = *FILE*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Active les ressources globales 54. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALUE*
    Spécifie la configuration active de la ressource globale 54 (et active les ressources globales) 54.

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *FILE*
    Spécifie le fichier de ressource globale 54 (et active les ressources globales) 54.

- Messages, erreurs, aide, timeout, version
  - error-format

--error-format = `text|shortxml|longxml`
Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

- error-limit

  --error-limit = *N | unlimited*
  Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

- info-limit

  --info-limit = *N | unlimited*
  Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  --help
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  --log-output = *FILE*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.1.2 valxml-withxsd (xsi)

La commande **`valxml-withdtd | xsi`** valide un ou plusieurs documents d'instance XML par rapport aux spécifications W3C XML Schema Definition Language (XSD) 1.0 et 1.1.

```
raptorxml valxml-withxsd | xsi [options] InputFile
```

- L'argument *`InputFile`* est le document XML à valider. L'option `--schemalocation-hints`[213] spécifie quel mécanisme est utilisé pour trouver le schéma. L'option `--xsd=FILE`[212] spécifie les schémas à utiliser si le fichier XML ne contient pas de référence de schéma.
- Pour valider de multiples documents, soit : (i) lister les fichiers à valider sur la CLI, chaque fichier étant séparé de l'autre par un espace ; ou (ii) lister les fichiers à valider dans un fichier de texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *`InputFile`* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).

Note : Si vous utilisez l'option `--script` pour exécuter des scripts Python[346], veuillez vous assurer de spécifier `--streaming=false`.

Exemples

Exemples de la commande **`valxml-withxsd`** :

- **`raptorxml`** `valxml-withxsd --schemalocation-hints=load-by-schemalocation --xsd=c:\MyXSD.xsd c:\HasNoXSDRef.xml`
- **`raptorxml`** `xsi c:\HasXSDRef.xml`
- **`raptorxml`** `xsi --xsd-version=1.1 --listfile=true c:\FileList.txt`

- Casing and slashes on the command line

  `RaptorXML` (and **`RaptorXMLServer`** for administration commands) *on Windows*
  `raptorxml` (and **`raptorxmlserver`** for administration commands) *on Windows and Unix (Linux, Mac)*

  * Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
  * Use forward slashes on Linux and Mac, backslashes on Windows.

- Backslashes, spaces, and special characters on Windows systems

  On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **`"My File"`**. Note, however, that

a backslash followed by a double-quotation mark (for example, `"C:\My directory\"`) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence `\"` stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: `\\"`. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: `"C:\My Directory\ \"`.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

Validation et traitement

assessment-mode

`--assessment-mode =` `lax|strict`
Spécifie le mode d'évaluation de validité de schéma tel que défini dans les spécifications XSD. La valeur par défaut est `strict`. Le document d'instance XML sera validé conformément au mode spécifié avec cette option.

ct-restrict-mode

`--ct-restrict-mode =` `1.0|1.1|default`
Spécifie comment contrôler les restrictions de type complexes. Une valeur de **`1.0`** contrôles les restrictions de type complexes telles que définies dans la spécification XSD 1.0 — même dans le mode de validation XSD 1.1. Une valeur de **`1.1`** contrôles les restrictions de type complexe telles que définies dans la spécification XSD 1.1 — même dans le mode de validation XSD 1.0. Une valeur de **`default`** contrôle les restrictions de type complexes telles que définies dans la spécification XSD du mode de validation actuel (1.0 ou 1.1). La valeur par défaut est **`default`**.

listfile

`--listfile =` `true|false`
Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

parallel-assessment [pa]

`--pa | --parallel-assessment =` `true|false`
Si configuré sur `true`, l'évaluation de validité de schéma est effectuée en parallèle. Cela signifie qu'il y

a plus de 128 éléments à tout niveau, ces éléments sont traités en parallèle, en utilisant plusieurs threads. Des fichiers XML très volumineux peuvent donc être traités plus rapidement si cette option est activée. L'évaluation parallèle a lieu un niveau hiérarchique à la fois, mais peut se produire à plusieurs niveaux différents dans une seule infobulle. Veuillez noter que l'évaluation parallèle ne fonctionne pas en mode streaming. C'est pour cette raison que l'option `--streaming` est ignorée si l'évaluation `--parallel-` est définie sur `true`. De même, les besoins en mémoire sont plus élevés lorsque l'option `--parallel-assessment` est utilisée. Le paramètre par défaut est `false`. La forme abrégée pour l'option est `--pa`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- recurse

  `--recurse =` `true|false`
  Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- report-import-namespace-mismatch-as-warning

  `--report-import-namespace-mismatch-as-warning =` `true|false`
  Réduit des erreurs de non-correspondance d'espace de noms ou d'espace de noms cible lors de l'importation de schémas avec **`xs:import`** et les fait passer d'erreurs à des avertissements. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- schema-imports

  `--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
  Spécifie le comportement des éléments `xs:import`, chacun d'entre eux ayant un attribut `namespace` optionnel et un attribut `schemaLocation` optionnel : `<import namespace="someNS" schemaLocation="someURL">`. L'option spécifie s'il faut charger un document de schéma ou juste mettre un espace de noms sous licence, et, si un document de schéma doit être chargé, quelle information doit être utilisée pour le trouver. Défaut : `load-preferring-schemalocation`.
  Le comportement est le suivant :

  - `load-by-schemalocation`: La valeur de l'attribut `schemaLocation` est utilisée pour situer le schéma, en prenant en compte les mappages de catalogue[47]. Si l'attribut de l'espace de nom est présent, l'espace de noms est importé (licencé).
  - `load-preferring-schemalocation`: Si l'attribut `schemaLocation` est présent, il est utilisé en prenant en compte les mappages de catalogue[47]. Si l'attribut `schemaLocation` est présent, la valeur de l'attribut `namespace` est utilisée via un mappage de catalogue[47]. C'est la **valeur par défaut**.
  - `load-by-namespace`: La valeur de l'attribut `namespace` est utilisée pour situer le schéma via un mappage de catalogue[47].

- `load-combining-both`: Si soit l'Attribut `namespace` ou l'attribut `schemaLocation` a un mappage de catalogue[47], le mappage est utilisé. Si les deux ont des mappages de catalogue[47], alors la valeur de l'option `--schema-mapping` (option XML/XSD[213]) décide de quel mappage utiliser. Si aucun mappage de catalogue[47] n'est présent, l'attribut `schemaLocation` est utilisé.
- `license-namespace-only`: Le nom d'espace est importé. Aucun document de schéma n'est importé.

schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`

Spécifie le comportement des attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` : S'il faut charger un document de schéma et, si oui, quelle information doit être utilisée pour la trouver. Défaut : `load-by-schemalocation`.

- La valeur `load-by-schemalocation` utilise l'URL de l'emplacement de schéma[397] dans les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` dans les documents d'instance XML. Il s'agit de la **valeur par défaut**.
- La valeur `load-by-namespace` prend la part d'espace de nom[397] `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` et localise le schéma par le biais d'un mappage de catalogue[47].
- Si `load-combining-both` est utilisé et si soit la partie espace de noms ou la partie URL a un mappage de catalogue[47], alors le mappage de catalogue[47] est utilisé. Si tous deux ont des mappages de catalogue[47], alors la valeur de l'option de `--schema-mapping` (option XML/XSD[213]) de quel mappage utiliser. Si ni l'espace de noms ni l'URL n'a un mappage de catalogue, l'URL est utilisée.
- Si la valeur de l'option est `ignore`, les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` seront ignorés tous les deux.

schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`

Si l'emplacement de schéma et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, spécifie lequel des deux doit être utilisé pendant la consultation du catalogue. (Si soit l'option `--schemalocation-hints` ou l'option `--schema-imports` a une valeur de `load-combining-both`, et si les parties d'espace de noms et d'URL impliquées ont toutes les deux des mappages de catalogue[47], alors la valeur de cette option spécifie lequel des deux mappages utiliser (mappage d'espace de noms ou mappage URL ; la valeur `prefer-schemalocation` réfère au mappage URL).) Défaut : `prefer-schemalocation`.

script

`--script =` *`FILE`*

Exécute le script Python dans le fichier soumis une fois que la validation a été terminée. Ajouter l'option plusieurs fois pour spécifier plus d'un seul script.

script-api-version

`--api, --script-api-version =` `1; 2; 2.1 to 2.4; 2.4.1; 2.5 to 2.8; 2.8.1 to 2.8.6; 2.9.0`

Spécifie la version API Python à utiliser pour le script. La valeur par défaut est la dernière version, actuellement **`2.9.0`**. À la place des valeurs entières `1` et `2`, vous pouvez aussi utiliser les valeurs correspondantes `1.0` et `2.0`. De la même manière, vous pouvez utiliser les trois chiffres `2.5.0` pour

les deux `2.5`. Voir aussi le chapitre Versions Python API 347.

- script-output

  `--script-output =` *`FILE`*
  Rédige la sortie standard du script dans le fichier nommé dans *`FILE`*.

- script-param

  `--script-param =` *`KEY:VALUE`*
  Des paramètres spécifiés par l'utilisateur supplémentaires qui peuvent être accédés au cours de l'exécution des scripts Python. Ajouter l'option plusieurs fois pour spécifier plus d'un paramètre de script.

- streaming

  `--streaming =` `true|false`
  Permet la validation de streaming. La valeur par défaut est `true`. Dans le mode streaming, les données stockées dans la mémoire sont minimisées et le traitement est plus rapide. L'inconvénient est que l'information que peut être requise par conséquent, par exemple, un modèle de données du document d'instance XML —ne sera pas disponible. Dans des situations où cela est significatif, le mode de streaming devra être éteint (en donnant à `--streaming` une valeur de `false`). En utilisant l'option `--script` avec la commande `valxml-withxsd`, désactiver le streaming. Veuillez noter que l'option `--streaming` est ignorée si`--parallel-assessment` est défini sur `true`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- xinclude

  `--xinclude =` `true|false`
  Active la prise en charge de XML Inclusions (XInclude). La valeur par défaut est `false`. Si `false`, les élément `include` d'XInclude sont ignorés.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- xml-mode

  `--xml-mode =` `wf|id|valid`
  Spécifie le mode de traitement XML à utiliser pour le document d'instance XML : `wf`=wellformed check (vérification de la bonne forme); `id`=bien formé avec des contrôles ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document d'instance chargé depuis les traitements référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

- xml-mode-for-schemas

  `--xml-mode-for-schemas =` `wf|id|valid`
  Spécifie le mode de traitement XML à utiliser pour les documents de schéma XML : `wf`=vérification de la bonne formation ; `id`=contrôles de la bonne formation avec ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document de schéma chargé pendant le traitement référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

- xsd

--xsd = *FILE*
Spécifie un ou plusieurs documents de Schéma XML à utiliser pour la validation des documents d'instance XML. Ajouter l'option plusieurs fois pour spécifier plus d'un document de schéma.

- xsd-version

  --xsd-version = `1.0|1.1|detect`
  Spécifie la version du langage de définition de Schéma W3C (XSD) à utiliser. Le défaut est `1.0`. Cette option peut être utile pour découvrir comment un schéma compatible à 1.0 n'est pas compatible à 1.1. L'option `detect` est une fonction spécifique à Altova. Elle permet la détection de la version du document de Schéma XML (`1.0` ou `1.1`) en lisant la valeur de l'attribut `vc:minVersion` de l'élément du document `<xs:schema>`. Si la valeur de l'attribut `@vc:minVersion` est `1.1`, le schéma est détecté comme étant de version `1.1`. Pour toute autre valeur, ou si l'attribut `@vc:minVersion` est absent, le schéma est détecté comme étant de version `1.0`.

- Catalogues et ressources globales
  - catalog

    --catalog = *FILE*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML [47], pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    --user-catalog = *FILE*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Active les ressources globales [54]. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALUE*
    Spécifie la configuration active de la ressource globale [54] (et active les ressources globales) [54].

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *FILE*
    Spécifie le fichier de ressource globale [54] (et active les ressources globales) [54].

- Messages, erreurs, aide, timeout, version
  - error-format

--error-format = `text|shortxml|longxml`
Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

- error-limit

--error-limit = *N | unlimited*
Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

- info-limit

--info-limit = *N | unlimited*
Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

--help
Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

--log-output = *FILE*
Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

--network-timeout = *VALEUR*
Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

--verbose = `true|false`
Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

--verbose-output = *FILE*
Écrit la sortie verbeuse sur *FILE*.

- version

--version
Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

▼ warning-limit

--warning-limit = *N | unlimited*
Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.1.3 valdtd (dtd)

La commande **valdtd | dtd** valide un ou plusieurs documents DTD conformément à la spécification XML 1.0 ou XML 1.1.

```
raptorxml valdtd | dtd [options] InputFile
```

- L'argument *InputFile* est le document DTD à valider.
- Pour valider de multiples documents, soit : (i) lister les fichiers à valider sur la CLI, chaque fichier étant séparé du suivant par un espace ; ou (ii) lister les fichiers à valider dans un fichier texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *InputFile* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).).

Exemples

Exemples de la commande **valdtd**

- **raptorxml** valdtd c:\Test.dtd
- **raptorxml** dtd --verbose=true c:\Test.dtd
- **raptorxml** dtd --listfile=true c:\FileList.txt

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Validation et traitement
  - listfile

    `--listfile =` `true|false`
    Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - recurse

    `--recurse =` `true|false`
    Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - script

    `--script =` *`FILE`*
    Exécute le script Python dans le fichier soumis une fois que la validation a été terminée. Ajouter l'option plusieurs fois pour spécifier plus d'un seul script.

  - script-api-version

    `--api, --script-api-version =` `1; 2; 2.1 to 2.4; 2.4.1; 2.5 to 2.8; 2.8.1 to 2.8.6; 2.9.0`
    Spécifie la version API Python à utiliser pour le script. La valeur par défaut est la dernière version, actuellement **`2.9.0`**. À la place des valeurs entières `1` et `2`, vous pouvez aussi utiliser les valeurs correspondantes `1.0` et `2.0`. De la même manière, vous pouvez utiliser les trois chiffres `2.5.0` pour les deux `2.5`. Voir aussi le chapitre Versions Python API 347.

- script-output

  --script-output = *FILE*
  Rédige la sortie standard du script dans le fichier nommé dans *FILE*.

- script-param

  --script-param = *KEY:VALUE*
  Des paramètres spécifiés par l'utilisateur supplémentaires qui peuvent être accédés au cours de l'exécution des scripts Python. Ajouter l'option plusieurs fois pour spécifier plus d'un paramètre de script.

- Catalogues et ressources globales
  - catalog

    --catalog = *FILE*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    --user-catalog = *FILE*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Active les ressources globales 54. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALUE*
    Spécifie la configuration active de la ressource globale 54 (et active les ressources globales) 54.

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *FILE*
    Spécifie le fichier de ressource globale 54 (et active les ressources globales) 54.

- Messages, erreurs, aide, timeout, version
  - error-format

    --error-format = `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

- error-limit

  --error-limit = *N | unlimited*
  Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

- info-limit

  --info-limit = *N | unlimited*
  Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  --help
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  --log-output = *FILE*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut

est `100`.

5.1.4 valxsd (xsd)

La commande **`valxsd | xsd`** valide un ou plusieurs documents de Schéma XML (documents XSD) conformément à la spécification de langue de définition de Schéma XML W3C (XSD) 1.0 ou 1.1. Veuillez noter que c'est le schéma lui-même qui est validé par rapport à la spécification de Schéma XML, et non pas un document d'instance XML par rapport à un Schéma XML.

```
raptorxml valxsd | xsd [options] InputFile
```

- L'argument *`InputFile`* est le document de Schéma XML à valider. L'option `--xsd-version=1.0|1.1|detect`[213] spécifie la version XSD par rapport à laquelle valider, le défaut étant `1.0`.
- Pour valider de multiples documents, soit : (i) recensez les fichiers à valider dans le CLI, avec chaque fichier séparé du prochain par un espace ; ou (ii) recensez les fichiers à valider dans un fichier texte (fichier `.txt`), avec un nom de fichier par ligne, et fournissez ce fichier texte en tant qu'argument *`InputFile`* ensemble avec l'option `--listfile`[211] définie comme `true` (*voir les Options dans la liste ci-dessous*).

Exemples

Exemples de la commande **`valxsd`**

- **`raptorxml`** `valxsd c:\Test.xsd`
- **`raptorxml`** `xsd --verbose=true c:\Test.xsd`
- **`raptorxml`** `xsd --listfile=true c:\FileList.txt`

▼ Casing and slashes on the command line

`RaptorXML` (and **`RaptorXMLServer`** for administration commands) *on Windows*
`raptorxml` (and **`raptorxmlserver`** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **`"My File"`**. Note, however, that a backslash followed by a double-quotation mark (for example, **`"C:\My directory\"`**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **`\"`** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **`\\"`**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **`"C:\My Directory\ \"`**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Validation et traitement
  - ct-restrict-mode

    `--ct-restrict-mode =` `1.0|1.1|default`
    Spécifie comment contrôler les restrictions de type complexes. Une valeur de **`1.0`** contrôles les restrictions de type complexes telles que définies dans la spécification XSD 1.0 — même dans le mode de validation XSD 1.1. Une valeur de **`1.1`** contrôles les restrictions de type complexe telles que définies dans la spécification XSD 1.1 — même dans le mode de validation XSD 1.0. Une valeur de **`default`** contrôle les restrictions de type complexes telles que définies dans la spécification XSD du mode de validation actuel (1.0 ou 1.1). La valeur par défaut est **`default`**.

  - listfile

    `--listfile =` `true|false`
    Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - recurse

    `--recurse =` `true|false`
    Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - report-import-namespace-mismatch-as-warning

    `--report-import-namespace-mismatch-as-warning =` `true|false`
    Réduit des erreurs de non-correspondance d'espace de noms ou d'espace de noms cible lors de l'importation de schémas avec **`xs:import`** et les fait passer d'erreurs à des avertissements. La valeur

par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

schema-imports

`--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Spécifie le comportement des éléments `xs:import`, chacun d'entre eux ayant un attribut `namespace` optionnel et un attribut `schemaLocation` optionnel : `<import namespace="someNS" schemaLocation="someURL">`. L'option spécifie s'il faut charger un document de schéma ou juste mettre un espace de noms sous licence, et, si un document de schéma doit être chargé, quelle information doit être utilisée pour le trouver. Défaut : `load-preferring-schemalocation`.
Le comportement est le suivant :

- `load-by-schemalocation`: La valeur de l'attribut `schemaLocation` est utilisée pour situer le schéma, en prenant en compte les mappages de catalogue[47]. Si l'attribut de l'espace de nom est présent, l'espace de noms est importé (licencé).
- `load-preferring-schemalocation`: Si l'attribut `schemaLocation` est présent, il est utilisé en prenant en compte les mappages de catalogue[47]. Si l'attribut `schemaLocation` est présent, la valeur de l'attribut `namespace` est utilisée via un mappage de catalogue[47]. C'est la **valeur par défaut**.
- `load-by-namespace`: La valeur de l'attribut `namespace` est utilisée pour situer le schéma via un mappage de catalogue[47].
- `load-combining-both`: Si soit l'Attribut `namespace` ou l'attribut `schemaLocation` a un mappage de catalogue[47], le mappage est utilisé. Si les deux ont des mappages de catalogue[47], alors la valeur de l'option `--schema-mapping` (option XML/XSD[213]) décide de quel mappage utiliser. Si aucun mappage de catalogue[47] n'est présent, l'attribut `schemaLocation` est utilisé.
- `license-namespace-only`: Le nom d'espace est importé. Aucun document de schéma n'est importé.

schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Spécifie le comportement des attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` : S'il faut charger un document de schéma et, si oui, quelle information doit être utilisée pour la trouver. Défaut : `load-by-schemalocation`.

- La valeur `load-by-schemalocation` utilise l'URL de l'emplacement de schéma[397] dans les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` dans les documents d'instance XML. Il s'agit de la **valeur par défaut**.
- La valeur `load-by-namespace` prend la part d'espace de nom[397] `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` et localise le schéma par le biais d'un mappage de catalogue[47].
- Si `load-combining-both` est utilisé et si soit la partie espace de noms ou la partie URL a un mappage de catalogue[47], alors le mappage de catalogue[47] est utilisé. Si tous deux ont des mappages de catalogue[47], alors la valeur de l'option de `--schema-mapping` (option XML/XSD[213]) de quel mappage utiliser. Si ni l'espace de noms ni l'URL n'a un mappage de catalogue, l'URL est utilisée.
- Si la valeur de l'option est `ignore`, les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` seront ignorés tous les deux.

- schema-mapping

  --schema-mapping = `prefer-schemalocation | prefer-namespace`
  Si l'emplacement de schéma et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, spécifie lequel des deux doit être utilisé pendant la consultation du catalogue. (Si soit l'option `--schemalocation-hints` ou l'option `--schema-imports` a une valeur de `load-combining-both`, et si les parties d'espace de noms et d'URL impliquées ont toutes les deux des mappages de catalogue[47], alors la valeur de cette option spécifie lequel des deux mappages utiliser (mappage d'espace de noms ou mappage URL ; la valeur `prefer-schemalocation` réfère au mappage URL).) Défaut : `prefer-schemalocation`.

- script

  --script = *FILE*
  Exécute le script Python dans le fichier soumis une fois que la validation a été terminée. Ajouter l'option plusieurs fois pour spécifier plus d'un seul script.

- script-api-version

  --api, --script-api-version = `1; 2; 2.1 to 2.4; 2.4.1; 2.5 to 2.8; 2.8.1 to 2.8.6; 2.9.0`
  Spécifie la version API Python à utiliser pour le script. La valeur par défaut est la dernière version, actuellement **2.9.0**. À la place des valeurs entières `1` et `2`, vous pouvez aussi utiliser les valeurs correspondantes `1.0` et `2.0`. De la même manière, vous pouvez utiliser les trois chiffres `2.5.0` pour les deux `2.5`. Voir aussi le chapitre Versions Python API[347].

- script-output

  --script-output = *FILE*
  Rédige la sortie standard du script dans le fichier nommé dans *FILE*.

- script-param

  --script-param = *KEY:VALUE*
  Des paramètres spécifiés par l'utilisateur supplémentaires qui peuvent être accédés au cours de l'exécution des scripts Python. Ajouter l'option plusieurs fois pour spécifier plus d'un paramètre de script.

- xinclude

  --xinclude = `true|false`
  Active la prise en charge de XML Inclusions (XInclude). La valeur par défaut est `false`. Si `false`, les élément `include` d'XInclude sont ignorés.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- xml-mode-for-schemas

  --xml-mode-for-schemas = `wf|id|valid`
  Spécifie le mode de traitement XML à utiliser pour les documents de schéma XML : `wf`=vérification de la bonne formation ; `id`=contrôles de la bonne formation avec ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document de schéma

chargé pendant le traitement référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

- xsd-version

  --xsd-version = `1.0|1.1|detect`
  Spécifie la version du langage de définition de Schéma W3C (XSD) à utiliser. Le défaut est `1.0`. Cette option peut être utile pour découvrir comment un schéma compatible à 1.0 n'est pas compatible à 1.1. L'option `detect` est une fonction spécifique à Altova. Elle permet la détection de la version du document de Schéma XML (`1.0` ou `1.1`) en lisant la valeur de l'attribut `vc:minVersion` de l'élément du document `<xs:schema>`. Si la valeur de l'attribut `@vc:minVersion` est `1.1`, le schéma est détecté comme étant de version `1.1`. Pour toute autre valeur, ou si l'attribut `@vc:minVersion` est absent, le schéma est détecté comme étant de version `1.0`.

Catalogues et ressources globales

- catalog

  --catalog = *FILE*
  Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

- user-catalog

  --user-catalog = *FILE*
  Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

- enable-globalresources

  --enable-globalresources = `true|false`
  Active les ressources globales 54. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Spécifie la configuration active de la ressource globale 54 (et active les ressources globales) 54.

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Spécifie le fichier de ressource globale 54 (et active les ressources globales) 54.

Messages, erreurs, aide, timeout, version

- error-format

  --error-format = `text|shortxml|longxml`
  Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des

formats XML, avec `longxml` générant plus de détails.

- error-limit

  `--error-limit =` *`N | unlimited`*
  Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

- info-limit

  `--info-limit =` *`N | unlimited`*
  Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  `--help`
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  `--log-output =` *`FILE`*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  `--network-timeout =` *`VALEUR`*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  `--verbose =` `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  `--verbose-output =` *`FILE`*
  Écrit la sortie verbeuse sur *`FILE`*.

- version

  `--version`
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

--warning-limit = *N | unlimited*
Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.2 Commandes de vérification de la bonne formation

Les commandes de vérification de la bonne formation peuvent être utilisées pour vérifier la bonne formation de documents XML et DTD. Ces commandes sont listées ci-dessous et sont décrites en détails dans les sous-sections de cette section :

- wfxml 81: Vérifie la bonne formation de documents XML
- wfdtd 85: Vérifie la bonne formation de DTD
- wfany 88: Vérifie la bonne formation d'un document XML ou de DTD. Le type est détecté automatiquement

5.2.1 wfxml

La commande **wfxml** contrôle la bonne formation d'un ou de plusieurs documents XML conformément à la spécification XML 1.0 ou XML 1.1.

```
raptorxml wfxml [options] InputFile
```

- L'argument *InputFile* est le document XML dont vous souhaitez vérifier la bonne formation.
- Si vous souhaitez contrôler plusieurs documents, il faut soit : (i) lister les fichiers à Coché sur la CLI, chaque fichier étant séparé du suivant par un espace ; ou (ii) lister les fichiers à Coché dans un fichier texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *InputFile* avec l'option `--listfile` 211 définie sur `true` (*voir la liste des Options ci-dessous*).).

Exemples

Exemples de la commande **wfxml** :

- **raptorxml** wfxml c:\Test.xml
- **raptorxml** wfxml --verbose=true c:\Test.xml
- **raptorxml** wfxml --listfile=true c:\FileList.txt

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to

escape this sequence of characters, use a preceding backslash, like this: `\\"`. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: `"C:\My Directory\ \"`.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

Validation et traitement

dtd

`--dtd =` *`FILE`*
Spécifie le document DTD externe à utiliser pour la validation. Si une référence à un DTD externe est présente dans le document XM, alors l'option CLI contourne la référence externe.

listfile

`--listfile =` `true|false`
Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

namespaces

`--namespaces =` `true|false`
Permet le traitement sensible aux espaces de noms. Cela est utile pour contrôler l'instance XML pour des erreurs dues à des espaces de noms incorrectes. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

recurse

`--recurse =` `true|false`
Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une

valeur.

- streaming

  --streaming = `true|false`
  Permet la validation de streaming. La valeur par défaut est `true`. Dans le mode streaming, les données stockées dans la mémoire sont minimisées et le traitement est plus rapide. L'inconvénient est que l'information que peut être requise par conséquent, par exemple, un modèle de données du document d'instance XML —ne sera pas disponible. Dans des situations où cela est significatif, le mode de streaming devra être éteint (en donnant à `--streaming` une valeur de `false`). En utilisant l'option `--script` avec la commande `valxml-withxsd`, désactiver le streaming. Veuillez noter que l'option `--streaming` est ignorée si`--parallel-assessment` est défini sur `true`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Catalogues et ressources globales
  - catalog

    --catalog = *FILE*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML [47], pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    --user-catalog = *FILE*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Active les ressources globales [54]. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALUE*
    Spécifie la configuration active de la ressource globale [54] (et active les ressources globales) [54].

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *FILE*
    Spécifie le fichier de ressource globale [54] (et active les ressources globales) [54].

- Messages, erreurs, aide, timeout, version
  - error-format

--error-format = `text|shortxml|longxml`
Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

- error-limit

  --error-limit = *N | unlimited*
  Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

- info-limit

  --info-limit = *N | unlimited*
  Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  --help
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  --log-output = *FILE*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

▼ warning-limit

--warning-limit = *N | unlimited*
Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.2.2 wfdtd

La commande **wfdtd** contrôle la bonne formation d'un ou de plusieurs documents DTD conformément à la spécification XML 1.0 ou XML 1.1.

```
raptorxml wfdtd [options] InputFile
```

- L'argument *InputFile* est le document DTD dont vous souhaitez vérifier la bonne formation.
- Si vous souhaitez contrôler plusieurs documents, il faut soit : (i) lister les fichiers à Coché sur la CLI, chaque fichier étant séparé du suivant par un espace ; ou (ii) lister les fichiers à Coché dans un fichier texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *InputFile* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).).

Exemples

Exemples de la commande **wfdtd** :

- **raptorxml** wfdtd c:\Test.dtd
- **raptorxml** wfdtd --verbose=true c:\Test.dtd
- **raptorxml** wfdtd --listfile=true c:\FileList.txt

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Validation et traitement
  - listfile

    `--listfile =` `true|false`
    Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - recurse

    `--recurse =` `true|false`
    Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Catalogues et ressources globales
  - catalog

    `--catalog =` *`FILE`*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    `--user-catalog =` *`FILE`*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section,

Catalogues XML[47], pour plus d'informations concernant le travail avec des catalogues.

- enable-globalresources

  --enable-globalresources = `true|false`
  Active les ressources globales[54]. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Spécifie la configuration active de la ressource globale[54] (et active les ressources globales)[54].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Spécifie le fichier de ressource globale[54] (et active les ressources globales)[54].

- Messages, erreurs, aide, timeout, version
  - error-format

    --error-format = `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    --error-limit = *N | unlimited*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    --info-limit = *N | unlimited*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - log-output

    --log-output = *FILE*
    Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.2.3 wfany

La commande **wfany** contrôle la bonne formation d'un document XML ou DTD conformément à la spécification respective. Le type de document est détecté automatiquement.

```
raptorxml wfany [options] InputFile
```

- L'argument *InputFile* est le document dont vous souhaitez vérifier la bonne formation.
- Veuillez noter qu'un seul document peut être soumis en tant que l'argument de la commande. Le type du document soumis est détecté automatiquement.

Exemples

Exemples de la commande **wfany** :

- **raptorxml** wfany c:\Test.xml
- **raptorxml** wfany --error-format=text c:\Test.xml

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (raptorxml and raptorxmlserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (RaptorXML) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est TRUE. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.
.

▼ Traitement

▼ listfile

--listfile = true|false
Si true, traiter l'argument *InputFile* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est false. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option --listfile s'applique uniquement aux arguments, et non pas aux options.
Note : les valeurs d'option booléennes sont configurées sur true si l'option est spécifiée sans une valeur.

▼ Catalogues et ressources globales

▼ catalog

--catalog = *FILE*
Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé

(`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML[47], pour plus d'informations à propos du travail avec les catalogues.

- user-catalog

  `--user-catalog =` *`FILE`*
  Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML[47], pour plus d'informations concernant le travail avec des catalogues.

- enable-globalresources

  `--enable-globalresources =` `true|false`
  Active les ressources globales[54]. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- globalresourceconfig [gc]

  `--gc | --globalresourceconfig =` *`VALUE`*
  Spécifie la configuration active de la ressource globale[54] (et active les ressources globales)[54].

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *`FILE`*
  Spécifie le fichier de ressource globale[54] (et active les ressources globales)[54].

- Messages, erreurs, aide, timeout, version
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

  - help

    `--help`
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  --log-output = *FILE*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.3 Commandes XQuery

Les commandes XQuery sont :

- xquery 92: pour exécuter des documents XQuery documents, en option, avec un document d'entrée
- xqueryupdate 100: pour exécuter un document XQuery update, utiliser un document XQuery et, en option, le document XML d'entrée à mettre à jour
- valxquery 108: pour valider des documents XQuery
- valxqueryupdate 114: pour valider un document XQuery (update)

5.3.1 xquery

La commande **xqueryupdate** prend un fichier XQuery en tant que son argument unique et l'exécute avec un fichier d'entrée optionnel pour produire un fichier de sortie mis à jour. Les fichiers d'entrée et de sortie sont spécifiés en tant qu'options.

```
raptorxml xquery [options] XQuery-File
```

- L'argument ***XQuery-File*** est le chemin et le nom du fichier XQuery à exécuter.
- Vous pouvez utiliser XQuery 1.0 ou 3.0. XQuery 3.0 est utilisé par défaut.

Exemples

Exemples de la commande **xquery** :

- **raptorxml** xquery --output=c:\Output.xml c:\TestQuery.xq
- **raptorxml** xquery --input=c:\Input.xml --output=c:\Output.xml --param=company:"Altova" --p=date:"2006-01-01" c:\TestQuery.xq
- **raptorxml** xquery --input=c:\Input.xml --output=c:\Output.xml --param=source:"doc('c:\test\books.xml')//book "
- **raptorxml** xquery --output=c:\Output.xml --omit-xml-declaration=false --output-encoding=ASCII c:\TestQuery.xq

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (raptorxml and raptorxmlserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (RaptorXML) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to

escape this sequence of characters, use a preceding backslash, like this: `\\"`. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: `"C:\My Directory\ \"`.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Traitement XQuery
  - indent-characters

    `--indent-characters =` *VALUE*
    Spécifie le string de caractère à utiliser en tant que retrait.

  - input

    `--input =` *FILE*
    L'URL du fichier XML à transformer.

  - omit-xml-declaration

    `--omit-xml-declaration =` `true|false`
    Option de sérialisation pour spécifier si la déclaration XML doit être omise de la sortie ou pas. Si `true`, il n'y aura pas de déclaration XML dans le document de sortie. Si `false`, une déclaration XML sera incluse. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - output, xsltoutput

    `output =` *FILE,* **`xsltoutput =`** *FILE*
    L'URL du fichier sortie-primaire. Par exemple, dans le cas d'une sortie HTML de fichier multiple, le fichier de sortie primaire sera l'emplacement du fichier HTML du point d'entrée. Les fichiers de sortie supplémentaires, comme des fichiers d'image générée, sont rapportés en tant que `xslt-additional-output-files`. Si aucune option `--output` ou `--xsltoutput` n'est spécifiée, la sortie est écrite sur la sortie standard.

  - output-encoding

    `--output-encoding =` *VALUE*
    La valeur de l'attribut d'encodage dans le document de sortie. Les valeurs valides sont des noms dans le registre d'ensemble de caractères IANA. La valeur par défaut est `UTF-8`.

  - output-indent

--output-indent = `true|false`
Si `true`, la sortie sera en retrait conformément à sa structure hiérarchique. Si `false`, il n'y aura pas de retrait hiérarchique. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ output-method

--output-method = `xml|html|xhtml|text`
Spécifie le format de sortie. La valeur par défaut est `xml`.

▼ param [p]

--p | --param = *KEY:VALUE*

⊟ ***XQuery***

Spécifie la valeur d'un paramètre externe. Un paramètre externe est déclaré dans le document XQuery avec la déclaration `declare variable` suivi par un nom de variable puis le mot-clé `external` suivi par le point-virgule de fin de ligne. Par exemple :

```
declare variable $foo as xs:string external;
```

Le mot-clé `external`, `$foo` devient un paramètre externe, dont la valeur est passée pendant le runtime depuis une source externe. Le paramètre externe reçoit une value avec la commande CLI. Par exemple :

```
--param=foo:'MyName'
```

Dans l'instruction de description ci-dessus, *KEY* est le nom de paramètre externe, *VALUE* est la valeur du paramètre externe, donné en tant qu'une expression XPath. Les noms de paramètres utilisés sur la CLI doivent être déclarés dans le document XQuery. Si plusieurs paramètres externes sont des valeurs passées sur la CLI, chacun doit être recevoir une option séparée `--param`. Les doubles guillemets doivent être utilisés si l'expression XPath contient des espaces.

⊟ ***XSLT***

Spécifie paramètre de feuille de style global. *KEY* est le nom de paramètre, *VALUE* est une expression XPath qui fournit la valeur de paramètre. Les noms de paramètre utilisés sur la CLI doivent être déclarés dans la feuille de style. Si plusieurs paramètres multiples sont utilisés, le changement `--param` doit être utilisé avant chaque paramètre. Les double guillemets doivent être utilisés autour de l'expression XPath si elle contient un espace ; que l'espace se trouve dans l'expression XPath elle-même ou dans un littéral de string dans l'expression. Par exemple :

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
```

▼ xpath-static-type-errors-as-warnings

--xpath-static-type-errors-as-warnings = `true|false`
Si `true`, passe à une version antérieure d'avertissements de tous les types d'erreurs qui sont détectés dans le contexte statique XPath. Alors qu'une erreur entraînerait l'échec de l'exécution, un avertissement permettrait la poursuite du traitement. Le défaut est `false`.

▼ xquery-version

--xquery-version = `1|1.0|3|3.0|3.1`
Spécifie si le processeur XQuery devrait utiliser XSLT XQuery 1.0 ou XQuery 3.0. La valeur par défaut est 3.1.

Schéma XML et instance XML

load-xml-with-psvi

--load-xml-with-psvi = `true|false`
Active la validation des fichiers XML d'entrée et génère des informations de post-schema-validation pour les fichiers. La valeur par défaut est : `true`.

schema-imports

--schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Spécifie le comportement des éléments `xs:import`, chacun d'entre eux ayant un attribut `namespace` optionnel et un attribut `schemaLocation` optionnel : `<import namespace="someNS" schemaLocation="someURL">`. L'option spécifie s'il faut charger un document de schéma ou juste mettre un espace de noms sous licence, et, si un document de schéma doit être chargé, quelle information doit être utilisée pour le trouver. Défaut : `load-preferring-schemalocation`.
Le comportement est le suivant :

- `load-by-schemalocation`: La valeur de l'attribut `schemaLocation` est utilisée pour situer le schéma, en prenant en compte les mappages de catalogue[47]. Si l'attribut de l'espace de nom est présent, l'espace de noms est importé (licencé).
- `load-preferring-schemalocation`: Si l'attribut `schemaLocation` est présent, il est utilisé en prenant en compte les mappages de catalogue[47]. Si l'attribut `schemaLocation` est présent, la valeur de l'attribut `namespace` est utilisée via un mappage de catalogue[47]. C'est la **valeur par défaut**.
- `load-by-namespace`: La valeur de l'attribut `namespace` est utilisée pour situer le schéma via un mappage de catalogue[47].
- `load-combining-both`: Si soit l'Attribut `namespace` ou l'attribut `schemaLocation` a un mappage de catalogue[47], le mappage est utilisé. Si les deux ont des mappages de catalogue[47], alors la valeur de l'option `--schema-mapping` (option XML/XSD[213]) décide de quel mappage utiliser. Si aucun mappage de catalogue[47] n'est présent, l'attribut `schemaLocation` est utilisé.
- `license-namespace-only`: Le nom d'espace est importé. Aucun document de schéma n'est importé.

schemalocation-hints

--schemalocation-hints = `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Spécifie le comportement des attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` : S'il faut charger un document de schéma et, si oui, quelle information doit être utilisée pour la trouver. Défaut : `load-by-schemalocation`.

- La valeur `load-by-schemalocation` utilise l'URL de l'emplacement de schéma[397] dans les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` dans les documents d'instance XML. Il s'agit de la **valeur par défaut**.
- La valeur `load-by-namespace` prend la part d'espace de nom[397] `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` et localise le schéma par le biais d'un mappage de catalogue[47].

- Si `load-combining-both` est utilisé et si soit la partie espace de noms ou la partie URL a un mappage de catalogue[47], alors le mappage de catalogue[47] est utilisé. Si tous deux ont des mappages de catalogue[47], alors la valeur de l'option de `--schema-mapping` (option XML/XSD[213]) de quel mappage utiliser. Si ni l'espace de noms ni l'URL n'a un mappage de catalogue, l'URL est utilisée.
- Si la valeur de l'option est `ignore`, les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` seront ignorés tous les deux.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si l'emplacement de schéma et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, spécifie lequel des deux doit être utilisé pendant la consultation du catalogue. (Si soit l'option `--schemalocation-hints` ou l'option `--schema-imports` a une valeur de `load-combining-both`, et si les parties d'espace de noms et d'URL impliquées ont toutes les deux des mappages de catalogue[47], alors la valeur de cette option spécifie lequel des deux mappages utiliser (mappage d'espace de noms ou mappage URL ; la valeur `prefer-schemalocation` réfère au mappage URL).) Défaut : `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`
Active la prise en charge de XML Inclusions (XInclude). La valeur par défaut est `false`. Si `false`, les élément `include` d'XInclude sont ignorés.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ xml-mode

`--xml-mode =` `wf|id|valid`
Spécifie le mode de traitement XML à utiliser pour le document d'instance XML : `wf`=wellformed check (vérification de la bonne forme); `id`=bien formé avec des contrôles ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document d'instance chargé depuis les traitements référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`
Spécifie le mode de traitement XML à utiliser pour les documents de schéma XML : `wf`=vérification de la bonne formation ; `id`=contrôles de la bonne formation avec ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document de schéma chargé pendant le traitement référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

▼ xml-validation-error-as-warning

`--xml-validation-error-as-warning =` `true|false`
Si `true`, traite les erreurs de validation en tant qu'avertissements. Si les erreurs sont traitées en tant qu'avertissements, les traitements supplémentaires, comme des transformations XSLT, continueront quelles que soient les erreurs. La valeur par défaut est `false`.

▼ xpath-static-type-errors-as-warnings

--xpath-static-type-errors-as-warnings = true|false
Si true, passe à une version antérieure d'avertissements de tous les types d'erreurs qui sont détectés dans le contexte statique XPath. Alors qu'une erreur entraînerait l'échec de l'exécution, un avertissement permettrait la poursuite du traitement. Le défaut est false.

- xsd

--xsd = *FILE*
Spécifie un ou plusieurs documents de Schéma XML à utiliser pour la validation des documents d'instance XML. Ajouter l'option plusieurs fois pour spécifier plus d'un document de schéma.

- xsd-version

--xsd-version = 1.0|1.1|detect
Spécifie la version du langage de définition de Schéma W3C (XSD) à utiliser. Le défaut est 1.0. Cette option peut être utile pour découvrir comment un schéma compatible à 1.0 n'est pas compatible à 1.1. L'option detect est une fonction spécifique à Altova. Elle permet la détection de la version du document de Schéma XML (1.0 ou 1.1) en lisant la valeur de l'attribut vc:minVersion de l'élément du document <xs:schema>. Si la valeur de l'attribut @vc:minVersion est 1.1, le schéma est détecté comme étant de version 1.1. Pour toute autre valeur, ou si l'attribut @vc:minVersion est absent, le schéma est détecté comme étant de version 1.0.

- Catalogues et ressources globales
  - catalog

--catalog = *FILE*
Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

--user-catalog = *FILE*
Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

--enable-globalresources = true|false
Active les ressources globales 54. La valeur par défaut est false.
Note : les valeurs d'option booléennes sont configurées sur true si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Spécifie la configuration active de la ressource globale 54 (et active les ressources globales) 54.

  - globalresourcefile [gr]

--gr | --globalresourcefile = *FILE*

Spécifie le fichier de ressource globale[54] (et active les ressources globales)[54].

- Extensions

  Ces options définissent la gestion des fonctions d'extension spéciales disponibles dans un certain nombre de produits Altova de niveau d'entreprise (comme XMLSpy Enterprise Edition). Leur utilisation est décrite dans les manuels d'utilisateur de ces produits.

  - chartext-disable

    --chartext-disable = `true|false`
    Désactive les extensions de graphique. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - dotnetext-disable

    --dotnetext-disable = `true|false`
    Désactive les extensions .NET. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - jvm-location

    --jvm-location = *FILE*
    FILE spécifie l'emplacement de la Java Virtual Machine (DLL sur Windows, objet partagé sur Linux). La JVM est nécessaire si vous utilisez les fonctions d'extension Java dans votre code XSLT/XQuery. La valeur par défaut est `false`.

  - javaext-barcode-location

    --javaext-barcode-location = *FILE*
    Spécifie le chemin vers le dossier qui contient le fichier d'extension du code-barres `AltovaBarcodeExtension.jar`. Le chemin doit être donné dans une des formes suivantes :

    - Un URI de fichier, par exemple : `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2024/etc/jar/"`
    - Un chemin Windows avec un échappement par barres obliques inversées, par exemple : `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2024\\etc\\jar\\"`

  - javaext-disable

    --javaext-disable = `true|false`
    Désactive les extensions Java. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Messages, erreurs, aide, timeout, version

  - error-format

    --error-format = `text|shortxml|longxml`

Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

- error-limit

  `--error-limit =` *`N | unlimited`*
  Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

- info-limit

  `--info-limit =` *`N | unlimited`*
  Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  `--help`
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  `--log-output =` *`FILE`*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  `--network-timeout =` *`VALEUR`*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  `--verbose =` `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  `--verbose-output =` *`FILE`*
  Écrit la sortie verbeuse sur *`FILE`*.

- version

  `--version`
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

--warning-limit = *N | unlimited*
Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.3.2 xqueryupdate

La commande **xqueryupdate** prend un fichier XQuery ou Xquery Update en tant que son argument unique et le valide. Si un fichier XML d'entrée optionnel est spécifié, alors ce fichier XML est traité avec les commandes XQuery Update soumises dans ***XQuery(Update)-File***. Dans ce cas, les mises à jour peuvent être directement appliquées dans le fichier d'entrée ou les données XML mises à jour peuvent être écrites dans un fichier XML de sortie. Les fichiers d'entrée et de sortie sont spécifiés en tant qu'options. Si le ***XQuery(Update)-File*** contient uniquement des instructions XQuery et pas d'instruction de XQuery Update, alors la commande réalise une exécution directe de XQuery.

raptorxml xqueryupdate [options] ***XQuery(Update)-File***

- L'argument ***XQuery(Update)-File*** est le chemin et le nom du fichier XQuery (**.xq**) ou du fichier XQuery Update (**.xqu**) à être exécuté. Si le fichier contient des instructions XQuery Update, alors celles-ci sont exécutées dans le fichier XML d'entrée. Autrement, la commande fonctionne comme commande d'exécution XQuery.
- Vous pouvez spécifier si vous préférez utiliser XQuery Update 1.0 ou 3.0. XQuery Update 3.0 est utilisé par défaut.

Exemples

Exemples de la commande **xqueryupdate** :

- **raptorxml** `xqueryupdate --output=c:\Output.xml c:\TestQuery.xq` (écrit la sortie du fichier XQuery dans le fichier de sortie.)
- **raptorxml** `xqueryupdate --input=c:\Input.xml --output=c:\Output.xml --updated-xml=asmainresult c:\UpdateFile.xqu` (met à jour `Input.xml` utilisant les instructions de mise à jour dans `UpdateFile.xqu`, et écrit la mise à jour dans `Output.xml`.)
- **raptorxml** `xqueryupdate --input=c:\Input.xml --output=c:\Output.xml --updated-xml=writeback c:\UpdateFile.xq` (met à jour `Input.xml` utilisant les instructions de mise à jour dans `UpdateFile.xq`. Le fichier `Output.xml` n'est pas créé.)
- **raptorxml** `xqueryupdate --input=c:\Input.xml --output=c:\Output.xml --updated-xml=discard c:\TestQuery.xqu` (les mises à jour sont ignorées. Le fichier d'entrée n'est pas modifié. Le fichier `Output.xml` sera créé mais il ne contiendra pas de XML mis à jour.)
- **raptorxml** `xqueryupdate --input=c:\Input.xml --output=c:\Output.xml c:\TestQuery.xqu` (les mises à jour sont ignorées dans l'exemple précédent. C'est parce que la valeur par défaut de l'option `--updated-xml` est `ignorée`.)

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*

raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (raptorxml and raptorxmlserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (RaptorXML) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est TRUE. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.
.

▼ Traitement XQuery Update

▼ indent-characters

--indent-characters = *VALUE*
Spécifie le string de caractère à utiliser en tant que retrait.

▼ input

--input = *FILE*
L'URL du fichier XML à transformer.

▼ omit-xml-declaration

--omit-xml-declaration = true|false
Option de sérialisation pour spécifier si la déclaration XML doit être omise de la sortie ou pas. Si true, il n'y aura pas de déclaration XML dans le document de sortie. Si false, une déclaration XML sera incluse. La valeur par défaut est false.
Note : les valeurs d'option booléennes sont configurées sur true si l'option est spécifiée sans une valeur.

▼ output, xsltoutput

output = *FILE*, **xsltoutput =** *FILE*
L'URL du fichier sortie-primaire. Par exemple, dans le cas d'une sortie HTML de fichier multiple, le fichier de sortie primaire sera l'emplacement du fichier HTML du point d'entrée. Les fichiers de sortie supplémentaires, comme des fichiers d'image générée, sont rapportés en tant que `xslt-additional-output-files`. Si aucune option `--output` ou `--xsltoutput` n'est spécifiée, la sortie est écrite sur la sortie standard.

▼ output-encoding

--output-encoding = *VALUE*
La valeur de l'attribut d'encodage dans le document de sortie. Les valeurs valides sont des noms dans le registre d'ensemble de caractères IANA. La valeur par défaut est `UTF-8`.

▼ output-indent

--output-indent = `true|false`
Si `true`, la sortie sera en retrait conformément à sa structure hiérarchique. Si `false`, il n'y aura pas de retrait hiérarchique. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ output-method

--output-method = `xml|html|xhtml|text`
Spécifie le format de sortie. La valeur par défaut est `xml`.

▼ param [p]

--p | --param = *KEY:VALUE*

⊟ ***XQuery***

Spécifie la valeur d'un paramètre externe. Un paramètre externe est déclaré dans le document XQuery avec la déclaration `declare variable` suivi par un nom de variable puis le mot-clé `external` suivi par le point-virgule de fin de ligne. Par exemple :

```
declare variable $foo as xs:string external;
```

Le mot-clé `external`, `$foo` devient un paramètre externe, dont la valeur est passée pendant le runtime depuis une source externe. Le paramètre externe reçoit une value avec la commande CLI. Par exemple :

```
--param=foo:'MyName'
```

Dans l'instruction de description ci-dessus, *KEY* est le nom de paramètre externe, *VALUE* est la valeur du paramètre externe, donné en tant qu'une expression XPath. Les noms de paramètres utilisés sur la CLI doivent être déclarés dans le document XQuery. Si plusieurs paramètres externes sont des valeurs passées sur la CLI, chacun doit être recevoir une option séparée `--param`. Les doubles guillemets doivent être utilisés si l'expression XPath contient des espaces.

⊟ ***XSLT***

Spécifie paramètre de feuille de style global. *KEY* est le nom de paramètre, *VALUE* est une expression XPath qui fournit la valeur de paramètre. Les noms de paramètre utilisés sur la CLI doivent être déclarés dans la feuille de style. Si plusieurs paramètres multiples sont utilisés, le changement `--param` doit être utilisé avant chaque paramètre. Les double guillemets doivent être utilisés autour de l'expression XPath si elle contient un espace ; que l'espace se trouve dans l'expression XPath elle-même ou dans un littéral de string dans l'expression. Par exemple

:

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
```

- xpath-static-type-errors-as-warnings

  --xpath-static-type-errors-as-warnings = `true|false`
  Si `true`, passe à une version antérieure d'avertissements de tous les types d'erreurs qui sont détectés dans le contexte statique XPath. Alors qu'une erreur entraînerait l'échec de l'exécution, un avertissement permettrait la poursuite du traitement. Le défaut est `false`.

- xquery-update-version

  --xquery-update-version = `1|1.0|3|3.0`
  Spécifie si le processeur XQuery devrait utiliser XQuery Update Facility 1.0 ou XQuery Update Facility 3.0. La valeur par défaut est `3`.

- keep-formatting

  --keep-formatting = `true|false`
  Garde le formatage du document cible autant que possible. La valeur par défaut est : `true`.

- updated-xml

  --updated-xml = `discard|writeback|asmainresult`
  Spécifie comment gérer le fichier XML mis à jour.

  - `discard` : la mise à jour est éliminée et pas écrites sur le fichier. Ni le fichier d'entrée, ni le fichier de sortie seront mis à jour. Veuillez noter que ceci est le défaut.
  - `writeback` : réécrit la mise à jour de retour dans le fichier XML d'entrée qui est spécifié avec l'option `--input`.
  - `asmainresult` : écrit la mise à jour dans le fichier XML de sortie qui est spécifié dans l'option `--output` option. Si l'option **`--output`** n'est pas spécifiée, alors la mise à jour est écrite dans la sortie standard. Dans les deux cas, le fichier XML d'entrée ne sera pas modifié ().

  La valeur par défaut est : `discard`.

- Schéma XML et instance XML
  - load-xml-with-psvi

    --load-xml-with-psvi = `true|false`
    Active la validation des fichiers XML d'entrée et génère des informations de post-schema-validation pour les fichiers. La valeur par défaut est : `true`.

  - schema-imports

    --schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Spécifie le comportement des éléments `xs:import`, chacun d'entre eux ayant un attribut `namespace`

optionnel et un attribut `schemaLocation` optionnel : `<import namespace="someNS" schemaLocation="someURL">`. L'option spécifie s'il faut charger un document de schéma ou juste mettre un espace de noms sous licence, et, si un document de schéma doit être chargé, quelle information doit être utilisée pour le trouver. Défaut : `load-preferring-schemalocation`.
Le comportement est le suivant :

- `load-by-schemalocation`: La valeur de l'attribut `schemaLocation` est utilisée pour situer le schéma, en prenant en compte les mappages de catalogue[47]. Si l'attribut de l'espace de nom est présent, l'espace de noms est importé (licencé).
- `load-preferring-schemalocation`: Si l'attribut `schemaLocation` est présent, il est utilisé en prenant en compte les mappages de catalogue[47]. Si l'attribut `schemaLocation` est présent, la valeur de l'attribut `namespace` est utilisée via un mappage de catalogue[47]. C'est la **valeur par défaut**.
- `load-by-namespace`: La valeur de l'attribut `namespace` est utilisée pour situer le schéma via un mappage de catalogue[47].
- `load-combining-both`: Si soit l'Attribut `namespace` ou l'attribut `schemaLocation` a un mappage de catalogue[47], le mappage est utilisé. Si les deux ont des mappages de catalogue[47], alors la valeur de l'option `--schema-mapping` (option XML/XSD[213]) décide de quel mappage utiliser. Si aucun mappage de catalogue[47] n'est présent, l'attribut `schemaLocation` est utilisé.
- `license-namespace-only`: Le nom d'espace est importé. Aucun document de schéma n'est importé.

▼ schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Spécifie le comportement des attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` : S'il faut charger un document de schéma et, si oui, quelle information doit être utilisée pour la trouver. Défaut : `load-by-schemalocation`.

- La valeur `load-by-schemalocation` utilise l'URL de l'emplacement de schéma[397] dans les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` dans les documents d'instance XML. Il s'agit de la **valeur par défaut**.
- La valeur `load-by-namespace` prend la part d'espace de nom[397] `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` et localise le schéma par le biais d'un mappage de catalogue[47].
- Si `load-combining-both` est utilisé et si soit la partie espace de noms ou la partie URL a un mappage de catalogue[47], alors le mappage de catalogue[47] est utilisé. Si tous deux ont des mappages de catalogue[47], alors la valeur de l'option de `--schema-mapping` (option XML/XSD[213]) de quel mappage utiliser. Si ni l'espace de noms ni l'URL n'a un mappage de catalogue, l'URL est utilisée.
- Si la valeur de l'option est `ignore`, les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` seront ignorés tous les deux.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si l'emplacement de schéma et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, spécifie lequel des deux doit être utilisé pendant la consultation du catalogue. (Si soit l'option `--schemalocation-hints` ou l'option `--schema-imports` a une valeur de `load-combining-both`, et si les parties d'espace de noms et d'URL impliquées ont toutes les deux des mappages de catalogue[47], alors la valeur de cette option spécifie lequel des deux mappages utiliser

(mappage d'espace de noms ou mappage URL ; la valeur `prefer-schemalocation` réfère au mappage URL).) Défaut : `prefer-schemalocation`.

- xinclude

  `--xinclude =` `true|false`
  Active la prise en charge de XML Inclusions (XInclude). La valeur par défaut est `false`. Si `false`, les élément `include` d'XInclude sont ignorés.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- xml-mode

  `--xml-mode =` `wf|id|valid`
  Spécifie le mode de traitement XML à utiliser pour le document d'instance XML : `wf`=wellformed check (vérification de la bonne forme); `id`=bien formé avec des contrôles ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document d'instance chargé depuis les traitements référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

- xml-mode-for-schemas

  `--xml-mode-for-schemas =` `wf|id|valid`
  Spécifie le mode de traitement XML à utiliser pour les documents de schéma XML : `wf`=vérification de la bonne formation ; `id`=contrôles de la bonne formation avec ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document de schéma chargé pendant le traitement référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

- xml-validation-error-as-warning

  `--xml-validation-error-as-warning =` `true|false`
  Si `true`, traite les erreurs de validation en tant qu'avertissements. Si les erreurs sont traitées en tant qu'avertissements, les traitements supplémentaires, comme des transformations XSLT, continueront quelles que soient les erreurs. La valeur par défaut est `false`.

- xsd

  `--xsd =` *`FILE`*
  Spécifie un ou plusieurs documents de Schéma XML à utiliser pour la validation des documents d'instance XML. Ajouter l'option plusieurs fois pour spécifier plus d'un document de schéma.

- xsd-version

  `--xsd-version =` `1.0|1.1|detect`
  Spécifie la version du langage de définition de Schéma W3C (XSD) à utiliser. Le défaut est `1.0`. Cette option peut être utile pour découvrir comment un schéma compatible à 1.0 n'est pas compatible à 1.1. L'option `detect` est une fonction spécifique à Altova. Elle permet la détection de la version du document de Schéma XML (`1.0` ou `1.1`) en lisant la valeur de l'attribut `vc:minVersion` de l'élément du document `<xs:schema>`. Si la valeur de l'attribut `@vc:minVersion` est `1.1`, le schéma est détecté comme étant de version `1.1`. Pour toute autre valeur, ou si l'attribut `@vc:minVersion` est absent, le schéma est détecté comme étant de version `1.0`.

- Catalogues et ressources globales
  - catalog

    `--catalog =` *FILE*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML [47], pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    `--user-catalog =` *FILE*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Active les ressources globales [54]. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *VALUE*
    Spécifie la configuration active de la ressource globale [54] (et active les ressources globales) [54].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *FILE*
    Spécifie le fichier de ressource globale [54] (et active les ressources globales) [54].

- Extensions

  Ces options définissent la gestion des fonctions d'extension spéciales disponibles dans un certain nombre de produits Altova de niveau d'entreprise (comme XMLSpy Enterprise Edition). Leur utilisation est décrite dans les manuels d'utilisateur de ces produits.
  - chartext-disable

    `--chartext-disable =` `true|false`
    Désactive les extensions de graphique. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - dotnetext-disable

    `--dotnetext-disable =` `true|false`
    Désactive les extensions .NET. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - jvm-location

    `--jvm-location =` *FILE*

FILE spécifie l'emplacement de la Java Virtual Machine (DLL sur Windows, objet partagé sur Linux). La JVM est nécessaire si vous utilisez les fonctions d'extension Java dans votre code XSLT/XQuery. La valeur par défaut est `false`.

- javaext-barcode-location

  --javaext-barcode-location = *FILE*
  Spécifie le chemin vers le dossier qui contient le fichier d'extension du code-barres `AltovaBarcodeExtension.jar`. Le chemin doit être donné dans une des formes suivantes :

  - Un URI de fichier, par exemple : `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2024/etc/jar/"`
  - Un chemin Windows avec un échappement par barres obliques inversées, par exemple : `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2024\\etc\jar\\"`

- javaext-disable

  --javaext-disable = `true|false`
  Désactive les extensions Java. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Messages, erreurs, aide, timeout, version

  - error-format

    --error-format = `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    --error-limit = *N | unlimited*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    --info-limit = *N | unlimited*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - log-output

--log-output = *FILE*
Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.3.3 valxquery

La commande **valxquery** prend un fichier XQuery en tant que son argument unique et le valide.

```
raptorxml valxquery [options] XQuery-File
```

- L'argument ***XQuery-File*** est le chemin et le nom du fichier XQuery à valider.

Exemples

Exemples de la commande **valxquery** :

- **raptorxml** valxquery c:\Test.xquery
- **raptorxml** valxquery --xquery-version=1 c:\Test.xquery

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (raptorxml and raptorxmlserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (RaptorXML) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est TRUE. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.
.

▼ Traitement XQuery

▼ omit-xml-declaration

--omit-xml-declaration = true|false
Option de sérialisation pour spécifier si la déclaration XML doit être omise de la sortie ou pas. Si true, il n'y aura pas de déclaration XML dans le document de sortie. Si false, une déclaration XML sera incluse. La valeur par défaut est false.
Note : les valeurs d'option booléennes sont configurées sur true si l'option est spécifiée sans une valeur.

▼ xquery-version

--xquery-version = 1|1.0|3|3.0|3.1
Spécifie si le processeur XQuery devrait utiliser XSLT XQuery 1.0 ou XQuery 3.0. La valeur par défaut est 3.1.

- Schéma XML et instance XML
  - load-xml-with-psvi

    --load-xml-with-psvi = `true|false`
    Active la validation des fichiers XML d'entrée et génère des informations de post-schema-validation pour les fichiers. La valeur par défaut est : `true`.

  - schema-imports

    --schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Spécifie le comportement des éléments `xs:import`, chacun d'entre eux ayant un attribut `namespace` optionnel et un attribut `schemaLocation` optionnel : `<import namespace="someNS" schemaLocation="someURL">`. L'option spécifie s'il faut charger un document de schéma ou juste mettre un espace de noms sous licence, et, si un document de schéma doit être chargé, quelle information doit être utilisée pour le trouver. Défaut : `load-preferring-schemalocation`.
    Le comportement est le suivant :

    - `load-by-schemalocation`: La valeur de l'attribut `schemaLocation` est utilisée pour situer le schéma, en prenant en compte les mappages de catalogue[47]. Si l'attribut de l'espace de nom est présent, l'espace de noms est importé (licencé).
    - `load-preferring-schemalocation`: Si l'attribut `schemaLocation` est présent, il est utilisé en prenant en compte les mappages de catalogue[47]. Si l'attribut `schemaLocation` est présent, la valeur de l'attribut `namespace` est utilisée via un mappage de catalogue[47]. C'est la **valeur par défaut**.
    - `load-by-namespace`: La valeur de l'attribut `namespace` est utilisée pour situer le schéma via un mappage de catalogue[47].
    - `load-combining-both`: Si soit l'Attribut `namespace` ou l'attribut `schemaLocation` a un mappage de catalogue[47], le mappage est utilisé. Si les deux ont des mappages de catalogue[47], alors la valeur de l'option `--schema-mapping` (option XML/XSD[213]) décide de quel mappage utiliser. Si aucun mappage de catalogue[47] n'est présent, l'attribut `schemaLocation` est utilisé.
    - `license-namespace-only`: Le nom d'espace est importé. Aucun document de schéma n'est importé.

  - schemalocation-hints

    --schemalocation-hints = `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
    Spécifie le comportement des attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` : S'il faut charger un document de schéma et, si oui, quelle information doit être utilisée pour la trouver. Défaut : `load-by-schemalocation`.

    - La valeur `load-by-schemalocation` utilise l'URL de l'emplacement de schéma[397] dans les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` dans les documents d'instance XML. Il s'agit de la **valeur par défaut**.
    - La valeur `load-by-namespace` prend la part d'espace de nom[397] `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` et localise le schéma par le biais d'un mappage de catalogue[47].
    - Si `load-combining-both` est utilisé et si soit la partie espace de noms ou la partie URL a un mappage de catalogue[47], alors le mappage de catalogue[47] est utilisé. Si tous deux ont des mappages de catalogue[47], alors la valeur de l'option de `--schema-mapping` (option XML/XSD[213]) de quel mappage utiliser. Si ni l'espace de noms ni l'URL n'a un mappage de catalogue, l'URL est utilisée.

- Si la valeur de l'option est `ignore`, les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` seront ignorés tous les deux.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si l'emplacement de schéma et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, spécifie lequel des deux doit être utilisé pendant la consultation du catalogue. (Si soit l'option `--schemalocation-hints` ou l'option `--schema-imports` a une valeur de `load-combining-both`, et si les parties d'espace de noms et d'URL impliquées ont toutes les deux des mappages de catalogue[47], alors la valeur de cette option spécifie lequel des deux mappages utiliser (mappage d'espace de noms ou mappage URL ; la valeur `prefer-schemalocation` réfère au mappage URL).) Défaut : `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`
Active la prise en charge de XML Inclusions (XInclude). La valeur par défaut est `false`. Si `false`, les élément `include` d'XInclude sont ignorés.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ xml-mode

`--xml-mode =` `wf|id|valid`
Spécifie le mode de traitement XML à utiliser pour le document d'instance XML : `wf`=wellformed check (vérification de la bonne forme); `id`=bien formé avec des contrôles ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document d'instance chargé depuis les traitements référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`
Spécifie le mode de traitement XML à utiliser pour les documents de schéma XML : `wf`=vérification de la bonne formation ; `id`=contrôles de la bonne formation avec ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document de schéma chargé pendant le traitement référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

▼ xpath-static-type-errors-as-warnings

`--xpath-static-type-errors-as-warnings =` `true|false`
Si `true`, passe à une version antérieure d'avertissements de tous les types d'erreurs qui sont détectés dans le contexte statique XPath. Alors qu'une erreur entraînerait l'échec de l'exécution, un avertissement permettrait la poursuite du traitement. Le défaut est `false`.

▼ xsd-version

`--xsd-version =` `1.0|1.1|detect`
Spécifie la version du langage de définition de Schéma W3C (XSD) à utiliser. Le défaut est `1.0`. Cette option peut être utile pour découvrir comment un schéma compatible à 1.0 n'est pas compatible à 1.1. L'option `detect` est une fonction spécifique à Altova. Elle permet la détection de la version du document de Schéma XML (`1.0` ou `1.1`) en lisant la valeur de l'attribut `vc:minVersion` de l'élément

du document `<xs:schema>`. Si la valeur de l'attribut `@vc:minVersion` est `1.1`, le schéma est détecté comme étant de version `1.1`. Pour toute autre valeur, ou si l'attribut `@vc:minVersion` est absent, le schéma est détecté comme étant de version `1.0`.

Catalogues et ressources globales

catalog

`--catalog =` *FILE*
Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML [47], pour plus d'informations à propos du travail avec les catalogues.

user-catalog

`--user-catalog =` *FILE*
Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

enable-globalresources

`--enable-globalresources =` `true|false`
Active les ressources globales [54]. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

globalresourceconfig [gc]

`--gc | --globalresourceconfig =` *VALUE*
Spécifie la configuration active de la ressource globale [54] (et active les ressources globales) [54].

globalresourcefile [gr]

`--gr | --globalresourcefile =` *FILE*
Spécifie le fichier de ressource globale [54] (et active les ressources globales) [54].

Extensions

Ces options définissent la gestion des fonctions d'extension spéciales disponibles dans un certain nombre de produits Altova de niveau d'entreprise (comme XMLSpy Enterprise Edition). Leur utilisation est décrite dans les manuels d'utilisateur de ces produits.

chartext-disable

`--chartext-disable =` `true|false`
Désactive les extensions de graphique. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

dotnetext-disable

`--dotnetext-disable =` `true|false`

Désactive les extensions .NET. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- jvm-location

  `--jvm-location =` *`FILE`*
  `FILE` spécifie l'emplacement de la Java Virtual Machine (DLL sur Windows, objet partagé sur Linux). La JVM est nécessaire si vous utilisez les fonctions d'extension Java dans votre code XSLT/XQuery. La valeur par défaut est `false`.

- javaext-barcode-location

  `--javaext-barcode-location =` *`FILE`*
  Spécifie le chemin vers le dossier qui contient le fichier d'extension du code-barres `AltovaBarcodeExtension.jar`. Le chemin doit être donné dans une des formes suivantes :

  - Un URI de fichier, par exemple : `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2024/etc/jar/"`
  - Un chemin Windows avec un échappement par barres obliques inversées, par exemple : `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2024\\etc\\jar\\"`

- javaext-disable

  `--javaext-disable =` `true|false`
  Désactive les extensions Java. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Messages, erreurs, aide, timeout, version

  - error-format

    `--error-format =` `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  --help
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  --log-output = *FILE*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.3.4 valxqueryupdate

La commande **valxqueryupdate** prend un fichier XQuery en tant que son argument unique et le valide.

```
raptorxml valxqueryupdate [options] XQuery-File
```

- L'argument ***XQuery-File*** est le chemin et le nom du fichier XQuery à valider.

Exemples

Exemples de la commande **valxqueryupdate**

- **raptorxml** valxqueryupdate c:\Test.xqu
- **raptorxml** valxqueryupdate --xquery-update-version=1 c:\Test.xqu

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (raptorxml et raptorxmlserver) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (RaptorXML) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est TRUE. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.
.

▼ Traitement XQuery

▼ omit-xml-declaration

--omit-xml-declaration = true|false
Option de sérialisation pour spécifier si la déclaration XML doit être omise de la sortie ou pas. Si true, il n'y aura pas de déclaration XML dans le document de sortie. Si false, une déclaration XML sera incluse. La valeur par défaut est false.

Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- xquery-update-version

  `--xquery-update-version =` `1|1.0|3|3.0`
  Spécifie si le processeur XQuery devrait utiliser XQuery Update Facility 1.0 ou XQuery Update Facility 3.0. La valeur par défaut est `3`.

- Schéma XML et instance XML
  - load-xml-with-psvi

    `--load-xml-with-psvi =` `true|false`
    Active la validation des fichiers XML d'entrée et génère des informations de post-schema-validation pour les fichiers. La valeur par défaut est : `true`.

  - schema-imports

    `--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Spécifie le comportement des éléments `xs:import`, chacun d'entre eux ayant un attribut `namespace` optionnel et un attribut `schemaLocation` optionnel : `<import namespace="someNS" schemaLocation="someURL">`. L'option spécifie s'il faut charger un document de schéma ou juste mettre un espace de noms sous licence, et, si un document de schéma doit être chargé, quelle information doit être utilisée pour le trouver. Défaut : `load-preferring-schemalocation`.
    Le comportement est le suivant :

    - `load-by-schemalocation`: La valeur de l'attribut `schemaLocation` est utilisée pour situer le schéma, en prenant en compte les mappages de catalogue[47]. Si l'attribut de l'espace de nom est présent, l'espace de noms est importé (licencé).
    - `load-preferring-schemalocation`: Si l'attribut `schemaLocation` est présent, il est utilisé en prenant en compte les mappages de catalogue[47]. Si l'attribut `schemaLocation` est présent, la valeur de l'attribut `namespace` est utilisée via un mappage de catalogue[47]. C'est la **valeur par défaut**.
    - `load-by-namespace`: La valeur de l'attribut `namespace` est utilisée pour situer le schéma via un mappage de catalogue[47].
    - `load-combining-both`: Si soit l'Attribut `namespace` ou l'attribut `schemaLocation` a un mappage de catalogue[47], le mappage est utilisé. Si les deux ont des mappages de catalogue[47], alors la valeur de l'option `--schema-mapping` (option XML/XSD[213]) décide de quel mappage utiliser. Si aucun mappage de catalogue[47] n'est présent, l'attribut `schemaLocation` est utilisé.
    - `license-namespace-only`: Le nom d'espace est importé. Aucun document de schéma n'est importé.

  - schemalocation-hints

    `--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
    Spécifie le comportement des attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` : S'il faut charger un document de schéma et, si oui, quelle information doit être utilisée pour la trouver. Défaut : `load-by-schemalocation`.

- La valeur `load-by-schemalocation` utilise l'URL de l'emplacement de schéma[397] dans les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` dans les documents d'instance XML. Il s'agit de la **valeur par défaut**.
- La valeur `load-by-namespace` prend la part d'espace de nom[397] `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` et localise le schéma par le biais d'un mappage de catalogue[47].
- Si `load-combining-both` est utilisé et si soit la partie espace de noms ou la partie URL a un mappage de catalogue[47], alors le mappage de catalogue[47] est utilisé. Si tous deux ont des mappages de catalogue[47], alors la valeur de l'option de `--schema-mapping` (option XML/XSD[213]) de quel mappage utiliser. Si ni l'espace de noms ni l'URL n'a un mappage de catalogue, l'URL est utilisée.
- Si la valeur de l'option est `ignore`, les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` seront ignorés tous les deux.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si l'emplacement de schéma et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, spécifie lequel des deux doit être utilisé pendant la consultation du catalogue. (Si soit l'option `--schemalocation-hints` ou l'option `--schema-imports` a une valeur de `load-combining-both`, et si les parties d'espace de noms et d'URL impliquées ont toutes les deux des mappages de catalogue[47], alors la valeur de cette option spécifie lequel des deux mappages utiliser (mappage d'espace de noms ou mappage URL ; la valeur `prefer-schemalocation` réfère au mappage URL).) Défaut : `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`
Active la prise en charge de XML Inclusions (XInclude). La valeur par défaut est `false`. Si `false`, les élément `include` d'XInclude sont ignorés.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ xml-mode

`--xml-mode =` `wf|id|valid`
Spécifie le mode de traitement XML à utiliser pour le document d'instance XML : `wf`=wellformed check (vérification de la bonne forme); `id`=bien formé avec des contrôles ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document d'instance chargé depuis les traitements référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`
Spécifie le mode de traitement XML à utiliser pour les documents de schéma XML : `wf`=vérification de la bonne formation ; `id`=contrôles de la bonne formation avec ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document de schéma chargé pendant le traitement référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

▼ xsd-version

`--xsd-version =` `1.0|1.1|detect`

Spécifie la version du langage de définition de Schéma W3C (XSD) à utiliser. Le défaut est `1.0`. Cette option peut être utile pour découvrir comment un schéma compatible à 1.0 n'est pas compatible à 1.1. L'option `detect` est une fonction spécifique à Altova. Elle permet la détection de la version du document de Schéma XML (`1.0` ou `1.1`) en lisant la valeur de l'attribut `vc:minVersion` de l'élément du document `<xs:schema>`. Si la valeur de l'attribut `@vc:minVersion` est `1.1`, le schéma est détecté comme étant de version `1.1`. Pour toute autre valeur, ou si l'attribut `@vc:minVersion` est absent, le schéma est détecté comme étant de version `1.0`.

Catalogues et ressources globales

catalog

`--catalog =` *FILE*
Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML [47], pour plus d'informations à propos du travail avec les catalogues.

user-catalog

`--user-catalog =` *FILE*
Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

enable-globalresources

`--enable-globalresources =` `true|false`
Active les ressources globales [54]. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

globalresourceconfig [gc]

`--gc | --globalresourceconfig =` *VALUE*
Spécifie la configuration active de la ressource globale [54] (et active les ressources globales) [54].

globalresourcefile [gr]

`--gr | --globalresourcefile =` *FILE*
Spécifie le fichier de ressource globale [54] (et active les ressources globales) [54].

Extensions

Ces options définissent la gestion des fonctions d'extension spéciales disponibles dans un certain nombre de produits Altova de niveau d'entreprise (comme XMLSpy Enterprise Edition). Leur utilisation est décrite dans les manuels d'utilisateur de ces produits.

chartext-disable

`--chartext-disable =` `true|false`
Désactive les extensions de graphique. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- dotnetext-disable

  --dotnetext-disable = `true|false`
  Désactive les extensions .NET. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- jvm-location

  --jvm-location = *FILE*
  FILE spécifie l'emplacement de la Java Virtual Machine (DLL sur Windows, objet partagé sur Linux). La JVM est nécessaire si vous utilisez les fonctions d'extension Java dans votre code XSLT/XQuery. La valeur par défaut est `false`.

- javaext-barcode-location

  --javaext-barcode-location = *FILE*
  Spécifie le chemin vers le dossier qui contient le fichier d'extension du code-barres `AltovaBarcodeExtension.jar`. Le chemin doit être donné dans une des formes suivantes :

  - Un URI de fichier, par exemple : `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2024/etc/jar/"`
  - Un chemin Windows avec un échappement par barres obliques inversées, par exemple : `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2024\\etc\\jar\\"`

- javaext-disable

  --javaext-disable = `true|false`
  Désactive les extensions Java. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Messages, erreurs, aide, timeout, version
  - error-format

    --error-format = `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    --error-limit = *N | unlimited*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    --info-limit = *N | unlimited*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement

continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  `--help`
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  `--log-output =` *`FILE`*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  `--network-timeout =` *`VALEUR`*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  `--verbose =` `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  `--verbose-output =` *`FILE`*
  Écrit la sortie verbeuse sur *`FILE`*.

- version

  `--version`
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.4 Commandes XSLT

Les commandes XSLT sont :

- xslt[121] : pour transformer les documents XML avec un document XSLT
- valxslt[129] : pour valider des documents XSLT

5.4.1 xslt

La commande **xslt** prend un fichier XSLT en tant que son argument unique et l'utilise pour transformer un fichier XML d'entrée pour produire un fichier de sortie. Les fichiers d'entrée et de sortie sont spécifiés en tant qu'options[218].

```
raptorxml xslt [options] XSLT-File
```

- L'argument *XSLT-File* est le chemin et le nom du fichier XSLT à utiliser pour la transformation.
- Un fichier XML d'entrée (--input[218]) ou un point d'entrée de modèle nommé (--template-entry-point[218]) est nécessaire.
- Pour transformer des données JSON, charger les données JSON via la fonction **json-doc($path)** de XPath 3.1, et utiliser l'option **--initial-match-selection** de la commande **xslt**. Voir le dernier item dans les exemples donnés ci-dessous.
- Si aucune option --output[218] n'est spécifiée, la sortie est écrite dans la sortie standard. Vous pouvez utiliser XSLT 1.0, 2.0 ou 3.0. XSLT 3.0 est la spécification utilisée par défaut.

Exemples

Exemples de la commande **xslt** :

- **raptorxml** xslt --input=c:\Test.xml --output=c:\Output.xml c:\Test.xslt
- **raptorxml** xslt --template-entry-point=StartTemplate --output=c:\Output.xml c:\Test.xslt
- **raptorxml** xslt --input=c:\Test.xml --output=c:\Output.xml --param=date://node[1]/@att1 --p=title:'stringwithoutspace' --param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
- **raptorxml** xslt --initial-match-selection=json-doc('MyData.json',**map**{'liberal':true()}) --output=c:\MyData.xml c:\Test.xslt
- **raptorxml** xslt --initial-match-selection="**json-doc(**'MyData.json',**map**{'liberal':true()}**)**" --output=c:\MyData.xml c:\Test.xslt (Si le string d'argument json-doc contient des espaces, alors mettez toute la valeur json-doc entre guillemets.)

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (raptorxml and raptorxmlserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (RaptorXML) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

- Backslashes, spaces, and special characters on Windows systems

  On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, `"My File"`. Note, however, that a backslash followed by a double-quotation mark (for example, `"C:\My directory\"`) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence `\"` stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: `\\"`. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: `"C:\My Directory\ \"`.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Traitement XSLT
  - indent-characters

    `--indent-characters =` *VALUE*
    Spécifie le string de caractère à utiliser en tant que retrait.

  - function-param

    `--function-param =` *VALUE*
    Spécifie les fonctions qui seront passées à la fonction initiale. Pour spécifier plus d'une fonction, utiliser l'option plusieurs fois. Attention, l'ordre est important.

  - global-context-item

    `--global-context-item =` *VALUE*
    Spécifie l'item de contexte qui doit être utilisé pour évaluer les variables globales.

  - initial-function

    `--initial-function =` *VALUE*
    Le nom d'une fonction qui doit être exécutée en tant que le point d'entrée de la transformation.

  - initial-match-selection

    `--initial-match-selection =` *VALUE*
    Spécifie la valeur (séquence) de la sélection de correspondance initiale.

  - initial-mode, template-mode

    `--initial-template, --template-entry-point =` *VALUE*

Spécifie le mode de modèle à utiliser pour la transformation.

- initial-template, template-entry-point

  `--initial-template, --template-entry-point =` *VALUE*
  Donne le nom d'un modèle nommé dans la feuille de style XSLT qui est le point d'entrée de la transformation.

- input

  `--input =` *FILE*
  L'URL du fichier XML à transformer.

- output, xsltoutput

  `output =` *FILE*, **`xsltoutput =`** *FILE*
  L'URL du fichier sortie-primaire. Par exemple, dans le cas d'une sortie HTML de fichier multiple, le fichier de sortie primaire sera l'emplacement du fichier HTML du point d'entrée. Les fichiers de sortie supplémentaires, comme des fichiers d'image générée, sont rapportés en tant que `xslt-additional-output-files`. Si aucune option `--output` ou `--xsltoutput` n'est spécifiée, la sortie est écrite sur la sortie standard.

- param [p]

  `--p | --param =` *KEY:VALUE*

  XQuery

  Spécifie la valeur d'un paramètre externe. Un paramètre externe est déclaré dans le document XQuery avec la déclaration `declare variable` suivi par un nom de variable puis le mot-clé `external` suivi par le point-virgule de fin de ligne. Par exemple :

  ```
  declare variable $foo as xs:string external;
  ```

  Le mot-clé `external`, `$foo` devient un paramètre externe, dont la valeur est passée pendant le runtime depuis une source externe. Le paramètre externe reçoit une value avec la commande CLI. Par exemple :

  ```
  --param=foo:'MyName'
  ```

  Dans l'instruction de description ci-dessus, *KEY* est le nom de paramètre externe, *VALUE* est la valeur du paramètre externe, donné en tant qu'une expression XPath. Les noms de paramètres utilisés sur la CLI doivent être déclarés dans le document XQuery. Si plusieurs paramètres externes sont des valeurs passées sur la CLI, chacun doit être recevoir une option séparée `--param`. Les doubles guillemets doivent être utilisés si l'expression XPath contient des espaces.

  XSLT

  Spécifie paramètre de feuille de style global. *KEY* est le nom de paramètre, *VALUE* est une expression XPath qui fournit la valeur de paramètre. Les noms de paramètre utilisés sur la CLI doivent être déclarés dans la feuille de style. Si plusieurs paramètres multiples sont utilisés, le changement `--param` doit être utilisé avant chaque paramètre. Les double guillemets doivent être utilisés autour de l'expression XPath si elle contient un espace ; que l'espace se trouve dans l'expression XPath elle-même ou dans un littéral de string dans l'expression. Par exemple :

  ```
  raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
  param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
  ```

```
param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
```

- streaming-serialization-enabled

  --streaming-serialization-enabled = `true|false`
  Active la sérialisation de streaming. La valeur par défaut est `true`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- template-param

  --template-param = *KEY:VALUE*
  Spécifie les paramètres qui seront passés uniquement au modèle initial (et pas à un appel de modèle descendant). Pour spécifier plusieurs paramètres, utiliser l'option une fois pour chaque paramètre.

- tunnel-param

  --tunnel-param = *KEY:VALUE*
  Spécifie les paramètres qui seront passés au modèle initial et aux appels de modèle descendants. Pour spécifier plusieurs paramètres, utiliser l'option une fois pour chaque paramètre.

- xpath-static-type-errors-as-warnings

  --xpath-static-type-errors-as-warnings = `true|false`
  Si `true`, passe à une version antérieure d'avertissements de tous les types d'erreurs qui sont détectés dans le contexte statique XPath. Alors qu'une erreur entraînerait l'échec de l'exécution, un avertissement permettrait la poursuite du traitement. Le défaut est `false`.

- xslt-version

  --xslt-version = `1|1.0|2|2.0|3|3.0|3.1`
  Spécifie si le processeur XSLT devrait utiliser XSLT 1.0, XSLT 2.0 ou XSLT 3.0. La valeur par défaut est `3`.

- Schéma XML et instance XML
  - load-xml-with-psvi

    --load-xml-with-psvi = `true|false`
    Active la validation des fichiers XML d'entrée et génère des informations de post-schema-validation pour les fichiers. La valeur par défaut est : `true`.

  - schema-imports

    --schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Spécifie le comportement des éléments `xs:import`, chacun d'entre eux ayant un attribut `namespace` optionnel et un attribut `schemaLocation` optionnel : `<import namespace="someNS" schemaLocation="someURL">`. L'option spécifie s'il faut charger un document de schéma ou juste mettre un espace de noms sous licence, et, si un document de schéma doit être chargé, quelle information doit être utilisée pour le trouver. Défaut : `load-preferring-schemalocation`.
    Le comportement est le suivant :

- `load-by-schemalocation`: La valeur de l'attribut `schemaLocation` est utilisée pour situer le schéma, en prenant en compte les mappages de catalogue[47]. Si l'attribut de l'espace de nom est présent, l'espace de noms est importé (licencé).
- `load-preferring-schemalocation`: Si l'attribut `schemaLocation` est présent, il est utilisé en prenant en compte les mappages de catalogue[47]. Si l'attribut `schemaLocation` est présent, la valeur de l'attribut `namespace` est utilisée via un mappage de catalogue[47]. C'est la **valeur par défaut**.
- `load-by-namespace`: La valeur de l'attribut `namespace` est utilisée pour situer le schéma via un mappage de catalogue[47].
- `load-combining-both`: Si soit l'Attribut `namespace` ou l'attribut `schemaLocation` a un mappage de catalogue[47], le mappage est utilisé. Si les deux ont des mappages de catalogue[47], alors la valeur de l'option `--schema-mapping` (option XML/XSD[213]) décide de quel mappage utiliser. Si aucun mappage de catalogue[47] n'est présent, l'attribut `schemaLocation` est utilisé.
- `license-namespace-only`: Le nom d'espace est importé. Aucun document de schéma n'est importé.

▼ schemalocation-hints

`--schemalocation-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Spécifie le comportement des attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` : S'il faut charger un document de schéma et, si oui, quelle information doit être utilisée pour la trouver. Défaut : `load-by-schemalocation`.

- La valeur `load-by-schemalocation` utilise l'URL de l'emplacement de schéma[397] dans les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` dans les documents d'instance XML. Il s'agit de la **valeur par défaut**.
- La valeur `load-by-namespace` prend la part d'espace de nom[397] `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` et localise le schéma par le biais d'un mappage de catalogue[47].
- Si `load-combining-both` est utilisé et si soit la partie espace de noms ou la partie URL a un mappage de catalogue[47], alors le mappage de catalogue[47] est utilisé. Si tous deux ont des mappages de catalogue[47], alors la valeur de l'option de `--schema-mapping` (option XML/XSD[213]) de quel mappage utiliser. Si ni l'espace de noms ni l'URL n'a un mappage de catalogue, l'URL est utilisée.
- Si la valeur de l'option est `ignore`, les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` seront ignorés tous les deux.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si l'emplacement de schéma et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, spécifie lequel des deux doit être utilisé pendant la consultation du catalogue. (Si soit l'option `--schemalocation-hints` ou l'option `--schema-imports` a une valeur de `load-combining-both`, et si les parties d'espace de noms et d'URL impliquées ont toutes les deux des mappages de catalogue[47], alors la valeur de cette option spécifie lequel des deux mappages utiliser (mappage d'espace de noms ou mappage URL ; la valeur `prefer-schemalocation` réfère au mappage URL).) Défaut : `prefer-schemalocation`.

▼ xinclude

--xinclude = `true|false`
Active la prise en charge de XML Inclusions (XInclude). La valeur par défaut est `false`. Si `false`, les élément `include` d'XInclude sont ignorés.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- xml-mode

  --xml-mode = `wf|id|valid`
  Spécifie le mode de traitement XML à utiliser pour le document d'instance XML : `wf`=wellformed check (vérification de la bonne forme); `id`=bien formé avec des contrôles ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document d'instance chargé depuis les traitements référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

- xml-mode-for-schemas

  --xml-mode-for-schemas = `wf|id|valid`
  Spécifie le mode de traitement XML à utiliser pour les documents de schéma XML : `wf`=vérification de la bonne formation ; `id`=contrôles de la bonne formation avec ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document de schéma chargé pendant le traitement référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

- xml-validation-error-as-warning

  --xml-validation-error-as-warning = `true|false`
  Si `true`, traite les erreurs de validation en tant qu'avertissements. Si les erreurs sont traitées en tant qu'avertissements, les traitements supplémentaires, comme des transformations XSLT, continueront quelles que soient les erreurs. La valeur par défaut est `false`.

- xsd

  --xsd = *FILE*
  Spécifie un ou plusieurs documents de Schéma XML à utiliser pour la validation des documents d'instance XML. Ajouter l'option plusieurs fois pour spécifier plus d'un document de schéma.

- xsd-version

  --xsd-version = `1.0|1.1|detect`
  Spécifie la version du langage de définition de Schéma W3C (XSD) à utiliser. Le défaut est `1.0`. Cette option peut être utile pour découvrir comment un schéma compatible à 1.0 n'est pas compatible à 1.1. L'option `detect` est une fonction spécifique à Altova. Elle permet la détection de la version du document de Schéma XML (`1.0` ou `1.1`) en lisant la valeur de l'attribut `vc:minVersion` de l'élément du document `<xs:schema>`. Si la valeur de l'attribut `@vc:minVersion` est `1.1`, le schéma est détecté comme étant de version `1.1`. Pour toute autre valeur, ou si l'attribut `@vc:minVersion` est absent, le schéma est détecté comme étant de version `1.0`.

- Catalogues et ressources globales
  - catalog

    --catalog = *FILE*

Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML [47], pour plus d'informations à propos du travail avec les catalogues.

- user-catalog

  `--user-catalog =` *`FILE`*
  Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

- enable-globalresources

  `--enable-globalresources =` `true|false`
  Active les ressources globales [54]. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- globalresourceconfig [gc]

  `--gc | --globalresourceconfig =` *`VALUE`*
  Spécifie la configuration active de la ressource globale [54] (et active les ressources globales) [54].

- globalresourcefile [gr]

  `--gr | --globalresourcefile =` *`FILE`*
  Spécifie le fichier de ressource globale [54] (et active les ressources globales) [54].

- Extensions

  Ces options définissent la gestion des fonctions d'extension spéciales disponibles dans un certain nombre de produits Altova de niveau d'entreprise (comme XMLSpy Enterprise Edition). Leur utilisation est décrite dans les manuels d'utilisateur de ces produits.

  - chartext-disable

    `--chartext-disable =` `true|false`
    Désactive les extensions de graphique. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - dotnetext-disable

    `--dotnetext-disable =` `true|false`
    Désactive les extensions .NET. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - jvm-location

    `--jvm-location =` *`FILE`*
    `FILE` spécifie l'emplacement de la Java Virtual Machine (DLL sur Windows, objet partagé sur Linux). La JVM est nécessaire si vous utilisez les fonctions d'extension Java dans votre code XSLT/XQuery. La valeur par défaut est `false`.

- javaext-barcode-location

  --javaext-barcode-location = *FILE*
  Spécifie le chemin vers le dossier qui contient le fichier d'extension du code-barres `AltovaBarcodeExtension.jar`. Le chemin doit être donné dans une des formes suivantes :

  - Un URI de fichier, par exemple : `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2024/etc/jar/"`
  - Un chemin Windows avec un échappement par barres obliques inversées, par exemple : `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2024\\etc\\jar\\"`

- javaext-disable

  --javaext-disable = `true|false`
  Désactive les extensions Java. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Messages, erreurs, aide, timeout, version
  - error-format

    --error-format = `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    --error-limit = *N | unlimited*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    --info-limit = *N | unlimited*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - log-output

    --log-output = *FILE*
    Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.4.2 valxslt

La commande **valxslt** prend un fichier XSLT en tant que son argument unique et le valide.

```
raptorxml valxslt [options] XSLT-File
```

- L'argument ***XSLT-File*** est le chemin et le nom du fichier XSLT à utiliser pour la validation.
- La validation peut s'effectuer conformément à la spécification XSLT 1.0, 2.0 ou 3.0. XSLT 3.0 est la spécification utilisée par défaut.

Exemples

Exemples de la commande **valxslt**

- **raptorxml** valxslt c:\Test.xslt

- **raptorxml** valxslt --xslt-version=2 c:\Test.xslt

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (raptorxml and raptorxmlserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (RaptorXML) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est TRUE. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.
.

▼ Traitement XSLT

▼ initial-mode, template-mode

--initial-template, --template-entry-point = *VALUE*
Spécifie le mode de modèle à utiliser pour la transformation.

▼ initial-template, template-entry-point

--initial-template, --template-entry-point = *VALUE*
Donne le nom d'un modèle nommé dans la feuille de style XSLT qui est le point d'entrée de la transformation.

▼ xslt-version

--xslt-version = 1|1.0|2|2.0|3|3.0|3.1
Spécifie si le processeur XSLT devrait utiliser XSLT 1.0, XSLT 2.0 ou XSLT 3.0. La valeur par défaut est 3.

- Schéma XML et instance XML
  - load-xml-with-psvi

    --load-xml-with-psvi = `true|false`
    Active la validation des fichiers XML d'entrée et génère des informations de post-schema-validation pour les fichiers. La valeur par défaut est : `true`.

  - schema-imports

    --schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
    Spécifie le comportement des éléments `xs:import`, chacun d'entre eux ayant un attribut `namespace` optionnel et un attribut `schemaLocation` optionnel : `<import namespace="someNS" schemaLocation="someURL">`. L'option spécifie s'il faut charger un document de schéma ou juste mettre un espace de noms sous licence, et, si un document de schéma doit être chargé, quelle information doit être utilisée pour le trouver. Défaut : `load-preferring-schemalocation`.
    Le comportement est le suivant :

    - `load-by-schemalocation`: La valeur de l'attribut `schemaLocation` est utilisée pour situer le schéma, en prenant en compte les mappages de catalogue[47]. Si l'attribut de l'espace de nom est présent, l'espace de noms est importé (licencé).
    - `load-preferring-schemalocation`: Si l'attribut `schemaLocation` est présent, il est utilisé en prenant en compte les mappages de catalogue[47]. Si l'attribut `schemaLocation` est présent, la valeur de l'attribut `namespace` est utilisée via un mappage de catalogue[47]. C'est la **valeur par défaut**.
    - `load-by-namespace`: La valeur de l'attribut `namespace` est utilisée pour situer le schéma via un mappage de catalogue[47].
    - `load-combining-both`: Si soit l'Attribut `namespace` ou l'attribut `schemaLocation` a un mappage de catalogue[47], le mappage est utilisé. Si les deux ont des mappages de catalogue[47], alors la valeur de l'option `--schema-mapping` (option XML/XSD[213]) décide de quel mappage utiliser. Si aucun mappage de catalogue[47] n'est présent, l'attribut `schemaLocation` est utilisé.
    - `license-namespace-only`: Le nom d'espace est importé. Aucun document de schéma n'est importé.

  - schemalocation-hints

    --schemalocation-hints = `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
    Spécifie le comportement des attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` : S'il faut charger un document de schéma et, si oui, quelle information doit être utilisée pour la trouver. Défaut : `load-by-schemalocation`.

    - La valeur `load-by-schemalocation` utilise l'URL de l'emplacement de schéma[397] dans les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` dans les documents d'instance XML. Il s'agit de la **valeur par défaut**.
    - La valeur `load-by-namespace` prend la part d'espace de nom[397] `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` et localise le schéma par le biais d'un mappage de catalogue[47].
    - Si `load-combining-both` est utilisé et si soit la partie espace de noms ou la partie URL a un mappage de catalogue[47], alors le mappage de catalogue[47] est utilisé. Si tous deux ont des mappages de catalogue[47], alors la valeur de l'option de `--schema-mapping` (option XML/XSD[213]) de quel mappage utiliser. Si ni l'espace de noms ni l'URL n'a un mappage de catalogue, l'URL est utilisée.

- Si la valeur de l'option est `ignore`, les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` seront ignorés tous les deux.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`
Si l'emplacement de schéma et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, spécifie lequel des deux doit être utilisé pendant la consultation du catalogue. (Si soit l'option `--schemalocation-hints` ou l'option `--schema-imports` a une valeur de `load-combining-both`, et si les parties d'espace de noms et d'URL impliquées ont toutes les deux des mappages de catalogue[47], alors la valeur de cette option spécifie lequel des deux mappages utiliser (mappage d'espace de noms ou mappage URL ; la valeur `prefer-schemalocation` réfère au mappage URL).) Défaut : `prefer-schemalocation`.

▼ xinclude

`--xinclude =` `true|false`
Active la prise en charge de XML Inclusions (XInclude). La valeur par défaut est `false`. Si `false`, les élément `include` d'XInclude sont ignorés.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ xml-mode

`--xml-mode =` `wf|id|valid`
Spécifie le mode de traitement XML à utiliser pour le document d'instance XML : `wf`=wellformed check (vérification de la bonne forme); `id`=bien formé avec des contrôles ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document d'instance chargé depuis les traitements référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`
Spécifie le mode de traitement XML à utiliser pour les documents de schéma XML : `wf`=vérification de la bonne formation ; `id`=contrôles de la bonne formation avec ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document de schéma chargé pendant le traitement référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

▼ xpath-static-type-errors-as-warnings

`--xpath-static-type-errors-as-warnings =` `true|false`
Si `true`, passe à une version antérieure d'avertissements de tous les types d'erreurs qui sont détectés dans le contexte statique XPath. Alors qu'une erreur entraînerait l'échec de l'exécution, un avertissement permettrait la poursuite du traitement. Le défaut est `false`.

▼ xsd-version

`--xsd-version =` `1.0|1.1|detect`
Spécifie la version du langage de définition de Schéma W3C (XSD) à utiliser. Le défaut est `1.0`. Cette option peut être utile pour découvrir comment un schéma compatible à 1.0 n'est pas compatible à 1.1. L'option `detect` est une fonction spécifique à Altova. Elle permet la détection de la version du document de Schéma XML (`1.0` ou `1.1`) en lisant la valeur de l'attribut `vc:minVersion` de l'élément

du document <xs:schema>. Si la valeur de l'attribut @vc:minVersion est 1.1, le schéma est détecté comme étant de version 1.1. Pour toute autre valeur, ou si l'attribut @vc:minVersion est absent, le schéma est détecté comme étant de version 1.0.

Catalogues et ressources globales

catalog

--catalog = *FILE*

Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

user-catalog

--user-catalog = *FILE*

Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

enable-globalresources

--enable-globalresources = true|false

Active les ressources globales 54. La valeur par défaut est false.

Note : les valeurs d'option booléennes sont configurées sur true si l'option est spécifiée sans une valeur.

globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*

Spécifie la configuration active de la ressource globale 54 (et active les ressources globales) 54.

globalresourcefile [gr]

--gr | --globalresourcefile = *FILE*

Spécifie le fichier de ressource globale 54 (et active les ressources globales) 54.

Extensions

Ces options définissent la gestion des fonctions d'extension spéciales disponibles dans un certain nombre de produits Altova de niveau d'entreprise (comme XMLSpy Enterprise Edition). Leur utilisation est décrite dans les manuels d'utilisateur de ces produits.

chartext-disable

--chartext-disable = true|false

Désactive les extensions de graphique. La valeur par défaut est false.

Note : les valeurs d'option booléennes sont configurées sur true si l'option est spécifiée sans une valeur.

dotnetext-disable

--dotnetext-disable = true|false

Désactive les extensions .NET. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- jvm-location

  `--jvm-location =` *`FILE`*
  `FILE` spécifie l'emplacement de la Java Virtual Machine (DLL sur Windows, objet partagé sur Linux). La JVM est nécessaire si vous utilisez les fonctions d'extension Java dans votre code XSLT/XQuery. La valeur par défaut est `false`.

- javaext-barcode-location

  `--javaext-barcode-location =` *`FILE`*
  Spécifie le chemin vers le dossier qui contient le fichier d'extension du code-barres `AltovaBarcodeExtension.jar`. Le chemin doit être donné dans une des formes suivantes :

  - Un URI de fichier, par exemple : `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2024/etc/jar/"`
  - Un chemin Windows avec un échappement par barres obliques inversées, par exemple : `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2024\\etc\\jar\\"`

- javaext-disable

  `--javaext-disable =` `true|false`
  Désactive les extensions Java. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Messages, erreurs, aide, timeout, version

  - error-format

    `--error-format =` `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  --help
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  --log-output = *FILE*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.5 Commandes JSON/Avro

Les commandes JSON peuvent être utilisées pour vérifier la validité et la bonne formation de schéma JSON et de documents d'instance. Ces commandes sont listées ci-dessous et sont décrites en détails dans les sous-sections de cette section :

- avroextractschema 136: Extrait le schéma Avro d'un fichier binaire Avro
- valavro 144: Valide les données dans un ou plusieurs binaires Avro par rapport au schéma Avro respectif de chaque binaire
- valavrojson 147: Valide un ou plusieurs fichiers de données JSON par rapport à un schéma Avro
- valavroschema 151: Valide un schéma Avro par rapport à la spécification de schéma Avro
- valjsonschema 154: Contrôle la validité des documents de schéma JSON
- valjson 158: Contrôle la validité des documents JSON
- wfjson 163: Contrôle la bonne formation des documents JSON

5.5.1 avroextractschema

Un fichier binaire Avro contient un bloc de données Avro précédé du schéma Avro qui définit la structure du bloc de données. La commande **avroextractschema** extrait le schéma Avro du binaire Avro et sérialise le schéma Avro en tant que JSON.

```
raptorxml avroextractschema [options] --avrooutput=AvroSchemaFile AvroBinaryFile
```

- L'argument *AvroBinaryFile* spécifie le fichier binaire Avro à partir duquel le schéma Avro sera extrait.
- L'option `--avrooutput` spécifie l'emplacement du schéma Avro extrait.

Exemple

Exemples de la commande **avroextractschema** :

- **raptorxml** avroextractschema --avrooutput=c:\MyAvroSchema.avsc c:\MyAvroBinary.avro

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to

escape this sequence of characters, use a preceding backslash, like this: `\\"`. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: `"C:\My Directory\ \"`.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Traitement
  - sortie, **`avrooutput`**

    `--output =` *`FILE`*, **`--avrooutput =`** *`FILE`*
    Définit l'emplacement du fichier de sortie Avro.

  - recurse

    `--recurse =` `true|false`
    Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Catalogues et ressources globales
  - catalog

    `--catalog =` *`FILE`*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML [47], pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    `--user-catalog =` *`FILE`*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

--enable-globalresources = `true|false`
Active les ressources globales[54]. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Spécifie la configuration active de la ressource globale[54] (et active les ressources globales)[54].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Spécifie le fichier de ressource globale[54] (et active les ressources globales)[54].

- Messages, erreurs, aide, timeout, version
  - error-format

    --error-format = `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    --error-limit = *N | unlimited*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    --info-limit = *N | unlimited*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - log-output

    --log-output = *FILE*
    Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

  - network-timeout

    --network-timeout = *VALEUR*
    Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.5.2 json2xml

La commande **json2xml** convertit a document d'instance JSON en un document XML.

```
raptorxml json2xml [options] JSONFile
```

- L'argument *XMLFile* est le fichier JSON à convertir.
- Utilisez l'option `--conversion-output` pour spécifier l'emplacement du fichier XML généré.

Exemple

Exemple de la commande **json2xml** :

- **raptorxml** `json2xml --conversion-output=c:\MyJSONData.json c:\MyXMLData.xml`

- Casing and slashes on the command line

  RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
  raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **`"My File"`**. Note, however, that a backslash followed by a double-quotation mark (for example, **`"C:\My directory\"`**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **`\"`** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **`\\"`**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **`"C:\My Directory\ \"`**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

▼ Options de conversion JSON en XML

Ces options définissent la gestion de détails liés à la conversion spécifiques dans les conversions entre XML et JSON.

▼ array-element

`--array-element =` `VALUE`
Précise le nom de l'élément à convertir en un item array.

▼ attributs

`--attributes =` `true|false`
Si défini comme **`true`**, alors la conversion entre les attributs XML et les propriétés préfixées JSON@ apparaît. Autrement, les attributs XML et propriétés JSON@ ne seront pas convertis. Le réglage par défaut est **`true`**.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ commentaires

`--comments =` `true|false`
Si défini comme **`true`**, alors la conversion entre les commentaires XML et les propriétés préfixées JSON# apparaît. Autrement, les attributs XML et propriétés JSON# ne seront pas convertis. Le réglage par défaut est **`true`**.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- conversion-output, o

  --o, --converson-output = FICHIER
  Défini le chemin et le nom du fichier auquel le résultat de la conversion est envoyé.

- create-array-container

  --create-array-container = true|false
  Si défini comme **true**, crée un élément conteneur dans le fichier XML généré pour chaque tableau JSON dans le document JSON source. Le réglage par défaut est **false**.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- encode-colons

  --encode-colons = true|false
  Si défini comme **true**, les colonnes dans les noms de propriété JSON sont encodées dans le document XML généré. Le réglage par défaut est **true**.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- json-type-hints

  --json-type-hints = true|false
  Si défini comme **true**, ajoute des attributs dans le document XML généré pour des « type-hints » dans le document JSON source. Le réglage par défaut est **true**.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- pi

  -- pi = true|false
  Si défini comme **true**, alors la conversion entre les instructions de traitement XML et les propriétés préfixées JSON ? apparaît. Autrement, les attributs XML et propriétés JSON ? ne seront pas convertis. Le réglage par défaut est **true**.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- pretty-printing

  --pp, --pretty-print = true|false
  Si défini comme **true**, ceci pretty-print le document de sortie généré. Le réglage par défaut est **false**.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- texte

  -- text = true|false
  Si défini comme **true**, alors la conversion entre les contenu texte XML et les propriétés préfixées JSON $ apparaît. Autrement, les attributs XML et propriétés JSON $ ne seront pas convertis. Le réglage par défaut est **true**.

Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Messages, erreurs, aide, timeout, version
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    `--info-limit =` *`N | unlimited`*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

  - help

    `--help`
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - log-output

    `--log-output =` *`FILE`*
    Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

  - network-timeout

    `--network-timeout =` *`VALEUR`*
    Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

  - verbose

    `--verbose =` `true|false`
    Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - verbose-output

    `--verbose-output =` *`FILE`*

Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

- Catalogues et ressources globales

  - catalog

    --catalog = *FILE*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    --user-catalog = *FILE*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    --enable-globalresources = `true|false`
    Active les ressources globales 54. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

    --gc | --globalresourceconfig = *VALUE*
    Spécifie la configuration active de la ressource globale 54 (et active les ressources globales) 54.

  - globalresourcefile [gr]

    --gr | --globalresourcefile = *FILE*
    Spécifie le fichier de ressource globale 54 (et active les ressources globales) 54.

5.5.3 valavro (avro)

La commande **valavro | avro** valide le bloc de données dans un ou plusieurs fichiers binaires Avro par rapport aux schéma Avro respectifs dans chaque fichier binaire.

```
raptorxml valavro | avro [options] AvroBinaryFile
```

- L'argument *AvroBinaryFile* spécifie un ou plusieurs fichiers binaires Avro à valider. En particulier, le bloc de données dans chaque fichier binaire Avro est validé par rapport au Schéma Avro dans ce fichier binaire.
- Pour valider plusieurs binaires Avro, il faut soit : (i) lister les fichiers à valider sur la CLI, chaque fichier étant séparé de l'autre par un espace ; ou (ii) lister les fichiers à valider dans un fichier de texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *AvroBinaryFile* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).

Exemples

Exemples de la commande **valavro** :

- **raptorxml** valavro c:\MyAvroBinary.avro
- **raptorxml** valavro c:\MyAvroBinary01.avro c:\MyAvroBinary02.avro
- **raptorxml** avro --listfile=true c:\MyFileList.txt

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué

dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Traitement
  - listfile

    `--listfile =` `true|false`
    Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - recurse

    `--recurse =` `true|false`
    Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Catalogues et ressources globales
  - catalog

    `--catalog =` *`FILE`*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    `--user-catalog =` *`FILE`*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Active les ressources globales 54. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Spécifie la configuration active de la ressource globale[54] (et active les ressources globales)[54].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Spécifie le fichier de ressource globale[54] (et active les ressources globales)[54].

- Messages, erreurs, aide, timeout, version
  - error-format

    --error-format = `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    --error-limit = *N | unlimited*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    --info-limit = *N | unlimited*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - log-output

    --log-output = *FILE*
    Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

  - network-timeout

    --network-timeout = *VALEUR*
    Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

  - verbose

    --verbose = `true|false`
    Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.

Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  `--verbose-output =` *`FILE`*
  Écrit la sortie verbeuse sur *`FILE`*.

- version

  `--version`
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.5.4 valavrojson (avrojson)

La commande **`valavrojson | avrojson`** valide un document JSON par rapport à un schéma Avro.

```
raptorxml valavrojson | avrojson [options] --avroschema=AvroSchema JSONFile
```

- L'argument *`JSONFile`* spécifie le document JSON à valider.
- L'option **`--avroschema`** spécifie le schéma Avro par rapport auquel le document JSON doit être validé.
- Pour valider plusieurs fichiers JSON, il faut soit (i) lister les fichiers à valider sur la CLI, chaque fichier étant séparé du suivant par un espace ; ou (ii) lister les fichiers à valider dans un fichier texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *`JSONFile`* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).

Exemples

Exemples de la commande **`valavrojson`** :

- **`raptorxml`** `valavrojson --avroschema=c:\MyAvroSchema.avsc c:\MyJSONDataFile.json`
- **`raptorxml`** `avrojson --avroschema=c:\MyAvroSchema.avsc c:\MyJSONDataFile.json`

- Casse et barres obliques sur la ligne de commande

  `RaptorXML(` et **`RaptorXMLServer`** pour des commandes d'administration) *sur Windows*
  `raptorxml(` et **`raptorxmlserver`** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **`"My File"`**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **`"C:\My directory\"`**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **`\"`** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **`\ \"`**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **`"C:\My Directory\\"`**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

▼ Traitement

▼ listfile

`--listfile =` `true|false`
Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ recurse

`--recurse =` `true|false`
Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.

Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Catalogues et ressources globales
  - catalog

    `--catalog =` *`FILE`*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    `--user-catalog =` *`FILE`*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Active les ressources globales 54. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *`VALUE`*
    Spécifie la configuration active de la ressource globale 54 (et active les ressources globales) 54.

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *`FILE`*
    Spécifie le fichier de ressource globale 54 (et active les ressources globales) 54.

- Messages, erreurs, aide, timeout, version
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    `--error-limit =` *`N | unlimited`*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

--info-limit = *N | unlimited*
Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  --help
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  --log-output = *FILE*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.5.5 valavroschema (avroschema)

La commande **valavroschema | avroschema** valide un ou plusieurs documents de schéma Avro par rapport à la spécification de schéma Avro.

```
raptorxml valavroschema | avroschema [options] AvroSchema
```

- L'argument *AvroSchema* spécifie le document de schéma Avro à valider.
- Pour valider plusieurs schémas Avro, il faut soit (i) lister les fichiers à valider sur la CLI, chaque fichier étant séparé du suivant par un espace ; ou (ii) lister les fichiers à valider dans un fichier texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *AvroSchema* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).

Exemples

Exemples de la commande **valavroschema** :

- **raptorxml** valavroschema c:\MyAvroSchema.avsc
- **raptorxml** valavroschema c:\MyAvroSchema01.avsc c:\MyAvroSchema02.avsc
- **raptorxml** avroschema --listfile=true c:\MyFileList.txt

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si

elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Traitement
  - listfile

    `--listfile =` `true|false`
    Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - recurse

    `--recurse =` `true|false`
    Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Catalogues et ressources globales
  - catalog

    `--catalog =` *`FILE`*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    `--user-catalog =` *`FILE`*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Active les ressources globales 54. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une

valeur.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Spécifie la configuration active de la ressource globale 54 (et active les ressources globales) 54.

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Spécifie le fichier de ressource globale 54 (et active les ressources globales) 54.

- Messages, erreurs, aide, timeout, version
  - error-format

    --error-format = `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    --error-limit = *N | unlimited*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    --info-limit = *N | unlimited*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - log-output

    --log-output = *FILE*
    Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

  - network-timeout

    --network-timeout = *VALEUR*
    Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

  - verbose

    --verbose = `true|false`

Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  `--verbose-output =` *`FILE`*
  Écrit la sortie verbeuse sur *`FILE`*.

- version

  `--version`
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.5.6 valjsonschema (jsonschema)

La commande **`valjsonschema | jsonschema`** valide un ou plusieurs documents de schéma JSON conformément aux diverses spécifications JSON (définies via l'option **`jsonschema-version`** option).

```
raptorxml valjsonschema | jsonschema [options] InputFile
```

- L'argument *`InputFile`* spécifie le document de schéma JSON à valider.
- Pour valider plusieurs documents, il faut soit (i) lister les fichiers à valider sur la CLI, chaque fichier étant séparé du suivant par un espace ; ou (ii) lister les fichiers à valider dans un fichier texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *`InputFile`* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).

Exemples

Exemples de la commande **`valjsonschema`** :

- **`raptorxml`** `valjsonschema c:\MyJSONSchema.json`
- **`raptorxml`** `jsonschema c:\MyJSONSchema-01.json c:\MyJSONSchema-02.json`
- **`raptorxml`** `jsonschema --listfile=true c:\FileList.txt`

- Casse et barres obliques sur la ligne de commande

  `RaptorXML(` et **`RaptorXMLServer`** pour des commandes d'administration) *sur Windows*

raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.
.

▼ Validation et traitement

▼ listfile

--listfile = `true|false`
Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ recurse

--recurse = `true|false`
Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml`

à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

Options de validation JSON

additional-schema

`--additional-schema =` *FILE*
Spécifie des URIs d'un document de schéma supplémentaire. Le schéma supplémentaire sera chargé par le schéma principal et peut être référencé depuis le schéma principal par des schémas supplémentaires **`id`** ou de propriété **`$id`**.

disable-format-checks

`--disable-format-checks =` `true|false`
Désactive la validation sémantique imposée par l'attribut de format. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

jsonschema-version

`--jsonschema-version =` `draft04|draft06|draft07|2019-09|2020-12|latest|detect`
Spécifie laquelle des versions d'ébauche de spécification JSON Schema utiliser. Le défaut est `detect`.

strict-integer-checks

`--strict-integer-checks =` `true|false`
Spécifie si des vérifications d'entier plus strictes de draft-04 devraient être utilisées avec des schémas ultérieurs - là où les vérifications d'entier sont moins strictes. Par exemple, **`1.0`** n'est pas un entier valide dans draft-04, mais est un entier valide dans des drafts ultérieurs. Cette option n'a pas d'effet pour les schémas draft-04. La valeur par défaut de l'option est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

Catalogues et ressources globales

catalog

`--catalog =` *FILE*
Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

user-catalog

`--user-catalog =` *FILE*

Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

- enable-globalresources

  --enable-globalresources = `true|false`
  Active les ressources globales [54]. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- globalresourceconfig [gc]

  --gc | --globalresourceconfig = *VALUE*
  Spécifie la configuration active de la ressource globale [54] (et active les ressources globales) [54].

- globalresourcefile [gr]

  --gr | --globalresourcefile = *FILE*
  Spécifie le fichier de ressource globale [54] (et active les ressources globales) [54].

- Messages, erreurs, aide, timeout, version
  - error-format

    --error-format = `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    --error-limit = *N | unlimited*
    Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

  - info-limit

    --info-limit = *N | unlimited*
    Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - log-output

    --log-output = *FILE*
    Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  --warning-limit = *N | unlimited*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.5.7 valjson (json)

La commande **`valjson | json`** valide un ou plusieurs documents d'instance JSON conformément au schéma JSON fournit avec l'option `--schema`.

```
raptorxml valjson | json [options] --jsonschema=File InputFile
```

- L'argument *InputFile* spécifie le document d'instance JSON à valider.
- Pour valider plusieurs documents, il faut soit (i) lister les fichiers à valider sur la CLI, chaque fichier étant séparé du suivant par un espace ; ou (ii) lister les fichiers à valider dans un fichier texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *InputFile* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).

Exemples

Exemples de la commande **valjson** :

- **raptorxml** valjson --jsonschema=c:\MyJSONSchema.json c:\MyJSONInstance.json
- **raptorxml** json --jsonschema=c:\MyJSONSchema.json c:\MyJSONInstance-01.json c:\MyJSONInstance-02.json
- **raptorxml** json --jsonschema=c:\MyJSONSchema.json --listfile=true c:\FileList.txt

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (raptorxml et raptorxmlserver) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (RaptorXML) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est TRUE. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.
.

▼ Validation et traitement

▼ schema**, jsonschema**

--schema = *FILE*, **--jsonschema =** *FILE*
Spécifie le chemin vers le document de Schéma JSON à utiliser pour la validation des documents d'instance JSON.

- listfile

  --listfile = `true|false`
  Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- additional-schema

  --additional-schema = *`FILE`*
  Spécifie des URIs d'un document de schéma supplémentaire. Le schéma supplémentaire sera chargé par le schéma principal et peut être référencé depuis le schéma principal par des schémas supplémentaires **`id`** ou de propriété **`$id`**.

- recurse

  --recurse = `true|false`
  Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- json5

  --json5 = `true|false`
  Active la prise en charge JSON5. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- jsonc

  --jsonc = `true|false`
  Active la prise en charge pour les commentaires dans JSON. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- jsonschema-version

  --jsonschema-version = `draft04|draft06|draft07|2019-09|2020-12|latest|detect`
  Spécifie laquelle des versions d'ébauche de spécification JSON Schema utiliser. Le défaut est `detect`.

- disable-format-checks

--disable-format-checks = `true|false`
Désactive la validation sémantique imposée par l'attribut de format. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- strict-integer-checks

--strict-integer-checks = `true|false`
Spécifie si des vérifications d'entier plus strictes de draft-04 devraient être utilisées avec des schémas ultérieurs - là où les vérifications d'entier sont moins strictes. Par exemple, `1.0` n'est pas un entier valide dans draft-04, mais est un entier valide dans des drafts ultérieurs. Cette option n'a pas d'effet pour les schémas draft-04. La valeur par défaut de l'option est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Catalogues et ressources globales
  - catalog

--catalog = *FILE*
Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML [47], pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

--user-catalog = *FILE*
Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

--enable-globalresources = `true|false`
Active les ressources globales [54]. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Spécifie la configuration active de la ressource globale [54] (et active les ressources globales) [54].

  - globalresourcefile [gr]

--gr | --globalresourcefile = *FILE*
Spécifie le fichier de ressource globale [54] (et active les ressources globales) [54].

- Messages, erreurs, aide, timeout, version
  - error-format

--error-format = `text|shortxml|longxml`
Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

- error-limit

  --error-limit = *N | unlimited*
  Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

- info-limit

  --info-limit = *N | unlimited*
  Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  --help
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  --log-output = *FILE*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  --network-timeout = *VALEUR*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  --verbose = `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  --verbose-output = *FILE*
  Écrit la sortie verbeuse sur *FILE*.

- version

  --version
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

▼ warning-limit

--warning-limit = *N | unlimited*
Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.5.8 wfjson

La commande **wfjson** contrôle un ou plusieurs documents JSON conformément à la spécification ECMA-404 concernant la bonne formation.

```
raptorxml wfjson [options] InputFile
```

- L'argument *InputFile* est le document JSON dont vous (schéma ou instance) pour vérifier la bonne formation.
- Si vous souhaitez contrôler plusieurs documents, il faut soit : (i) lister les fichiers à cocher sur la CLI, chaque fichier étant séparé du suivant par un espace ; ou (ii) lister les fichiers à Coché dans un fichier texte (fichier `.txt`), avec un nom de fichier par ligne, et fournir ce fichier de texte en tant que l'argument *InputFile* avec l'option `--listfile`[211] définie sur `true` (*voir la liste des Options ci-dessous*).).

Exemples

Exemples de la commande **wfjson** :

- **raptorxml** wfjson c:\MyJSONFile.json
- **raptorxml** wfjson c:\MyJSONFile-01.json c:\MyJSONFile-02.json
- **raptorxml** wfjson --listfile=true c:\FileList.txt

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory**

\".

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Validation et traitement
  - json5

    `--json5 =` `true|false`
    Active la prise en charge JSON5. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - jsonc

    `--jsonc =` `true|false`
    Active la prise en charge pour les commentaires dans JSON. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - json-lines

    `--json-lines =` `true|false`
    Permet la prise en charge pour JSON Lines (c'est à dire, une valeur JSON par ligne). Valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - listfile

    `--listfile =` `true|false`
    Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - recurse

    `--recurse =` `true|false`
    Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP.

Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Catalogues et ressources globales
  - catalog

    `--catalog =` *`FILE`*
    Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML [47], pour plus d'informations à propos du travail avec les catalogues.

  - user-catalog

    `--user-catalog =` *`FILE`*
    Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML [47], pour plus d'informations concernant le travail avec des catalogues.

  - enable-globalresources

    `--enable-globalresources =` `true|false`
    Active les ressources globales [54]. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - globalresourceconfig [gc]

    `--gc | --globalresourceconfig =` *`VALUE`*
    Spécifie la configuration active de la ressource globale [54] (et active les ressources globales) [54].

  - globalresourcefile [gr]

    `--gr | --globalresourcefile =` *`FILE`*
    Spécifie le fichier de ressource globale [54] (et active les ressources globales) [54].

- Messages, erreurs, aide, timeout, version
  - error-format

    `--error-format =` `text|shortxml|longxml`
    Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

  - error-limit

    `--error-limit =` *`N | unlimited`*

Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

- info-limit

  `--info-limit =` *`N | unlimited`*
  Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- help

  `--help`
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- log-output

  `--log-output =` *`FILE`*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  `--network-timeout =` *`VALEUR`*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  `--verbose =` `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  `--verbose-output =` *`FILE`*
  Écrit la sortie verbeuse sur *`FILE`*.

- version

  `--version`
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.6 Commandes Signature XML

Les commandes Signature XML peuvent être utilisées pour signer un document XML et pour vérifier un document signé. Ces commandes sont listées ci-dessous et sont décrites en détails dans les sous-sections de cette section :

- xmlsignature-sign 168: Crée un document de sortie de signature XML à partir d'un document d'entrée
- xmlsignature-verify 173: Vérifie un document de signature XML
- xmlsignature-update 168: Met à jour la signature d'un document XML (modifié)
- xmlsignature-remove 168: Supprime la signature d'un document XML

5.6.1 xmlsignature-sign

La commande **xmlsignature-sign | xsign** prend un document XML en tant qu'entrée et crée un document de sortie à signature XML utilisant les options de signature spécifiées.

```
raptorxml xmlsignature-sign [options] --output=File --signature-type=Value --
signature-canonicalization-method=Value --certname=Value|hmackey=Value InputFile
```

- L'argument *InputFile* est le document XML à signer.
- L'option **--output** spécifie l'emplacement du document qui contient la signature XML.

Exemples

Exemples de la commande **xmlsignature-sign** :

- **raptorxml** xsign --output=c:\SignedFile.xml --signature-type=enveloped --signature-canonicalization-method=xml-c14n11 --hmackey=secretpassword c:\SomeUnsigned.xml

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (raptorxml et raptorxmlserver) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (RaptorXML) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par

exemple, `"C:\My directory\"`) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement `\"` représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : `\ \"`. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : `"C:\My Directory\\"`.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Options communes
  - output

    `output =` *FILE*
    L'URL du document de sortie qui est créé avec la nouvelle signature XML.

  - verbose

    `--verbose =` `true|false`
    Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Options de Signature XML
  - absolute-reference-uri

    `--absolute-reference-uri =` `true|false`
    Spécifie si l'URI du document signé doit être lu en tant qu'absolu (`true`) ou relatif (`false`). La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

  - certname, certificate-name

    `--certname, --certificate-name =` *VALUE*
    Le nom du certificat utilisé pour signer.

    Windows
    Il s'agit du nom du **Sujet** d'un certificat depuis le **`--certificate-store`** sélectionné.

xemple de liste des certificats (sous PowerShell)

```
% ls cert://CurrentUser/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

xemple : **--certificate-name==certificate1**

Linux/MacOS
--certname spécifie le nom de fichier d'un certificat X.509v3 à encodage PEM avec la clé privée. Ces fichiers portent généralement l'extension **.pem**.

xemple : **--certificate-name==/path/to/certificate1.pem**

certstore, certificate-store

--certstore, --certificate-store = *VALUE*
L'emplacement où le certificat spécifié par **--certificate-name** est stocké.

Windows
Le nom d'un magasin de certificat sous **cert://CurrentUser**. Les magasins de certificat disponibles peuvent être listés (sous PowerShell) à l'aide de **% ls cert://CurrentUser/**. Les certificats seraient donc listés comme suit :

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

xemple : **--certificate-store==MyCertStore**

Linux/MacOS
L'option **--certstore** n'est actuellement pas prise en charge.

- digest, digest-method

  --digest, --digest-method = sha1|sha256|sha384|sha512
  L'algorithme utilisé pour calculer la valeur de digest par rapport au fichier XML d'entrée.

- hmackey, hmac-secret-key

  --hmackey, --hmac-secret-key = *VALUE*
  La clé secrète partagée HMAC ; doit avoir une longueur minimum de six caractères.

  Exemple : **--hmackey=secretpassword**

- hmaclen, hmac-output-length

  --hmaclen, --hmac-output-length = *LENGTH*
  Tronque la sortie de l'algorithme HMAC à des bits LENGTH. Si spécifiée, cette valeur doit être
  - un multiple de 8
  - supérieur à 80
  - supérieur à la moitié de la longueur de sortie de l'algorithme hash sous-jacent

- keyinfo, append-keyinfo

  --keyinfo, --append-keyinfo = true|false
  Spécifie si vous souhaitez inclure l'élément **KeyInfo** dans la signature ou pas. La valeur par défaut est **false**.

- sigc14nmeth, signature-canonicalization-method

  --sigc14nmeth, --signature-canonicalization-method = *VALUE*
  Spécifie l'algorithme de canonicalisation à appliquer à l'élément **SignedInfo**. La valeur doit être une des suivantes :

  - REC-xml-c14n-20010315
  - xml-c14n11
  - xml-exc-c14n#

- sigmeth, signature-method

  --sigmeth, --signature-method = *VALUE*
  Spécifie l'algorithme à utiliser pour générer la signature.

  Lorsqu'un certificat est utilisé
  Si un certificat est spécifié, alors--signature-method est optionnel et la valeur pour ce paramètre est dérivé du certificat. Lorsqu'il est spécifié, il doit correspondre à l'algorithme utilisé par le certificat.

  xemple : **--signature-method=rsa-sha256**

  Lorsque --hmac-secret-key est utilisé
  Lorsque --hmac-secret-key est utilisé, cette option est obligatoire. La valeur doit être un des algorithmes HMAC pris en charge :
  - hmac-sha256

- hmac-sha386
- hmac-sha512
- hmac-sha1 (découragé par la spécification)

xemple : **--signature-method=hmac-sha256**

- sigtype, signature-type

  --sigtype, --signature-type = detached | enveloping | enveloped
  Spécifie le type de signature à générer.

- transforms

  --transforms = *VALUE*
  Spécifie les transformations de Signature XML appliquées au document d'entrée. Cette option peut être spécifiée plusieurs fois. Dans ce cas, l'ordre de la spécification est important. La première transformation spécifiée reçoit le document d'entrée. La dernière transformation spécifiée est utilisée immédiatement avant le calcul de la valeur digest.

  Les valeurs prises en charge sont :

  - REC-xml-c14n-20010315 pour Canonical XML 1.0 (sans commentaires)
  - xml-c14n11 pour Canonical XML 1.1 (sans comments)
  - xml-exc-c14n# pour Exclusive XML Canonicalization 1.0 (sans commentaires)
  - REC-xml-c14n-20010315#WithComments pour Canonical XML 1.0 (avec commentaires)
  - xml-c14n11#WithComments pour Canonical XML 1.1 (avec commentaires)
  - xml-exc-c14n#WithComments pour Exclusive XML Canonicalization 1.0 (avec commentaires)
  - base64
  - strip-whitespaces extension Altova

  xemple : **--transforms=xml-c14n11**

  Note: Cette option peut être spécifiée plusieurs fois. Si elle est spécifiée plusieurs fois, l'ordre de spécification est significatif. La première transformation spécifiée reçoit le document d'entrée. La dernière transformation spécifiée est utilisée juste avant le calcul de la valeur de digest.

- write-default-attributes

  --write-default-attributes = true|false
  Spécifie s'il faut inclure les valeur d'attribut par défaut depuis le DTD dans le document signé.

- Options d'aide et de version
  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, valany --h. (en alternative, la commande help peut être utilisée avec un argument. Par exemple : help valany.)

  - version

    --version

Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

5.6.2 xmlsignature-verify

La commande **`xmlsignature-update | xupdate`** met à jour la signature XML dans le fichier d'entrée signé.

```
raptorxml xmlsignature-verify [options] InputFile
```

- L'argument *`InputFile`* est le document XML signé à vérifier.
- Si la vérification est réussie, un message **`result="OK"`** est affiché, sinon, un message **`result="Failed"`** est affiché.

Exemple

Exemples de la commande **`xmlsignature-verify`** :

- **`raptorxml`** `xverify c:\SignedFile.xml`

▼ Casing and slashes on the command line

`RaptorXML` (and **`RaptorXMLServer`** for administration commands) *on Windows*
`raptorxml` (and **`raptorxmlserver`** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **`"My File"`**. Note, however, that a backslash followed by a double-quotation mark (for example, **`"C:\My directory\"`**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **`\"`** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **`\\"`**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **`"C:\My Directory\ \"`**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans

guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Options communes
  - verbose

    `--verbose =` `true|false`
    Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Options de Signature XML
  - certname, certificate-name

    `--certname, --certificate-name =` *VALUE*
    Le nom du certificat utilisé pour signer.

    Windows
    Il s'agit du nom du **Sujet** d'un certificat depuis le **`--certificate-store`** sélectionné.

    xemple de liste des certificats (sous PowerShell)

```
% ls cert://CurrentUser/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

    xemple : **`--certificate-name==certificate1`**

    Linux/MacOS
    `--certname` spécifie le nom de fichier d'un certificat X.509v3 à encodage PEM avec la clé privée. Ces fichiers portent généralement l'extension **`.pem`**.

    xemple : **`--certificate-name==/path/to/certificate1.pem`**

  - certstore, certificate-store

    `--certstore, --certificate-store =` *VALUE*
    L'emplacement où le certificat spécifié par **`--certificate-name`** est stocké.

    Windows
    Le nom d'un magasin de certificat sous **`cert://CurrentUser`**. Les magasins de certificat disponibles peuvent être listés (sous PowerShell) à l'aide de **`% ls cert://CurrentUser/`**. Les

certificats seraient donc listés comme suit :

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

xemple : **--certificate-store==MyCertStore**

Linux/MacOS
L'option **--certstore** n'est actuellement pas prise en charge.

- hmackey, hmac-secret-key

  --hmackey, --hmac-secret-key = *VALUE*
  La clé secrète partagée HMAC ; doit avoir une longueur minimum de six caractères.

  Exemple : **--hmackey=secretpassword**

- ignore-certificate-errors

  --i, --ignore-certificate-errors = `true|false`
  Si défini comme **true**, ignore les erreurs de certificats pendant la vérification des signatures XML (les éléments **SignedInfo**) dans un document XML. Le réglage par défaut est **false**.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Options d'aide et de version
  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - version

    --version
    Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

5.6.3 xmlsignature-update

La commande **xmlsignature-update | xupdate** met à jour la signature XML dans le fichier d'entrée signé. Si le document a été modifié, la signature XML mise à jour sera différente ; sinon, la signature mise à jour sera la même que celle de la signature précédente.

```
raptorxml xmlsignature-update [options] --output=File SignedFile
```

- L'argument *SignedFile* est le document XML signé à mettre à jour.
- Il faut spécifier soit (i) l'option hmac-secret-key soit (ii) les options certificate-name et certificate-store.
- Si les options certificate-name et certificate-store sont spécifiées, elles doivent correspondre à celles qui ont été utilisées pour signer le document XML précédemment. (Veuillez noter que l'option certificate-store n'est actuellement pas pris en charge sur Linux et macOS.)

Exemples

Exemples de la commande **xmlsignature-update** :

- **raptorxml** xupdate --output=c:\UpdatedSignedFile.xml --certname=certificate1 --certstore=MyCertStore c:\SomeSignedFile.xml
- **raptorxml** xupdate --output=c:\UpdatedSignedFile.xml --hmackey=SecretPassword c:\SomeSignedFile.xml

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (raptorxml et raptorxmlserver) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (RaptorXML) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Options communes
  - output

    `output =` *FILE*
    L'URL du document de sortie qui est créé avec la nouvelle signature XML.

  - verbose

    `--verbose =` `true|false`
    Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Options de Signature XML
  - certname, certificate-name

    `--certname, --certificate-name =` *VALUE*
    Le nom du certificat utilisé pour signer.

    Windows
    Il s'agit du nom du **Sujet** d'un certificat depuis le **`--certificate-store`** sélectionné.

    xemple de liste des certificats (sous PowerShell)

```
% ls cert://CurrentUser/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

    xemple : **`--certificate-name==certificate1`**

    Linux/MacOS
    `--certname` spécifie le nom de fichier d'un certificat X.509v3 à encodage PEM avec la clé privée. Ces

fichiers portent généralement l'extension **.pem**.

xemple : **--certificate-name==/path/to/certificate1.pem**

- certstore, certificate-store

  --certstore, --certificate-store = *VALUE*
  L'emplacement où le certificat spécifié par **--certificate-name** est stocké.

  Windows
  Le nom d'un magasin de certificat sous **cert://CurrentUser**. Les magasins de certificat disponibles peuvent être listés (sous PowerShell) à l'aide de **% ls cert://CurrentUser/**. Les certificats seraient donc listés comme suit :

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

  xemple : **--certificate-store==MyCertStore**

  Linux/MacOS
  L'option **--certstore** n'est actuellement pas prise en charge.

- hmackey, hmac-secret-key

  --hmackey, --hmac-secret-key = *VALUE*
  La clé secrète partagée HMAC ; doit avoir une longueur minimum de six caractères.

  Exemple : **--hmackey=secretpassword**

- Options d'aide et de version
  - help

    --help
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

▼ version

--version
Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

5.6.4 xmlsignature-remove

La commande **xmlsignature-remove | xremove** supprime la signature XML dans le fichier d'entrée signé et enregistre le document résultant non signé dans un emplacement de sortie que vous spécifiez.

```
raptorxml xmlsignature-remove [options] --output=File SignedFile
```

- L'argument *SignedFile* est le document XML signé à partir duquel vous souhaitez supprimer la signature XML.
- L'option `--output` spécifie l'emplacement du document XML non-signé qui est généré.

Exemples

Exemples de la commande **xmlsignature-remove** :

- **raptorxml** xremove --output=c:\UnsignedFile.xml c:\SignedFile.xml

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **`--option=value`**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **`--h, --help`** pour afficher des informations à propos de la commande.
.

- Options communes
  - output

    `output =` *FILE*
    L'URL du document de sortie qui est créé avec la signature XML supprimée.

  - verbose

    `--verbose =` `true|false`
    Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par défaut est `false`.
    Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- Options d'aide et de version
  - help

    `--help`
    Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

  - version

    `--version`
    Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

5.7 Commandes générales

Cette section contient une description des commandes générales suivantes :

- valany[181]: valide le document soumis conformément à son type
- script[182]: exécute un script Python
- help[183]: affiche des informations concernant la commande nommée

5.7.1 valany

La commande **valany** est une commande générale qui valide un document sur la base du type de document dont il s'agit. Le type du document d'entrée est détecté automatiquement et la validation correspondante est effectuée conformément à la spécification respective. L'argument *InputFile* spécifie le document à valider. Veuillez noter qu'un seul document peut être soumis en tant que l'argument de la commande.

```
raptorxml valany [options] InputFile
```

La commande **valany** couvre les types de validation suivants. Ces options sont celles qui sont disponibles pour la commande de validation individuelle correspondante. Voir la description des commandes de validation respectives pour une liste de leurs options respectives.

- valdtd (dtd)[70]
- valxsd (xsd)[74]
- valxml-withdtd (xml)[59]
- valxml-withxsd (xsi)[63]
- valxslt[129]
- valxquery[108]
- valavrojson (avrojson)[147]

Exemples

- **raptorxml** valany c:\Test.xsd

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Options

Voir la description des commandes de validation respective pour une liste de leurs options respectives. Veuillez noter, néanmoins qu'alors que la plupart des commandes de validation individuelles acceptent des documents d'entrée multiples, la commande **valany** n'accepte qu'un seul document d'entrée. C'est pourquoi les options de type `--listfile` ne s'appliqueront pas à **valany**.

5.7.2 script

La commande **script** exécute un script 3.11.5 qui utilise l'API Python RaptorXML.

```
raptorxml script [options] PythonScriptFile
```

L'argument *File* est le chemin vers le script Python que vous souhaitez exécuter. Des options supplémentaires sont disponibles pour cette commande. Pour obtenir une liste de ces options, exécutez la commande suivante :

```
raptorxml script [-h | --help]
```

Exemples

- **raptorxml** script c:\MyPythonScript.py
- **raptorxml** script -h
- **raptorxml** script # *Sans fichier de script, un shell Python interactif est démarré*
- **raptorxml** script -m pip # *Charge et exécute le module* pip *; voir la section des Options ci-dessous*

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (raptorxml and raptorxmlserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (RaptorXML) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

- Backslashes, spaces, and special characters on Windows systems

  On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, `"My File"`. Note, however, that a backslash followed by a double-quotation mark (for example, `"C:\My directory\"`) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence `\"` stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: `\\"`. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: `"C:\My Directory\ \"`.

Options

Toutes les options et arguments se trouvant après la commande `script` seront envoyés directement à l'interprète Python. Veuillez consulter la page de documentation Python https://docs.python.org/3.7/using/cmdline.html pour une liste complète des options disponibles.

5.7.3 help

Syntaxe et description

La commande **`help`** prend un seul argument (`Command`), qui est le nom de la commande pour laquelle l'aide est requise. Elle affiche la syntaxe de la commande, ses options, et d'autres informations pertinentes. Si l'argument `Command` n'est pas spécifié, toutes les commandes du programme d'exécution sont recensés, chacun présentant une brève description de texte. La commande **`help`** peut être appelée depuis leur programme d'exécution : **`raptorxml`** ou **`raptorxmlserver`**.

```
raptorxml help Command
raptorxmlserver help Command
```

- Casse et barres obliques sur la ligne de commande

  `RaptorXML(` et **`RaptorXMLServer`** pour des commandes d'administration) *sur Windows*
  `raptorxml(` et **`raptorxmlserver`** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

  * Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
  * Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Exemple

Exemples de la commande **`help`** pour afficher des informations concernant la commande **`licenserver`** (cette commande est disponible dans les deux programmes d'exécution):

```
raptorxml help licenseserver
raptorxmlserver help licenseserver
```

L'option --help

L'information Help à propos d'une commande est également disponible en utilisant l'option `--help` avec la commande pour laquelle l'information d'aide est nécessaire. Les deux commandes ci-dessous produisent les mêmes résultats :

```
raptorxml licenseserver --help
```

La commande ci-dessus utilise l'option `--help` de la commande **`licenserver`**.

```
raptorxml help licenseserver
```

La commande **`help`** prend `licenseserver` en tant que son argument.

Les deux commandes affichent des informations d'aide concernant la commande **`licenseserver`**.

5.8 Commandes de localisation

Vous pouvez créer une version localisée de l'application RaptorXML pour la langue de votre choix. Cinq versions localisées (anglais, allemand, espagnol, français et japonais) sont disponibles dans le dossier `<ProgramFilesFolder>\Altova\RaptorXMLServer2024\bin\` . Ces cinq versions ne devront donc pas être créées.

Pour créer une version localisée dans une autre langue :

1. Générer un fichier XML contenant les strings de ressource. Pour ce faire, utiliser la commande `exportresourcestrings`[185]. Les strings de ressource dans le fichier XML généré seront dans une des cinq langues : l'anglais (`en`), l'allemand (`de`), l'espagnol (`es`), le français (`fr`) ou le japonais (`ja`), conformément à l'argument utilisé avec la commande.
2. Traduire les strings de ressource depuis la langue du fichier XML généré dans la langue cible. Les strings de ressource sont les contenus des éléments `<string>` dans le fichier XML. Ne pas traduire les variables dans les accolades, comme `{option}` ou `{product}`.
3. Contacter l'Assistance Altova pour générer un fichier DLL RaptorXML localisé depuis votre fichier XML traduit.
4. Une fois avoir reçu votre fichier DLL localisé de la part de l'Assistance Altova, enregistrer la DLL dans le dossier `<ProgramFilesFolder>\Altova\RaptorXMLServer2024\bin\` . Votre fichier DLL aura un nom de la forme `RaptorXMLServer_lc.dll`. La partie `_lc` du nom contient le code de langage. Par exemple, dans `RaptorXMLServer_de.dll`, la partie `de` est la langue de code pour l'allemand (Deutsch).
5. Exécuter la commande `setdeflang`[187] pour définir votre fichier DLL localisé en tant que l'application RaptorXML à utiliser. En ce qui concerne l'argument de la commande `setdeflang`[187], utiliser le code de langage qui fait partie du nom de DLL.

Note : Altova RaptorXML Server est fourni avec une prise en charge de cinq langues : anglais, allemand, espagnol, français et japonais. Donc vous n'aurez pas besoin de créer une version localisée pour ces langues-là. Pour définir une de ces cinq langues en tant que la langue de défaut, utiliser la commande de CLI `setdeflang`[187].

5.8.1 exportresourcestrings

Syntaxe et description

La commande **`exportresourcestrings`** sort un fichier XML contenant les strings de ressource de l'application RaptorXML Server dans la langue spécifiée. Les langues d'exportation disponibles sont allemand (`de`), anglais (`en`), espagnol (`es`), français (`fr`) et japonais (`ja`).

```
raptorxml exportresourcestrings [options] LanguageCode XMLOutputFile
raptorxmlserver exportresourcestrings [options] LanguageCode XMLOutputFile
```

- L'argument ***`LanguageCode`*** donne la langue des strings de ressource dans le fichier XML de sortie ; il s'agit de la langue d'exportation *export language*. Les langues d'exportation autorisées (avec leurs codes de langue respectifs entre parenthèses) sont : allemand (`de`), anglais (`en`), espagnol (`es`), français (`fr`) et japonais (`ja`).
- L'argument ***`XMLOutputFile`*** spécifie le chemin et le nom du fichier XML de sortie .

- La commande **exportresourcestrings** peut être appelée depuis un programme d'exécution : **raptorxml** ou **%SERVER-EXENAME-LC%>**.

Pour créer des localisations, veuillez noter les points suivants.

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Exemples

Exemples de la commande **exportresourcestrings** :

```
raptorxml exportresourcestrings de c:\Strings.xml
raptorxmlserver exportresourcestrings de c:\Strings.xml
```

- La première commande ci-dessus crée un fichier appelé `Strings.xml` sous `c:\` qui contient les strings de ressource de RaptorXML Server en allemand.
- La deuxième commande appelle le server-executable à procéder de la même manière que dans le premier exemple.

Créer des versions localisées de RaptorXML Server

Vous pouvez créer une version localisée de RaptorXML Server pour toute langue de votre choix. Cinq versions localisées (allemand, anglais, espagnol, français et japonais) sont déjà disponibles sous le dossier `C:\Program Files (x86)\Altova\RaptorXMLServer2024\bin` , et n'ont donc pas besoin d'être créées.

Créer une version localisée comme suit :

1. Générer un fichier XML contenant des strings de ressource en utilisant la commande `exportresourcestrings` (*voir la syntaxe de commande ci-dessus*). Les strings de ressource dans ce

fichier XML seront ceux d'une des cinq langues prises en charge : allemand (`de`), anglais (`en`), espagnol (`es`), français (`fr`) ou japonais (`ja`), conformément à l'argument ***LanguageCode*** utilisé avec la commande.
2. Traduire les strings de ressource à partir de l'une des cinq langues prises en charge dans la langue cible. Les strings de ressource sont les contenus des éléments `<string>` dans le fichier XML. Ne pas traduire les variables dans les accolades comme `{option}` ou `{product}`.
3. Contactez Altova Support pour générer un fichier DLL RaptorXML Server localisé à partir de votre fichier XML traduit.
4. Après avoir reçu votre fichier DLL localisé de Altova Support, enregistrez le fichier DLL sous le dossier `C:\Program Files (x86)\Altova\RaptorXMLServer2024\bin` . Votre fichier DLL aura un nom de la forme `RaptorXML2024_lc.dll`. La partie `_lc` du nom contient le code de langue. Par exemple, dans `RaptorXML2024_de.dll`, la partie `de` est le code de langue pour l'allemand (Deutsch).
5. Exécutez la commande `setdeflang` pour définir votre fichier DLL localisé DLL en tant qu'application RaptorXML Server à utiliser. Pour l'argument de la commande `setdeflang`, utilisez le code de langue qui fait partie du nom DLL.

Note : Altova RaptorXML Server est livré avec une prise en charge pour les cinq langues : allemand, anglais, espagnol, français et japonais. Donc vous ne devez pas créer de version localisée pour ces langues. Pour définir une de ces langues en tant que langue par défaut, utilisez la commande RaptorXML Serverde `setdeflang`.

5.8.2 setdeflang

Syntaxe et description

La commande **`setdeflang`** (la forme abrégée est **`sdl`**) définit la langue par défaut de RaptorXML Server. Les langues disponibles sont anglais (`en`), allemand (`de`), espagnol (`es`), français (`fr`) et japonais (`ja`). La commande prend un argument *`LanguageCode`* obligatoire.

`raptorxml setdeflang [options]` *LanguageCode*
`raptorxmlserver setdeflang [options]` *LanguageCode*

- L'argument *`LanguageCode`* est requis et définit la langue par défaut de RaptorXML Server. Les valeurs respectives à utiliser sont : `en`, `de`, `es`, `fr`, `ja`.
- La commande **`setdeflang`** peut être appelée depuis un des programmes d'exécution : **`raptorxml`** or **`raptorxmlserver`**.
- Utiliser l'option **`--h, --help`** pour afficher les informations concernant la commande.

▼ Casse et barres obliques sur la ligne de commande

`RaptorXML(` et **`RaptorXMLServer`** pour des commandes d'administration) *sur Windows*
`raptorxml(` et **`raptorxmlserver`** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Exemples

Exemples de la commande **setdeflang** (**sdl**) :

```
raptorxml sdl de
raptorxml setdeflang es
raptorxmlserver setdeflang es
```

- La première commande définit le langage par défaut de RaptorXML Server en allemand.
- La seconde commande définit le langage par défaut de RaptorXML Server en espagnol.
- La troisième commande est la même que la seconde commande, mais est exécutée par le programme d'exécution de serveur.

Options

Use the **--h, --help** option to display information about the command.

5.9 Commandes de licence

Cette section décrit des commandes qui peuvent être utilisées pour la mise sous licence RaptorXML Server:

- licenseserver 189 pour enregistrer RaptorXML Server avec Altova LicenseServer sur votre réseau
- assignlicense 190 pour charger un fichier de licence sur LicenseServer (uniquement Windows)
- verifylicense 192 pour vérifier si RaptorXML Server est mis sous licence (uniquement Windows)

Note : Ces commandes peuvent aussi être exécutées par le biais d'exécutables de serveur pour les commandes d'administration 194.

Pour plus d'informations concernant la mise sous licence des produits Altova avec Altova LicenseServer, voir la documentation Altova LicenseServer.

5.9.1 licenseserver

Syntaxe et description

La commande **`licenseserver`** enregistre RaptorXML Server avec Altova LicenseServer spécifié par l'argument *`Server-Or-IP-Address`*. Pour pouvoir exécuter la commande `licenseserver` correctement, les deux serveurs (RaptorXML Server et LicenseServer) doivent être connectés sur le réseau et le LicenseServer doit fonctionner. Vous devez posséder des privilèges d'administrateur pour pouvoir enregistrer RaptorXML Server auprès du LicenseServer.

`raptorxml licenseserver [options]` *Server-Or-IP-Address*
`raptorxmlserver licenseserver [options]` *Server-Or-IP-Address*

- L'argument *`Server-Or-IP-Address`* prend le nom ou l'adresse IP de l'appareil de LicenseServer.
- La commande **`licenseserver`** peut être appelée depuis un des programmes d'exécution : **`raptorxml`** ou **`%SERVER-EXENAME-LC%`**.

Une fois que RaptorXML Server a été enregistré avec succès auprès de LicenseServer, vous recevrez un message. Le message affichera aussi l'URL du LicenseServer. Vous pouvez maintenant vous rendre sur LicenseServer pour attribuer une licence à RaptorXML Server. Pour plus de détails concernant la licence, voir la documentation LicenseServer (https://www.altova.com/manual/fr/licenseserver/3.14/).

▼ Casing and slashes on the command line

`RaptorXML` (and **`RaptorXMLServer`** for administration commands) *on Windows*
`raptorxml` (and **`raptorxmlserver`** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder

names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Exemples

Exemples de la commande **licenseserver** :

```
raptorxml licenseserver DOC.altova.com
raptorxml licenseserver localhost
raptorxml licenseserver 127.0.0.1
raptorxmlserver licenseserver 127.0.0.1
```

Les commandes ci-dessus spécifient, respectivement, la machine nommée `DOC.altova.com`, et la machine de l'utilisateur (`localhost` et `127.0.0.1`) qui fait marcher Altova LicenseServer. Dans tous les cas, la commande enregistre RaptorXML Server avec le LicenseServer sur la machine spécifiée. La dernière commande appelle le programme d'exécution de serveur pour exécuter la commande.

Options

Options are listed in short form (if available) and long form. You can use one or two dashes for both short and long forms. An option may or may not take a value. If it takes a value, it is written like this: **--option=value**. Values can be specified without quotes except in two cases: (i) when the value string contains spaces, or (ii) when explicitly stated in the description of the option that quotes are required. If an option takes a Boolean value and no value is specified, then the option's default value is `TRUE`. Use the **--h, --help** option to display information about the command.

- json [j]

  --j, --json = `true|false`

  Les valeurs sont `true|false`. Si `true`, imprime le résultat de la tentative d'enregistrement en tant qu'objet JSON parsable par machine.

5.9.2 assignlicense (uniquement Windows)

Syntaxe et description

La commande **assignlicense** télécharge un fichier de licence vers Altova LicenseServer avec lequel RaptorXML Server est enregistré (voir la commande `licenseserver`), et attribue la licence à RaptorXML Server. Elle prend le chemin d'un fichier de licence en tant que son argument. La commande vous permet aussi de tester la validité d'une licence.

```
raptorxml assignlicense [options] FILE
raptorxmlserver assignlicense [options] FILE
```


- L'argument *FILE* prend le chemin du fichier de licence.
- L'option `--test-only` charge le fichier de licence sur LicenseServer et valider la licence, mais n'attribue pas la licence à RaptorXML Server.
- La commande **assignlicense** peut être appelée depuis l'un des programme d'exécution : **raptorxml** ou **%SERVER-EXENAME-LC%>**.

Pour plus de détails concernant la licence, voir la documentation LicenseServer (https://www.altova.com/manual/fr/licenseserver/3.14/).

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Exemples

Des exemples de la commande `assignlicense` :

```
raptorxml assignlicense C:\licensepool\mylicensekey.altova_licenses
raptorxmlserver assignlicense C:\licensepool\mylicensekey.altova_licenses
raptorxml assignlicense --test-only=true C:\licensepool\mylicensekey.altova_licenses
```

- La première commande ci-dessus charge la licence spécifiée sur LicenseServer et l'attribue à RaptorXML Server.
- La deuxième commande appelle le programme exécution de serveur pour effectuer la même chose que la première commande.
- La troisième commande charge la licence spécifiée sur LicenseServer et la valide, sans l'attribuer à RaptorXML Server.

Options

Options are listed in short form (if available) and long form. You can use one or two dashes for both short and long forms. An option may or may not take a value. If it takes a value, it is written like this: **--option=value**. Values can be specified without quotes except in two cases: (i) when the value string contains spaces, or (ii) when explicitly stated in the description of the option that quotes are required. If an option takes a Boolean

value and no value is specified, then the option's default value is `TRUE`. Use the **`--h, --help`** option to display information about the command.

- test-only [t]

  `--t, --test-only =` `true|false`
  Les valeurs sont `true|false`. Si `true`, alors le fichier de licence est chargé sur LicenseServer et validé, mais pas attribué.

5.9.3 verifylicense (uniquement Windows)

Syntaxe et description

La commande **`verifylicense`** contrôle si le produit actuel est mis sous licence. De plus, l'option `--license-key` vous permet de vérifier si une clé de licence spécifique est assignée au produit.

```
raptorxml verifylicense [options]
raptorxmlserver verifylicense [options]
```

- Pour contrôler si une licence spécifique est attribuée à RaptorXML Server, fournir la clé de licence en tant que la valeur de l'option `--license-key`.
- La commande **`verifylicense`** peut être appelée depuis un des programmes d'exécution : **`raptorxml`** ou **`%SERVER-EXENAME-LC%>`**.

Pour plus de détails concernant la licence, voir la documentation LicenseServer (https://www.altova.com/manual/fr/licenseserver/3.14/).

- Casing and slashes on the command line

  `RaptorXML` (and **`RaptorXMLServer`** for administration commands) *on Windows*
  `raptorxml` (and **`raptorxmlserver`** for administration commands) *on Windows and Unix (Linux, Mac)*

  * Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
  * Use forward slashes on Linux and Mac, backslashes on Windows.

Exemples

Exemple de la commande **`verifylicense`** :

```
raptorxml verifylicenseserver
raptorxml verifylicenseserver --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-
ABCD123
raptorxmlserver verifylicenseserver --license-key=ABCD123-ABCD123-ABCD123-ABCD123-
ABCD123-ABCD123
```

- La première commande contrôle si RaptorXML Server est mis sous licence.
- La seconde commande contrôle si RaptorXML Server est mis sous licence avec la clé de licence

spécifiée avec l'option `--license-key`.

- La troisième commande est la même que la seconde commande, mais est exécutée par le programme d'exécution de serveur.

Options

Options are listed in short form (if available) and long form. You can use one or two dashes for both short and long forms. An option may or may not take a value. If it takes a value, it is written like this: **`--option=value`**. Values can be specified without quotes except in two cases: (i) when the value string contains spaces, or (ii) when explicitly stated in the description of the option that quotes are required. If an option takes a Boolean value and no value is specified, then the option's default value is `TRUE`. Use the **`--h, --help`** option to display information about the command.

▼ license-key [l]

`--l, --license-key =` *Value*

Contrôle si RaptorXML Server est mis sous licence avec la clé de licence spécifiée en tant que la valeur de cette option.

5.10 Commandes d'administration

Les commande d'administration (comme les commandes installation-en-tant-que-service et les commandes de licence) sont émises dans l'exécutable du serveur de RaptorXML Server (nommé **RaptorXMLServer**). Cet exécutable se situe par défaut dans :

Windows	`<ProgramFilesFolder>\Altova\RaptorXMLServer2024\bin\`**RaptorXMLServer.exe**
Linux	`/opt/Altova/RaptorXMLServer2024/bin/`**raptorxmlserver**
Mac	`/usr/local/Altova/RaptorXMLServer2024/bin/`**raptorxmlserver**

Usage

La syntaxe de ligne de commande :

```
raptorxmlserver --h | --help | --version | <command> [options] [arguments]
```

- `--help` (short form `--h`) affiche le texte d'aide de la commande donnée. Si aucune commande n'est nommée, alors toutes les commandes de l'exécutable sont regroupées, chacune présentant une brève description de la commande.
- `--version` affiche le numéro de version RaptorXML Server.
- `<command>` est la commande à exécuter. Les commandes sont affichées dans les sous-sections de cette section *(voir liste ci-dessous)*.
- `[options]` sont les options d'une commande ; elles sont regroupées et décrites avec leurs commandes respectives.
- `[arguments]` sont les arguments d'une commande ; elles sont regroupées et décrites avec leurs commandes respectives.

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Commandes d'administration

Les commandes de l'exécutable du serveur fournissent des fonctions d'administration. Elles sont regroupées ci-dessous dans les sous-sections de cette section :

5.10.1 install

Syntaxe et description

La commande `install` installe RaptorXML Server en tant que service sur la machine de serveur.

```
raptorxmlserver install [options]
```

- Veuillez noter qu'une installation de RaptorXML Server en tant que service ne permet pas de lancer automatiquement le service. Pour le lancer, utiliser la commande **`start`**.
- Pour désinstaller RaptorXML Server en tant que service, utiliser la commande **`uninstall`**.
- Utiliser l'option **`--h, --help`** pour afficher les informations concernant la commande

▼ Casse et barres obliques sur la ligne de commande

`RaptorXML(` et **`RaptorXMLServer`** pour des commandes d'administration) *sur Windows*
`raptorxml(` et **`raptorxmlserver`** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Exemple

Exemple de la commande **`install`** :

```
raptorxmlserver install
```

5.10.2 uninstall

Syntaxe et description

La commande **`uninstall`** désinstalle RaptorXML Server en tant que service sur la machine de serveur.

```
raptorxmlserver uninstall [options]
```

Pour réinstaller RaptorXML Server en tant que service, utiliser la commande **`install`**.

- Casse et barres obliques sur la ligne de commande

  `RaptorXML(` et **`RaptorXMLServer`** pour des commandes d'administration) *sur Windows*
  `raptorxml(` et **`raptorxmlserver`** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

  * Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
  * Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Exemple

Exemple de la commande **`uninstall`** :

```
raptorxmlserver uninstall
```

5.10.3 start

Syntaxe et description

La commande `start` lance RaptorXML Server en tant que service sur la machine de serveur.

```
raptorxmlserver start [options]
```

- Si RaptorXML Server n'est pas installé en tant que service, l'installer tout d'abord avec la commande **`install`** (avant de le démarrer).
- Pour désinstaller RaptorXML Server en tant que service, utiliser la commande **`uninstall`**.
- Utiliser l'option **`--h, --help`** pour afficher les informations concernant la commande.

- Casse et barres obliques sur la ligne de commande

  `RaptorXML(` et **`RaptorXMLServer`** pour des commandes d'administration) *sur Windows*
  `raptorxml(` et **`raptorxmlserver`** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

  * Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
  * Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

- Barre oblique inversée et des espaces sur des systèmes Windows

  Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **`"My File"`**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par

exemple, `"C:\My directory\"`) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement `\"` représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : `\\"`. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : `"C:\My Directory\\"`.

Exemple

Exemple de la commande **start** :

```
raptorxmlserver start
```

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.

▾ config [c]

--c, --config = *File*
Spécifie le chemin dans un fichier de configuration.

▾ fork

--fork = `true|false`
Fournit la capacité à fourcher lors de l'utilisation de `init` sur des serveurs Unix. Le réglage par défaut est `false`.

▾ port

--port = *PortNumber*
Le numéro de port dans l'instance de débogage de RaptorXML Server.

5.10.4 setdeflang

Syntaxe et description

La commande **setdeflang** (la forme abrégée est **sdl**) définit la langue par défaut de RaptorXML Server. Les langues disponibles sont anglais (`en`), allemand (`de`), espagnol (`es`), français (`fr`) et japonais (`ja`). La commande prend un argument *LanguageCode* obligatoire.

```
raptorxml setdeflang [options] LanguageCode
```

```
raptorxmlserver setdeflang [options] LanguageCode
```

- L'argument *LanguageCode* est requis et définit la langue par défaut de RaptorXML Server. Les valeurs respectives à utiliser sont : en, de, es, fr, ja.
- La commande **setdeflang** peut être appelée depuis un des programmes d'exécution : **raptorxml** or **raptorxmlserver**.
- Utiliser l'option **--h, --help** pour afficher les informations concernant la commande.

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (raptorxml et raptorxmlserver) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (RaptorXML) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Exemples

Exemples de la commande **setdeflang** (**sdl**) :

```
raptorxml sdl de
raptorxml setdeflang es
raptorxmlserver setdeflang es
```

- La première commande définit le langage par défaut de RaptorXML Server en allemand.
- La seconde commande définit le langage par défaut de RaptorXML Server en espagnol.
- La troisième commande est la même que la seconde commande, mais est exécutée par le programme d'exécution de serveur.

Options

Use the **--h, --help** option to display information about the command.

5.10.5 licenseserver

Syntaxe et description

La commande **licenseserver** enregistre RaptorXML Server avec Altova LicenseServer spécifié par l'argument *Server-Or-IP-Address*. Pour pouvoir exécuter la commande licenseserver correctement, les deux serveurs (RaptorXML Server et LicenseServer) doivent être connectés sur le réseau et le LicenseServer doit fonctionner. Vous devez posséder des privilèges d'administrateur pour pouvoir enregistrer RaptorXML Server auprès du LicenseServer.

```
raptorxml licenseserver [options] Server-Or-IP-Address
```



```
raptorxmlserver licenseserver [options] Server-Or-IP-Address
```

- L'argument *Server-Or-IP-Address* prend le nom ou l'adresse IP de l'appareil de LicenseServer.
- La commande **licenseserver** peut être appelée depuis un des programmes d'exécution : **raptorxml** ou **%SERVER-EXENAME-LC%>**.

Une fois que RaptorXML Server a été enregistré avec succès auprès de LicenseServer, vous recevrez un message. Le message affichera aussi l'URL du LicenseServer. Vous pouvez maintenant vous rendre sur LicenseServer pour attribuer une licence à RaptorXML Server. Pour plus de détails concernant la licence, voir la documentation LicenseServer (https://www.altova.com/manual/fr/licenseserver/3.14/).

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (raptorxml and raptorxmlserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (RaptorXML) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

▼ Backslashes, spaces, and special characters on Windows systems

On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **"My File"**. Note, however, that a backslash followed by a double-quotation mark (for example, **"C:\My directory\"**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **\"** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **\\"**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **"C:\My Directory\ \"**.

Exemples

Exemples de la commande **licenseserver** :

```
raptorxml licenseserver DOC.altova.com
raptorxml licenseserver localhost
raptorxml licenseserver 127.0.0.1
raptorxmlserver licenseserver 127.0.0.1
```

Les commandes ci-dessus spécifient, respectivement, la machine nommée DOC.altova.com, et la machine de l'utilisateur (localhost et 127.0.0.1) qui fait marcher Altova LicenseServer. Dans tous les cas, la commande enregistre RaptorXML Server avec le LicenseServer sur la machine spécifiée. La dernière commande appelle le programme d'exécution de serveur pour exécuter la commande.

Options

Options are listed in short form (if available) and long form. You can use one or two dashes for both short and long forms. An option may or may not take a value. If it takes a value, it is written like this: **--option=value**. Values can be specified without quotes except in two cases: (i) when the value string contains spaces, or (ii)

when explicitly stated in the description of the option that quotes are required. If an option takes a Boolean value and no value is specified, then the option's default value is `TRUE`. Use the **`--h, --help`** option to display information about the command.

- json [j]

  `--j, --json =` `true|false`
  Les valeurs sont `true|false`. Si `true`, imprime le résultat de la tentative d'enregistrement en tant qu'objet JSON parsable par machine.

5.10.6 assignlicense (uniquement Windows)

Syntaxe et description

La commande **`assignlicense`** télécharge un fichier de licence vers Altova LicenseServer avec lequel RaptorXML Server est enregistré (voir la commande `licenseserver`), et attribue la licence à RaptorXML Server. Elle prend le chemin d'un fichier de licence en tant que son argument. La commande vous permet aussi de tester la validité d'une licence.

```
raptorxml assignlicense [options] FILE
raptorxmlserver assignlicense [options] FILE
```

- L'argument *`FILE`* prend le chemin du fichier de licence.
- L'option `--test-only` charge le fichier de licence sur LicenseServer et valider la licence, mais n'attribue pas la licence à RaptorXML Server.
- La commande **`assignlicense`** peut être appelée depuis l'un des programme d'exécution : **`raptorxml`** ou **`%SERVER-EXENAME-LC%`**.

Pour plus de détails concernant la licence, voir la documentation LicenseServer (https://www.altova.com/manual/fr/licenseserver/3.14/).

- Casing and slashes on the command line

  `RaptorXML` (and **`RaptorXMLServer`** for administration commands) *on Windows*
  `raptorxml` (and **`raptorxmlserver`** for administration commands) *on Windows and Unix (Linux, Mac)*

  * Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
  * Use forward slashes on Linux and Mac, backslashes on Windows.

- Backslashes, spaces, and special characters on Windows systems

  On Windows systems: When spaces or special characters occur in strings (for example in file or folder names, or company, person or product names), use quotes: for example, **`"My File"`**. Note, however, that a backslash followed by a double-quotation mark (for example, **`"C:\My directory\"`**) might not be read correctly. This is because the backslash character is also used to indicate the start of an escape sequence, and the escape sequence **`\"`** stands for the double-quotation mark character. If you want to escape this sequence of characters, use a preceding backslash, like this: **`\\"`**. To summarize: If you need to write a file path that contains spaces or an end backslash, write it like this: **`"C:\My Directory\\"`**.

Exemples

Des exemples de la commande `assignlicense` :

```
raptorxml assignlicense C:\licensepool\mylicensekey.altova_licenses
raptorxmlserver assignlicense C:\licensepool\mylicensekey.altova_licenses
raptorxml assignlicense --test-only=true C:\licensepool\mylicensekey.altova_licenses
```

- La première commande ci-dessus charge la licence spécifiée sur LicenseServer et l'attribue à RaptorXML Server.
- La deuxième commande appelle le programme exécution de serveur pour effectuer la même chose que la première commande.
- La troisième commande charge la licence spécifiée sur LicenseServer et la valide, sans l'attribuer à RaptorXML Server.

Options

Options are listed in short form (if available) and long form. You can use one or two dashes for both short and long forms. An option may or may not take a value. If it takes a value, it is written like this: **`--option=value`**. Values can be specified without quotes except in two cases: (i) when the value string contains spaces, or (ii) when explicitly stated in the description of the option that quotes are required. If an option takes a Boolean value and no value is specified, then the option's default value is `TRUE`. Use the **`--h, --help`** option to display information about the command.

▼ test-only [t]

`--t, --test-only =` `true|false`

Les valeurs sont `true|false`. Si `true`, alors le fichier de licence est chargé sur LicenseServer et validé, mais pas attribué.

5.10.7 verifylicense (uniquement Windows)

Syntaxe et description

La commande **`verifylicense`** contrôle si le produit actuel est mis sous licence. De plus, l'option `--license-key` vous permet de vérifier si une clé de licence spécifique est assignée au produit.

```
raptorxml verifylicense [options]
raptorxmlserver verifylicense [options]
```

- Pour contrôler si une licence spécifique est attribuée à RaptorXML Server, fournir la clé de licence en tant que la valeur de l'option `--license-key`.
- La commande **`verifylicense`** peut être appelée depuis un des programmes d'exécution : **`raptorxml`** ou **`%SERVER-EXENAME-LC%`**.

Pour plus de détails concernant la licence, voir la documentation LicenseServer (https://www.altova.com/manual/fr/licenseserver/3.14/).

▼ Casing and slashes on the command line

RaptorXML (and **RaptorXMLServer** for administration commands) *on Windows*
raptorxml (and **raptorxmlserver** for administration commands) *on Windows and Unix (Linux, Mac)*

* Note that lowercase (`raptorxml` and `raptorxmlserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`RaptorXML`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

Exemples

Exemple de la commande **verifylicense** :

```
raptorxml verifylicenseserver
raptorxml verifylicenseserver --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-ABCD123
raptorxmlserver verifylicenseserver --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-ABCD123
```

- La première commande contrôle si RaptorXML Server est mis sous licence.
- La seconde commande contrôle si RaptorXML Server est mis sous licence avec la clé de licence spécifiée avec l'option `--license-key`.
- La troisième commande est la même que la seconde commande, mais est exécutée par le programme d'exécution de serveur.

Options

Options are listed in short form (if available) and long form. You can use one or two dashes for both short and long forms. An option may or may not take a value. If it takes a value, it is written like this: **--option=value**. Values can be specified without quotes except in two cases: (i) when the value string contains spaces, or (ii) when explicitly stated in the description of the option that quotes are required. If an option takes a Boolean value and no value is specified, then the option's default value is `TRUE`. Use the **--h, --help** option to display information about the command.

▼ license-key [l]

--l, --license-key = *Value*
Contrôle si RaptorXML Server est mis sous licence avec la clé de licence spécifiée en tant que la valeur de cette option.

5.10.8 createconfig

Syntaxe et description

La commande **createconfig** écrase le fichier de configuration de serveur avec des valeurs par défaut.

```
raptorxmlserver createconfig [options]
```

- L'option **--lang** spécifie le langage par défaut du fichier de configuration de serveur.

Pour plus d'informations concernant les fichiers de configuration de serveur, voir Configurer le serveur 231.

- Casse et barres obliques sur la ligne de commande

  RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
  raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

  * Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
  * Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Examples

Exemples de la commande **createconfig** :

```
raptorxml createconfig
raptorxml createconfig --lang=de
```

Options

- lang

  --lang = `en|de|es|fr|ja`
  Spécifie la langue par défaut du fichier de configuration de serveur. Les options suivantes sont disponibles : anglais (`en`), allemand (`de`), espagnol (`es`), français (`fr`), japanais (`ja`). Si l'option n'est pas spécifiée, l'anglais est choisi en tant que langage par défaut.

Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.

5.10.9 exportresourcestrings

Syntaxe et description

La commande **exportresourcestrings** sort un fichier XML contenant les strings de ressource de l'application RaptorXML Server dans la langue spécifiée. Les langues d'exportation disponibles sont allemand (`de`), anglais (`en`), espagnol (`es`), français (`fr`) et japonais (`ja`).

```
raptorxml exportresourcestrings [options] LanguageCode XMLOutputFile
raptorxmlserver exportresourcestrings [options] LanguageCode XMLOutputFile
```

- L'argument ***LanguageCode*** donne la langue des strings de ressource dans le fichier XML de sortie ; il s'agit de la langue d'exportation *export language*. Les langues d'exportation autorisées (avec leurs codes de langue respectifs entre parenthèses) sont : allemand (`de`), anglais (`en`), espagnol (`es`), français (`fr`) et japonais (`ja`).
- L'argument ***XMLOutputFile*** spécifie le chemin et le nom du fichier XML de sortie .
- La commande **exportresourcestrings** peut être appelée depuis un programme d'exécution : **raptorxml** ou **%SERVER-EXENAME-LC%>**.

Pour créer des localisations, veuillez noter les points suivants.

▼ Casse et barres obliques sur la ligne de commande

RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

▼ Barre oblique inversée et des espaces sur des systèmes Windows

Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Exemples

Exemples de la commande **exportresourcestrings** :

```
raptorxml exportresourcestrings de c:\Strings.xml
raptorxmlserver exportresourcestrings de c:\Strings.xml
```

- La première commande ci-dessus crée un fichier appelé `Strings.xml` sous `c:\` qui contient les strings de ressource de RaptorXML Server en allemand.
- La deuxième commande appelle le server-executable à procéder de la même manière que dans le premier exemple.

Créer des versions localisées de RaptorXML Server

Vous pouvez créer une version localisée de RaptorXML Server pour toute langue de votre choix. Cinq versions localisées (allemand, anglais, espagnol, français et japonais) sont déjà disponibles sous le dossier `C:\Program Files (x86)\Altova\RaptorXMLServer2024\bin` , et n'ont donc pas besoin d'être créées.

Créer une version localisée comme suit :

1. Générer un fichier XML contenant des strings de ressource en utilisant la commande `exportresourcestrings` (*voir la syntaxe de commande ci-dessus*). Les strings de ressource dans ce fichier XML seront ceux d'une des cinq langues prises en charge : allemand (`de`), anglais (`en`), espagnol (`es`), français (`fr`) ou japonais (`ja`), conformément à l'argument ***LanguageCode*** utilisé avec la commande.
2. Traduire les strings de ressource à partir de l'une des cinq langues prises en charge dans la langue cible. Les strings de ressource sont les contenus des éléments `<string>` dans le fichier XML. Ne pas traduire les variables dans les accolades comme `{option}` ou `{product}`.
3. Contactez Altova Support pour générer un fichier DLL RaptorXML Server localisé à partir de votre fichier XML traduit.
4. Après avoir reçu votre fichier DLL localisé de Altova Support, enregistrez le fichier DLL sous le dossier `C:\Program Files (x86)\Altova\RaptorXMLServer2024\bin` . Votre fichier DLL aura un nom de la forme `RaptorXML2024_lc.dll`. La partie `_lc` du nom contient le code de langue. Par exemple, dans `RaptorXML2024_de.dll`, la partie `de` est le code de langue pour l'allemand (Deutsch).
5. Exécutez la commande `setdeflang` pour définir votre fichier DLL localisé DLL en tant qu'application RaptorXML Server à utiliser. Pour l'argument de la commande `setdeflang`, utilisez le code de langue qui fait partie du nom DLL.

Note : Altova RaptorXML Server est livré avec une prise en charge pour les cinq langues : allemand, anglais, espagnol, français et japonais. Donc vous ne devez pas créer de version localisée pour ces langues. Pour définir une de ces langues en tant que langue par défaut, utilisez la commande RaptorXML Serverde `setdeflang`.

5.10.10 debug

Syntaxe et description

La commande **`debug`** lance RaptorXML Server pour le débogage, pas en tant que service. Pour arrêter RaptorXML Server dans ce mode, appuyer sur **Ctrl+C**.

```
raptorxmlserver debug [options]
```

▼ Casse et barres obliques sur la ligne de commande

`RaptorXML(` et **`RaptorXMLServer`** pour des commandes d'administration) *sur Windows*
`raptorxml(` et **`raptorxmlserver`** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

* Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
* Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers

l'arrière sur Windows.

- Barre oblique inversée et des espaces sur des systèmes Windows

  Dans les systèmes Windows : lorsque des espaces se produisent (par exemple, dans les noms de fichier ou de dossier, ou les noms d'entreprise, de personne ou de produit), utiliser des guillemets : par exemple, **"My File"**. Veuillez noter, néanmoins qu'une barre oblique inversée suivie par un double guillemet (par exemple, **"C:\My directory\"**) peut ne pas être lue correctement. Cela est dû au fait que le caractère de barre oblique inversée est également utilisé pour indiquer le début d'une séquence d'échappement, et la séquence d'échappement **\"** représente le caractère de marque de double guillemet. Si vous souhaitez échapper cette séquence des caractères, utiliser une barre oblique inversée précédente comme ceci : **\ \"**. Pour résumer : Si vous souhaitez écrire un chemin de fichier qui contient des espaces et des barre oblique inversée de fin, l'écrire comme ceci : **"C:\My Directory\\"**.

Exemple

Exemple de la commande de **debug** :

```
raptorxmlserver debug
```

Options

Les options sont recensées dans la forme abrégée (si disponible) et dans la forme longue. Vous pouvez utiliser un ou deux tirets pour les formes abrégées et longues. Une option peut ou ne peut pas prendre une valeur. Si elle prend une valeur, elle est écrite comme ceci : **--option=value**. Des valeurs peuvent être spécifiées sans guillemets sauf dans deux cas : (i) si le string valeur contient des espace, ou (ii) s'il est explicitement indiqué dans la description que des guillemets sont exigés. Si une option prend une valeur booléenne et qu'aucune valeur n'est spécifiée, alors la valeur par défaut de l'option est `TRUE`. Utiliser l'option **--h, --help** pour afficher des informations à propos de la commande.

- config [c]

  --c, --config = *File*
  Spécifie le chemin d'un fichier de configuration.

- port

  --port = *PortNumber*
  Le numéro de port de l'instance de débogage de RaptorXML Server

5.10.11 help

Syntaxe et description

La commande **help** prend un seul argument (`Command`), qui est le nom de la commande pour laquelle l'aide est requise. Elle affiche la syntaxe de la commande, ses options, et d'autres informations pertinentes. Si l'argument `Command` n'est pas spécifié, toutes les commandes du programme d'exécution sont recensés, chacun présentant une brève description de texte. La commande **help** peut être appelée depuis leur

programme d'exécution : **raptorxml** ou **raptorxmlserver**.

```
raptorxml help Command
raptorxmlserver help Command
```

- Casse et barres obliques sur la ligne de commande

  RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
  raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

  * Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
  * Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Exemple

Exemples de la commande **help** pour afficher des informations concernant la commande **licenserver** (cette commande est disponible dans les deux programmes d'exécution):

```
raptorxml help licenseserver
raptorxmlserver help licenseserver
```

L'option --help

L'information Help à propos d'une commande est également disponible en utilisant l'option `--help` avec la commande pour laquelle l'information d'aide est nécessaire. Les deux commandes ci-dessous produisent les mêmes résultats :

```
raptorxml licenseserver --help
```

La commande ci-dessus utilise l'option `--help` de la commande **licenserver**.

```
raptorxml help licenseserver
```

La commande **help** prend `licenseserver` en tant que son argument.

Les deux commandes affichent des informations d'aide concernant la commande **licenseserver**.

5.10.12 version

Syntaxe et description

La commande **version** affiche le numéro de version de RaptorXML Server. Il peut être appelé depuis un des programmes d'exécution : **raptorxml** ou **raptorxmlserver**.

```
raptorxml version
raptorxmlserver version
```

- Casse et barres obliques sur la ligne de commande

  RaptorXML(et **RaptorXMLServer** pour des commandes d'administration) *sur Windows*
  raptorxml(et **raptorxmlserver** pour des commandes d'administration) *sur Windows et Unix (Linux, Mac)*

  * Veuillez noter que la casse minuscule (`raptorxml` et `raptorxmlserver`) fonctionne sur toutes les plate-formes (Windows, Linux et Mac), alors que la casse majuscule (`RaptorXML`) fonctionne uniquement sur Windows et Mac.
  * Utiliser des barres obliques basculée vers l'avant sur Linux et Mac, des barres obliques basculées vers l'arrière sur Windows.

Exemple

Exemples de la commande `version` :

```
raptorxml version
raptorxmlserver version
```

5.11 Options

Cette section contient une description de toutes les options de CLI, organisées par les fonctions. Pour découvrir quelles options peuvent être utilisées avec chaque commande, voir la description des commandes respectives.

5.11.1 Catalogues, Ressources globales, fichiers ZIP

▼ catalog

--catalog = *FILE*
Spécifie le chemin absolu vers un fichier de catalogue root qui n'est pas le fichier catalogue root installé. La valeur par défaut est le chemin absolu vers le fichier de catalogue root installé (`<installation-folder>\Altova\RaptorXMLServer2024\etc\RootCatalog.xml`). Voir la section, Catalogues XML 47, pour plus d'informations à propos du travail avec les catalogues.

▼ user-catalog

--user-catalog = *FILE*
Spécifie le chemin absolu vers un catalogue XML à utiliser en plus du catalogue root. Voir la section, Catalogues XML 47, pour plus d'informations concernant le travail avec des catalogues.

▼ enable-globalresources

--enable-globalresources = `true|false`
Active les ressources globales 54. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Spécifie la configuration active de la ressource globale 54 (et active les ressources globales) 54.

▼ globalresourcefile [gr]

--gr | --globalresourcefile = *FILE*
Spécifie le fichier de ressource globale 54 (et active les ressources globales) 54.

▼ recurse

--recurse = `true|false`

Utilisé pour sélectionner des fichiers dans le cadre de sous-répertoires, y compris des archives ZIP. Si `true`, l'argument *`InputFile`* de la commande sélectionnera aussi le fichier spécifié dans les sous-répertoires. Par exemple : "`test.zip|zip\test.xml`" sélectionnera des fichiers nommés `test.xml` à tous les niveaux de dossier du dossier zip. Les références aux fichiers ZIP doivent être indiquées entre guillemets. Les caractères génériques `*` et `?` peuvent être utilisés. Donc, `*.xml` sélectionnera tous les fichiers `.xml` dans le dossier (zip). La valeur par défaut de l'option est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

5.11.2 Messages, Erreurs, Aide, Timeout, Version

- error-format

  `--error-format =` `text|shortxml|longxml`
  Spécifie le format de la sortie d'erreur. La valeur par défaut est `text`. Les autres options génèrent des formats XML, avec `longxml` générant plus de détails.

- error-limit

  `--error-limit =` *`N | unlimited`*
  Spécifie la limite d'erreur avec une plage de valeur de `1` à `999` ou `unlimited`. La valeur par défaut est `100`. Le traitement s'arrête lorsque la limite d'erreur est atteinte. Utile pour limiter l'utilisation du processeur pendant la validation/transformation.

- help

  `--help`
  Affiche le texte d'aide pour la commande. Par exemple, `valany --h`. (en alternative, la commande `help` peut être utilisée avec un argument. Par exemple : `help valany`.)

- info-limit

  `--info-limit =` *`N | unlimited`*
  Spécifie la limite de message d'information dans la plage `1-65535` ou `unlimited`. Le traitement continue si la limite d'info spécifiée est atteinte, mais les messages ultérieurs ne sont pas rapportés. La valeur par défaut est `100`.

- log-output

  `--log-output =` *`FILE`*
  Écrit la sortie de log dans l'URL de fichier spécifié. Veuillez vous assurer que la CLI a une permission d'écriture dans l'emplacement de la sortie.

- network-timeout

  `--network-timeout =` *`VALEUR`*
  Spécifie le timeout en secondes pour les opérations I/O à distance. La valeur par défaut est : `40000`.

- verbose

  `--verbose =` `true|false`
  Une valeur de `true` permet la sortie d'informations supplémentaires pendant la validation. La valeur par

défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- verbose-output

  `--verbose-output =` *`FILE`*
  Écrit la sortie verbeuse sur *`FILE`*.

- version

  `--version`
  Affiche la version de RaptorXML Server. Si utilisé avec une commande, placer `--version` avant la commande.

- warning-limit

  `--warning-limit =` *`N | unlimited`*
  Spécifie la limite d'avertissement dans la plage `1-65535` ou `unlimited`. Le traitement se poursuit si cette limite a été atteinte, mais d'autres avertissements ne sont pas rapportés. La valeur par défaut est `100`.

5.11.3 Traitement

- listfile

  `--listfile =` `true|false`
  Si `true`, traiter l'argument *`InputFile`* de la commande en tant que fichier de texte contenant un nom de fichier par ligne. La valeur par défaut est `false`. (Une alternative est de lister les fichiers sur la CLI avec un espace en tant que séparateur. Veuillez noter, néanmoins, que les CLI ont une limitation de caractères maximum.) Veuillez noter que l'option `--listfile` s'applique uniquement aux arguments, et non pas aux options.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- parallel-assessment [pa]

  `--pa | --parallel-assessment =` `true|false`
  Si configuré sur `true`, l'évaluation de validité de schéma est effectuée en parallèle. Cela signifie qu'il y a plus de 128 éléments à tout niveau, ces éléments sont traités en parallèle, en utilisant plusieurs threads. Des fichiers XML très volumineux peuvent donc être traités plus rapidement si cette option est activée. L'évaluation parallèle a lieu un niveau hiérarchique à la fois, mais peut se produire à plusieurs niveaux différents dans une seule infobulle. Veuillez noter que l'évaluation parallèle ne fonctionne pas en mode streaming. C'est pour cette raison que l'option `--streaming` est ignorée si l'évaluation `--parallel-` est définie sur `true`. De même, les besoins en mémoire sont plus élevés lorsque l'option `--parallel-assessment` est utilisée. Le paramètre par défaut est `false`. La forme abrégée pour l'option est `--pa`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- script

  `--script =` *`FILE`*
  Exécute le script Python dans le fichier soumis une fois que la validation a été terminée. Ajouter l'option plusieurs fois pour spécifier plus d'un seul script.

- script-api-version

  --api, --script-api-version = `1; 2; 2.1 to 2.4; 2.4.1; 2.5 to 2.8; 2.8.1 to 2.8.6; 2.9.0`
  Spécifie la version API Python à utiliser pour le script. La valeur par défaut est la dernière version, actuellement **`2.9.0`**. À la place des valeurs entières `1` et `2`, vous pouvez aussi utiliser les valeurs correspondantes `1.0` et `2.0`. De la même manière, vous pouvez utiliser les trois chiffres `2.5.0` pour les deux `2.5`. Voir aussi le chapitre Versions Python API 347.

- script-param

  --script-param = *KEY:VALUE*
  Des paramètres spécifiés par l'utilisateur supplémentaires qui peuvent être accédés au cours de l'exécution des scripts Python. Ajouter l'option plusieurs fois pour spécifier plus d'un paramètre de script.

- streaming

  --streaming = `true|false`
  Permet la validation de streaming. La valeur par défaut est `true`. Dans le mode streaming, les données stockées dans la mémoire sont minimisées et le traitement est plus rapide. L'inconvénient est que l'information que peut être requise par conséquent, par exemple, un modèle de données du document d'instance XML —ne sera pas disponible. Dans des situations où cela est significatif, le mode de streaming devra être éteint (en donnant à `--streaming` une valeur de `false`). En utilisant l'option `--script` avec la commande `valxml-withxsd`, désactiver le streaming. Veuillez noter que l'option `--streaming` est ignorée si`--parallel-assessment` est défini sur `true`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- xml-validation-error-as-warning

  --xml-validation-error-as-warning = `true|false`
  Si `true`, traite les erreurs de validation en tant qu'avertissements. Si les erreurs sont traitées en tant qu'avertissements, les traitements supplémentaires, comme des transformations XSLT, continueront quelles que soient les erreurs. La valeur par défaut est `false`.

5.11.4 XML

- assessment-mode

  --assessment-mode = `lax|strict`
  Spécifie le mode d'évaluation de validité de schéma tel que défini dans les spécifications XSD. La valeur par défaut est `strict`. Le document d'instance XML sera validé conformément au mode spécifié avec cette option.

- dtd

  --dtd = *FILE*
  Spécifie le document DTD externe à utiliser pour la validation. Si une référence à un DTD externe est présente dans le document XM, alors l'option CLI contourne la référence externe.

- load-xml-with-psvi

  --load-xml-with-psvi = `true|false`
  Active la validation des fichiers XML d'entrée et génère des informations de post-schema-validation pour les fichiers. La valeur par défaut est : `true`.

- namespaces

  --namespaces = `true|false`
  Permet le traitement sensible aux espaces de noms. Cela est utile pour contrôler l'instance XML pour des erreurs dues à des espaces de noms incorrectes. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- xinclude

  --xinclude = `true|false`
  Active la prise en charge de XML Inclusions (XInclude). La valeur par défaut est `false`. Si `false`, les élément `include` d'XInclude sont ignorés.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- xml-mode

  --xml-mode = `wf|id|valid`
  Spécifie le mode de traitement XML à utiliser pour le document d'instance XML : `wf`=wellformed check (vérification de la bonne forme); `id`=bien formé avec des contrôles ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document d'instance chargé depuis les traitements référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

- xml-validation-error-as-warning

  --xml-validation-error-as-warning = `true|false`
  Si `true`, traite les erreurs de validation en tant qu'avertissements. Si les erreurs sont traitées en tant qu'avertissements, les traitements supplémentaires, comme des transformations XSLT, continueront quelles que soient les erreurs. La valeur par défaut est `false`.

- xsd

  --xsd = *FILE*
  Spécifie un ou plusieurs documents de Schéma XML à utiliser pour la validation des documents d'instance XML. Ajouter l'option plusieurs fois pour spécifier plus d'un document de schéma.

5.11.5 XSD

- assessment-mode

  --assessment-mode = `lax|strict`
  Spécifie le mode d'évaluation de validité de schéma tel que défini dans les spécifications XSD. La valeur par défaut est `strict`. Le document d'instance XML sera validé conformément au mode spécifié avec cette option.

- ct-restrict-mode

--ct-restrict-mode = `1.0|1.1|default`
Spécifie comment contrôler les restrictions de type complexes. Une valeur de **1.0** contrôles les restrictions de type complexes telles que définies dans la spécification XSD 1.0 — même dans le mode de validation XSD 1.1. Une valeur de **1.1** contrôles les restrictions de type complexe telles que définies dans la spécification XSD 1.1 — même dans le mode de validation XSD 1.0. Une valeur de **default** contrôle les restrictions de type complexes telles que définies dans la spécification XSD du mode de validation actuel (1.0 ou 1.1). La valeur par défaut est **default**.

▼ namespaces

--namespaces = `true|false`
Permet le traitement sensible aux espaces de noms. Cela est utile pour contrôler l'instance XML pour des erreurs dues à des espaces de noms incorrectes. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ report-import-namespace-mismatch-as-warning

--report-import-namespace-mismatch-as-warning = `true|false`
Réduit des erreurs de non-correspondance d'espace de noms ou d'espace de noms cible lors de l'importation de schémas avec **xs:import** et les fait passer d'erreurs à des avertissements. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ schema-imports

--schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Spécifie le comportement des éléments `xs:import`, chacun d'entre eux ayant un attribut `namespace` optionnel et un attribut `schemaLocation` optionnel : `<import namespace="someNS" schemaLocation="someURL">`. L'option spécifie s'il faut charger un document de schéma ou juste mettre un espace de noms sous licence, et, si un document de schéma doit être chargé, quelle information doit être utilisée pour le trouver. Défaut : `load-preferring-schemalocation`.
Le comportement est le suivant :

- `load-by-schemalocation`: La valeur de l'attribut `schemaLocation` est utilisée pour situer le schéma, en prenant en compte les mappages de catalogue[47]. Si l'attribut de l'espace de nom est présent, l'espace de noms est importé (licencé).
- `load-preferring-schemalocation`: Si l'attribut `schemaLocation` est présent, il est utilisé en prenant en compte les mappages de catalogue[47]. Si l'attribut `schemaLocation` est présent, la valeur de l'attribut `namespace` est utilisée via un mappage de catalogue[47]. C'est la **valeur par défaut**.
- `load-by-namespace`: La valeur de l'attribut `namespace` est utilisée pour situer le schéma via un mappage de catalogue[47].
- `load-combining-both`: Si soit l'Attribut `namespace` ou l'attribut `schemaLocation` a un mappage de catalogue[47], le mappage est utilisé. Si les deux ont des mappages de catalogue[47], alors la valeur de l'option `--schema-mapping` (option XML/XSD[213]) décide de quel mappage utiliser. Si aucun mappage de catalogue[47] n'est présent, l'attribut `schemaLocation` est utilisé.
- `license-namespace-only`: Le nom d'espace est importé. Aucun document de schéma n'est importé.

▼ schemalocation-hints

--schemalocation-hints = `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
Spécifie le comportement des attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` : S'il

faut charger un document de schéma et, si oui, quelle information doit être utilisée pour la trouver. Défaut : `load-by-schemalocation`.

- La valeur `load-by-schemalocation` utilise l'URL de l'emplacement de schéma[397] dans les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` dans les documents d'instance XML. Il s'agit de la **valeur par défaut**.
- La valeur `load-by-namespace` prend la part d'espace de nom[397] `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` et localise le schéma par le biais d'un mappage de catalogue[47].
- Si `load-combining-both` est utilisé et si soit la partie espace de noms ou la partie URL a un mappage de catalogue[47], alors le mappage de catalogue[47] est utilisé. Si tous deux ont des mappages de catalogue[47], alors la valeur de l'option de `--schema-mapping` (option XML/XSD[213]) de quel mappage utiliser. Si ni l'espace de noms ni l'URL n'a un mappage de catalogue, l'URL est utilisée.
- Si la valeur de l'option est `ignore`, les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` seront ignorés tous les deux.

▼ schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`

Si l'emplacement de schéma et l'espace de noms sont tous les deux utilisés pour trouver un document de schéma, spécifie lequel des deux doit être utilisé pendant la consultation du catalogue. (Si soit l'option `--schemalocation-hints` ou l'option `--schema-imports` a une valeur de `load-combining-both`, et si les parties d'espace de noms et d'URL impliquées ont toutes les deux des mappages de catalogue[47], alors la valeur de cette option spécifie lequel des deux mappages utiliser (mappage d'espace de noms ou mappage URL ; la valeur `prefer-schemalocation` réfère au mappage URL).) Défaut : `prefer-schemalocation`.

▼ xml-mode-for-schemas

`--xml-mode-for-schemas =` `wf|id|valid`

Spécifie le mode de traitement XML à utiliser pour les documents de schéma XML : `wf`=vérification de la bonne formation ; `id`=contrôles de la bonne formation avec ID/IDREF ; `valid`=validation. La valeur par défaut est `wf`. Veuillez noter qu'une valeur de `valid` exige que chaque document de schéma chargé pendant le traitement référence un DTD. Si aucun DTD n'existe, une erreur est rapportée.

▼ xsd-version

`--xsd-version =` `1.0|1.1|detect`

Spécifie la version du langage de définition de Schéma W3C (XSD) à utiliser. Le défaut est `1.0`. Cette option peut être utile pour découvrir comment un schéma compatible à 1.0 n'est pas compatible à 1.1. L'option `detect` est une fonction spécifique à Altova. Elle permet la détection de la version du document de Schéma XML (`1.0` ou `1.1`) en lisant la valeur de l'attribut `vc:minVersion` de l'élément du document `<xs:schema>`. Si la valeur de l'attribut `@vc:minVersion` est `1.1`, le schéma est détecté comme étant de version `1.1`. Pour toute autre valeur, ou si l'attribut `@vc:minVersion` est absent, le schéma est détecté comme étant de version `1.0`.

5.11.6 XQuery

▼ indent-characters

`--indent-characters =` *`VALUE`*

Spécifie le string de caractère à utiliser en tant que retrait.

▼ input

--input = *FILE*
L'URL du fichier XML à transformer.

▼ keep-formatting

--keep-formatting = `true|false`
Garde le formatage du document cible autant que possible. La valeur par défaut est : `true`.

▼ omit-xml-declaration

--omit-xml-declaration = `true|false`
Option de sérialisation pour spécifier si la déclaration XML doit être omise de la sortie ou pas. Si `true`, il n'y aura pas de déclaration XML dans le document de sortie. Si `false`, une déclaration XML sera incluse. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ output, xsltoutput

output = *FILE,* **xsltoutput =** *FILE*
L'URL du fichier sortie-primaire. Par exemple, dans le cas d'une sortie HTML de fichier multiple, le fichier de sortie primaire sera l'emplacement du fichier HTML du point d'entrée. Les fichiers de sortie supplémentaires, comme des fichiers d'image générée, sont rapportés en tant que `xslt-additional-output-files`. Si aucune option `--output` ou `--xsltoutput` n'est spécifiée, la sortie est écrite sur la sortie standard.

▼ output-encoding

--output-encoding = *VALUE*
La valeur de l'attribut d'encodage dans le document de sortie. Les valeurs valides sont des noms dans le registre d'ensemble de caractères IANA. La valeur par défaut est `UTF-8`.

▼ output-indent

--output-indent = `true|false`
Si `true`, la sortie sera en retrait conformément à sa structure hiérarchique. Si `false`, il n'y aura pas de retrait hiérarchique. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ output-method

--output-method = `xml|html|xhtml|text`
Spécifie le format de sortie. La valeur par défaut est `xml`.

▼ param [p]

--p | --param = *KEY:VALUE*

⊟ ***XQuery***

Spécifie la valeur d'un paramètre externe. Un paramètre externe est déclaré dans le document XQuery avec la déclaration `declare variable` suivi par un nom de variable puis le mot-clé `external` suivi par le point-virgule de fin de ligne. Par exemple :


```
declare variable $foo as xs:string external;
```

Le mot-clé `external`, `$foo` devient un paramètre externe, dont la valeur est passée pendant le runtime depuis une source externe. Le paramètre externe reçoit une value avec la commande CLI. Par exemple :

```
--param=foo:'MyName'
```

Dans l'instruction de description ci-dessus, *KEY* est le nom de paramètre externe, *VALUE* est la valeur du paramètre externe, donné en tant qu'une expression XPath. Les noms de paramètres utilisés sur la CLI doivent être déclarés dans le document XQuery. Si plusieurs paramètres externes sont des valeurs passées sur la CLI, chacun doit être recevoir une option séparée `--param`. Les doubles guillemets doivent être utilisés si l'expression XPath contient des espaces.

XSLT

Spécifie paramètre de feuille de style global. *KEY* est le nom de paramètre, *VALUE* est une expression XPath qui fournit la valeur de paramètre. Les noms de paramètre utilisés sur la CLI doivent être déclarés dans la feuille de style. Si plusieurs paramètres multiples sont utilisés, le changement `--param` doit être utilisé avant chaque paramètre. Les double guillemets doivent être utilisés autour de l'expression XPath si elle contient un espace ; que l'espace se trouve dans l'expression XPath elle-même ou dans un littéral de string dans l'expression. Par exemple :

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --param=title:"'string
with spaces'" --p=amount:456 c:\Test.xslt
```

updated-xml

`--updated-xml =` `discard|writeback|asmainresult`

Spécifie comment gérer le fichier XML mis à jour.

- `discard` : la mise à jour est éliminée et pas écrites sur le fichier. Ni le fichier d'entrée, ni le fichier de sortie seront mis à jour. Veuillez noter que ceci est le défaut.
- `writeback` : réécrit la mise à jour de retour dans le fichier XML d'entrée qui est spécifié avec l'option `--input`.
- `asmainresult` : écrit la mise à jour dans le fichier XML de sortie qui est spécifié dans l'option `--output` option. Si l'option **`--output`** n'est pas spécifiée, alors la mise à jour est écrite dans la sortie standard. Dans les deux cas, le fichier XML d'entrée ne sera pas modifié ().

La valeur par défaut est : `discard`.

xpath-static-type-errors-as-warnings

`--xpath-static-type-errors-as-warnings =` `true|false`

Si `true`, passe à une version antérieure d'avertissements de tous les types d'erreurs qui sont détectés dans le contexte statique XPath. Alors qu'une erreur entraînerait l'échec de l'exécution, un avertissement permettrait la poursuite du traitement. Le défaut est `false`.

xquery-update-version

`--xquery-update-version =` `1|1.0|3|3.0`

Spécifie si le processeur XQuery devrait utiliser XQuery Update Facility 1.0 ou XQuery Update Facility 3.0. La valeur par défaut est `3`.

▼ xquery-version

--xquery-version = `1|1.0|3|3.0|3.1`
Spécifie si le processeur XQuery devrait utiliser XSLT XQuery 1.0 ou XQuery 3.0. La valeur par défaut est 3.1.

5.11.7 XSLT

▼ chartext-disable

--chartext-disable = `true|false`
Désactive les extensions de graphique. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ dotnetext-disable

--dotnetext-disable = `true|false`
Désactive les extensions .NET. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ indent-characters

--indent-characters = *VALUE*
Spécifie le string de caractère à utiliser en tant que retrait.

▼ input

--input = *FILE*
L'URL du fichier XML à transformer.

▼ javaext-barcode-location

--javaext-barcode-location = *FILE*
Spécifie le chemin vers le dossier qui contient le fichier d'extension du code-barres `AltovaBarcodeExtension.jar`. Le chemin doit être donné dans une des formes suivantes :

- Un URI de fichier, par exemple : `--javaext-barcode-location="file:///C:/Program Files/Altova/RaptorXMLServer2024/etc/jar/"`
- Un chemin Windows avec un échappement par barres obliques inversées, par exemple : `--javaext-barcode-location="C:\\Program Files\\Altova\\RaptorXMLServer2024\\etc\\jar\\"`

▼ javaext-disable

--javaext-disable = `true|false`
Désactive les extensions Java. La valeur par défaut est `false`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ output, xsltoutput

output = *FILE*, **xsltoutput =** *FILE*
L'URL du fichier sortie-primaire. Par exemple, dans le cas d'une sortie HTML de fichier multiple, le fichier de sortie primaire sera l'emplacement du fichier HTML du point d'entrée. Les fichiers de sortie

supplémentaires, comme des fichiers d'image générée, sont rapportés en tant que `xslt-additional-output-files`. Si aucune option `--output` ou `--xsltoutput` n'est spécifiée, la sortie est écrite sur la sortie standard.

▼ param [p]

`--p | --param =` *KEY:VALUE*

▣ ***XQuery***

Spécifie la valeur d'un paramètre externe. Un paramètre externe est déclaré dans le document XQuery avec la déclaration `declare variable` suivi par un nom de variable puis le mot-clé `external` suivi par le point-virgule de fin de ligne. Par exemple :

```
declare variable $foo as xs:string external;
```

Le mot-clé `external`, `$foo` devient un paramètre externe, dont la valeur est passée pendant le runtime depuis une source externe. Le paramètre externe reçoit une value avec la commande CLI. Par exemple :

```
--param=foo:'MyName'
```

Dans l'instruction de description ci-dessus, *KEY* est le nom de paramètre externe, *VALUE* est la valeur du paramètre externe, donné en tant qu'une expression XPath. Les noms de paramètres utilisés sur la CLI doivent être déclarés dans le document XQuery. Si plusieurs paramètres externes sont des valeurs passées sur la CLI, chacun doit être recevoir une option séparée `--param`. Les doubles guillemets doivent être utilisés si l'expression XPath contient des espaces.

▣ ***XSLT***

Spécifie paramètre de feuille de style global. *KEY* est le nom de paramètre, *VALUE* est une expression XPath qui fournit la valeur de paramètre. Les noms de paramètre utilisés sur la CLI doivent être déclarés dans la feuille de style. Si plusieurs paramètres multiples sont utilisés, le changement `--param` doit être utilisé avant chaque paramètre. Les double guillemets doivent être utilisés autour de l'expression XPath si elle contient un espace ; que l'espace se trouve dans l'expression XPath elle-même ou dans un littéral de string dans l'expression. Par exemple :

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --param=title:"'string
with spaces'" --p=amount:456 c:\Test.xslt
```

▼ streaming

`--streaming =` `true|false`

Permet la validation de streaming. La valeur par défaut est `true`. Dans le mode streaming, les données stockées dans la mémoire sont minimisées et le traitement est plus rapide. L'inconvénient est que l'information que peut être requise par conséquent, par exemple, un modèle de données du document d'instance XML —ne sera pas disponible. Dans des situations où cela est significatif, le mode de streaming devra être éteint (en donnant à `--streaming` une valeur de `false`). En utilisant l'option `--script` avec la commande `valxml-withxsd`, désactiver le streaming. Veuillez noter que l'option `--streaming` est ignorée si`--parallel-assessment` est défini sur `true`.
Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

▼ initial-template, template-entry-point

`--initial-template, --template-entry-point =` *VALUE*

Donne le nom d'un modèle nommé dans la feuille de style XSLT qui est le point d'entrée de la transformation.

- initial-mode, template-mode

  --initial-template, --template-entry-point = *VALUE*
  Spécifie le mode de modèle à utiliser pour la transformation.

- xpath-static-type-errors-as-warnings

  --xpath-static-type-errors-as-warnings = `true|false`
  Si `true`, passe à une version antérieure d'avertissements de tous les types d'erreurs qui sont détectés dans le contexte statique XPath. Alors qu'une erreur entraînerait l'échec de l'exécution, un avertissement permettrait la poursuite du traitement. Le défaut est `false`.

- xslt-version

  --xslt-version = `1|1.0|2|2.0|3|3.0|3.1`
  Spécifie si le processeur XSLT devrait utiliser XSLT 1.0, XSLT 2.0 ou XSLT 3.0. La valeur par défaut est `3`.

5.11.8 JSON/Avro

- additional-schema

  --additional-schema = *FILE*
  Spécifie des URIs d'un document de schéma supplémentaire. Le schéma supplémentaire sera chargé par le schéma principal et peut être référencé depuis le schéma principal par des schémas supplémentaires **id** ou de propriété **$id**.

- disable-format-checks

  --disable-format-checks = `true|false`
  Désactive la validation sémantique imposée par l'attribut de format. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- jsonc

  --jsonc = `true|false`
  Active la prise en charge pour les commentaires dans JSON. La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- json-lines

  --json-lines = `true|false`
  Permet la prise en charge pour JSON Lines (c'est à dire, une valeur JSON par ligne). Valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

5.11.9 Signatures XML

- absolute-reference-uri

  --absolute-reference-uri = `true|false`
  Spécifie si l'URI du document signé doit être lu en tant qu'absolu (`true`) ou relatif (`false`). La valeur par défaut est `false`.
  Note : les valeurs d'option booléennes sont configurées sur `true` si l'option est spécifiée sans une valeur.

- certname, certificate-name

  --certname, --certificate-name = *VALUE*
  Le nom du certificat utilisé pour signer.

  Windows
  Il s'agit du nom du **Sujet** d'un certificat depuis le **--certificate-store** sélectionné.

  xemple de liste des certificats (sous PowerShell)

```
% ls cert://CurrentUser/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

  xemple : **--certificate-name==certificate1**

  Linux/MacOS
  --certname spécifie le nom de fichier d'un certificat X.509v3 à encodage PEM avec la clé privée. Ces fichiers portent généralement l'extension **.pem**.

  xemple : **--certificate-name==/path/to/certificate1.pem**

- certstore, certificate-store

  --certstore, --certificate-store = *VALUE*
  L'emplacement où le certificat spécifié par **--certificate-name** est stocké.

  Windows
  Le nom d'un magasin de certificat sous **cert://CurrentUser**. Les magasins de certificat disponibles peuvent être listés (sous PowerShell) à l'aide de **% ls cert://CurrentUser/**. Les certificats seraient donc listés comme suit :

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
```

```
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

xemple : **--certificate-store==MyCertStore**

Linux/MacOS
L'option **--certstore** n'est actuellement pas prise en charge.

▼ digest, digest-method

--digest, --digest-method = sha1|sha256|sha384|sha512
L'algorithme utilisé pour calculer la valeur de digest par rapport au fichier XML d'entrée.

▼ hmackey, hmac-secret-key

--hmackey, --hmac-secret-key = *VALUE*
La clé secrète partagée HMAC ; doit avoir une longueur minimum de six caractères.

Exemple : **--hmackey=secretpassword**

▼ hmaclen, hmac-output-length

--hmaclen, --hmac-output-length = *LENGTH*
Tronque la sortie de l'algorithme HMAC à des bits LENGTH. Si spécifiée, cette valeur doit être
- un multiple de 8
- supérieur à 80
- supérieur à la moitié de la longueur de sortie de l'algorithme hash sous-jacent

▼ keyinfo, append-keyinfo

--keyinfo, --append-keyinfo = true|false
Spécifie si vous souhaitez inclure l'élément **KeyInfo** dans la signature ou pas. La valeur par défaut est **false**.

▼ sigc14nmeth, signature-canonicalization-method

--sigc14nmeth, --signature-canonicalization-method = *VALUE*
Spécifie l'algorithme de canonicalisation à appliquer à l'élément **SignedInfo**. La valeur doit être une des suivantes :

- REC-xml-c14n-20010315
- xml-c14n11
- xml-exc-c14n#

▼ sigmeth, signature-method

--sigmeth, --signature-method = *VALUE*

Spécifie l'algorithme à utiliser pour générer la signature.

Lorsqu'un certificat est utilisé
Si un certificat est spécifié, alors`--signature-method` est optionnel et la valeur pour ce paramètre est dérivé du certificat. Lorsqu'il est spécifié, il doit correspondre à l'algorithme utilisé par le certificat.

xemple : **`--signature-method=rsa-sha256`**

Lorsque --hmac-secret-key est utilisé
Lorsque `--hmac-secret-key` est utilisé, cette option est obligatoire. La valeur doit être un des algorithmes HMAC pris en charge :

- `hmac-sha256`
- `hmac-sha386`
- `hmac-sha512`
- hmac-sha1 (découragé par la spécification)

xemple : **`--signature-method=hmac-sha256`**

▼ sigtype, signature-type

`--sigtype, --signature-type =` `detached | enveloping | enveloped`
Spécifie le type de signature à générer.

▼ transforms

`--transforms =` *`VALUE`*
Spécifie les transformations de Signature XML appliquées au document d'entrée. Cette option peut être spécifiée plusieurs fois. Dans ce cas, l'ordre de la spécification est important. La première transformation spécifiée reçoit le document d'entrée. La dernière transformation spécifiée est utilisée immédiatement avant le calcul de la valeur digest.

Les valeurs prises en charge sont :

- `REC-xml-c14n-20010315` pour Canonical XML 1.0 (sans commentaires)
- `xml-c14n11` pour Canonical XML 1.1 (sans comments)
- `xml-exc-c14n#` pour Exclusive XML Canonicalization 1.0 (sans commentaires)
- `REC-xml-c14n-20010315#WithComments` pour Canonical XML 1.0 (avec commentaires)
- `xml-c14n11#WithComments` pour Canonical XML 1.1 (avec commentaires)
- `xml-exc-c14n#WithComments` pour Exclusive XML Canonicalization 1.0 (avec commentaires)
- `base64`
- `strip-whitespaces` extension Altova

xemple : **`--transforms=xml-c14n11`**

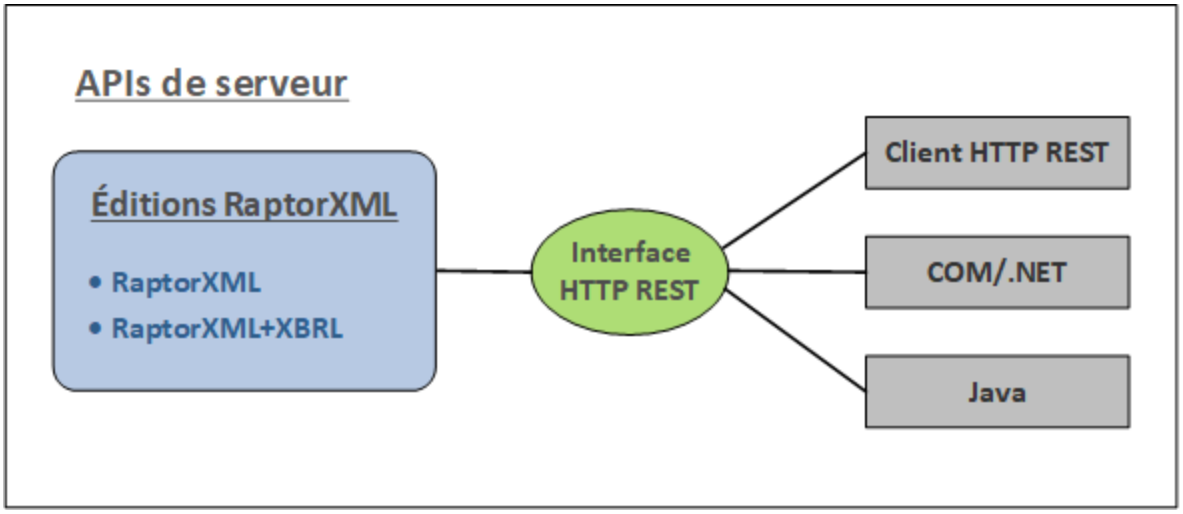
Note: Cette option peut être spécifiée plusieurs fois. Si elle est spécifiée plusieurs fois, l'ordre de spécification est significatif. La première transformation spécifiée reçoit le document d'entrée. La dernière transformation spécifiée est utilisée juste avant le calcul de la valeur de digest.

▼ write-default-attributes

--write-default-attributes = `true|false`
Spécifie s'il faut inclure les valeur d'attribut par défaut depuis le DTD dans le document signé.

6 API serveur : REST HTTP, COM/.NET, Java

RaptorXML Server définit une interface REST HTTP, qui est utilisée par des clients pour répartir les tâches de validation sur le serveur. Les clients peuvent soit accéder directement à l'interface REST HTTP soit utiliser des API de haut niveau COM/.NET et de serveur Java. Ces API fournissent des classes COM/.NET et Java d'utilisation simple qui gèrent la création et la répartition des requêtes REST HTTP. La figure ci-dessous montre un sommaire des méthodes client REST HTTP disponibles pour communiquer avec le serveur RaptorXML.



Il existe trois API de serveur qui peuvent être utilisées pour communiquer avec RaptorXML par le biais de l'interface REST HTTP (*voir aussi la figure ci-dessus*).

- Interface client HTTP REST 227
- API COM/.NET 268
- API Java 274

Note : les API de serveur proposent des fonctions semblables dans l'interface de ligne de commande (CLI) 57 . Elle permet également d'effectuer une validation et des transformations de document. Si vous souhaitez utiliser des fonctions avancées, comme la lecture, l'extraction et l'analyse des données, utiliser les API de moteur. Les API de moteur peuvent fournir des informations supplémentaires comme le décompte des éléments, leurs positions dans le document, et un accès et une manipulation des données XBRL complexes.

Utilisation

RaptorXML Server devrait être installé sur un appareil accessible par les clients sur le réseau local. Une fois que le service RaptorXML Server a été démarré, les clients peuvent se connecter au serveur et émettre des commandes. Les méthodes d'accès suivantes sont libellées en tant qu'API de serveur car elles fournissent une manière de communiquer avec le serveur RaptorXML À distance.

- *HTTP REST client interface* 227: Les requêtes client sont effectuées en format JSON comme décrit dans la section Interface Client REST HTTP 227. Un répertoire de tâche est attribué à chaque requête sur le serveur, dans lequel des fichiers de sortie sont enregistrés. Le serveur répond au client avec toutes les informations pertinentes à la tâche.

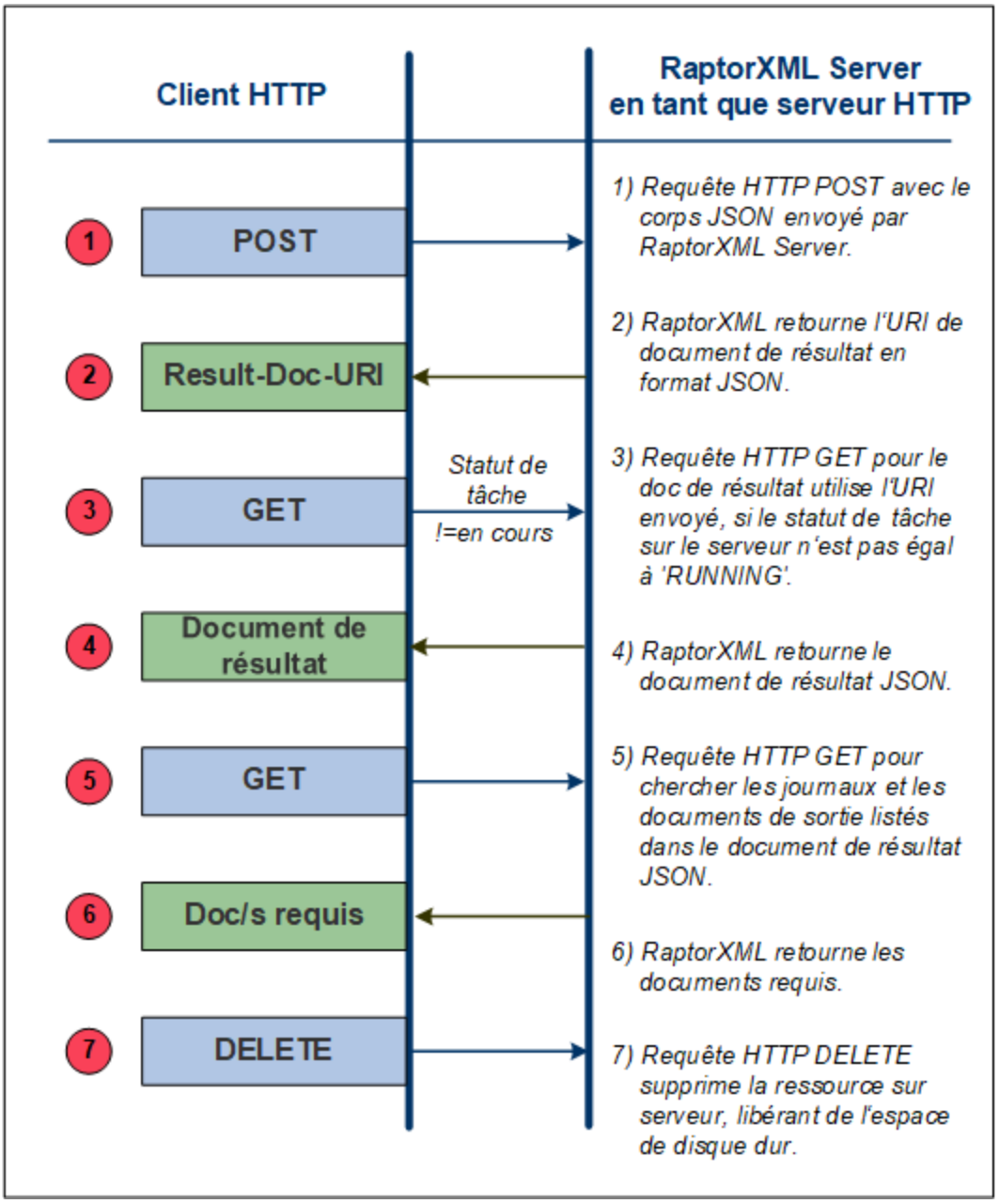
- *COM/.NET API*[268] *et Java API*[274] : Les applications et scripts dans langages de programmation COM/.NET[268] et les applications Java[274] utilisent des objets de API de serveur RaptorXML[277] pour accéder à la fonctionnalité de RaptorXML Server. L'API de Serveur RaptorXML[277] émettra les requêtes REST HTTP correspondantes pour le client. Voir les sous-sections respectives pour plus d'informations.

Licence

RaptorXML Server est mis sous licence sur l'appareil sur lequel il est installé. Les connexions à RaptorXML Server sont établies par le biais de HTTP.

6.1 Interface Client REST HTTP

RaptorXML Server accepte les tâches soumises par HTTP (ou HTTPS 236). La description de la tâche ainsi que les résultats sont échangés dans le format JSON. Le flux de travail de base est tel que montré dans le diagramme ci-dessous.



Problèmes de sécurité liés à l'interface REST HTTP

L'interface REST HTTP, par défaut, permet aux documents de résultat d'être rédigés sur tout emplacement spécifié par le client (qui est accessible avec le protocole HTTP). C'est pourquoi il est important de considérer ces aspects de sécurité lors de la configuration de RaptorXML Server.

Si vous estimez que la sécurité ait pu être compromise ou que l'interface ait pu être utilisée à mauvais escient, le serveur peut être configuré pour écrire les documents de résultat dans un répertoire de sortie réservé sur le serveur lui-même. Cela est spécifié en configurant l'option `server.unrestricted-filesystem-access`[233] du fichier de configuration du serveur sur `false`. Lorsque l'accès est limité de cette manière, le client peut télécharger les documents de résultat depuis le répertoire de sortie réservé avec des requêtes `GET`. En alternative, un administrateur peut copier/charger les fichiers de document de résultat depuis le serveur vers l'emplacement cible.

Dans cette section

Avant d'envoyer une requête client, RaptorXML Server doit être lancé et configuré correctement. La procédure est décrite dans la section Configuration de serveur[228]. La section Requêtes Client[240] décrit comment envoyer des requêtes client. Enfin, la section C# Example pour REST API[264] fournit une description de l'exemple de fichier API REST qui est installé avec votre package RaptorXML Server.

6.1.1 Configuration de serveur

RaptorXML est mis sous licence sur l'appareil sur lequel il est installé. Cette installation peut ensuite être accédée par le biais d'une Interface REST HTTP[227]. Pour configurer correctement RaptorXML Server, procéder comme suit. Nous assumons que RaptorXML Server a déjà été installé[46] et mis sous licence[46] correctement.

1. RaptorXML Server doit soit être lancé en tant qu'un service ou une application[229] pour pouvoir être accédé correctement par le biais de HTTP ou HTTPS. La procédure diffère selon le système d'exploitation et est décrite ici : sur Windows[229], sur Linux[229], sur macOS[230].
2. Utiliser la configuration de serveur initiale[231] pour tester la connexion au serveur[230]. (La configuration de serveur initiale[231] est la configuration par défaut que vous obtenez lors de l'installation.) Vous pouvez utiliser une requête HTTP simple`GET` comme `http://localhost:8087/v1/version` pour tester la connexion. (La requête peut aussi être saisie dans la barre d'adresse d'une fenêtre de navigateur.) Si le service est en cours d'exécution vous devez obtenir une réponse à une requête de test HTTP comme la requête de version ci-dessus.
3. Consultez le fichier de configuration de serveur[231], `server_config.xml`. Si vous souhaitez changer des paramètres[233] dans le fichier, éditer le fichier de configuration de serveur et enregistrer les changements. HTTPS est désactivé par défaut et devront être activés dans le fichier de configuration[231].
4. Si vous avez édité le fichier de configuration de serveur[231], redémarrer RaptorXML Server en tant que service de manière à ce que les nouveaux paramètres de configuration sont appliqués. Tester à nouveau de la connexion pour vous assurer que le service est en cours et accessible.

Note : Les erreurs de démarrage de serveur, le fichier de configuration de serveur qui est utilisé, et les erreurs de licence sont rapportés dans le journal de système. Veuillez donc vous référer au journal de système[233] s'il y a des problèmes avec le serveur.

Pour plus d'informations concernant HTTPS, voir la section Paramètres HTTPS[236].

6.1.1.1 Démarrer le serveur

Cette section :

- Emplacement de l'exécutable de serveur[229]
- Démarrer RaptorXML en tant que service sur Windows[229]
- Démarrer RaptorXML en tant que service sur Linux[229]
- Démarrer RaptorXML en tant que service sur macOS[230]

Emplacement du fichier exécutable du serveur

L'exécutable RaptorXML Server est installé par défaut dans le dossier :

```
<ProgramFilesFolder>\Altova\RaptorXMLServer2024\bin\RaptorXML.exe
```

L'exécutable peut être utilisé pour lancer RaptorXML Server en tant que service.

Démarrage en tant que service sur Windows

Le processus d'installation aura enregistré RaptorXML Server en tant qu'un service sur Windows. Vous devez, néanmoins, **lancer** RaptorXML Server en tant qu'un service. Vous pouvez procéder des manières suivantes :

- Par le biais d'Altova ServiceController, que vous trouverez sous la forme d'une icône dans la zone de notification. Si l'icône n'est pas disponible, vous pouvez lancer Altova ServiceController et ajouter son icône à la zone de notification en vous rendant dans le menu **Start**, en sélectionnant **All Programs | Altova | Altova LicenseServer | Altova ServiceController**.
- Par le biais du Windows Services Management Console: **Control Panel | All Control Panel Items | Administrative Tools | Services**.
- Par le biais de l'invite de commandes que vous pouvez lancer si vous disposez des droits d'administrateur. Utilisez la commande suivante sous n'importe quel répertoire : **`net start "AltovaRaptorXMLServer"`**
- Par le biais de l'exécutable RaptorXML Server dans une fenêtre d''invite de commandes : **`RaptorXMLServer.exe debug`**. Cela démarre le serveur, les informations d'activité du serveur allant directement dans la fenêtre d'invite de commandes. L'affichage des informations d'activité du serveur peut être allumé et éteint avec le paramètre `http.log-screen`[233] du fichier de configuration du serveur[231]. Pour arrêter le serveur, appuyer sur **Ctrl+Break** (ou **Ctrl+Pause**). Lorsque le serveur est lancé de cette manière, au lieu de le lancer en tant qu'un service tel que décrit dans les trois premières étapes précédentes ; le serveur s'arrêtera lorsque la console de ligne de commande est fermée ou lorsque l'utilisateur se déconnecte.

Démarrage en tant que service sur Linux

Démarrer RaptorXML Server en tant qu'un service avec la commande suivante :

[< Debian 8]	sudo /etc/init.d/raptorxmlserver start
[≥ Debian 8]	sudo systemctl start raptorxmlserver
[< CentOS 7]	sudo initctl start raptorxmlserver
[≥ CentOS 7]	sudo systemctl start raptorxmlserver
[< Ubuntu 15]	sudo initctl start raptorxmlserver
[≥ Ubuntu 15]	sudo systemctl start raptorxmlserver
[RedHat]	sudo initctl start raptorxmlserver

Si, à tout moment, vous devez arrêter RaptorXML Server, utiliser :

[< Debian 8]	sudo /etc/init.d/raptorxmlserver stop
[≥ Debian 8]	sudo systemctl stop raptorxmlserver
[< CentOS 7]	sudo initctl stop raptorxmlserver
[≥ CentOS 7]	sudo systemctl stop raptorxmlserver
[< Ubuntu 15]	sudo initctl stop raptorxmlserver
[≥ Ubuntu 15]	sudo systemctl stop raptorxmlserver
[RedHat]	sudo initctl stop raptorxmlserver

Démarrage en tant que service sur macOS

Démarrer RaptorXML Server en tant qu'un service avec la commande suivante :

```
sudo launchctl load /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

Si, à tout moment, vous devez arrêter RaptorXML Server, utiliser :

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.RaptorXMLServer2024.plist
```

6.1.1.2 Tester la connexion

Cette section :

Requête GET pour tester la connexion

Une fois que RaptorXML Server a été lancé, tester la connexion en utilisant une requête `GET`. (Vous pouvez aussi saisir cette requête dans la barre d'adresse d'une fenêtre de navigateur.)

```
http://localhost:8087/v1/version
```

Note : L'interface et le numéro port de RaptorXML Server est spécifié dans le fichier de configuration de serveur, `server_config.xml`, qui est décrit dans la section suivante, Configuration de serveur 231.

Réponse de serveur et liste de structure de données JSON

Lorsque le service est exécuté et le serveur est correctement configuré, la requête ne devrait jamais échouer. RaptorXML Server retournera son information de version en tant qu'une structure de donnée JSON (*liste ci-dessous*).

```
{
    "copyright": "Copyright (c) 1998-2013 Altova GmbH. ...",
    "name": "Altova RaptorXML+XBRL Server 2013 rel. 2 sp1",
    "eula": "http://www.altova.com/server_software_license_agreement.html"
}
```

Note : Si vous modifiez la configuration de serveur—, lorsque vous éditez le fichier de configuration de serveur 231—, vous devriez tester à nouveau la connexion.

6.1.1.3 Configurer le serveur

Cette section :

Fichier de configuration de serveur : paramètres de base

RaptorXML Server est configuré au moyen d'un fichier de configuration appelé `server_config.xml`, qui est situé par défaut sous :

```
C:\Program Files (x86)\Altova\RaptorXMLServer2024\etc\server_config.xml
```

La configuration initiale pour RaptorXML Server définit les éléments suivants :

- Un numéro de port de `8087` en tant que le port de serveur.
- Que le serveur écoute uniquement les connexions locales (`localhost`).
- Que le serveur écrit la sortie sous `C:\ProgramData\Altova\RaptorXMLServer2024\Output\`.

Les autres paramètres par défaut sont affichés dans la liste 232 de `server_config.xml` ci-dessous.

Fichier de configuration de serveur : modifier les paramètres de base, retourner aux paramètres de base

Si vous souhaitez changer les paramètres de base, vous devez éditer le fichier de configuration de serveur, `server_config.xml` (voir la liste ci-dessous 232), l'enregistrer, puis redémarrer RaptorXML Server en tant que service.

Si vous souhaitez recréer le fichier de configuration de serveur original (de manière à ce que le serveur soit à nouveau configuré avec les paramètres de base), exécuter la commande `createconfig`:

```
RaptorXMLServer.exe createconfig
```

Lors de l'exécution de cette commande, le fichier de paramètres de base sera recréé et écrasera le fichier `server_config.xml`. La commande `createconfig` est utile si vous souhaitez réinitialiser la configuration de serveur pour les paramètres de base.

Fichier de configuration de serveur : liste et paramètres

Le fichier de configuration de serveur, **`server_config.xml`**, est listé ci-dessous avec les paramètres de base. Les paramètres disponibles s'y trouvant sont expliqués en-dessous de la liste.

`server_config.xml`

```
<config xmlns="http://www.altova.com/schemas/altova/raptorxml/config"
    xsi:schemaLocation="http://www.altova.com/schemas/altova/raptorxml/config
    http://www.altova.com/schemas/altova/raptorxml/config.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <language>en</language>
  <server.unrestricted-filesystem-access>true</server.unrestricted-filesystem-access>
  <server.output-root-dir>C:\ProgramData\Altova\RaptorXMLServer2024\output\</server.output-
root-dir>
  <server.script-root-dir>C:\Program
Files\Altova\RaptorXMLServer2024\etc\scripts\</server.script-root-dir>
  <!--<server.default-script-api-version>2</server.default-script-api-version>-->
    <!--<server.catalog-file>catalog.xml</server.catalog-file>-->
  <!--<server.log-file>C:
\ProgramData\Altova\RaptorXMLServer2024\Log\server.log</server.log-file>-->

  <http.enable>true</http.enable>
    <http.environment>production</http.environment>
    <http.socket-host>127.0.0.1</http.socket-host>
    <http.socket-port>8087</http.socket-port>
  <http.log-screen>true</http.log-screen>
    <http.access-file>C:\ProgramData\Altova\RaptorXMLServer2024\Log\access.log</http.access-
    file>
    <http.error-file>C:\ProgramData\Altova\RaptorXMLServer2024\Log\error.log</http.error-
    file>

  <https.enable>false</https.enable>
  <https.socket-host>127.0.0.1</https.socket-host>
```



```
  <https.socket-port>443</https.socket-port>
  <https.private-key>C:\Program
Files\Altova\RaptorXMLServer2024\etc\cert\key.pem</https.private-key>
  <https.certificate>C:\Program
Files\Altova\RaptorXMLServer2024\etc\cert\cert.pem</https.certificate>
  <!--<https.certificate-chain>/path/to/chain.pem</https.certificate-chain>-->

</config>
```

Paramètres

language
Configure la langue des messages de serveur, dans un élément `language` optionnel. La valeur par défaut est `en` (anglais). Les valeurs autorisées sont `en|de|es|fr|ja` (anglais, allemand, espagnol, français et japonais, respectivement). Voir les Commandes de localisation 185 pour une description de la localisation de RaptorXML.

server.unrestricted-filesystem-access

- Une fois définis sur `true` (la valeur par défaut), les fichiers de sortie seront écrits directement dans l'emplacement spécifié par l'utilisateur et dans les scripts Python (écrasant éventuellement les fichiers existants du même nom). Veuillez noter, néanmoins, que les chemins de fichiers locaux ne peuvent pas être utilisés pour accéder aux fichiers depuis une machine à distance par HTTP. Donc, si RaptorXML Server est exécuté sur une machine à distance, régler la valeur de cette option sur **false**. Le réglage de la valeur sur **true** est uniquement viable si le client et le serveur se trouvent sur la même machine et que vous souhaitez écrire les fichiers de sortie sur un répertoire dans cette machine.
- Si définis sur `false`, les fichiers seront écrits dans le répertoire de la tâche dans le répertoire de sortie 233, et les URI de ces fichiers seront inclus dans le document de résultat 258. La configuration de la valeur sur `false` fournit un niveau de sécurité, puisque les fichiers peuvent uniquement être écrits sur le disque dans un répertoire de tâche réservé et connu sur le serveur. Les fichiers de sortie de tâche peuvent ensuite être copiés par des moyens autorisés dans d'autres emplacements.

server.output-root-dir
Le répertoire dans lequel la sortie de toutes les tâches soumises est enregistrée.

server.script-root-dir
Le répertoire dans lequel scripts Python 346 de confiance doivent être enregistrés. L'option `script`, lorsqu'utilisée par le biais de l'interface HTTP, fonctionnera uniquement si des scripts provenant de ce répertoire de confiance sont utilisés. La spécification d'un script Python provenant d'un autre répertoire entraînera un erreur. *Voir Rendre les Scripts Python sûrs' 346*.

server.default-script-api-version
La version API Python par défaut utilisés pour exécuter les scripts Python. Par défaut, la version la plus récente de l'API Python est utilisée. Les valeurs prises en charge actuellement sont 1 et 2.

server.catalog-file

L'URL du fichier catalogue XML à utiliser. Par défaut, le fichier catalogue `RootCatalog.xml`, qui est situé dans le dossier `<ProgramFilesFolder>\Altova\RaptorXMLServer2024\etc`, sera utilisé. Utiliser le paramètre `server.catalog-file` uniquement si vous souhaitez modifier le fichier de catalogue par défaut.

`server.log-file`
Le nom et l'emplacement du fichier de log de serveur. Les événements sur le serveur, comme *Server started/stopped*, sont connectez en continu dans le journal de l'événement du système et affiché dans un visionneur de l'événement de système comme par exemple Windows Event Viewer. Outre l'affichage du visionneur, les messages de journal peuvent aussi être écrits sur le fichier spécifié avec l'option `server.log-file`. Le fichier de journal de serveur contiendra des informations concernant des activités sur le serveur, y compris les erreurs de lancement de serveur, le fichier de configuration utilisé, et les erreurs de licence.

`http.enable`
Une valeur booléenne pour activer ou désactiver HTTP: `true | false`. HTTP peut être activé/désactivé indépendamment ou HTTPS, et les deux peuvent être actifs en parallèle.

`http.environment`
Les environnements interne de raptorxml: `production | development`. L'environnement de Développement sera plus adapté aux besoin des développeurs, permettant un débogage plus simple que lorsque l'environnement de Production est utilisé.

`http.socket-host`
L'interface par le biais duquel RaptorXML Server est accédé. Si vous souhaitez que RaptorXML Server accepte les connexions depuis les machines à distance, décommenter l'élément et définir son contenu sur : `0.0.0.0`, comme cela : `<https.socket-host>0.0.0.0</https.socket-host>`. Cela héberge le service sur toute interface adressable sur l'appareil de serveur. Dans ce cas, s'assurer que les paramètres du pare-feu sont configurés correctement. Les exceptions de pare-feu entrantes pour les produits Altova doivent être enregistrées comme suit : Altova LicenseServer : port `8088`; Altova RaptorXML Server: port `8087`; Altova FlowForce Server : port `8082`.

`http.socket-port`
Le port sur lequel le service est accédé. Le port doit être fixe et connu de manière à ce que les requêtes HTTP puissent être adressées correctement au service.

`http.log-screen`
Si RaptorXML Server est lancé avec la commande `RaptorXMLServer.exe debug`, (voir Lancer le serveur 229) et si `http.log-screen` est configuré sur `true`, alors l'activité du serveur est affichée dans la console de la ligne de commande. Sinon, l'activité du serveur n'est pas affichée. L'écran journal est affiché en plus de l'écriture dans les fichiers journaux.

`http.access-file`
Le nom et l'emplacement du fichier d'accès HTTP. Le fichier d'accès contient des informations concernant l'activité liée aux accès. Il contient des informations utiles pour résoudre les problèmes de connexion.

http.error-file
Le nom et l'emplacement du fichier d'erreur HTTP. Le fichier d'erreur contient des erreurs liées au trafic provenant de et allant vers le serveur. En cas de problèmes de connexion, ce fichier peut fournir des informations utiles concernant la résolution d'erreur.

http.max_request_body_size
Cette option spécifie la taille maximum du corps de requête, en octets, que RaptorXML Server accepte. La valeur par défaut est de 100MO. Si la taille d'un corps de requête est plus grande que la valeur spécifiée pour cette option, alors le serveur répond avec `HTTP Error 413: Request entity too large`. La valeur de l'option doit être supérieure à ou égale à zéro. La limite peut être désactivée avec `http.max_request_body_size=0`.

https.enable
Une valeur booléenne pour activer ou désactiver HTTPS: `true | false`. HTTPS peut être activé/désactivé indépendamment de HTTP, et tous les deux peuvent être actives en parallèle. La prise en charge HTTPS est désactivée par défaut et doit être activée en changeant la valeur de ce paramètre sur **`true`**.

https.socket-host
Prend une valeur string qui est l'adresse hôte sur laquelle les connexions HTTP sont acceptées. Pour accepter les connexions depuis l'hôte local uniquement, définir **`localhost`** ou **`127.0.0.1`**. Si vous souhaitez que RaptorXML Server accepte les connexions provenant de toutes les machines à distance, définir la valeur sur : `0.0.0.0`, comme ceci : `<http.socket-host>0.0.0.0</http.socket-host>`. Cela héberge le service sur chaque interface adressable de la machine du serveur. Dans ce cas, veuillez vous assurer que les paramètres de pare-feu sont correctement configurés. Les exceptions pare-feu entrantes pour les produits Altova doivent être enregistrés tels que suit : Altova LicenseServer : port `8088`; Altova RaptorXML Server : port `8087`; Altova FlowForce Server : port `8082`. Vous pouvez aussi utiliser des adresses IPv6 comme : `'::'`.

https.socket-port
Une valeur de nombre entier qui est le port sur lequel HTTPS est accepté. Le port doit être fixé et connu de manière à ce que les requêtes HTTP peuvent être adressées correctement au service.

https.private-key, https.certificate
Les URI qui sont les chemins, respectivement, à la clé privée du serveur et les fichiers de certificat. Les deux sont nécessaires. Voir Paramètres HTTPS 236 et Configurer l'encodage SSL 237 pour plus d'informations. Sur les appareils Windows, vous pouvez aussi utiliser des chemins Windows.

https.certificate-chain
Un paramètre optionnel. il s'agit d'un URI qui situe le fichier de certificat intermédiaire. Si vous avec deux certificats intermédiaires (primaire et secondaire), puis de les combiner en un fichier tel que décrit dans l'étape 7 de Configurer l'encodage SSL 237. Voir Paramètres HTTPS 236 et Configurer l'encodage SSL 237 pour plus d'informations.

L'adresse **RaptorXML Server**
L'adresse HTTP du serveur consiste en : socket-host et socket-port :

```
http://{socket-host}:{socket-port}/
```

L'adresse telle que configurée avec la configuration de base sera :

```
http://localhost:8087/
```

Pour changer l'adresse, modifier les paramètres `http.socket-host` et `http.socket-port` dans le fichier de configuration de serveur, `server_config.xml`. Si par exemple, le serveur a une adresse IP de `123.12.123.1`, et que les paramètres de configuration de serveur suivants ont été fait :

```
<http.socket-host>0.0.0.0</http.socket-host>
<http.socket-port>8087</http.socket-port>
```

RaptorXML Server peut ensuite être adressé avec :

```
http://123.12.123.1:8087/
```

Note : Une fois que `server_config.xml` aura été modifié, RaptorXML Server doit être redémarré pour que les nouvelles valeurs soient appliquées.

Note : Si vous être confronté à des problèmes de connexion à RaptorXML Server, l'information dans les fichiers nommé dans `http.access-file` et `http.error-file` peut aider à résoudre des problèmes.

Note : Les messages soumis à RaptorXML Server doivent contenir les noms de chemin qui sont valides sur le serveur. Les documents sur le serveur peuvent être accédés soit localement soit à distance (dans le deuxième cas avec des URI HTTP, par exemple).

6.1.1.4 Paramètres HTTPS

RaptorXML Server prend en charge le démarrage non seulement en tant qu'un serveur HTTP, mais aussi en tant qu'un serveur HTTPS. Les deux types de connexion peuvent être active parallèlement.

Activer HTTPS

La prise en charge HTTPS est désactivée par défaut. Pour activer HTTPS, dans le fichier de configuration de serveur[231], `server_config.xml`, mettre le paramètre **`https.enable`** sur **`true`**. Modifier les différents paramètres HTTPS du fichier de configuration[231] conformément à vos exigences de serveur.

Clé privée et certificat

Vous pouvez obtenir une clé privée et des fichiers de certificat d'une des manières suivantes :

- Depuis une autorité de certificat : suivez les étapes décrites dans la section Configurer le chiffrage Up SSL[237].
- Créer un certificat auto-signé en utilisant la commande OpenSSL suivante (modifié en conséquence pour votre environnement) :


```
openssl req -x509 -newkey rsa:4096 -nodes -keyout key.pem -out cert.pem -days 365 -
subj "/C=AT/ST=vienna/L=vienna/O=Altova Gmbh/OU=dev/CN=www.altova.com"
```

Tester la connexion

Un bon moyen de tester votre connexion s'effectue par le biais de l'outil de ligne de commande curl pour transmettre les données avec des URL. Vous pouvez utiliser la commande suivante :

```
curl.exe https://localhost:443/v1/version
```

Si le certificat n'est pas approuvé, utiliser l'option `-k`, comme ceci :

```
curl.exe -k https://localhost:443/v1/version
```

La commande suivante exécute l'exemple Python HTTP qui est distribué avec RaptorXML Server:

```
python3.exe examples\ServerAPI\python\RunRaptorXML.py --host localhost -p 443 -s
```

6.1.1.5 Configurer le cryptage SSL

Si vous souhaitez encoder vos transferts de données RaptorXML Server à l'aide du protocole SSL, vous devez :

- Générer une clé privée SSL et créer un fichier de certificat de clé publique SSL
- Configurer RaptorXML Server pour la communication SSL.

Vous trouverez ci-dessous les étapes à suivre.

Cette méthode utilise le kit outil OpenSSL open-source pour gérer l'encodage SSL. Les étapes listées ci-dessous, doivent donc être effectuées sur un ordinateur sur lequel OpenSSL est disponible. Généralement, OpenSSL est pré-installé sur la plupart des distributions Linux et sur des appareils macOS. Il peut donc être installé sur les ordinateurs Windows. Pour télécharger des liens pour les binaires d'installation, voir le Wiki OpenSSL.

Pour générer une clé privée et obtenir un certificat auprès d'une autorité de certification, procéder comme suit :

1. Générer une clé privée

   SSL nécessite l'installation d'une **clé privée** sur RaptorXML Server. Cette clé privée sera utilisée pour encoder toutes les données RaptorXML Server. Pour créer la clé privée, utiliser la commande OpenSSL suivante :

   ```
   openssl genrsa -out private.key 2048
   ```

   Un fichier nommé `private.key` est créé, qui contient votre clé privée. Veuillez noter où vous avez enregistré le fichier. Il vous faudra la clé privée pour (i) générer la Certificate Signing Request (CSR), et (ii) l'installer sur RaptorXML Server.

2. Requête de signature de certificat (CSRs)

Une Requête de signature de certificat (Certificate Signing Request) (CSR) est envoyée à l'Autorité de certification (CA), comme VeriSign ou Thawte, pour demander un certificat de clé public. La CSR est basée sur votre clé privée et contient des informations concernant votre organisation. Créer une CSR avec la commande OpenSSL suivante (qui fournit le fichier privé-clé, `private.key`, qui a été créé dans l'étape 1, en tant que l'un de ces paramètres) :

```
openssl req -new -nodes -key private.key -out my.csr
```

Au cours de la génération de la CSR vous devrez donner des informations concernant votre organisation, comme celle que vous trouverez ci-dessous. Cette information sera utilisée par l'autorité de certification pour vérifier l'identité de votre entreprise.

- *Pays*
- *Lieu* (la ville où votre commerce se situe)
- *Organisation* (le nom de votre entreprise). Ne pas utiliser des caractères spéciaux ; ils invalideront votre certificat
- *Nom commun* (le nom DNS de votre serveur). Cela doit correspondre exactement au nom officiel de votre serveur, c'est à dire, le nom DNS que les applis client utiliseront pour se connecter au serveur
- *Un mot de passe.* Garder cette entrée vierge !

3. Acheter un certificat SSL

Acheter un certificat SSL auprès d'une autorité de certificat (CA) reconnu, comme VeriSign ou Thawte. Pour le reste de ces instructions, nous suivons la procédure VeriSign. La procédure avec d'autres CA est semblable.

- Se rendre sur le site web VeriSign.
- Cliquer sur **Buy SSL Certificates**.
- Plusieurs types de certificats SSL sont disponibles. En ce qui concerne RaptorXML Server, les certificats Secure Site ou Secure Site Pro sont suffisants. Une EV (vérification étendue) n'est pas nécessaire, puisque les utilisateurs ne verront pas de "barre d'adresse verte".
- Suivez les étapes d'inscription et saisir les informations nécessaires pour passer votre commande.
- Lorsque vous êtes invité à indiquer la CSR (*créée dans l'étape 2*), copier et coller le contenu du fichier `my.csr` dans le formulaire de commande.
- Payer le certificat avec votre carte de crédit.

Permettre du temps pour obtenir un certificat

L'obtention des certificats de clé publiques auprès de l'autorité de certificat (CA) SSL prend généralement **deux ou trois jours ouvrés**. Veuillez y penser lorsque vous configurez votre RaptorXML Server.

4. Recevoir clé publique de la CA

Votre autorité de certificat achèvera le processus d'inscription en deux ou trois jours ouvrés. Pendant ce temps, vous pourrez recevoir des e-mails ou des appels téléphoniques pour vérifier si vous êtes autorisé à demander un certificat SSL pour votre domaine DNS. Veuillez travailler avec l'autorité pour achever le processus.

Une fois que le processus d'autorisation et d'inscription est achevé, vous obtiendrez un e-mail contenant la **clé publique** de votre certificat SSL. La clé publique sera en texte clair ou attaché en tant que fichier **.cer**.

5. Enregistrer la clé publique dans le fichier

   Pour pouvoir être utilisé avec RaptorXML Server, la clé publique doit être enregistrée dans un fichier **.cer**. Si la clé publique était fournie en tant que texte, copier-coller toutes les lignes depuis

   ```
   --BEGIN CERTIFICATE--
     ...
   --END CERTIFICATE--
   ```

   dans un fichier de texte que nous appellerons **mycertificate.cer**.

6. Enregistrer les certificats intermédiaires de la CA sous le fichier

   Pour terminer votre certificat SSL, il vous faut deux certificats supplémentaires : le **certificat intermédiaire primaire** et **secondaire**. Votre autorité de certificat (CA) établira la liste du contenu des certificats intermédiaires sur son site web.

   - Les certificats intermédiaires de Verisign : https://knowledge.verisign.com/support/ssl-certificates-support/index?page=content&id=AR657&actp=LIST&viewlocale=en_US
   - Les certificats intermédiaires de Verisign pour son produit Secure Site : https://knowledge.verisign.com/support/ssl-certificates-support/index?page=content&id=AR1735

   Copier-coller les deux certificats intermédiaires (primaire et secondaire) dans des fichiers de texte séparés et les enregistrer sur votre ordinateur.

7. En option, combiner des certificats dans un fichier de certificat de clé publique

   Vous disposez maintenant de trois fichiers de certificat :

   - Clé publique (`mycertificate.cer`)
   - Certificat intermédiaire secondaire
   - Certificat intermédiaire primaire

   Vous pouvez intégrer vos certificats intermédiaires dans votre certificat de clé publique si vous le souhaitez. Vous trouverez ci-dessous la description de la procédure à suivre. (En alternative, vous pouvez utiliser les paramètres du fichier de configuration[233] **https.certificate-chain** pour spécifier l'emplacement des certificats intermédiaires.)

   Chacun contient des blocs de texte contenus dans des tirets :

   ```
   --BEGIN CERTIFICATE--
     ...
   --END CERTIFICATE--
   ```

À présent, copier-coller les trois certificats dans un fichier de manière à ce qu'ils se trouvent dans le bon ordre : l'ordre de la séquence est important : (i) clé publique, (ii) certificat intermédiaire secondaire, (iii) certificat intermédiaire primaire. Veuillez vous assurer qu'il n'y a pas de lignes entre les certificats.

```
--BEGIN CERTIFICATE--
  clé publique  provenant de mycertificate.cer (voir étape 5)
--END CERTIFICATE--
--BEGIN CERTIFICATE--
  certificat intermédiaire secondaire (voir étape 6)
--END CERTIFICATE--
--BEGIN CERTIFICATE--
  certificat intermédiaire primaire (voir étape 6)
--END CERTIFICATE--
```

Enregistrer le texte de certificat combiné en résultat sous un fichier nommé **`publickey.cer`** . Il s'agit du *fichier de certificat de clé publique* de votre certificat SSL. Il inclut votre certificat de clé publique et la chaîne complète de confiance sous la forme des certificats intermédiaires qui ont été utilisés par la CA pour signer votre certificat.

6.1.2 Requêtes Client

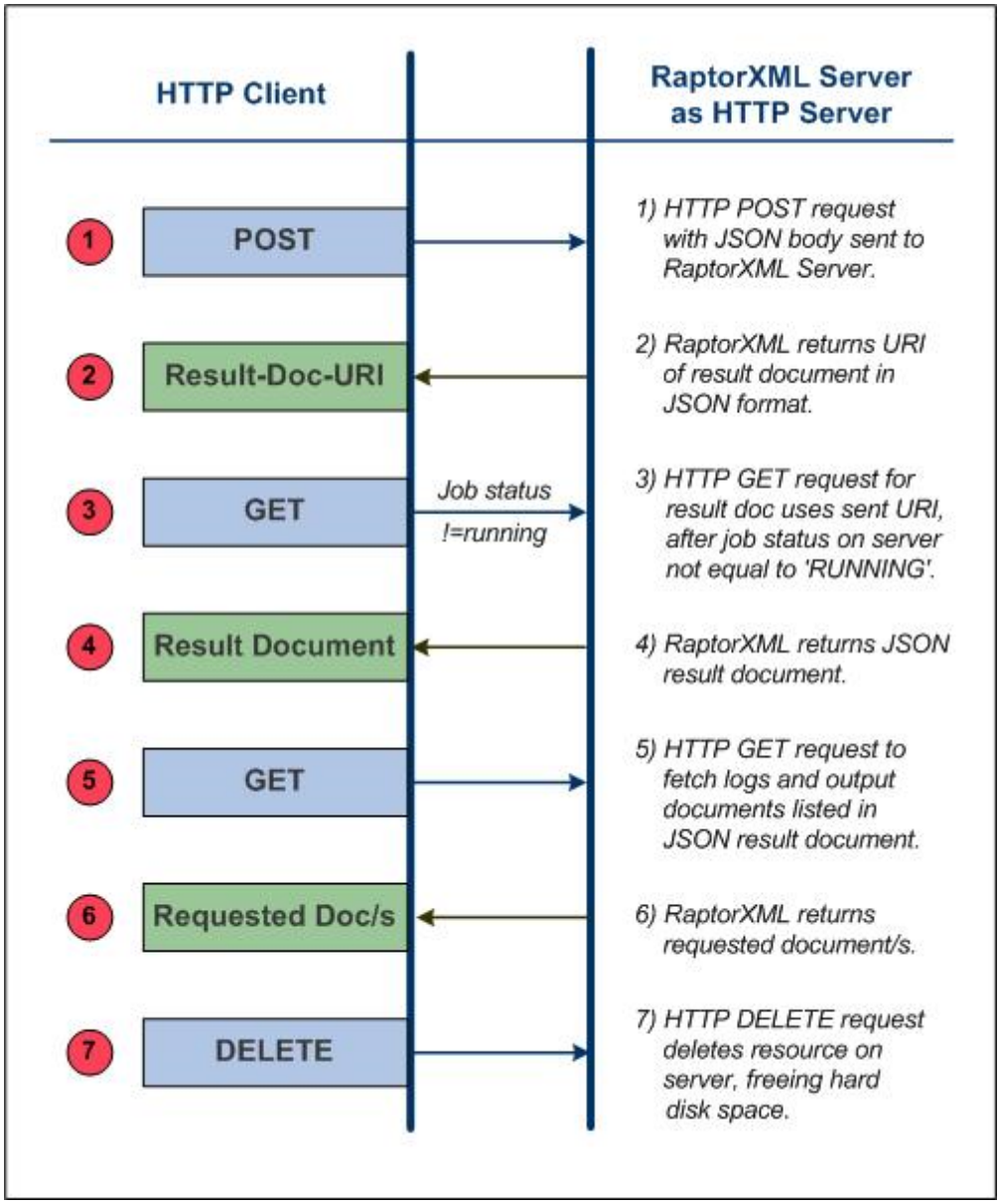
Une fois que RaptorXML Server a été lancé en tant qu'un service[229], sa fonction peut être accédée par tout client HTTP capable de :

- utiliser les méthodes HTTP `GET`, `PUT`, `POST` et `DELETE`
- Définissez le champ de l'en-tête `Content-Type`

Un client HTTP facile à utiliser

Il existe plusieurs clients web disponibles au téléchargement sur Internet. Le client web facile à utiliser et fiable que nous avons trouvé est RESTClient de Mozilla, qui peut être ajouté en tant qu'un plugin Firefox. Il est facile à utiliser, prend en charge les méthodes HTTP requises par RaptorXML et fournit une coloration syntaxique JSON suffisamment bonne. Si vous n'avez pas d'expérience préalable avec des clients HTTP, vous pouvez essayer RESTClient. Veuillez noter, néanmoins, que l'installation et l'utilisation de RESTClient est entièrement à vos risques.

Une requête client typique consiste en une série d'étapes telles qu'affichées dans le diagramme ci-dessous.



Les points importants caractérisant chaque étape sont notés ci-dessous. Les termes-clés sont gras.

1. Une méthode `POST` HTTP est utilisée pour faire une requête[242], le corps de la requête est en **format JSON**. La requête peut s'adresser à toute fonction de RaptorXML Server. Par exemple, la requête peut être pour une validation, ou pour une transformation XSLT. Les commandes, arguments et options utilisés dans la requête sont les mêmes que ceux utilisés dans la ligne de commande[57]. La requête est postée sur: **`http://localhost:8087/v1/queue`**, assumant que **`localhost:8087`** est l'adresse de RaptorXML Server (the adresse initiale du serveur[235]). Une telle requête est appelée une tâche **RaptorXML Server** .

2. Si la requête est reçue et acceptée pour le traitement par RaptorXML Server, un **document de résultat** contenant les résultats de l'action de serveur sera créé une fois que le job a été traité. L'**URI**

de ce document de résultat (le Result-Doc-URI dans le diagramme ci-dessus), est retourné sur le client[258]. Veuillez noter que l'URI sera retournée immédiatement après que la tâche ait été acceptée (queued) pour le traitement et même si le traitement n'a pas été achevé.

3. Le client envoie une requête pour le document de résultat[258] (en utilisant l'URI de document de résultat) dans une méthode `GET` sur le serveur. Si le traitement de la tâche n'a pas encore démarré ou n'a pas encore été achevé au moment de la réception de la requête, le serveur retourne un statut de *Running*. La requête `GET` doit être répétée jusqu'à ce que le traitement de la tâche ait été achevé et que le document de résultat ait été créé.

4. RaptorXML Server retourne le document de résultat dans le format JSON[259]. Le document de résultat peut contenir les **URI des documents d'erreurs ou de sortie** produits par le traitement de la requête d'origine par RaptorXML Server. Les journaux d'erreur sont retournés, par exemple, si une validation a retourné des erreurs. Des documents de sortie primaire, comme le résultat d'une transformation XSLT sont retournés si une tâche de production de sortie est achevée avec succès.

5. Le client envoie les URI des documents de sortie[262] reçu dans l'étape 4 par le biais d'une méthode `GET` HTTP vers le serveur. Chaque requête est envoyée dans une méthode `GET` séparée.

6. RaptorXML Server retourne les documents requis[262] en réponse aux requêtes `GET` effectués dans l'Étape 5.

7. Le client peut supprimer des documents non souhaités sur le serveur[264] qui ont été générés en tant qu'un résultat d'une requête de tâche. Cela s'effectue en soumettant, dans une méthode `DELETE` HTTP, l'URI du document de résultat en question. Tous les fichiers sur disque liés à cette tâche sont supprimés. Cela comprend le fichier de document de résultat, tout fichier temporaire et les fichiers de document d'erreur et de sortie. Cette étape est utile pour libérer de l'espace sur le disque dur du serveur.

Les détails de chaque étape sont décrits dans les sous-sections de cette section.

6.1.2.1 Initier les tâches avec POST

Cette section :

- Envoyer la requête[242]
- Syntaxe JSON pour les requêtes POST[242]
- Charger les fichiers avec la requête POST[245]
- Charger des archives ZIP[245]

Envoyer la requête

Une tâche RaptorXML Server est initiée avec la méthode `POST` HTTP

Méthode HTTP	URI	Type de contenu	Corps
`POST`	`http://localhost:8087/v1/queue/`	`application/json`	JSON

Veuillez prendre note des points suivants :

- L'URI ci-dessus a une adresse de serveur qui utilise les paramètres de la configuration initiale[231].
- L'URI a un chemin `/2024.2/queue/`, qui doit être présent dans l'URI. Il peut être considéré comme un dossier abstrait dans la mémoire dans laquelle la tâche est placée.
- Le numéro de version correct `/vN` est celui que le serveur retourne (et pas nécessairement celui dans cette documentation). Le nombre que le serveur retourne est le numéro de version de l'interface HTTP actuelle. Les numéros de version précédents indiquent des versions plus anciennes de l'interface HTTP, qui sont toujours pris en charge pour la rétrocompatibilité.
- L'en-tête doit contenir le champ : `Content-Type: application/json`. Néanmoins, si vous souhaitez charger des fichiers dans le corps de la requête `POST`, alors le type de contenu de l'en-tête du message doit être configuré sur `multipart/form-data` (par ex. `Content-Type: multipart/form-data`). Voir la section Charger les fichiers avec la requête POST[245] pour plus de détails.
- Le corps de la requête doit être en format JSON.
- Les fichiers à traiter doivent se trouver sur le serveur. Donc les fichiers doivent soit être copiés sur le serveur avant d'effectuer une requête, ou être chargés avec la requête POST[245]. Dans ce cas, le type de contenu de l'en-tête de message doit être configuré sur `multipart/form-data`. Voir la section Charger les fichiers avec la requête POST[245] ci-dessous pour plus de détails.

Pour vérifier la bonne formation d'un fichier XML, la requête dans le format JSON ressemblerait à l'exemple suivant :

```
{
    "command": "wfxml", "args": [ "file:///c:/Test/Report.xml" ]
}
```

Les commandes valides, et leurs arguments et options, sont telles que documenté dans la section de Ligne de commande[57].

Syntaxe JSON pour les requêtes HTTP POST

```
{
   "command": "Command-Name",
   "options": {"opt1": "opt1-value", "opt2": "opt2-value"},
   "args"   : ["file:///c:/filename1", "file:///c:/filename2"]
}
```

- Tout le texte noir est fixe et doit être inclus. Cela contient toutes les accolades, double guillemets, virgules, double-point et crochets. L'espace blanc peut être normalisé.
- Les italiques bleus sont des espaces réservés et sont synonymes de noms de commande, options et valeurs d'option et de valeurs d'argument. Veuillez vous référer à la section de ligne de commande[57] pour une description des commandes.
- Les clés `command` et `args` sont obligatoires. La clé `options` est optionnelle. Certaines clés `options` ont des valeurs par défaut ; donc parmi ces options, seules celles dont les valeurs par défaut doivent être changées doivent être spécifiées.

- Tous les strings doivent être inclus dans des doubles guillemets. Les valeurs booléennes et les nombres n'ont pas de guillemets. Donc : {"error-limit": "unlimited"} et {"error-limit": 1} est l'usage correct.
- Notez que les URI fichier —plutôt que les chemins de fichier—sont recommandés et qu'ils utilisent des barres obliques. Les chemins de fichier Windows, si utilisés, prennent des barres obliques vers l'arrière. De plus, les barres obliques file-path Windows doivent être dans une séquence d'échappement dans JSON (avec des séquences d'échappement de barre oblique ; comme "c:\\dir\\filename"). Veuillez noter que les URI de fichier et les chemins de fichier sont des strings et doivent donc se trouver entre guillemets.

Voici un exemple avec des options. Veuillez noter que certaines options (comme input ou xslt-version) prennent une valeur d'option directe, alors que d'autres (comme param) prennent une paire de valeur-clé en tant que leur valeur et nécessitent donc une syntaxe différente.

```
{
   "command": "xslt",
   "args": [
           "file:///C:/Work/Test.xslt"
           ],
   "options": {
              "input": "file:///C:/Work/Test.xml",
              "xslt-version": "1",
              "param": {
                       "key": "myTestParam",
                       "value": "SomeParamValue"
                       },
              "output": "file:///C:/temp/out2.xml"
              }
}
```

L'exemple ci-dessous montre un troisième type d'option : celle d'un table de valeurs (comme pour l'option xsd ci-dessous). Dans ce cas, la syntaxe à utiliser est celle d'un tableau JSON.

```
{
   "command": "xsi",
   "args": [
           "file:///C:/Work/Test.xml"
           ],
   "options": {
              "xsd" : ["file:///C:/Work/File1.xsd", "file:///C:/Work/File2.xsd"]
              }
}
```


Charger les fichiers avec la requête POST

Les fichiers à traiter peuvent être téléchargés dans le corps de la requête `POST`. Dans ce cas, la requête `POST` doit être effectuée comme suit.

En-tête de requête
Dans l'en-tête de requête, définir `Content-Type: multipart/form-data` et spécifier tout string arbitraire en tant que la limite. Voici une en-tête d'exemple :

```
Content-Type: multipart/form-data; boundary=---PartBoundary
```

L'objectif de la limite est de définir des limites des différentes parties données de forme dans le corps de la requête (*voir ci-dessous*).

Corps de requête : Partie de message
Le corps de la requête détient les parties données de forme suivantes, séparées par le string de limite spécifié dans l'en-tête de requête (*voir ci-dessus*) :

- *Parties données de forme obligatoires :* **`msg`**, qui spécifie l'action traitée requise, et **`args`**, qui contient les fichiers à télécharger comme argument/s de la commande spécifiée dans la partie form-data `msg`. *Voir la liste ci-dessous.*
- *Partie données de forme optionnelle :* Un nom de partie form-data **`additional-files`**, qui contient des fichiers référencés de fichiers dans `msg` ou des parties form-data`args`. Des parties de données de forme supplémentaires nommées selon une option de la commande peut aussi contenir des fichiers à charger.

Note : tous les fichiers chargés sont créés dans un seul répertoire virtuel.

Voir Exemple-1 (avec des légendes) : Valider XML 246 pour une explication détaillée du code, et Exemple-2 : Utiliser un catalogue pour trouver le schéma 247.

Tester avec CURL

Vous pouvez utiliser une application de transfert de données tiers comme CURL (http://curl.haxx.se/) pour tester la requête POST. CURL propose une option de trace précieuse qui génère et liste les limites des parties des requêtes. Cela vous épargnera la tâche de créer manuellement les limites de partie. La section Tester avec CURL 250 décrit comment utiliser CURL .

Charger des archives ZIP

Des archives ZIP peuvent aussi être chargés et des fichiers se trouvant dans un ZIP peuvent être référencés en utilisant le schéma **`additional-files`**. Par exemple :

```
additional-files:///mybigarchive.zip%7Czip/biginstance.xml
```

Note : La partie `|zip/` doit être échappée par URI en tant que `%7Czip/` pour pouvoir se conformer à l'URI RFC puisque le symbole `|` n'est pas directement autorisé. L'utilisation de motifs glob (`*` et `?`) est aussi autorisée. Vous pouvez donc utiliser quelque chose comme cela pour valider tous les fichiers XML dans l'archive ZIP :

```
{"command": "xsi", "args": ["additional-files:///mybigarchive.zip%7Czip/*.xml"],
"options": {...}}
```

Voir Exemple-3: Utiliser les archives ZIP 248 pour une liste du code d'exemple.

6.1.2.1.1 Exemple-1 (avec légendes): valider XML

Ci-dessous, vous trouverez une liste du corps d'une requête `POST`. Elle contient des légendes numérotées qui sont expliquées ci-dessous. La commande soumise dans la requête de liste aurait l'équivalent CLI suivant :

```
raptorxml xsi First.xml Second.xml --xsd=Demo.xsd
```

La requête est destinée à la validation de deux fichiers XML conformément à un schéma. Le corps de la requête ressemble à l'exemple suivant, en partant du principe que `---PartBoundary` a été spécifié dans l'en-tête en tant que le string de limite (*voir En-tête de requête 245 ci-dessus*).

```
-----PartBoundary                                                          1
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "options": {} , "args": []}                             2

-----PartBoundary                                                          3
Content-Disposition: attachment; filename="First.xml"; name="args"
Content-Type: application/octet-stream

    <?xml version="1.0" encoding="UTF-8"?>                                 4
    <test xsi:noNamespaceSchemaLocation="Demo.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">42</test>

-----PartBoundary                                                          5
Content-Disposition: attachment; filename="Second.xml"; name="args"
Content-Type: application/octet-stream

    <?xml version="1.0" encoding="UTF-8"?>                                 6
    <test xsi:noNamespaceSchemaLocation="Demo.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">35</test>

-----PartBoundary                                                          7
Content-Disposition: attachment; filename="Demo.xsd"; name="additional-files"
Content-Type: application/octet-stream
```



```
<?xml version="1.0" encoding="UTF-8"?>                                    8
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="test" type="xs:int"/>
</xs:schema>

-----PartBoundary--                                                       9
```

1 Le nom des limites de partie de données de forme principales sont déclarés dans l'en-tête de requête[245]. Le séparateur de limite de partie doit être un string unique qui ne se produira nulle part ailleurs dans les documents imbriqués. Il est préfixé avec deux tirets et est utilisé pour séparer les différentes parties. La première partie de données de forme dans cet exemple est **msg**. Veuillez noter que le type de contenu est `application/json`.

2 Voici la syntaxe standard pour les requêtes POST HTTP[243]. Si `args` contient une référence à un fichier et si des fichiers supplémentaires sont chargés, les deux sets de files seront passés au serveur.

3 Le premier membre du tableau `args` est un attache de fichier appelé `First.xml`.

4 Le texte du fichier `First.xml`. Il contient une référence au schéma appelé `Demo.xsd`, qui sera aussi chargé—dans la partie de donnée de forme `additional-files`.

5 Le second membre du tableau `args` est un attache appelé `Second.xml`.

6 Le texte du fichier `Second.xml`. Lui aussi contient une référence au schéma `Demo.xsd`. *Voir légende 7.*

7 Les premiers fichiers supplémentaires contiennent les métadonnées d'attache `Demo.xsd`.

8 Le texte du fichier `Demo.xsd`.

9 La fin de la partie des fichiers supplémentaires `Demo.xsd`, et la partie de données de forme `additional-files`. Veuillez noter que le dernier séparateur de limite de partie est aussi bien préfixé que postfixé avec deux tirets.

6.1.2.1.2 Exemple-2: utiliser un catalogue pour trouver le schéma

Dans cet exemple, un fichier de catalogue est utilisé pour trouver le schéma XML qui est référencé par les fichiers XML à valider.

```
-----PartBoundary
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "args": ["additional-files:///First.xml", "additional-
files:///Second.xml"], "options": {"user-catalog": "additional-files:///catalog.xml"}}

-----PartBoundary
Content-Disposition: attachment; filename="First.xml"; name="additional-files"
```

```
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<test xsi:noNamespaceSchemaLocation="http://example.com/Demo.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">42</test>

-----PartBoundary
Content-Disposition: attachment; filename="Second.xml"; name="additional-files"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<test xsi:noNamespaceSchemaLocation="http://example.com/Demo.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">35</test>

-----PartBoundary
Content-Disposition: attachment; filename="Demo.xsd"; name="additional-files"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:element name="test" type="xs:int"/>
</xs:schema>

-----PartBoundary
Content-Disposition: attachment; filename="catalog.xml"; name="additional-files"
Content-Type: application/octet-stream

<?xml version='1.0' encoding='UTF-8'?>
<catalog xmlns='urn:oasis:names:tc:entity:xmlns:xml:catalog'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd'>
                  <uri name="http://example.com/Demo.xsd" uri="additional-
files:///Demo.xsd"/>
</catalog>

-----PartBoundary--
```

6.1.2.1.3 Exemple-3 : utiliser des archives ZIP

Des archives ZIP peuvent aussi être chargés et des fichiers se trouvant dans un ZIP peuvent être référencés en utilisant le schéma **additional-files**. Par exemple :

```
additional-files:///mybigarchive.zip%7Czip/biginstance.xml
```

Note : La partie `|zip/` doit être échappée par URI en tant que `%7Czip/` pour pouvoir se conformer à l'URI RFC puisque le symbole `|` n'est pas directement autorisé. L'utilisation de motifs glob (`*` et `?`) est aussi autorisée. Vous pouvez donc utiliser quelque chose comme cela pour valider tous les fichiers XML dans l'archive ZIP :

```
{"command": "xsi", "args": ["additional-files:///mybigarchive.zip%7Czip/*.xml"],
"options": {…}}
```


Note : `'Content-Disposition: form-data'` est aussi valide, outre `'Content-Disposition: attachment'`. Puisque plusieurs outils génèrent `form-data` en tant que disposition de contenu, la valeur `form-data` est acceptée comme étant valide.

Exemple : Valider tous les fichiers XML dans un archive ZIP

Dans cet exemple, on part du principe que toutes les références de schéma sont des chemins relatifs et que tous les schémas sont contenus dans le zip.

```
-----PartBoundary
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "args": ["additional-files:///Demo.zip%7Czip/*.xml"], "options": {}}

-----PartBoundary
Content-Disposition: attachment; filename="Demo.zip"; name="additional-files"
Content-Type: application/octet-stream

Binary content of Demo.zip archive

-----PartBoundary--
```

Exemple : Valider les fichiers XML dans un archive ZIP contenant des références à des schémas externes

Dans cet exemple, les fichiers XML contenus dans un archive ZIP sont validés en utilisant des références à un schéma externe, qui est fourni dans un second archive ZIP.

```
-----PartBoundary
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "args": ["additional-files:///Instances.zip%7Czip/*.xml"], "options":
{"user-catalog": "additional-files:///Schemas.zip%7Czip/catalog.xml"}}

-----PartBoundary
Content-Disposition: attachment; filename="Instances.zip"; name="additional-files"
Content-Type: application/octet-stream

Binary content of Instances.zip archive

-----PartBoundary
Content-Disposition: attachment; filename="Schemas.zip"; name="additional-files"
Content-Type: application/octet-stream

Binary content of Schemas.zip archive

-----PartBoundary--
```

6.1.2.1.4 Tester avec CURL

L'application tierce CURL (http://curl.haxx.se/) est un outil de ligne de commande que vous pouvez utiliser pour tester la requête `POST`. CURL propose une option de trace très utile qui génère et liste les limites de partie des requêtes qui peut être utilisé directement dans vos requêtes ou en tant qu'une référence.

Ci-dessous, vous trouverez un scénario test dans lequel un fichier XML est validé par rapport à un schéma XML. Considérant que :

- les commandes ci-dessous sont exécutées depuis le dossier dans lequel se trouvent les fichiers à soumettre à la validation ; (cela nous permet d'écrire des chemins relatifs simples vers ces fichiers). Si vous avez installé l'application XMLSpy d'Altova, les fichiers utilisés dans cet exemple peuvent être trouvés dans le dossier `Examples` de l'application, qui se trouve par défaut sous : `C:\Users\<username>\Documents\Altova\XMLSpy2024\Examples`
- RaptorXML Server est exécuté localement sur le port 8087

Pour plus d'informations concernant les options de ligne de commande CURL, voir l'aide CURL.

Appeler CURL avec la commande de validation sur

```
[input: powershell]
    \path\to\curl.exe -F 'msg={\"command\": \"xsi\", \"args\":[\"additional-
    files:///PurchaseOrder.zip%7Czip/ipo.xml\"], \"options\":{}};type=application/json' -F
    "additional-files=@PurchaseOrder.zip;type=application/octet-stream"
    http://localhost:8087/v1/queue
```

Note : dans powershell, si des guillemets apparaissent dans des guillemets, il faudra utiliser un type différent de guillemets (simple/double).

```
[input: cmd]
    \path\to\curl.exe -F "msg={\"command\": \"xsi\", \"args\":[\"additional-
    files:///PurchaseOrder.zip%7Czip/ipo.xml\"], \"options\":{}};type=application/json" -F
    "additional-files=@PurchaseOrder.zip;type=application/octet-stream"
    http://localhost:8087/v1/queue

[output]
    {"jobid": "058F9E97-CB95-43EF-AC0A-496CD3AC43A3", "result": "/v1/results/058F9E97-CB95-
    43EF-AC0A-496CD3AC43A3"}
```

Utiliser l'URL de "result" pour obtenir le résultat

```
[input]
    \path\to\curl.exe http://localhost:8087/v1/results/058F9E97-CB95-43EF-AC0A-496CD3AC43A3

[output]
    {"jobid":"058F9E97-CB95-43EF-AC0A-496CD3AC43A3","state":"OK","error":{},"jobs":
    [{"file":"additional-files:///PurchaseOrder.zip%7Czip/ipo.xml","jobid":"D4B91CB0-CF03-
    4D29-B563-B6506E123A06","output":{},"state":"OK","error":{}}]}
```


L'option trace de CURL

CURL dispose d'une option trace (`--trace-ascii`), qui trace le trafic HTTP envoyé de et vers le serveur. L'option est très utile puisqu'elle liste les limites de partie qui sont requises pour lancer des tâches avec POST. Vous pouvez utiliser l'information contenue dans trace, soit directement soit en tant que référence, pour créer des limites de partie. La liste ci-dessous montre la trace obtenue en exécutant la commande donnée ci-dessus.

- *Liste de trace*

```
== Info:   Trying ::1...
== Info: Connected to localhost (::1) port 8087 (#0)
=> Send header, 217 bytes (0xd9)
0000: POST /v1/queue HTTP/1.1
0019: Host: localhost:8087
002f: User-Agent: curl/7.42.1
0048: Accept: */*
0055: Content-Length: 2939
006b: Expect: 100-continue
0081: Content-Type: multipart/form-data; boundary=--------------------
00c1: ----d887ed58324015c3
00d7:
<= Recv header, 23 bytes (0x17)
0000: HTTP/1.1 100 Continue
=> Send data, 393 bytes (0x189)
0000: --------------------------d887ed58324015c3
002c: Content-Disposition: form-data; name="msg"
0058: Content-Type: application/json
0078:
007a: {"command": "xsi", "args":["additional-files:///PurchaseOrder.zi
00ba: p%7Czip/ipo.xml"], "options":{}}
00dc: --------------------------d887ed58324015c3
0108: Content-Disposition: form-data; name="additional-files"; filenam
0148: e="PurchaseOrder.zip"
015f: Content-Type: application/octet-stream
0187:
=> Send data, 2498 bytes (0x9c2)
0000: PK........"..6}.c.....M.......ipo.xsd.T.N.@.}N....O 5v.}..S....(
0040:  .JU/...$Y..5{.E.•.....I*...g...Y...\....Z..~......P.A.ct....•y.
...
0940: .........."..6]g......l.............  .......address.xsdPK.......
0980: ..."..6I..v..................  .......ipo.xmlPK..................
09c0: ..
=> Send data, 48 bytes (0x30)
0000:
0002: --------------------------d887ed58324015c3--
<= Recv header, 22 bytes (0x16)
0000: HTTP/1.1 201 Created
<= Recv header, 13 bytes (0xd)
0000: Allow: POST
<= Recv header, 32 bytes (0x20)
0000: Content-Type: application/json
<= Recv header, 37 bytes (0x25)
0000: Date: Fri, 24 Jul 2015 16:58:08 GMT
```

```
<= Recv header, 24 bytes (0x18)
0000: Server: CherryPy/3.6.0
<= Recv header, 21 bytes (0x15)
0000: Content-Length: 111
<= Recv header, 2 bytes (0x2)
0000:
<= Recv data, 111 bytes (0x6f)
0000: {"jobid": "058F9E97-CB95-43EF-AC0A-496CD3AC43A3", "result": "/v1
0040: /results/058F9E97-CB95-43EF-AC0A-496CD3AC43A3"}
== Info: Connection #0 to host localhost left intact
```

Note : Veuillez noter, dans la liste supérieure `'Content-Disposition: form-data'` est aussi valide, outre `'Content-Disposition: attachment'`.

Appeler CURL avec la commande de bonne formation sur Linux

```
/path/to/curl -F 'msg={"command": "wfxml", "args":[]};type=application/json' -F
"args=@ipo.xml;type=application/octet-stream" http://localhost:8087/v1/queue

/path/to/curl -F 'msg={"command": "wfxml", "args":["additional-files:///ipo.zip%
7Czip/ipo.xml"]};type=application/json' -F "additional-
files=@ipo.zip;type=application/octet-stream" http://localhost:8087/v1/queue
```

6.1.2.1.5 Example-6: XQuery Execution

Cet exemple utilise PowerShell sur Windows pour exécuter un document XQuery sur un document XML. Les deux documents sont situés dans le dossier **`exemples`** de votre dossier d'application (RaptorXMLServer2024).

Note : l'utilisation des guillemets peut être différente sur d'autres shells ('bash' fonctionne avec l'exemple quand il utilise 'curl' à la place de 'curl.exe').

Soumettre la requête POST de validation XBRL Inline utilisant CURL

Ci-dessous, vous trouverez un exemple de la commande CURL pour soumettre une requête de validation XBRL Inline.

```
curl.exe -F 'msg={"command": "xquery", "args": ["additional-files:///CopyInput.xq"],
"options": {"input": "additional-files:///simple.xml", "output":
"MyQueryResult"}};type=application/json' -F "additional-
files=@CopyInput.xq;type=text/plain"  -F "additional-
files=@simple.xml;type=application/xml" http://localhost:8087/v1/queue
```

Pour une meilleure lisibilité :

```
(1) -F 'msg={
(2)     "command": "xquery",
(3)     "args": ["additional-files:///CopyInput.xq"],
```



```
(4)     "options": {"input": "additional-files:///simple.xml", "output":
"MyQueryResult"}
(5) };type=application/json'
(6) -F "additional-files=@CopyInput.xq;type=text/plain"
(7) -F "additional-files=@simple.xml;type=application/ xml"
(7) http://localhost:8087/v1/queue
```

Entrée
La différentes parties de la commande CURL sont expliquées ci-dessous, verrouillées aux callouts dans la liste ci-dessus.

(1) **`-F 'msg={...}'`** spécifie un champ de formulaire avec le nom **`'msg'`**

L'option **`-F`** : (i) fait que CURL génère un « multipart form post » avec **`Content-Type: multipart/form-data`** et (ii) fait que ce champ de formulaire est automatiquement ajouté à l'en-tête de la requête. Nous utilisons un objet JSON pour décrire la commande que RaptorXML Server devrait exécuter.

```
Content-Type: multipart/form-data; boundary=--------------------...
```

CURL traduit cette option en une requête HTTP en :

```
Content-Disposition: form-data; name="msg"
Content-Type: application/json
{"command": "xquery", "args": ["additional-files:///CopyInput.xq"], "options": {"input":
"additional-files:///simple.xml", "output": "MyQueryResult"}}
```

(2) la commande RaptorXML Server à exécuter sur le serveur. Voir la section Command Line Interface (CLI) 57 pour des informations sur les commandes qui peuvent être acceptées ici. Dans notre exemple, la commande pour l'exécution XQuery est **`XQuery`** 92.

(3) Les arguments de la commande (tels qu'acceptés par la ligne de commande RaptorXML Server) sont encodés comme array JSON. L'RaptorXML Server utilise un schéma explicite **`additional-files://`** pour référencer des ressources supplémentaires à l'intérieur du champ de formulaire **`additional-files`** séparé. Dans notre exemple, nous référençons le document XQuery **`CopyInput.xq`**.

Note : toutes les ressources dans l'array **`args`** doivent être disponibles sur le serveur ou soumises avec la requête similaire à (6 and 7).

(4) les options de la commande (tel qu'accepté par la ligne de commande RaptorXML Server) sont encodées comme objet JSON. Si les valeurs par défaut des options sont comme vous les voulez (voir la section CLI 57), alors cette partie peut être omise. Dans notre exemple, nous spécifions (i) le fichier XML sur lequel XQuery doit être exécuté et (ii) le fichier sur lequel la sortie de l'exécution sera stockée.

(5) Le Content-Type du champ de formulaire **msg** est spécifié après la définition du champ de formulaire et est séparé de celui-ci par un point-virgule. Dans notre exemple, le Content-Type de **msg** est donné par : **type=application/json**.

(6, 7) Les fichiers qui contiennent des ressources supplémentaires pour la commande peuvent être spécifiés utilisant le champ de formulaire **additional-files**. Dans notre exemple, nous spécifions deux ressources supplémentaires : (i) **@CopyInput.xq**, suivi d'un séparateur de point-virgule et puis son Content-Type, que nous indiquons comme **type=text/plain**; (i) **simple.xml**, suivi d'un séparateur de point-virgule et puis son Content-Type, que nous indiquons comme **type=application/xml**.

Note : donnez un préfixe au nom de fichier avec **@** pour instruire CURL pour (i) utiliser le nom de fichier comme la valeur de la propriété **filename** et (ii) le contenu du fichier comme la valeur du formulaire. Le champ de formulaire des fichiers supplémentaires peut être fourni de multiples fois, une fois pour chaque ressource supplémentaire requise par la commande. CURL traduit cette option dans la requête HTTP suivante :

```
Content-Disposition: form-data; name="additional-files"; filename="CopyInput.xq"
Content-Type: text/plain
<<content of CopyInput.xq>>

Content-Disposition: form-data; name="additional-files"; filename="CopyInput.xq"
Content-Type: application/ xml
<<content of simple.xml>>
```

Note : les fichiers dans d'autres dossiers peuent être fournis en mettant le chemin relatif devant le nom de fichier comme ceci : `-F "additional-files=@Examples/CopyInput.xq;type=text/plain"`. Toutefois, quand un fichier additionnel depuis un autre dossier est spécifié de cette manière, il doit être référencé en utilisant le nom du fichier uniquement Par exemple :

```
curl.exe -F 'msg={"command": "xquery", "args": ["additional-files:///CopyInput.xq"],
"options": {"output": "MyQueryResult"}};type=application/json' -F "additional-
files=@Examples/CopyInput.xq;type=text/plain"  http://localhost:8087/v1/queue
```

Si vous souhaitez préserver une structure de dossier, mettez les fichiers dans un dossier ZIP et référencez les fichiers de manière usuelle aux dossiers ZIP 248.

Sortie
La sortie RaptorXML Server est un objet JSON :

```
{"jobid": "42B8A75E-0180-4E05-B28F-7B46C6A0C686", "result": "/v1/results/42B8A75E-0180-
4E05-B28F-7B46C6A0C686"}
```

L'objet JSON contient une clé **jobid** et une clé **result**. La valeur de la clé **result** est le chemin au résultat. Ce chemin doit être ajouté à la partie **<scheme>://<host>:<port>** utilisée pour soumettre la requête. Dans notre exemple, le résultat entier de l'URL serait : **http://localhost:8087/v1/results/42B8A75E-0180-4E05-B28F-7B46C6A0C686**. Le résultat de l'URL de résultat est aussi utilisé pour demander le résultat de l'exécution de la commande. *Voir Obtenir le résultat du document 258.*

Obtenir l'erreur/le message/la sortie de la requête POST

La commande d'entrée qui est envoyée pour recevoir l'erreur/le message/la sortie de la requête POST (*voir* *Obtenir Documents d'erreur/de message/de sortie* 262) serait comme suit :

```
curl.exe http://localhost:8087/v1/results/42B8A75E-0180-4E05-B28F-7B46C6A0C686
```

Dans notre exemple, cette commande retourne l'objet JSON suivant :

```
{"jobid":"42B8A75E-0180-4E05-B28F-7B46C6A0C686","state":"OK","error":{},"jobs":
[{"file":"additional-files:///simple.xml","jobid":"768656F9-F4A1-4492-9676-
C6226E30D998","output":{"result.trace_file":["/v1/results/768656F9-F4A1-4492-9676-
C6226E30D998/output/trace.log"],"xquery.main_output_files":["/v1/results/768656F9-F4A1-
4492-9676-C6226E30D998/output/1"],"xquery.additional_output_files":
[]},"state":"OK","output-mapping":{"/v1/results/768656F9-F4A1-4492-9676-
C6226E30D998/output/1":"file:///C:/ProgramData/Altova/RaptorXMLXBRLServer2016/Output/768
656F9-F4A1-4492-9676-C6226E30D998/MyQueryResult"},"error":{}}]}
```

Ceci est transcrit sur des lignes séparées dessous pour une lisibilité plus facile et avec des callouts pour un référencement plus facile :

```
(1) {
(2)   "jobid":"42B8A75E-0180-4E05-B28F-7B46C6A0C686",
(3)   "state":"OK",
(4)   "error":{},
(5)   "jobs":[{
(6)      "file":["additional-files:///simple.xml"],
(7)      "jobid":"768656F9-F4A1-4492-9676-C6226E30D998",
(8)      "output":{
(9)         "result.trace_file":["/v1/results/768656F9-F4A1-4492-9676-
C6226E30D998/output/trace.log"],
(10)        "xquery.main_output_files":["/v1/results/768656F9-F4A1-4492-9676-
C6226E30D998/output/1"],
(11)        "xquery.additional_output_files":[]},
(12)        "state":"OK",
(13)        "output-mapping":{
(14)           "/v1/results/768656F9-F4A1-4492-9676-C6226E30D998/output/1":
(15)           "file:///C:/ProgramData/Altova/RaptorXMLXBRLServer2016/Output/768656F9-
F4A1-4492-9676-C6226E30D998/MyQueryResult"
(16)          },
(17)      "error":{}
(18)    }]
(19) }
```

Ci-dessous, vous trouverez une explication de cette liste :

(1) Le résultat est retourné en tant qu'un objet JSON.

(2) Le jobid au premier niveau est l'identifiant principal de la tâche.

(3) L'état pour cette tâche est OK. Les états possibles sont : *none ; Dispatched ; Running ; Canceled ; Crashed ; OK ; Failed*.

(4) L'objet d'erreur JSON dans notre exemple est vide. Il peut contenir une sérialisation JSON de l'erreur comme rapporté par RaptorXML Server.

(5) La tâche principale (au premier niveau) génère les sous-tâches (par exemple, un par argument).

(6) L'argument pour cette tâche est le fichier d'instance XML : **`additional-files:///simple.xml`**.

(7) Les sous-tâches ont également un identifiant de tâche qui peut être utilisé pour interroger l'état de ou chercher les résultats. L'exécution de la tâche est asynchrone. En guise de résultat, des tâches courtes soumises après de longues tâches peuvent être terminées plus tôt.

(8) à (16) L'objet de la **`sortie`** JSON contient des clés pour les fichiers de sortie générés par le serveur qui peuvent être demandées via HTTP. Quelques clés (telles que **`xquery.main_output_files`**) spécifient les URL vers les fichiers générés stockés sur le serveur. Ces chemins server-local peuvent être mappés vers les noms, qui peuvent être utilisés comme objets JSON **`output-mapping`** dans les URL HTTP. De tels URL sont utilisés pour récupérer des fichiers de sortie via HTTP et sont constitués comme suit :

```
<scheme>://<host>:<port>/<output-mapping-value>
```

Notre exemple pour récupérer le fichier de sortie principal XQuery aurait donc l'air de ceci :

```
curl.exe http://localhost:8087/v1/results/768656F9-F4A1-4492-9676-C6226E30D998/output/1
```

Notez que dans l'objet **`output-mapping`** (13), la première valeur (14) est la valeur de mappage que nous avons saisi dans la sortie XQuery (15), **`file:///C:/ProgramData/Altova/RaptorXMLXBRLServer2016/Output/768656F9-F4A1-4492-9676-C6226E30D998/MyQueryResult`**. Ceci nous permet d'utiliser la valeur de mappage pour référencer le fichier.

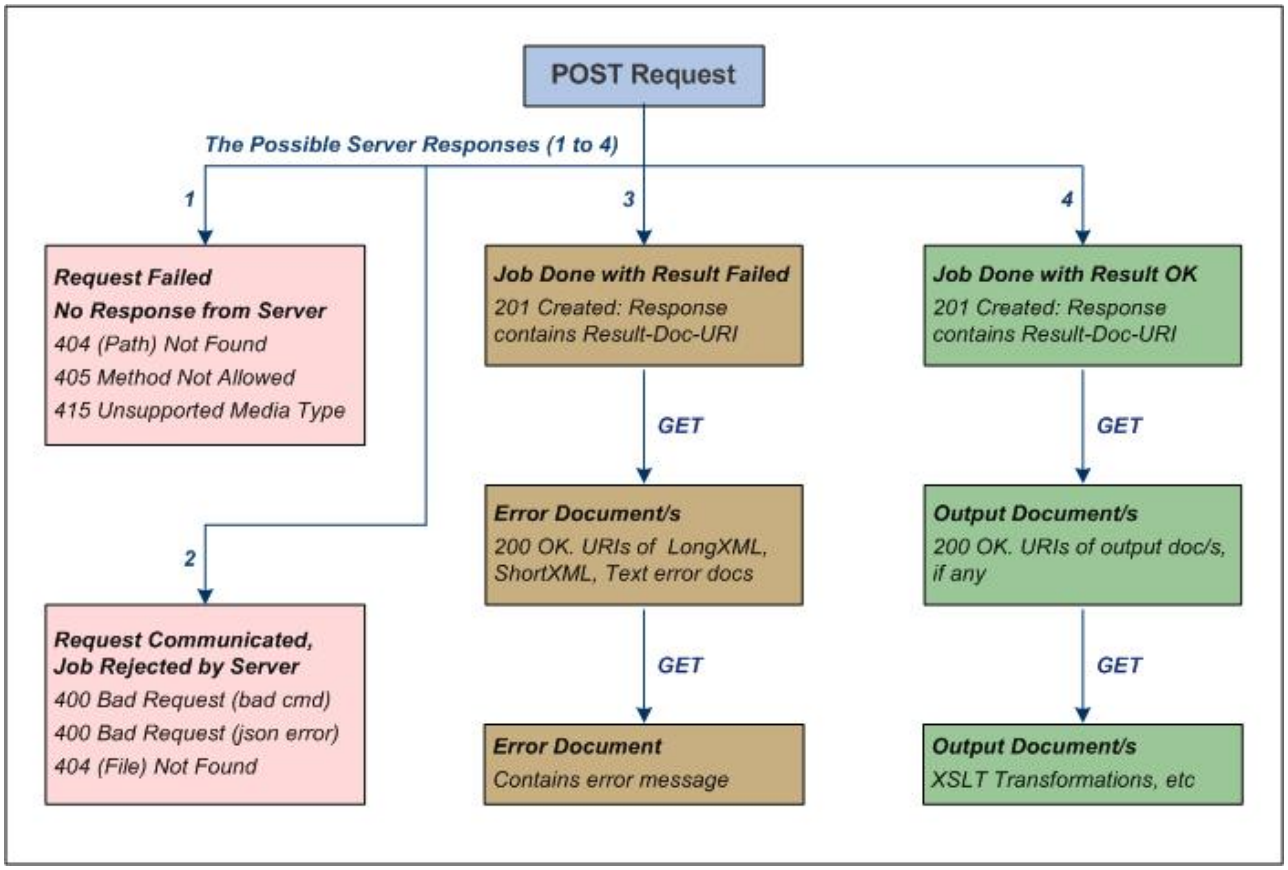
6.1.2.2 Réponse de serveur à une requête POST

Cette section :

Lorsqu'une requête `POST` est effectuée avec succès auprès du serveur, la tâche est placée dans la file d'attente du serveur. Un message `201 Created` et un URI de document de résultat sont retournés. La tâche sera traitée au plus tôt. Entre-temps, si le document de résultat est requis 258, un message `"status":` Le message `"Running"` est retourné si la tâche a démarré mais n'a pas été terminée ; le client devra essayer à nouveau plus tard. Un état `Dispatched` indique que la tâche se trouve dans la file d'attente du serveur mais n'a pas encore été démarrée.

Le résultat de la tâche (par exemple, une requête de validation) peut être négative (échec de la validation) ou positive (validation réussie). Dans les deux cas, un message `201 Created` est retourné et un document de

résultat est généré. Il est également possible que la requête `POST` n'ait pas été communiquée au serveur (*Échec de la requête*), ou bien, la requête a été communiquée mais la tâche a été rejetée par le serveur (*Requête communiquée, mais tâche rejetée*). Les différents résultats possibles sont montrés dans le diagramme ci-dessous.



Échec de la requête, aucune réponse du serveur

En cas d'échec de requête sur le serveur, les erreurs les plus communes sont les suivantes :

Message	Explication
404 Introuvable	Le chemin correct est : `http://localhost:8087/v1/queue/`
405 Méthode non autorisée	La méthode spécifiée est invalide pour cette ressource. Utiliser la méthode `POST`.
415 Format de requête non pris en charge	L'en-tête de message doit être `Content-Type:application/json`.

Requête communiquée, mais la tâche est rejetée par le serveur

Lorsque les requêtes sont effectuées avec succès sur le serveur, celui-ci peut les rejeter pour les raisons suivantes :

Message	Explication
400 Bad Request (*bad cmd*)	La commande RaptorXML 57 est incorrecte.
400 Bad Request (*json error*)	Le corps de requête contient une erreur de syntaxe JSON 243.
404 File Not Found	Contrôler syntaxe d'URI de fichier (ou chemin de fichier) 243 de tous les fichiers nommés dans la commande.

Tâche exécutée (avec un résultat positif ou négatif)

Lorsqu'une tâche (par exemple, une tâche de validation) est exécutée, son résultat peut être positif (*OK*) ou négatif (*Échec*). Par exemple, le résultat d'une tâche de validation est positif (*OK*) lorsque le document à valider est valide, négatif (*Échec*) si le document est invalide.

Dans les deux cas, la tâche est exécutée, mais avec des résultats différents. Un message `201 Created` est retourné dans les deux cas dès que la tâche a été placée avec succès dans la file d'attente. De même, dans les deux cas, un URI de document de résultat est retourné vers le client HTTP qui a effectué la requête. Après avoir créé le document de résultat, il peut être récupéré avec une requête HTTP `GET`.

Le document de résultat lui-même n'a peut-être pas encore été créé si la tâche n'a pas encore été démarrée ou terminée. Si le document de résultat est demandé à ce moment-là, un message `« statut » : "Running"` est retourné si la tâche a été démarrée mais pas encore terminée ; un statut `Dispatched` indique que la tâche se trouve dans le serveur mais pas encore été lancée.

En plus du document de résultat, d'autres documents peuvent être générés, comme suit :

- *Tâche exécutée avec le résultat « Échec » :* Un journal d'erreur est créé dans trois formats : texte, XML long et XML court. Les URI de ces trois documents sont envoyés dans le document de résultat (qui est en format JSON). `Les URI peuvent ensuite être utilisés dans une requête` GETpour aller chercher les documents d'erreur. 262
- *Tâche exécutée avec le résultat « OK » :* La tâche est traitée avec succès et les documents de sortie —tels que la sortie produite par une transformation XSLT —sont créés. Si des fichiers de sortie ont été générés, leurs URI sont envoyés dans le document de résultat de format JSON. Les URI peuvent ensuite être utilisés dans une requête `GET` HTTP pour aller chercher les documents de sortie. Veuillez noter que toutes les tâches n'auront pas forcément de fichiers de sortie ; par exemple, une tâche de validation. Ainsi, une tâche peut se terminer avec un état '*OK*', mais il peut s'être produit des avertissements et/ou des messages qui ont été rédigés des fichiers d'erreur. Dans ce cas, les URI de fichier d'erreur sont aussi envoyés dans le document de résultat (c'est à dire, en plus des documents de sortie).

Voir Obtenir le document de résultat 258 et Obtenir les documents d'erreur/sortie 262 pour une description de ces documents et de leur accès.

6.1.2.3 Obtenir le document de résultat

Cette section :

- L'URI de document de résultat 259

L'URI de document de résultat

Un document de résultat sera créé à chaque fois que qu'une tâche est créée, quel que soit le résultat d'une tâche (par exemple, une validation) est positive (document valide) ou négative (document invalide). Dans les deux cas, un message `201 Created` est retourné. Ce message sera en format JSON et contiendra un URI relatif du document de résultat. Le fragment JSON ressemblera à l'exemple ci-dessous :

```
{
   "result": "/2024.2/results/E6C4262D-8ADB-49CB-8693-990DF79EABEB",
   "jobid": "E6C4262D-8ADB-49CB-8693-990DF79EABEB"
}
```

L'objet `result` contient l'URI relatif du document de résultat. L'URI est relatif à l'adresse de serveur 235. Par exemple, si l'adresse de serveur est `http://localhost:8087/` (l'adresse de configuration initiale 231), alors l'URI agrandi du document de résultat spécifié dans la liste ci-dessus sera :

```
http://localhost:8087/2024.2/results/E6C4262D-8ADB-49CB-8693-990DF79EABEB
```

Note : le numéro de version correct `/vN` est celui que le serveur retourne (et n'est pas nécessairement celui dans cette documentation). Le numéro que le serveur retourne est le numéro de version de l'interface HTTP actuel. Les numéros de version précédents indiquent des versions plus anciens de l'interface HTTP, qui, néanmoins, sont toujours pris en charge pour la rétrocompatibilité.

Récupérer le document de résultat

Pour obtenir le document de résultat, soumettre l'URI agrandi du document (*voir ci-dessus* 259), dans une requête `GET` HTTP. Le document de résultat est retourné et pourrait être un des types génériques décrits ci-dessous.

Note : Lorsqu'une tâche est placée avec succès dans la file d'attente du serveur, le serveur retourne l'URI du document de résultat. Si le client requête le résultat avant que la tâche ait été lancée (elle se trouve toujours dans la file d'attente), un message `"status": "Dispatched"` sera retourné. Si la tâche a été lancée mais n'a pas été terminée (par exemple parce qu'il s'agit d'une tâche volumineuse), un message `"status": "Running"` sera retourné. Dans ces deux situations, le client devrait attendre un certain temps avant de faire une requête fraîche pour le document de résultat.

Note : Tous les documents d'exemple ci-dessous ont un accès de client limité 227. Donc les documents d'erreur, les documents de message et les documents de sortie sont tous présumés être enregistrés dans le répertoire de tâche pertinent sur le serveur. Les URI les concernant dans le document de résultat sont donc tous des URI relatifs. Aucun n'est un URI de fichier (qui serait le type d'URI généré dans les cas d'accès client non limités 227). En ce qui concerne les détails de ces URI, voir la section Obtenir des documents Erreur/Message/Sortie 262.

Document de résultat contenant les URI des documents d'erreur

Si la tâche requise s'est terminée avec un état *Échec*, alors la tâche a retourné un résultat négatif. Par exemple, une tâche de validation a retourné un résultat de document invalide. Les erreurs rencontrées lors de l'exécution de la tâche sont stockées dans le journal d'erreur, créé en trois formats de fichiers : (i) texte, (ii) XML long (journal d'erreur détaillé), et (iii) XML court (journal d'erreur moins détaillé). Voir la liste JSON ci-dessous.


```
{
    "jobid": "6B4EE31B-FAC9-4834-B50A-582FABF47B58",
    "state": "Failed",
    "error":
    {
      "text": "/2024.2/results/6B4EE31B-FAC9-4834-B50A-582FABF47B58/error/error.txt",
      "longxml": "/2024.2/results/6B4EE31B-FAC9-4834-B50A-582FABF47B58/error/long.xml",
      "shortxml": "/2024.2/results/6B4EE31B-FAC9-4834-B50A-582FABF47B58/error/short.xml"
    },
    "jobs":
    [
      {
        "file": "file:///c:/Test/ExpReport.xml",
        "jobid": "20008201-219F-4790-BB59-C091C276FED2",
        "output":
        {
        },
        "state": "Failed",
        "error":
        {
          "text": "/2024.2/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt",
          "longxml": "/2024.2/results/20008201-219F-4790-BB59-C091C276FED2/error/long.xml",
          "shortxml": "/2024.2/results/20008201-219F-4790-BB59-
C091C276FED2/error/short.xml"
        }
      }
    ]
}
```


Veuillez noter :

- Les tâches ont des sous-tâches.
- Les erreurs au niveau de sous-tâche se propagent jusqu'à la tâche de niveau supérieur. L'état de la tâche de niveau supérieur sera seulement *OK* si toutes ses sous-tâches ont un état *OK*.
- Chaque tâche sous sous-tâche a son propre journal d'erreur.
- Les journaux d'erreur comprennent des journaux d'avertissement. Donc, même si une tâche se termine avec un état *OK*, il pourrait avoir des URI de fichiers d'erreur.
- Les URI des fichiers d'erreur sont relatifs à l'adresse de serveur (*voir ci-dessus* 259).

Document de résultat contenant les URI des documents de sortie

Si la tâche requise s'est terminée avec un état *OK*, la tâche a retourné un résultat positif. Par exemple, une tâche de validation a retourné un résultat de document valide. Si la tâche a produit un document de sortie, par exemple, le résultat d'une transformation XSLT, alors l'URI du document de sortie est retourné. Voir la liste JSON ci-dessous.


```
{
    "jobid": "5E47A3E9-D229-42F9-83B4-CC11F8366466",
    "state": "OK",
    "error":
    {
```

```
    },
    "jobs":
    [
      {
        "file": "file:///c:/Test/SimpleExample.xml",
        "jobid": "D34B5684-C6FF-4A7A-BF35-EBB9A8A8C2C8",
        "output":
        {
          "xslt-output-file":
          [
            "/2024.2/results/D34B5684-C6FF-4A7A-BF35-EBB9A8A8C2C8/output/1"
          ]
        },
        "state": "OK",
        "output-mapping":
        {
         "/2024.2/results/D34B5684-C6FF-4A7A-BF35-EBB9A8A8C2C8/output/1":
         "file:///c:/temp/test.html"
        },
        "error":
        {
        }
      }
    ]
}
```


Veuillez noter :

- Le fichier de sortie dans le dossier `output` de la tâche. Vous pouvez utiliser son URI relatif pour accéder au fichier.
- Les URI des fichiers de sortie sont relatifs à l'adresse de serveur (*voir ci-dessus* 258).
- L'item `output-mapping` mappe le document de sortie dans le répertoire de tâche sur le serveur dans l'emplacement de fichier spécifié par le client dans la requête de tâche. Veuillez noter que seuls les documents de sortie spécifiés par le client dans la requête de tâche ont un mappage ; les fichiers liés aux tâches générés par le serveur (comme des fichiers d'erreur) n'ont pas de mappage.
- En alternative, il est possible de récupérer tous les documents de résultat générés pour une tâche spécifique en tant qu'un archive zip utilisant l'URL "`/v1/results/`*`JOBID`*`/output/zip`". Cette fonction n'est pas disponible dans le mode de filesystem non limité. Veuillez noter que l'archive zip contiendra des noms de fichier endommagé, qui nécessite d'être mappé à nouveau sur le nom réel en utilisant l'objet `output-mapping`.

Document de résultat ne contenant aucun URI
Si la tâche requise s'est terminée avec un état *OK*, la tâche a retourné un résultat positif. Par exemple, une tâche de validation a retourné un résultat de document valide. Certaines tâches, comme une validation ou un test bien formé, ne produit aucun document de sortie. Si une tâche de ce type se termine avec un état des *OK*, alors le document de résultat n'aura ni l'URI d'un document de sortie ni l'URI d'un journal d'erreur. Voir la liste de JSON ci-dessous.


```
{
    "jobid": "3FC8B90E-A2E5-427B-B9E9-27CB7BB6B405",
    "state": "OK",
    "error":
    {
    },
    "jobs":
    [
      {
        "file": "file:///c:/Test/SimpleExample.xml",
        "jobid": "532F14A9-F9F8-4FED-BCDA-16A17A848FEA",
```

```
        "output":
        {
        },
        "state": "OK",
        "error":
        {
        }
      }
    ]
}
```

Veuillez noter :

- Aussi bien les composants de sortie et d'erreur de la sous-tâche dans la liste ci-dessus sont vides.
- Une tâche pourrait se terminer avec un état *OK* mais contient tout de même encore des avertissements ou d'autres messages, qui sont inscrits dans les fichiers d'erreur. Dans ces cas, le document de résultat contiendra les URI de fichiers d'erreur même si la tâche s'est terminée avec l'état *OK*.

Accéder aux documents d'erreur et de sortie listés dans le document de résultat

Les documents d'erreur et de sortie peuvent être accédés avec des requêtes `GET` HTTP. Ils sont décrits dans la section suivante, Obtenir des documents Erreur/Sortie 262.

6.1.2.4 Obtenir les documents Erreur/Message/Sortie

Un document de résultat 258 peut contenir les URI de fichier ou URI relatifs des documents d'erreur 259, documents de message (tels que les logs), et/ou documents de sortie 260. (Il existe certaines situations 261 dans laquelle un document de résultat peut ne pas contenir d'URI.) Les différents types d'URI sont décrits ci-dessous 262.

Pour accéder ces documents via HTTP, procéder comme suit :

1. Agrandir l'URI relatif 263 du fichier dans le document de résultat à son URI absolu
2. Utiliser l'URI agrandi dans une requête GET HTTP 263 pour accéder au fichier

Les URI (dans le document de résultat) de documents d'erreur/de message/ de sortie

Le document de résultat contient des URI d'erreur, de message et/ou des documents de sortie. Les documents d'erreur et de message sont des documents liés aux tâches qui sont générés par le serveur ; ils sont toujours enregistrés dans le répertoire de la tâche sur le serveur. Les documents de sortie (tels que les transformations de sortie XSLT) peuvent être enregistrés dans un des emplacements suivants :

- Dans n'importe quel emplacement de fichier accessible au serveur. Pour que les fichiers de sortie soient enregistrés dans un emplacement quelconque, le serveur doit être configuré pour permettre au client un accès non limité 233 (le paramètre par défaut).
- Dans le répertoire de tâche sur le serveur. Le serveur est configuré 231 pour limiter l'accès au client.

Si un client spécifie qu'un fichier de sortie soit créé, l'emplacement dans lequel le fichier de sortie est enregistré sera déterminé par l'option `server.unrestricted-filesystem-access` 233 du fichier de configuration du serveur.

- Si l'accès n'est pas limité, le fichier sera enregistré sur l'emplacement spécifié par le client et l'URI retourné pour le document sera un URI de fichier.
- Si l'accès est limité, le fichier sera enregistré sur le répertoire de tâche et son URI sera un URI relatif. De plus, il y aura un mappage de cet URI relatif ver l'URL de fichier spécifiée par le client. (*Voir la liste du document actuel contenant les URI des documents de sortie.*[260])

En résumé, pour cette raison les types suivants des URI seront rencontrés :

Les URI de fichier des documents d'erreur/message
Ces documents sont enregistrés dans le répertoire de la tâche sur le serveur. Les URI de fichier ont la forme suivante :

`file:///<output-root-dir>/JOBID/message.doc`

Les URI de fichier ont la forme suivante :
Ces documents sont enregistrés sous tout emplacement. Les URI de fichier ont la forme suivante :

`file:///<path-to-file>/output.doc`

URI HTTP des documents d'erreur/message/sortie
Ces documents sont enregistrés dans le répertoire de la tâche sur le serveur. Les URI sont relatifs par rapport à l'adresse du serveur et doivent être élargis à l'URI HTTP entier. La forme relative aura la forme :

`/vN/results/JOBID/error/error.txt` pour les documents de message
`/vN/results/JOBID/output/verbose.log` pour les documents de message
`/vN/results/JOBID/output/1` pour les documents de sortie

Dans le cas de documents de sortie, les mappages de sortie sont donnés (*voir la liste d'exemple*[260]). Ces mappages mappent chaque document de sortie URI dans le document de résultat du document au document correspondant dans la requête client.

Élargir l'URI relatif

Élargir l'URI relatif dans le document de résultat[258] vers une URI HTTP absolue en préfixant l'URI relatif avec l'adresse de serveur. Par exemple, si l'adresse de serveur est :

`http://localhost:8087/` (l'adresse de configuration initiale[231])

et que l'URI relatif d'un fichier d'erreur dans le document de résultat[258] est :

`/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt`

alors l'adresse absolue agrandie sera

`http://localhost:8087/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt`

Pour plus d'informations, voir les sections : Configurer le serveur[231] et Obtenir le document de résultat[258].

Utiliser une requête `GET` HTTP pour accéder au fichier

Utiliser l'URI agrandie dans une requête `GET` HTTP pour obtenir le fichier requis. RaptorXML Server retourne le document requis.

6.1.2.5 Libérer des ressources de serveur après le traitement

RaptorXML Server garde le fichier de document de résultat, des fichiers temporaires et des fichiers de document de sortie liés à une tâche traitée sur disque dur. Ces fichiers peuvent être supprimés d'une des deux manières :

- En fournissant le document de résultat d'URI[259] avec la méthode `DELETE` HTTP. Cela supprime tous les fichiers liés à la tâche indiquée par l'URI de document de résultat soumis, y compris des documents d'erreur et de sortie.
- Suppression manuelle des fichiers individuels sur le serveur par un administrateur.

La structure de l'URI à utiliser avec la méthode HTTP `DELETE` est affichée ci-dessous. Notez que l'URI complet consiste en l'adresse de serveur plus l'URI relatif du document de résultat.

Méthode HTTP	URI
SUPPRIMER	http://localhost:8087/v1/result/D405A84A-AB96-482A-96E7-4399885FAB0F

Pour situer le répertoire d'une tâche sur un disque, construire l'URI comme suit :

[<server.output-root-dir> *voir fichier de configuration de serveur*[232]] + [jobid[258]]

Note : Puisqu'un grand nombre de fichiers de document d'erreur et de sortie, il est recommandé de surveiller l'utilisation du disque dur et de programmer les suppressions conformément à votre environnement et vos exigences.

6.1.3 C# Example for REST API

Votre installation RaptorXML Server contient un projet C# qui accède à l'interface client REST de RaptorXML Server pour exécuter un ensemble de tâches. L'exemple de projet est constitué de deux parties :

- **`RaptorXMLREST.cs`** : Une classe wrapper dans C# qui met en œuvre le mécanisme REST pour communiquer avec RaptorXML Server via HTTP.
- **`Program.cs`** : Le code de programme C# qui définit les tâches à envoyer à RaptorXML Server par le biais du wrapper REST.

Ces deux parties sont décrites dans les sous-sections de cette section : C# Wrapper pour API REST[265] et Code de programme pour requêtes REST[266].

Veuillez noter que vous pouvez utiliser tout wrapper REST adapté pour le code C#. La raison principale pour laquelle nous avons créé notre propre wrapper est pour pouvoir intégrer le code de programme C# plus étroitement dans la classe de wrapper, facilitant ainsi la compréhension de l'interface REST de RaptorXML Server.

Emplacement et utilisation de l'exemple C#

Cet exemple de projet est situé dans le dossier `C:\Program Files (x86)\Altova\RaptorXML Server2024\examples\REST_API\C#_RaptorREST_API`.

L'exemple de projet a été créé en utilisant Visual Studio 2019, donc vous devriez utiliser cette version ou ultérieure et exécuter le projet. Veuillez noter que les fichiers d'exemple C# sont situés dans le dossier des fichiers de programme, donc vous allez devoir ouvrir Visual Studio avec des droits d'administrateur afin d'accéder les fichiers. En alternative, vous pouvez copier l'exemple de projet vers un autre emplacement et apporter des amendements pertinents au projet.

6.1.3.1 C# Wrapper for REST API

La classe wrapper est définie dans le nom de fichier C# **`RaptorXMLREST.cs`**, et est dénommée **`RaptorXMLRESTAPI`**.

Elle définit les classes clés suivantes pour envoyer les requêtes HTTP et recevoir les réponses HTTP via REST :

- `Commande`
- `MultiPartCommand`
- `CommandResponse`
- `ResultDocument`

Elle définit les fonctions suivantes :

- `pollCommandResult`
- `fetchCommandResult`
- `sendRequest`
- `cleanupResults`

Pour voir comment le wrapper met en œuvre l'API REST, lisez la section Requêtes Client 240 pour comprendre comment fonctionne l'API REST. Ensuite, vous pouvez lire le code C# de la classe wrapper pour voir comment le wrapper met en œuvre le code C# pour l'API REST.

Par exemple, si vous voulez voir comment une commande est envoyée à RaptorXML Server depuis le code C#, vous pourriez faire la chose suivante :

- L'interface REST permet d'envoyer une commande à RaptorXML Server par le biais de la requête **`POST`** HTTP. Ce mécanisme est décrit au chapitre Initier des tâches avec POST 242.
- La prochaine question est de savoir : Comment le wrapper passerait-il la commande à l'API REST ? Le mécanisme à cette fin est défini dans la classe **`Commande`** du wrapper. Ouvrez le fichier **`RaptorXMLREST.cs`** pour voir le code de la classe **`Commande`**.
- Ensuite, pour voir comment le code de programme 266 instancie la classe **`Commande`** du wrapper, voir le code des trois tâches dans le code de programme 266.

6.1.3.2 Program Code for REST Requests

Le code de programme C# contenant des tâches pour RaptorXML Server est défini dans le fichier C# dénommé **Program.cs**. Le code utilise les classes définies dans le Wrapper C# pour API REST[265] pour créer les requêtes REST qui sont envoyées à RaptorXML Server.

Dans le code de programme, il existe trois cases d'utilisation pour démontrer comment utiliser l'API REST de RaptorXML Server :

- Validation d'un fichier XML référencé[266] avec la commande **valany**[181] de RaptorXML Server. Le fichier de schéma est référencé depuis l'intérieur du fichier XML et n'a pas besoin d'être fourni comme argument de la commande.
- Deux fichiers XML sont validés[266] utilisant la commande **valany**[181] de RaptorXML Server. Les deux fichiers XML, tout comme le fichier schéma utilisé pour la validation, sont chargés avec la commande en tant que string en pièces jointes. Le résultat des validations est retourné ensemble une fois que les deux validations ont été terminées.
- Un fichier XML est téléchargé et transformé par un fichier XSLT[267]. Les deux fichiers sont téléchargés par le biais de REST. La commande utilisée est **xslt**[121] de RaptorXML Server. Le document résultant de la transformation est extrait par le programme.

Le code pour ces trois cas d'utilisation est discuté plus en détail ci-dessous.

Gestion d'erreur

Si une erreur est retournée, une fonction de gestion d'erreur (dénommée **HandleError**) en bas du code extrait le message d'erreur depuis la réponse du serveur[256].

Cas 1 : Valider un fichier XML référencé (commande simple)

Le code de programme pour ce cas utilise les classes et fonctions depuis le wrapper API REST pour définir et exécuter la communication HTTP avec RaptorXML Server. La logique du code est comme suit :

RaptorXMLRESTAPI.Command	Spécifie la commande de RaptorXML Server à appeler, qui est **valany**, et le fichier à soumettre comme l'argument de la commande `valany`.
RaptorXMLRESTAPI.CommandResponse	Met la réponse du serveur de la requête de validation dans la variable **jsonResponse**. Veuillez noter que les tâches de validation sont rapportées comme «OK» ou «Échec»[258].
RaptorXMLRESTAPI.ResultDocument	Récupère le document de résultat retourné par le serveur et, s'il n'y a pas d'erreurs, affiche les résultats de validation.

Cas 2 : Valider deux fichiers XML téléchargés par rapport à un XSD téléchargé (commande multipartie)

Le code de programme pour ce cas utilise la classe **MultiPartCommand** du wrapper API REST pour définir et exécuter la communication HTTP avec RaptorXML Server. Puisque nous voulons charger des fichiers au sein de la requête de `POST`, l'en-tête de message doit avoir son set de type de contenu défini comme

multipart/form-data[242]. La classe de wrapper **MultiPartCommand** est utilisée pour définir la communication HTTP REST en conséquence. Le code pour ce cas d'utilisation est organisé comme suit :

RaptorXMLRESTAPI.MultiPartCommand	Spécifie la commande RaptorXML Server à appeler, qui est **valany**, et utilise ensuite la fonction **AppendAttachment** de la classe pour télécharger les deux fichiers XML et le fichier de schéma. Les fichiers sont soumis en tant que strings. La réponse du serveur retourne le résultat de validation des deux fichiers et sa réponse est stockée dans la variable **jsonResponse**
RaptorXMLRESTAPI.fetchCommandResult	Récupère le document de résultat retourné par le serveur et, s'il n'y a pas d'erreurs, affiche les résultats de validation.
RaptorXMLRESTAPI.cleanupResults	Cette fonction de wrapper utilise la méthode **DELETE** de l'HTTP pour supprimer le fichier de document du résultat, les fichiers temporaires et les fichiers de document de sortie et d'erreur liés à la tâche.

Cas 3 : Transformation XSLT pour XML et XSLT téléchargés (commande multipartie)

Le code de programme pour ce cas est similaire à celui du cas 2 ci-dessus. Il utilise la classe **MultiPartCommand** pour définir la transformation XSLT et afficher le document de résultat dans une zone de message. Les fichiers XML et XSLT pour la transformation sont téléchargés avec la requête. De plus, la commande **XSLT** de RaptorXML Server prend également des options, donc ce cas vous montre comment vous pourriez ajouter des options par le biais de l'interface REST (dans l'exemple, ceci est fait avec la fonction **RaptorXMLRESTAPI.AppendOption**. Les points importants en ce qui concerne le code sont données ci-dessous.

RaptorXMLRESTAPI.MultiPartCommand	Spécifie la commande de RaptorXML Server à appeler, qui est **XSLT**, et utilise ensuite (i) la fonction **AppendAttachment** de la classe pour télécharger les fichiers XML et XSLT, et (ii) la fonction **AppendOption** pour fournir des options pour la ligne de commande RaptorXML Server. Les fichiers téléchargés sont soumis en tant que strings. La réponse du serveur retourne le résultat de validation des deux fichiers et sa réponse est stockée dans la variable **jsonResponse**
RaptorXMLRESTAPI.fetchCommandResult	Récupère le document de résultat retourné par le serveur et, s'il n'y a pas d'erreurs, affiche les résultats de validation.
RaptorXMLRESTAPI.cleanupResults	Cette fonction de wrapper utilise la méthode **DELETE** de l'HTTP pour nettoyer le fichier du document de résultat, les fichiers temporaires et les fichiers de document de sortie et d'erreur liés à la tâche.

6.2 API COM/.NET

RaptorXML est mis sous licence sur l'appareil sur lequel il est installé. L'interface .NET est construite en tant que wrapper autour de l'interface COM. Les interfaces COM et .NET de RaptorXML Server utilisent une simple API: COM/.NET API de RaptorXML Server (référence d'objet ici[277]).

Vous pouvez utiliser RaptorXML Server avec :

- des langages de script, comme JavaScript, par le biais de l'interface COM
- des langages de programmation, comme C#, par le biais de l'interface .NET Framework

6.2.1 Interface COM

RaptorXML Server est automatiquement enregistrée en tant qu'objet de serveur COM lorsque RaptorXML Server est installé. Il peut donc être invoqué depuis les applications et les langages de script qui proposent une prise en charge de la programmation pour les appels COM. Si vous souhaitez changer l'emplacement du package d'installation RaptorXML Server, il vaut mieux désinstaller RaptorXML Server puis le réinstaller dans l'emplacement requis. De cette manière, la désinscription et l'enregistrement nécessaires sont effectués par le processus de l'installateur.

Contrôler le succès de l'inscription

Si l'enregistrement a réussi, le Registre contiendra les classes `RaptorXML.Server`. Ces classes se trouvent généralement sous `HKEY_LOCAL_MACHINE\SOFTWARE\Classes`.

Exemples de code

- Un exemple de VBScript[268] montrant comment l'API RaptorXML peut être utilisée par le biais de son interface COM est listé dans la section suivante
- Un fichier d'exemple correspondant à cette liste est disponible dans le dossier `examples/API` du dossier d'application RaptorXML.

6.2.2 Exemple COM : VBScript

L'exemple VBScript ci-dessous est structuré dans les parties suivantes :

- Configurer et initialiser l'objet COM RaptorXML[268]
- Valider un fichier XML[269]
- Effectuer une transformation XSLT, retourner le résultat en tant que string[269]
- Traiter un document XQuery, enregistrer le résultat dans un fichier[270]
- Configurer la séquence d'exécution du script et de son point d'entrée[270]

```
' The RaptorXML COM object
dim objRaptor
```



```
' Initialize the RaptorXML COM object
sub Init
	objRaptor = Null
	On Error Resume Next
	' Try to load the 32-bit COM object; do not throw exceptions if object is not found
	Set objRaptor = WScript.GetObject( "", "RaptorXML.Server" )
	On Error Goto 0
	if ( IsNull( objRaptor ) ) then
		' Try to load the 64-bit object (exception will be thrown if not found)
		Set objRaptor = WScript.GetObject( "", "RaptorXML_x64.Server" )
	end if
	' Configure the server: error reporting, HTTP server name and port (IPv6 localhost
in this example)
	objRaptor.ErrorLimit = 1
	objRaptor.ReportOptionalWarnings = true
	objRaptor.ServerName = "::1"
	objRaptor.ServerPort = 8087
end sub


' Validate one file
sub ValidateXML
	' Get a validator instance from the Server object
	dim objXMLValidator
	Set objXMLValidator = objRaptor.GetXMLValidator()

	' Configure input data
	objXMLValidator.InputFileName = "MyXMLFile.xml"

	' Validate; in case of invalid file report the problem returned by RaptorXML
	if ( objXMLValidator.IsValid() ) then
		MsgBox( "Input string is valid" )
	else
		MsgBox( objXMLValidator.LastErrorMessage )
	end if
end sub


' Perform a transformation; return the result as a string
sub RunXSLT
	' Get an XSLT engine instance from the Server object
	dim objXSLT
	set objXSLT = objRaptor.GetXSLT

	' Configure input data
	objXSLT.InputXMLFileName = "MyXMLFile.xml"
	objXSLT.XSLFileName = "MyTransformation.xsl"

	' Run the transformation; in case of success the result will be returned, in case of
errors the engine returns an error listing
	MsgBox( objXSLT.ExecuteAndGetResultAsString() )
end sub
```

```
' Execute an XQuery; save the result in a file
sub RunXQuery
	' Get an XQuery engine instance from the Server object
	dim objXQ
	set objXQ = objRaptor.GetXQuery()

	' Configure input data
	objXQ.InputXMLFileName = "MyXMLFile.xml"
	objXQ.XQueryFileName = "MyQuery.xq"

	' Configure serialization (optional - for fine-tuning the result's formatting)
	objXQ.OutputEncoding = "UTF8"
	objXQ.OutputIndent = true
	objXQ.OutputMethod = "xml"
	objXQ.OutputOmitXMLDeclaration = false

	' Run the query; the result will be serialized to the given path
	call objXQ.Execute( "MyQueryResult.xml" )
end sub


' Perform all sample functions
sub main
	Init
	ValidateXML
	RunXSLT
	RunXQuery
end sub

' Script entry point; run the main function
main
```

6.2.3 Interface .NET

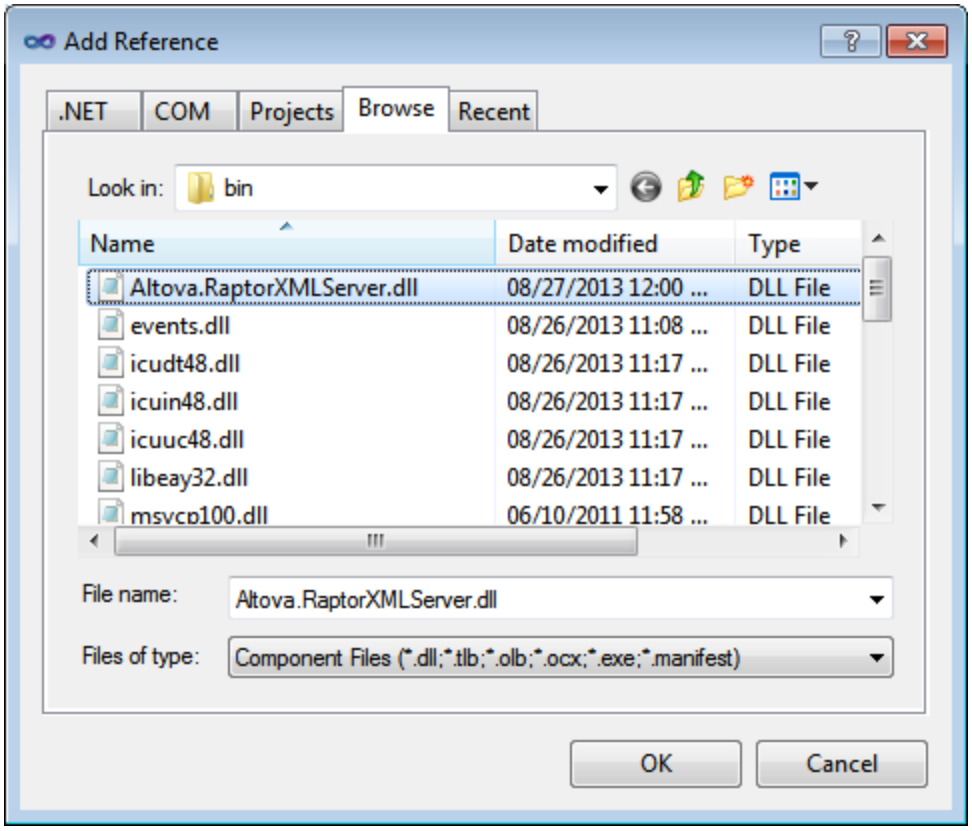
L'interface .NET est construite en tant qu'un wrapper autour de l'interface COM de RaptorXML Server. Elle est fournie en tant qu'assembly interop primaire signée par Altova ; elle utilise l'espace de noms `Altova.RaptorXMLServer`.

Ajouter la DLL RaptorXML en tant qu'une référence à un projet .NET Visual Studio

Pour pouvoir utiliser RaptorXML Server dans votre projet .NET, ajouter une référence à la DLL RaptorXML (`Altova.RaptorXMLServer.dll`) à votre projet. Votre installation RaptorXML Server contient un fichier DLL signé, nommé `Altova.RaptorXMLServer.dll`. Ce fichier DLL sera ajouté automatiquement au cache d'assembly global (GAC) lorsque RaptorXML Server est installé à l'aide de l'installateur RaptorXML Server. Le GAC se trouve généralement dans le dossier : `C:\WINDOWS\assembly`.

Pour ajouter la DLL de RaptorXML en tant que référence dans un projet .NET, procéder comme suit :

1. Lorsque le projet .NET est ouvert, cliquer sur **Project | Add reference**. Le dialogue Ajouter Référence (*capture d'écran ci-dessous*) s'affiche.



2. Dans l'onglet Navigateur, se rendre dans le dossier : `<RaptorXML application folder>/bin`, choisir la DLL RaptorXML `Altova.RaptorXMLServer.dll`, et cliquer sur **OK**.
3. Choisir la commande View | Object Browser pour voir les objets de l'API de RaptorXML.

Une fois que `Altova.RaptorXMLServer.dll` est disponible sur l'interface .NET et que RaptorXML a été enregistré en tant qu'objet de serveur COM, la fonction RaptorXML sera disponible dans votre projet .NET.

Note : RaptorXML sera enregistré automatiquement en tant qu'objet de serveur COM pendant l'installation. Un enregistrement manuel n'est pas nécessaire.

Note : Si vous recevez une erreur d'accès, contrôlez que les permissions sont correctement définies. Aller au Component Services et accorder les permissions au même compte qui exécute le pool d'applications contenant RaptorXML.

Exemples de code

Un exemple C# et un exemple .NET Visual Basic[272] montrant comment l'API RaptorXML peut être utilisée par le biais de son interface .NET sont listés dans les rubriques suivantes. Les fichiers correspondant à ces listes sont disponibles dans le dossier `examples/serverAPI` du dossier d'application RaptorXML.

6.2.4 Exemple .NET : Visual Basic .NET

L'exemple Visual Basic ci-dessous effectue les choses suivantes :

- Configurer et initialiser l'objet .NET RaptorXML 272
- Valider un fichier XML 272
- Effectuer une transformation XSLT, retourner le résultat en tant que string 272
- Traiter un document XQuery, enregistrer le résultat dans un fichier 273
- Configurer la séquence d'exécution du code et de son point d'entrée 273

```
Option Explicit On
Imports Altova.RaptorXMLServer

Module RaptorXMLRunner

   ' The RaptorXML .NET object
     Dim objRaptor As Server

   ' Initialize the RaptorXML .NET object
     Sub Init()

       ' Allocate a RaptorXML object
         objRaptor = New Server()

       ' Configure the server: error reporting, HTTP server name and port (IPv6 localhost in
this example)
         objRaptor.ErrorLimit = 1
         objRaptor.ReportOptionalWarnings = True
         objRaptor.ServerName = "::1"
         objRaptor.ServerPort = 8087
     End Sub


   ' Validate one file
     Sub ValidateXML()

       ' Get a validator instance from the RaptorXML object
         Dim objXMLValidator As XMLValidator
         objXMLValidator = objRaptor.GetXMLValidator()

       ' Configure input data
         objXMLValidator.InputFileName = "MyXMLFile.xml"

       ' Validate; in case of invalid file report the problem returned by RaptorXML
         If (objXMLValidator.IsValid()) Then
            Console.WriteLine("Input string is valid")
         Else
            Console.WriteLine(objXMLValidator.LastErrorMessage)
         End If
     End Sub


   ' Perform a transformation; return the result as a string
```



```
    Sub RunXSLT()

      ' Get an XSLT engine instance from the Server object
        Dim objXSLT As XSLT
        objXSLT = objRaptor.GetXSLT()

      ' Configure input data
        objXSLT.InputXMLFileName = "MyXMLFile.xml"
        objXSLT.XSLFileName = "MyTransformation.xsl"

      ' Run the transformation; in case of success the result will be returned, in case of
errors the engine returns an error listing
        Console.WriteLine(objXSLT.ExecuteAndGetResultAsString())
    End Sub


  ' Execute an XQuery; save the result in a file
    Sub RunXQuery()

      ' Get an XQuery engine instance from the Server object
        Dim objXQ As XQuery
        objXQ = objRaptor.GetXQuery()

      ' Configure input data
        objXQ.InputXMLFileName = "MyXMLFile.xml"
        objXQ.XQueryFileName = "MyQuery.xq"

      ' Configure serialization (optional - for fine-tuning the result's formatting)
        objXQ.OutputEncoding = "UTF8"
        objXQ.OutputIndent = true
        objXQ.OutputMethod = "xml"
        objXQ.OutputOmitXMLDeclaration = false

      ' Run the query; the result will be serialized to the given path
        objXQ.Execute( "MyQueryResult.xml" )
    End Sub


    Sub Main()
      ' Entry point; perform all sample functions
        Init()
        ValidateXML()
        RunXSLT()
        RunXQuery()
    End Sub

End Module
```

6.3 API Java

L'API de RaptorXML Server peut être accédée depuis le code Java. Pour accéder à RaptorXML Server depuis le code Java, les bibliothèques listées ci-dessous doivent résider dans le classpath. Ces bibliothèques sont installées dans le dossier `bin` du dossier d'installation.

- `RaptorXMLServer.jar`: La bibliothèque qui communique avec le serveur RaptorXML en utilisant les requêtes HTTP
- `RaptorXMLServer_JavaDoc.zip`: un fichier Javadoc contenant la documentation d'aide pour l'API Java

Note : Pour pouvoir utiliser l'API Java, le fichier Jar doit se trouver sur le Java Classpath. Vous pouvez copier le fichier Jar dans un des emplacements si cela convient mieux à votre configuration de projet que de le référencer depuis l'emplacement installé.

6.3.1 Aperçu de l'interface

L'API Java est contenue dans le package **`com.altova.raptorxml`**. La classe **`RaptorXML`** permet une méthode de point d'entrée appelée **`getFactory()`**, qui retourne des objets `RaptorXMLFactory`[277]. Donc, une instance `RaptorXMLFactory` peut être créée avec l'appel : `RaptorXML.getFactory()`.

L'interface `RaptorXMLFactory`[277] constitue une méthode pour obtenir des objets de moteur pour validation et d'autres fonctions de traitement (comme une transformation XSLT).

RaptorXMLFactory

L'interface publique de `RaptorXMLFactory`[277] est décrite dans la liste suivante :

```
public interface RaptorXMLFactory
{
   public XMLValidator[295] getXMLValidator();
    public XMLDSig[287] getXMLDSig();
   public XQuery[307] getXQuery();
   public XSLT[319] getXSLT();
   public void setServerName(String name) throws RaptorXMLException[287];
   public void setServerPath(String path) throws RaptorXMLException[287];
   public void setServerPort(int port) throws RaptorXMLException[287];
   public void setGlobalCatalog(String catalog);
   public void setUserCatalog(String catalog);
   public void setGlobalResourcesFile(String file);
   public void setGlobalResourceConfig(String config);
   public void setErrorFormat(RaptorXMLException[287] format);
   public void setErrorLimit(int limit);
   public void setReportOptionalWarnings(boolean report);
}
```

Pour plus de détails, voir les descriptions de `RaptorXMLFactory`[277] et les méthodes Java respectives. Voir aussi le Projet d'exemple Java[275].

6.3.2 Projet d'exemple Java

Le code Java listé ci-dessous montre comment accéder aux fonctions de base. Il est structuré dans les différentes parties suivantes :

- Situer le dossier d'exemple, et créer une instance d'objet COM RaptorXML 275
- Valider un fichier XML 275
- Effectuer une transformation XSLT, retourner le résultat en un string 275
- Traiter un document XQuery, retourner le résultat en un string 276
- Exécuter le projet 276

Cette fonction de base est incluse dans les fichiers dans le dossier `examples/API` du dossier d'application RaptorXML Server.

```
public class RunRaptorXML
{
    // Locate samples installed with the product
    // (will be two levels higher from examples/API/Java)
    // REMARK: You might need to modify this path
       static final String strExamplesFolder = System.getProperty("user.dir") + "/../../" ;

       static com.altova.raptorxml.RaptorXMLFactory rxml;



       static void ValidateXML() throws com.altova.raptorxml.RaptorXMLException
       {
           com.altova.raptorxml.XMLValidator xmlValidator = rxml.getXMLValidator();
           System.out.println("RaptorXML Java - XML validation");
           xmlValidator.setInputFromText( "<!DOCTYPE root [ <!ELEMENT root (#PCDATA)> ]>
<root>simple input document</root>" );
              if( xmlValidator.isWellFormed() )
                  System.out.println( "The input string is well-formed" );
              else
                  System.out.println( "Input string is not well-formed: " +
xmlValidator.getLastErrorMessage() );

              if( xmlValidator.isValid() )
                  System.out.println( "The input string is valid" );
              else
                  System.out.println( "Input string is not valid: " +
xmlValidator.getLastErrorMessage() );
       }



       static void RunXSLT() throws com.altova.raptorxml.RaptorXMLException
       {
           System.out.println("RaptorXML Java - XSL Transformation");
           com.altova.raptorxml.XSLT xsltEngine = rxml.getXSLT();
           xsltEngine.setInputXMLFileName( strExamplesFolder + "simple.xml" );
           xsltEngine.setXSLFileName( strExamplesFolder + "transform.xsl" );
```

```
            String result = xsltEngine.executeAndGetResultAsString();
            if( result == null )
                System.out.println( "Transformation failed: " +
xsltEngine.getLastErrorMessage() );
            else
                System.out.println( "Result is " + result );
        }



        static void RunXQuery() throws com.altova.raptorxml.RaptorXMLException
        {
            System.out.println("RaptorXML Java - XQuery execution");
            com.altova.raptorxml.XQuery xqEngine = rxml.getXQuery();
            xqEngine.setInputXMLFileName( strExamplesFolder + "simple.xml" );
            xqEngine.setXQueryFileName( strExamplesFolder + "CopyInput.xq" );
            System result = xqEngine.executeAndGetResultAsString();
            if( result == null )
                System.out.println( "Execution failed: " + xqEngine.getLastErrorMessage() );
            else
                System.out.println( "Result is " + result );
        }



        public static void main(String[] args)
        {
            try
            {
                rxml = com.altova.raptorxml.RaptorXML.getFactory();
                rxml.setErrorLimit( 3 );

                ValidateXML();
                RunXSLT();
                RunXQuery();
            }

            catch( com.altova.raptorxml.RaptorXMLException e )
            {
                e.printStackTrace();
            }

        }

}
```

6.4 Server API Reference

Cette section décrit l'API de RaptorXML Server : son modèle d'objet et les détails de ses interfaces et énumérations. La description API s'applique aussi bien aux interfaces COM/.NET et Java. Alors que la structure de l'API est la même pour les deux interfaces, les noms des méthodes et des propriétés sont différents. C'est pourquoi, chaque méthode, propriété et énumération est décrite avec une signature séparée pour COM/.NET et Java.

Le point de départ pour utiliser la fonction de RaptorXML Server est l'interface `IServer` 277 (COM/.NET) ou la classe `RaptorXMLFactory` 277 (Java).

Le point de départ pour utiliser la fonction de RaptorXML Server est l'interface `IServer` 277 (COM/.NET) ou la classe `RaptorXMLFactory` 277 (Java). Cet objet contient les objets qui fournissent la fonction RaptorXML : validation XML, traitement de document XQuery et XML Signature, et transformations XSLT.

La hiérarchie du model d'objet est affiché ci-dessous, et les interfaces sont décrites en détail dans les sections correspondantes. Les méthodes et les propriétés de chaque interface sont décrites dans la section pour cette interface.

```
IServer (COM/.NET) / RaptorXMLFactory (Java)
  |-- IXMLDSig(COM/.NET) / XMLDSig (Java)
  |-- IXMLValidator (COM/.NET) / XMLValidator (Java)
  |-- IXSLT (COM/.NET) / XSLT (Java)
  |-- IXQuery (COM/.NET) / XQuery (Java)
```

6.4.1 Interfaces/Classes

Le point de départ pour utiliser les fonctions de RaptorXML est l'interface `IServer` 277 (COM/.NET) ou la classe `RaptorXMLFactory` 277 (Java). Cet objet contient les objets qui proposent les fonctions RaptorXML : validation XML, document XQuery et traitement de XML Signature et transformations XSLT.

La hiérarchie du modèle d'objet est affichée ci-dessous et les interfaces sont décrites en détail dans les sections correspondantes. Les méthodes et les propriétés de chaque interface sont décrites dans la section de cette interface.

```
IServer (COM/.NET) / RaptorXMLFactory (Java)
  |-- IXMLDSig(COM/.NET) / XMLDSig (Java)
  |-- IXMLValidator (COM/.NET) / XMLValidator (Java)
  |-- IXSLT (COM/.NET) / XSLT (Java)
  |-- IXQuery (COM/.NET) / XQuery (Java)
```

6.4.1.1 IServer/RaptorXMLFactory

Utiliser l'interface **`IServer/RaptorXMLFactory`** pour accéder au moteur RaptorXML que vous souhaitez. Veuillez noter que le nom de l'interface dans l'API COM/.NET est différent de celui contenu dans l'interface de l'API Java :

- Dans COM/.NET : **IServer**
- Dans Java : **RaptorXMLFactory**

Les méthodes et les propriétés de **IServer/RaptorXMLFactory** sont décrites dans cette section.

Méthode de point d'entrée API Java

L'API Java est contenue dans le package **com.altova.raptorxml**. La classe **RaptorXML** permet une méthode de point d'entrée appelée **getFactory()**, qui retourne des objets RaptorXMLFactory[277]. Donc, une instance RaptorXMLFactory peut être créée avec l'appel : RaptorXML.getFactory().

6.4.1.1.1 Méthodes

Les méthodes des interfaces **IServer** (COM/.NET) et **RaptorXMLFactory** (Java) retournent une instance du moteur RaptorXML ou de la classe respectif : XMLDSig, XML Validator, XSLT et XQuery.

COM/.NET	Java
GetXMLDsig[278] *(pour les Signatures XML)*	getXMLDsig[278] *(pour les Signatures XML)*
GetXMLValidator[279]	getXMLValidator[279]
GetXQuery[279]	getXQuery[279]
GetXSLT[279]	getXSLT[279]

6.4.1.1.1.1 GetXMLDsig (for XML Signatures)

Retourne une instance de l'interface/classe de la Signature XML (**XMLDSig**[287])

COM et .NET

Signature: **IXMLDSig**[287] **GetXMLDSig()**

Java

Signature: **public XMLDSig**[287] **getXMLDSig()**

6.4.1.1.1.2 GetXMLValidator

Retourne une instance du moteur XML Validator.

COM et .NET

Signature: **IXMLValidator**[295] **GetXMLValidator()**

Java

Signature: **public XMLValidator**[295] **getXMLValidator()**

6.4.1.1.1.3 GetXQuery

Retourne une instance du moteur XQuery.

COM et .NET

Signature: **IXQuery**[307] **GetXQuery()**

Java

Signature: **public XQuery**[307] **getXQuery()**

6.4.1.1.1.4 GetXSLT

Retourne une instance du moteur XSLT.

COM et .NET

Signature: **IXSLT**[319] **GetXSLT()**

Java

Signature: **public XSLT**[319] **getXSLT()**

6.4.1.1.2 Propriétés

Les propriétés des interfaces **IServerIServer** (COM/.NET) et **RaptorXMLFactory** (Java) sont décrites dans cette section.

COM/.NET	Java
APIMajorVersion 280	getAPIMajorVersion 280
APIMinorVersion 281	getAPIMinorVersion 281
APIServicePackVersion 281	getAPIServicePackVersion 281
ErrorFormat 281	setErrorFormat 281
ErrorLimit 282	setErrorLimit 282
GlobalCatalog 282	setGlobalCatalog 282
GlobalResourceConfig 282	setGlobalResourceConfig 282
GlobalResourcesFile 283	setGlobalResourcesFile 283
Is64Bit 283	ss64Bit 283
MajorVersion 283	getMajorVersion 283
MinorVersion 284	getMinorVersion 284
ProductName 284	getProductName 284
ProductNameAndVersion 284	getProductNameAndVersion 284
ReportOptionalWarnings 285	setReportOptionalWarnings 285
ServerName 285	setServerName 285
ServerPath 285	setServerPath 285
ServerPort 286	setServerPort 286
ServicePackVersion 286	getServicePackVersion 286
UserCatalog 286	setUserCatalog 286

6.4.1.1.2.1 APIMajorVersion

Retourne la version majeure de l'API en tant qu'un entier. La version majeure de l'API peut différer de la version majeure du produit 283 si l'API est connectée à un autre serveur.

COM et .NET

Signature: `int APIMajorVersion()`

Java

Signature: `public int getAPIMajorVersion()`

6.4.1.1.2.2 APIMinorVersion

Retourne la version mineure de l'API en tant qu'un entier. La version mineure de l'API peut différer de la version mineure du produit [284] si l'API est connectée à un autre serveur.

COM et .NET

Signature: `int APIMinorVersion()`

Java

Signature: `public int getAPIMinorVersion()`

6.4.1.1.2.3 APIServicePackVersion

Retourne la version du pack de service de l'API en tant qu'un entier. La version du pack de service de l'API peut différer de la version du pack de service du produit [286] si l'API est connectée à un autre serveur.

COM et .NET

Signature: `int APIServicePackVersion()`

Java

Signature: `public int getAPIServicePackVersion()`

6.4.1.1.2.4 ErrorFormat

Définit le formant d'erreur RaptorXML sur un des littéraux `ENUMErrorFormat` (Text, ShortXML, LongXML).

COM et .NET

Signature: `ErrorFormat(ENUMErrorFormat [331] format)`

Java

Signature: `public void setErrorFormat(ENUMErrorFormat format)`

6.4.1.1.2.5 ErrorLimit

Définit la limite d'erreur de validation de RaptorXML. Le paramètre `limit` est de type `int` (Java), `uint` (COM/.NET). Il spécifie le nombre d'erreurs à rapporter avant que l'exécution soit stoppée. Utiliser `-1` pour que `limit` ne soit pas limité (c'est à dire que toutes les erreurs seront rapportées). La valeur par défaut est `100`.

COM et .NET

Signature: `ErrorLimit(uint limit)`

Java

Signature: `public int setErrorLimit(int limit)`

6.4.1.1.2.6 GlobalCatalog

Définit l'emplacement, en tant qu'une URL, du fichier de catalogue (point-entrée) principal. Le string fourni doit être une URL absolue qui donne l'emplacement exact du fichier de catalogue principal à utiliser.

COM et .NET

Signature: `GlobalCatalog(string catalog)`

Java

Signature: `public void setGlobalCatalog(string catalog)`

6.4.1.1.2.7 GlobalResourceConfig

Définit la configuration active de la ressource globale. Le paramètre `config` est de type `String`, et spécifie le nom de la configuration utilisée par la ressource globale active.

COM et .NET

Signature: `GlobalResourceConfig(string config)`

Java

Signature: `public void setGlobalResourceConfig(string config)`

6.4.1.1.2.8 GlobalResourcesFile

Définit l'emplacement, en tant qu'une URL, du fichier XML de Ressources globales. Le string fourni doit être une URL absolue qui donne l'emplacement exact du fichier XML de Ressources globales.

COM et .NET

Signature: `GlobalResourcesFile(string url)`

Java

Signature: `public void setGlobalResourcesFile(string url)`

6.4.1.1.2.9 Is64Bit

Vérifie si l'application est un exécutable 64-bit. Retourne une booléenne `true` si l'application est 64 bit, `false` si ce n'est pas le cas. *Exemple :* Pour `Altova RaptorXML Server 2024r2sp1(x64)`, retourne `true`. Si une erreur se produit, une **RaptorXMLException** [287] est soulevée.

COM et .NET

Signature: `boolean Is64Bit()`

Java

Signature: `public boolean is64Bit()`

6.4.1.1.2.10 MajorVersion

Retourne la version majeure du produit en tant qu'un entier. *Exemple :* Pour `Altova RaptorXML Server 2014r2sp1(x64)`, retourne `16` (la différence entre la version majeure (`2014`) et l'année initiale `1998`). Si une erreur se produit, une **RaptorXMLException** [287] est soulevée.

COM et .NET

Signature: `int MajorVersion()`

Java

Signature: `public int getMajorVersion()`

6.4.1.1.2.11 MinorVersion

Retourne la version mineure du produit en tant qu'un entier. *Exemple :* Pour `Altova RaptorXML Server 2024r2sp1(x64)`, retourne `2` (depuis la version mineure `r2`). Si une erreur se produit, une **RaptorXMLException** 287 est soulevée.

COM et .NET

Signature: `int MinorVersion()`

Java

Signature: `public int getMinorVersion()`

6.4.1.1.2.12 ProductName

Retourne le nom du produit en tant que string. *Exemple :* Pour `Altova RaptorXML Server 2024r2sp1(x64)`, retourne `Altova RaptorXML Server`. Si une erreur se produit, une **RaptorXMLException** 287 est soulevée.

COM et .NET

Signature: `string ProductName()`

Java

Signature: `public string getProductName()`

6.4.1.1.2.13 ProductNameAndVersion

Retourne la version de pack de service du produit en tant qu'un entier. *Exemple :* pour `Altova RaptorXML Server 2024r2sp1(x64)`, retourne `Altova RaptorXML Server 2024r2sp1(x64)`. Si une erreur se produit, une **RaptorXMLException** 287 est soulevée.

COM et .NET

Signature: `string ProductNameAndVersion()`

Java

Signature: `public string getProductNameAndVersion()`

6.4.1.1.2.14 ReportOptionalWarnings

Active/désactive le rapport des avertissements. Une valeur de `true` active les avertissements ; `false` les désactive.

COM et .NET

Signature: `ReportOptionalWarnings(boolean report)`

Java

Signature: `public void setReportOptionalWarnings(boolean report)`

6.4.1.1.2.15 ServerName

Définit le nom du serveur HTTPà partir duquel la connexion à RaptorXML Server sera établie. Le paramètre d'entrée est un string qui donne le nom du serveur HTTP. Si une erreur se produit, une **RaptorXMLException** 287 est soulevée.

COM et .NET

Signature: `ServerName(string name)`

Java

Signature: `public void setServerName(string name)`

6.4.1.1.2.16 ServerPath

Spécifie, sous la forme d'une URL, le chemin vers le serveur HTTP.

COM et .NET

Signature: `ServerPath(string path)`

Java

Signature: `public void setServerPath(string path)`

6.4.1.1.2.17 ServerPort

Définit le port sur le serveur HTTP par le biais duquel le service est accédé. Le port doit être fixé et connu de manière à ce que les requêtes HTTP puissent être adressées correctement au service. Le paramètre d'entrée est un entier qui spécifie le port d'accès sur le serveur HTTP. Si une erreur se produit, une **RaptorXMLException** 287 est soulevée.

COM et .NET

Signature: `ServerPort(int port)`

Java

Signature: `public void setServerPort(int port)`

6.4.1.1.2.18 ServicePackVersion

Retourne la version du pack de service du produit en tant qu'un entier. *Exemple :* pour `RaptorXML Server 2024r2sp1(x64)`, retourne `1` (du pack de service du numéro de version `sp1`). Si une erreur se produit, une **RaptorXMLException** 287 est soulevée.

COM et .NET

Signature: `int ServicePackVersion()`

Java

Signature: `public int getServicePackVersion()`

6.4.1.1.2.19 UserCatalog

Définit l'emplacement, en tant qu'une URL, du fichier de catalogue d'utilisateur personnalisé. Le string fourni doit être une URL absolue qui donne l'emplacement exact du fichier de catalogue d'utilisateur personnalisé.

COM et .NET

Signature: `UserCatalog(string userCatalog)`

Java

Signature: `public void setUserCatalog(string userCatalog)`

6.4.1.2 RaptorXMLException

Génère une exception qui contient des informations concernant une erreur qui se produit pendant le traitement. Le paramètre `message` fournit des informations concernant l'erreur.

COM et .NET

Signature: `RaptorXMLException(string message)`

Java

Signature: `public void RaptorXMLException(string message)`

6.4.1.3 XMLDSig (pour des signatures XML)

Les méthodes de l'interface/class de **`IXMLDSig/XMLDSig`** peuvent être utilisées pour signer les documents XML, vérifier les documents signés, mettre à jour (avec une nouvelle signature) les documents signés précédemment qui ont été modifiés et supprimer les signatures.

Veuillez noter que le nom de l'interface dans l'API COM/.NET est différent de celui de la classe de l'API Java :

- Dans COM/.NET : **`IXMLDSig`**
- Dans Java : **`XMLDSig`**

6.4.1.3.1 Méthodes

Les méthodes de l'interface **`IXMLDSig`** (COM/.NET) et de la classe **`XMLDSig`** (Java) sont décrites dans cette section.

6.4.1.3.1.1 ExecuteRemove

Supprime la signature XML du fichier XML signé, et enregistre le document non signé résultant dans un emplacement de sortie défini par `outputPath`, qui est un string qui fournit l'URL de l'emplacement de fichier. Le résultat est `true` en cas de succès, `false` en cas d'échec.

COM et .NET

Signature: `boolean ExecuteRemove(string outputPath)`

Java

Signature: `public boolean executeRemove(string outputPath)`

6.4.1.3.1.2 ExecuteSign

Signe le document XML conformément aux options de signature spécifiées (donné dans les paramètres `signatureType` et `canonicalizationMethod` ; *voir la commande CLI xmlsignature-sign*[168] *pour des valeurs disponibles*). Le fichier de sortie est défini par `outputPath`, qui est un string qui fourni l'URL du fichier de sortie. Le résultat est `true` en cas de succès, `false` en cas d'échec.

COM et .NET

Signature: `boolean ExecuteSign(string outputPath, string signatureType,string canonicalizationMethod)`

Java

Signature: `public boolean executeSign(string outputPath, string signatureType,string canonicalizationMethod)`

6.4.1.3.1.3 ExecuteUpdate

Met à jour la signature XML dans le fichier XML signé. Si le document a été modifié, la signature XML mise à jour sera différente ; sinon, la signature mise à jour sera la même que celle de la signature précédente.
Le fichier de sortie est spécifié avec `outputPath`, qui est un string qui fourni l'URL du fichier avec la signature mise à jour. Le résultat est `true` en cas de succès, `false` en cas d'échec.

Il est nécessaire de spécifier soit (i) la propriété HMAC secret key[292] ou (ii) les propriétés certificate-name[290] et certificate-store[291] doivent être spécifiées. Si les options de certificat sont spécifiées, elles doivent correspondre à celles utilisées pour signer le document XML précédemment. (Veuillez noter que l'option certificate-store n'est actuellement pas prise en charge par Linux et macOS.)

COM et .NET

Signature: `boolean ExecuteUpdate(string outputPath)`

Java

Signature: `public boolean executeUpdate(string outputPath)`

6.4.1.3.1.4 ExecuteVerify

Retourne le résultat de la vérification de la signature : **`true`** si la vérification est réussie, **`false`** sinon.

COM et .NET

Signature: `boolean ExecuteVerify()`

Java

Signature: `public boolean executeVerify()`

6.4.1.3.2 Propriétés

Les propriétés de l'interface **`IXMLDSig`** (COM/.NET) et de la classe **`XMLDSig`** (Java) sont décrites dans cette section.

6.4.1.3.2.1 AbsoluteReferenceUri

Spécifie si l'URI du document signé doit être lue en tant qu'absolu (`true`) ou relatif (`false`). La valeur par défaut est `false`.

COM et .NET

Signature: `AbsoluteReferenceUri(boolean absoluteuri)`

Java

Signature: `public void setAbsoluteReferenceUri(boolean absoluteuri)`

6.4.1.3.2.2 AppendKeyInfo

Spécifie s'il faut inclure l'élément `KeyInfo` dans la signature ou pas. La valeur par défaut est `false`.

COM et .NET

Signature: **`AppendKeyInfo(boolean include)`**

Java

Signature: **`public void setAppendKeyInfo(boolean inlude)`**

6.4.1.3.2.3 CertificateName

Le nom du certificat utilisé pour signer.

Windows
Il s'agit du nom **Sujet** d'un certificat depuis le **`--certificate-store`** sélectionné.

Exemple pour lister les certificats (sous PowerShell)

```
% ls cert://CurrentUser/My
PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My
Thumbprint Subject
---------- -------
C9DF64BB0AAF5FA73474D78B7CCFFC37C95BFC6C CN=certificate1
... CN=...
```

Exemple : **`--certificate-name==certificate1`**

Linux/MacOS
`--certname` spécifie le nom de fichier d'un certificat encodé en PEM X.509v3 avec la clé privée. Ce type de fichier a généralement l'extension **`.pem`**.

Exemple : **`--certificate-name==/path/to/certificate1.pem`**

COM et .NET

Signature: **`CertificateName(string name)`**

Java

Signature: **`public void setCertificateName(string name)`**

6.4.1.3.2.4 CertificateStore

L'emplacement dans lequel le certificat spécifié avec **--certificate-name** est stocké.

Windows
Le nom d'un stockage de certificat sous **cert://CurrentUser**. Les stockages de certificat disponibles peuvent être listés (sous PowerShell) en utilisant **% ls cert://CurrentUser/**. Les certificats seront ensuite listés comme suit :

```
Name : TrustedPublisher
Name : ClientAuthIssuer
Name : Root
Name : UserDS
Name : CA
Name : ACRS
Name : REQUEST
Name : AuthRoot
Name : MSIEHistoryJournal
Name : TrustedPeople
Name : MyCertStore
Name : Local NonRemovable Certificates
Name : SmartCardRoot
Name : Trust
Name : Disallowed
```

Exemple : **--certificate-store==MyCertStore**

Linux/MacOS
L'option **--certstore** n'est actuellement pas prise en charge.

COM et .NET

Signature: **CertificateStore(string filelocation)**

Java

Signature: **public** void **setCertificateStore(string filelocation)**

6.4.1.3.2.5 DigestMethod

L'algorithme qui est utilisé pour calculer la valeur digest sur le fichier XML d'entrée. Les valeurs disponibles sont : `sha1|sha256|sha384|sha512`.

COM et .NET

Signature: `DigestMethod(string algo)`

Java

Signature: `public void setDigestMethod(string algo)`

6.4.1.3.2.6 HMACOutputLength

Réduit la sortie de l'algorithme HMAC à `length` bits. Si spécifiée, cette valeur doit être

- un multiple de 8
- plus grand que 80
- plus grand que la moitié de la longueur de sortie de l'algorithme hash sous-jacent

COM et .NET

Signature: `HMACOutputLength(int length)`

Java

Signature: `public void setHMACOutputLength(int length)`

6.4.1.3.2.7 HMACSecretKey

La clé secrète HMAC partagée ; doit avoir une longueur minimum de six caractères.

COM et .NET

Signature: `HMACSecretKey(string key)`

Java

Signature: `public void setHMACSecretKey(string key)`

6.4.1.3.2.8 InputXMLFileName

Définit l'emplacement, en tant qu'une URL, du document XML à traiter. Le string fourni doit être une URL absolue qui donne l'emplacement exact du fichier XML.

COM et .NET

Signature: `InputXMLFileName(string filepath)`

Java

Signature: `public void setInputXMLFileName(string filepath)`

6.4.1.3.2.9 LastErrorMessage

Récupère un string qui est le dernier message d'erreur provenant du moteur RaptorXML.

COM et .NET

Signature: `string LastErrorMessage()`

Java

Signature: `public string getLastErrorMessage()`

6.4.1.3.2.10 SignatureMethod

Spécifie l'algorithme à utiliser pour générer la signature.

Lorsqu'un certificat est utilisé
Si un certificat est spécifié, alors `SignatureMethod` est optionnel et la valeur pour ce paramètre est dérivé du certificat. Lorsqu'il est spécifié, il doit correspondre à l'algorithme utilisé par le certificat. Exemple: `rsa-sha256`.

Exemple : `--signature-method=rsa-sha256`

Lorsque --hmac-secret-key est utilisé
Lorsque `HMACSecretKey` est utilisé, `SignatureMethod` est obligatoire. La valeur doit être un des algorithmes HMAC pris en charge :

- `hmac-sha256`
- `hmac-sha386`
- `hmac-sha512`
- hmac-sha1 (découragé par la spécification)

Exemple : `hmac-sha256`

COM et .NET

Signature: `SignatureMethod(string algo)`

Java

Signature: `public void setSignatureMethod(string algo)`

6.4.1.3.2.11 Transforms

Spécifie les transformations de Signature XML appliquées au document d'entrée. Les valeurs prises en charge sont :

- `REC-xml-c14n-20010315` pour Canonical XML 1.0 (sans commentaires)
- `xml-c14n11` pour Canonical XML 1.1 (sans comments)
- `xml-exc-c14n#` pour Exclusive XML Canonicalization 1.0 (sans commentaires)
- `REC-xml-c14n-20010315#WithComments` pour Canonical XML 1.0 (avec commentaires)
- `xml-c14n11#WithComments` pour Canonical XML 1.1 (avec commentaires)
- `xml-exc-c14n#WithComments` pour Exclusive XML Canonicalization 1.0 (avec commentaires)
- `base64`
- `strip-whitespaces` extension Altova

COM et .NET

Signature: `Transforms(string value)`

Java

Signature: `public void setTransforms(string value)`

6.4.1.3.2.12 WriteDefaultAttributes

Spécifie s'il faut inclure les valeurs d'attribut par défaut depuis le DTD dans le document signé.

COM et .NET

Signature: `WriteDefaultAttributes(boolean write)`

Java

Signature: `public void setWriteDefaultAttributes(boolean write)`

6.4.1.4 XMLValidator

L'interface/classe **IXMLValidator/XMLValidator** fournit des méthodes pour (i) valider plusieurs types de documents, (ii) contrôler la bonne formation des documents, et (iii) extraire un schéma Avro depuis un binaire Avro. Vous pouvez aussi fournir un traitement supplémentaire par le biais d'un script Python.

Veuillez noter que le nom de l'interface dans l'API COM/.NET est différent de celui de la classe dans l'API Java :

- Dans COM/.NET : **IXMLValidator**
- Dans Java : **XMLValidator**

6.4.1.4.1 Méthodes

Les méthodes de l'interface **IXMLValidator** (COM/.NET) et de la classe **XMLValidator** (Java) sont décrites dans cette section.

6.4.1.4.1.1 AddPythonScriptFile

Définit l'emplacement du fichier de script Python qui propose un traitement supplémentaire du fichier soumis à la validation. Le string fourni doit être une URL absolue du script Python. Celui-ci sera traité avec un paquet Python qui est fourni avec RaptorXML Server. Ce paquet Python est la version 3.11.5.

COM et .NET

Signature: **AddPythonScriptFile(**string **filepath)**

Java

Signature: **public** void **addPythonScriptFile(**string **filepath)**

6.4.1.4.1.2 ClearPythonScriptFile

Supprime les fichiers de script Python ajoutés avec la méthode **AddPythonScriptFile** ou la propriété **PythonScriptFile**.

COM et .NET

Signature: **ClearPythonScriptFile()**

Java

Signature: `public void clearPythonScriptFile()`

6.4.1.4.1.3 ExtractAvroSchema

Extrait un schéma Avro depuis un fichier binaire. Le paramètre `outputPath` est une URL absolue qui spécifie l'emplacement de sortie. Le résultat est `true` en cas de succès, `false` en cas d'échec. Si une erreur se produit, une **RaptorXMLException** 287 est soulevée. Utiliser `LastErrorMessage` pour accéder aux informations supplémentaires.

COM et .NET

Signature: `ExtractAvroSchema(string outputPath)`

Java

Signature: `public void extractAvroSchema(string outputPath)`

6.4.1.4.1.4 IsValid

Retourne le résultat de validation du document XML, le document de schéma, ou le document DTD. Le type de document à valider est spécifié par le paramètre `type`, qui prend un littéral ENUMValidationType 335 en tant que cette valeur. Le résultat est `true` en cas de succès, `false` en cas d'échec. Si une erreur se produit, une **RaptorXMLException** 287 est soulevée. Utiliser `LastErrorMessage` pour accéder à des informations supplémentaires.

COM et .NET

Signature: `boolean IsValid(ENUMValidationType 335 type)`

Java

Signature: `public boolean isValid(ENUMValidationType 335 type)`

6.4.1.4.1.5 IsWellFormed

Retourne le résultat du contrôle du document XML ou du document DTD de bonne formation. Le type de document à contrôler est spécifié avec le paramètre `type`, qui prend un littéral ENUMWellformedCheckType 336 en tant que sa valeur. Le résultat est `true` en cas de succès, `false` en cas d'échec. Si une erreur se produit,

une **RaptorXMLException** 287 est soulevée. Utiliser `LastErrorMessage` pour accéder à des informations supplémentaires.

COM et .NET

Signature: `boolean` **`isWellFormed(`**ENUMWellformedCheckType 335 **`type)`**

Java

Signature: **`public`** `boolean` **`isWellFormed(`**ENUMWellformedCheckType 335 **`type)`**

6.4.1.4.2 Propriétés

Les propriétés de l'interface **`IXMLValidator`** (COM/.NET) et de la classe **`XMLValidator`** (Java) sont décrites dans cette section.

6.4.1.4.2.1 AssessmentMode

Définit le mode d'évaluation de la validation XML (`Strict/Lax`), qui est donnée par un littéral ENUMAssessmentMode 330.

COM et .NET

Signature: **`AssessmentMode(`**ENUMAssessmentMode 330 **`mode)`**

Java

Signature: **`public`** `void` **`setAssessmentMode(`**ENUMAssessmentMode 330 **`mode)`**

6.4.1.4.2.2 AvroSchemaFileName

Définit l'emplacement, en tant qu'URL, du Schéma Avro externe à utiliser. Le string fournit doit être une URL absolue qui donne l'emplacement exact du fichier de schéma Avro.

COM et .NET

Signature: **`AvroSchemaFileName(`**`string` **`url)`**

Java

Signature: `public void setAvroSchemaFileName(string url)`

6.4.1.4.2.3 AvroSchemaFromText

Apporte le contenu du document de Schéma Avro à utiliser. Le string fourni est le contenu du document de Schéma Avro à utiliser.

COM et .NET

Signature: `AvroSchemaFromText(string avroschema)`

Java

Signature: `public void setAvroSchemaFromText(string avroschema)`

6.4.1.4.2.4 DTDFileName

Définit l'emplacement en tant qu'une URL, du document DTD à utiliser pour la validation. Le string fourni doit être une URL absolue qui indique l'emplacement exact du document DTD.

COM et .NET

Signature: `DTDFileName(string url)`

Java

Signature: `public void setDTDFileName(string url)`

6.4.1.4.2.5 DTDFromText

Apporte le contenu du document DTD à utiliser pour la validation. Le string fourni est le contenu du document DTD à utiliser.

COM et .NET

Signature: `DTDFromText(string dtdtext)`

Java

Signature: `public void setDTDFromText(string dtdtext)`

6.4.1.4.2.6 EnableNamespaces

Active le traitement reconnaissant l'espace de noms. Cela est utile pour contrôler la présence d'erreurs dues à des espaces de noms incorrects dans l'instance XML. Une valeur `true` permet le traitement reconnaissant les espaces de noms ; une valeur `false` la désactive. La valeur par défaut est `false`.

COM et .NET

Signature: **`EnableNamespaces(boolean enableNS)`**

Java

Signature: **`public void setEnableNamespaces(boolean enableNS)`**

6.4.1.4.2.7 InputFileArray

Fournit un tableau des URL des fichiers XML à utiliser en tant que données d'entrée. La propriété fournit un objet contenant, en tant que strings, les URL absolus de chaque fichiers XML.

COM et .NET

Signature: **`InputFileArray(object fileArray)`**

Java

Signature: **`public void setInputFileArray(object fileArray)`**

6.4.1.4.2.8 InputFileName

Définit l'emplacement, en tant qu'une URL, des données XML d'entrées à traiter. Le string fourni doit être une URL absolue qui donne l'emplacement du fichier d'entrée.

COM et .NET

Signature: **`InputFileName(string filepath)`**

Java

Signature: **`public void setInputFileName(string filepath)`**

6.4.1.4.2.9 InputFromText

Apporte le contenu du document XML à traiter. Le string fourni est le contenu du document XML à traiter.

COM et .NET

Signature: `InputFromText(string doc)`

Java

Signature: `public void setInputFromText(string doc)`

6.4.1.4.2.10 InputTextArray

Fournit un tableau des URL des fichiers texte à utiliser en tant que données d'entrée. La propriété fournit un objet contenant, en tant que strings, les URL absolus de chaque fichiers texte.

COM et .NET

Signature: `InputTextArray(object textfileArray)`

Java

Signature: `public void setInputTextArray(object textfileArray)`

6.4.1.4.2.11 InputXMLFileName

Définit l'emplacement, en tant qu'une URL, du document XML à traiter. Le string fourni doit être une URL absolue qui donne l'emplacement exact du fichier XML.

COM et .NET

Signature: `InputXMLFileName(string url)`

Java

Signature: `public void setInputXMLFileName(string url)`

6.4.1.4.2.12 InputXMLFromText

Apporte le contenu du document XML à traiter. Le string fourni est le contenu du document XML à traiter.

COM et .NET

Signature: `InputXMLFromText(string xml)`

Java

Signature: `public void setInputXMLFromText(string xml)`

6.4.1.4.2.13 Json5

Si elle est réglé sur `true`, active la prise en charge JSON 5.

COM et .NET

Signature: `Json5(boolean json5)`

Java

Signature: `public void setJson5(boolean json5)`

6.4.1.4.2.14 JSONSchemaFileName

Définit l'emplacement, en tant qu'URL, du fichier de Schéma JSON qui sera utilisé pour la validation instance-document JSON. Le string fournit doit être une URL absolu qui donne l'emplacement exact du fichier de Schéma JSON.

COM et .NET

Signature: `JSONSchemaFileName(string url)`

Java

Signature: `public void setJSONSchemaFileName(string url)`

6.4.1.4.2.15 JSONSchemaFromText

Fournit un string qui est le contenu textuel du document de Schéma JSON qui sera utilisé pour la validation du document d'instance JSON.

COM et .NET

Signature: `JSONSchemaFromText(string jsonschema)`

Java

Signature: `public void setJSONSchemaFromText(string jsonschema)`

6.4.1.4.2.16 LastErrorMessage

Récupère un string qui est le dernier message d'erreur provenant du moteur RaptorXML.

COM et .NET

Signature: `string LastErrorMessage()`

Java

Signature: `public string getLastErrorMessage()`

6.4.1.4.2.17 ParallelAssessment

Active/désactive estimation de validité de schéma parallèle[211].

COM et .NET

Signature: `ParallelAssessment(boolean enable)`

Java

Signature: `public void setParallelAssessment(boolean enable)`

6.4.1.4.2.18 PythonScriptFile

Définit l'emplacement du fichier de script Python qui propose un traitement supplémentaire du fichier soumis à la validation. Le string fourni doit être une URL absolue du script Python. Celui-ci sera traité avec un paquet Python qui est fourni avec RaptorXML Server. Ce paquet Python est la version 3.11.5.

COM et .NET

Signature: `PythonScriptFile(string filepath)`

Java

Signature: `public void setPythonScriptFile(string filepath)`

6.4.1.4.2.19 SchemaFileArray

Fournit la collection des fichiers de Schéma XML qui sera utilisée en tant que Schémas XML externe. Les fichiers sont identifiés par leurs URL. L'entrée est une collection de strings, chacun d'entre eux est l'URL absolue d'un fichier de Schéma XML.

COM et .NET

Signature: `SchemaFileArray(object urlArray)`

Java

Signature: `public void setSchemaFileArray(object urlArray)`

6.4.1.4.2.20 SchemaFileName

Définit l'emplacement, en tant qu'une URL, du document de Schéma XML à utiliser pour la validation. Le string fourni doit être une URL absolue qui donne l'emplacement exact du fichier de Schéma XML.

COM et .NET

Signature: `SchemaFileName(string filepath)`

Java

Signature: `public void setSchemaFileName(string filepath)`

6.4.1.4.2.21 SchemaFromText

Fournit les contenus du document de Schéma XML à utiliser. Le string fourni est le contenu du document de Schéma XML à utiliser.

COM et .NET

Signature: **SchemaFileName(**string **xsdText)**

Java

Signature: **public** void **setSchemaFileName(**string **xsdText)**

6.4.1.4.2.22 SchemaImports

Spécifie comment traiter les importations de schéma sur la base des valeurs d'attribut des éléments `xs:import`. Le type de gestion est spécifié par le littéral `ENUMSchemaImports` qui est soumis.

COM et .NET

Signature: **SchemaImports(**ENUMSchemaImports 333 **importOption)**

Java

Signature: **public** void **setSchemaImports(**ENUMSchemaImports 333 **importOption)**

6.4.1.4.2.23 SchemalocatiionHints

Spécifie le mécanisme à utiliser pour situer le schéma. Le mécanisme est spécifié par le littéral `ENUMLoadSchemalocation` qui a été sélectionné.

COM et .NET

Signature: **SchemalocationHints(**ENUMLoadSchemalocation 331 **hint)**

Java

Signature: **public** void **setSchemalocationHints(**ENUMLoadSchemalocation 331 **hint)**

6.4.1.4.2.24 SchemaMapping

Définit quel mappage utiliser pour localiser le schéma. Le mappage est spécifié par le littéral `ENUMSchemaMapping` qui est sélectionné.

COM et .NET

Signature: **SchemaMapping(**ENUMSchemaMapping[334] **mappingOption)**

Java

Signature: **public** void **setSchemaMapping(**ENUMSchemaMapping[334] **mappingOption)**

6.4.1.4.2.25 SchemaTextArray

Fournit le contenu de plusieurs fichiers de Schéma XML. L'entrée est une collection de strings, dont chacun est le contenu d'un document de Schéma XML.

COM et .NET

Signature: **SchemaTextArray(**object **schemaDocs)**

Java

Signature: **public** void **setSchemaTextArray(**object **schemaDocs)**

6.4.1.4.2.26 Streaming

Active la validation de streaming. Dans le mode streaming, les données qui sont stockées en mémoire sont minimisées et le traitement est plus rapide. Une valeur de `true` active le streaming; `false` le désactive. Le défaut est `true`.

COM et .NET

Signature: **Streaming(**boolean **enable)**

Java

Signature: **public** void **setStreaming(**boolean **enable)**

6.4.1.4.2.27 XincludeSupport

Active ou désactive l'utilisation des éléments `XInclude`. Une valeur de `true` active la prise en charge XInclude ; `false` le désactive. La valeur par défaut est `false`.

COM et .NET

Signature: **`XincludeSupport(boolean xinclude)`**

Java

Signature: **`public void setXincludeSupport(boolean xinclude)`**

6.4.1.4.2.28 XMLValidationMode

Définit le mode de validation XML, qui est une énumération littérale de ENUMXSDVersion[340] qui détermine s'il faut contrôler la validité ou la bonne formation.

COM et .NET

Signature: **`XMLValidationMode(`**ENUMXMLValidationMode[337] **`valMode)`**

Java

Signature: **`public void setXMLValidationMode(`**ENUMXMLValidationMode[337] **`valMode)`**

6.4.1.4.2.29 XSDVersion

Définit la version de schéma XML par rapport à laquelle le document XML sera validé. La valeur est une énumération littérale de ENUMXSDVersion[340].

COM et .NET

Signature: **`XSDVersion(`**ENUMXSDVersion[340] **`version)`**

Java

Signature: **`public void setXSDVersion(`**ENUMXSDVersion[340] **`version)`**

6.4.1.5 XQuery

L'interface/classe **IXQuery/XQuery** fournit les méthodes pour (i) exécuter des documents XQuery et des mises à jour XQuery, et (ii) valider des documents liés à XQuery. Vous pouvez aussi fournir des données pour les exécutions par le biais des variables externes.

Veuillez noter que le nom de l'interface dans l'API COM/.NET est différent que celui de la classs dans l'API Java :

- Dans COM/.NET : **IXQuery**
- Dans Java : **XQuery**

6.4.1.5.1 Méthodes

Les méthodes de l'interface **IXQuery** (COM/.NET) et de la classe **XQuery** (Java) sont décrites dans cette section.

6.4.1.5.1.1 AddExternalVariable

Ajoute le nom et la valeur d'une nouvelle variable externe. Chaque variable externe et sa valeur doivent être spécifiées dans un appel séparé à la méthode. Les variables doivent être déclarées dans le document XQuery (avec une déclaration de type optionnelle). Si la valeur variable est un string, entourer la valeur dans des guillemets simples. Le paramètre `name` contient le nom de la variable, qui est un QName, en tant que string. Le paramètre `value` contient la valeur de la variable en tant que string.

COM et .NET

Signature: **AddExternalVariable(**string **name,** string **value)**

Java

Signature: **public** void **addExternalVariable(**string **name,** string **value)**

6.4.1.5.1.2 ClearExternalVariableList

Supprime la liste des variables externes créée par la méthode AddExternalVariable[307].

COM et .NET

Signature: **ClearExternalVariableList()**

Java

Signature: `public void clearExternalVariableList()`

6.4.1.5.1.3 Execute

Exécute la transformation XQuery conformément à la version XQuery nommée dans la propriété `EngineVersion`[311], et enregistre le résultat dans un fichier de sortie nommé dans le paramètre `outputFile`. Le paramètre est un string qui fournit l'emplacement (chemin et le nom de fichier) du fichier de sortie. Le résultat est `true` en cas de succès, `false` en cas d'échec. Si une erreur se produit, une `RaptorXMLException`[287] est soulevée. Utiliser l'opération de la propriété `LastErrorMessage` pour accéder aux informations supplémentaires.

COM et .NET

Signature: `boolean Execute(string outputFile)`

Java

Signature: `public boolean Execute(string outputFile)`

6.4.1.5.1.4 ExecuteAndGetResultAsString

Exécute la mise à jour XQuery conformément à la spécification XQuery Update nommée dans la propriété `EngineVersion`[311], et retourne le résultat en tant que string. Cette méthode ne produit pas de fichiers de résultat supplémentaire, comme des graphiques ou les résultats secondaires. De même, il ne contient pas de résultats binaires comme des fichiers OOXML `.docx` . Si des fichiers de sortie supplémentaires sont nécessaires, utiliser la méthode `Execute`[308].

COM et .NET

Signature: `string ExecuteAndGetResultAsString()`

Java

Signature: `public string executeAndGetResultAsString()`

6.4.1.5.1.5 ExecuteUpdate

Exécute la mise à jour XQuery conformément à la spécification XQuery Update nommée dans la propriété `XQueryUpdateVersion`[318], et enregistre le résultat dans le fichier de sortie nommé dans le paramètre `outputFile`. Le paramètre est un string qui fournit l'emplacement (chemin et nom de fichier) du fichier de sortie. Le résultat est `true` en cas de succès, `false` en cas d'échec. Si une erreur se produit, une `RaptorXMLException`[287] est soulevée. Utiliser la propriété `LastErrorMessage` pour accéder à l'information supplémentaire.

COM et .NET

Signature: `boolean ExecuteUpdate(string outputFile)`

Java

Signature: `public boolean executeUpdate(string outputFile)`

6.4.1.5.1.6 ExecuteUpdateAndGetResultAsString

Exécute la mise à jour XQuery conformément à la spécification XQuery Update nommée dans la propriété `XQueryUpdateVersion`[318], et retourne le résultat en tant que string. Cette méthode ne produit pas de fichiers de résultat supplémentaires, comme des graphiques ou des résultats secondaires. Elle ne contient pas non plus de résultats binaires comme des fichiers OOXML `.docx`.

COM et .NET

Signature: `string ExecuteUpdateAndGetResultAsString()`

Java

Signature: `public string executeUpdateAndGetResultAsString()`

6.4.1.5.1.7 IsValid

Retourne le résultat de validation du document XQuery conformément à la spécification XQuery nommée dans la propriété `EngineVersion`[311]. Le résultat est `true` en cas de succès, `false` en cas d'échec. Si une erreur se produit, une `RaptorXMLException`[287] est soulevée. Utiliser la propriété `LastErrorMessage` pour accéder à l'information supplémentaire.

COM et .NET

Signature: `boolean IsValid()`

Java

Signature: `public boolean isValid()`

6.4.1.5.1.8 IsValidUpdate

Retourne le résultat de validation du document XQuery Update conformément à la spécification XQuery Update nommée dans la propriété `XQueryUpdateVersion`[318]. Le résultat est `true` en cas de succès, `false` en cas d'échec. Si une erreur se produit, une `RaptorXMLException`[287] est soulevée. Utiliser la propriété `LastErrorMessage` pour accéder à l'information supplémentaire.

COM et .NET

Signature: `boolean IsValidUpdate()`

Java

Signature: `public boolean isValidUpdate()`

6.4.1.5.2 Propriétés

Les propriétés de l'interface **`IXQuery`** (COM/.NET) et de la classe **`XQuery`** (Java) sont décrites dans cette section.

6.4.1.5.2.1 AdditionalOutputs

Retourne les sorties supplémentaires de la dernière tâche exécutée.

COM et .NET

Signature: `string AdditionalOutputs()`

Java

Signature: `public string getAdditionalOutputs()`

6.4.1.5.2.2 ChartExtensionsEnabled

Active ou désactive les fonctions d'extension de graphique d'Altova. Une valeur de `true` active les extensions de graphique ; `false` les désactive. La valeur par défaut est `true`.

COM et .NET

Signature: **`ChartExtensionsEnabled(boolean enable)`**

Java

Signature: **`public void setChartExtensionsEnabled(boolean enable)`**

6.4.1.5.2.3 DotNetExtensionsEnabled

Active ou désactive les fonctions d'extension de .NET. Une valeur de `true` active les extensions .NET ; `false` les désactive. La valeur par défaut est `true`.

COM et .NET

Signature: **`DotNetExtensionsEnabled(boolean enable)`**

Java

Signature: **`public void setDotNetExtensionsEnabled(boolean enable)`**

6.4.1.5.2.4 EngineVersion

Spécifie la version XQuery à utiliser. La valeur de propriété est un littéral ENUMXQueryVersion[339].

COM et .NET

Signature: **`EngineVersion(`**ENUMXQueryVersion[339] **`version)`**

Java

Signature: **`public void setEngineVersion(`**ENUMXQueryVersion[339] **`version)`**

6.4.1.5.2.5 IndentCharacters

Soumet le string de caractère qui sera utilisé en tant que retrait dans la sortie.

COM et .NET

Signature: `IndentCharacters(string indentChars)`

Java

Signature: `public void setIndentCharacters(string indentChars)`

6.4.1.5.2.6 InputXMLFileName

Définit l'emplacement, en tant qu'une URL, du document XML à traiter. Le string fourni doit être une URL absolue qui donne l'emplacement exact du fichier XML.

COM et .NET

Signature: `InputXMLFileName(string url)`

Java

Signature: `public void setInputXMLFileName(string url)`

6.4.1.5.2.7 InputXMLFromText

Apporte le contenu du document XML à traiter. Le string fourni est le contenu du document XML à traiter.

COM et .NET

Signature: `InputXMLFromText(string xml)`

Java

Signature: `public void setInputXMLFromText(string xml)`

6.4.1.5.2.8 JavaBarcodeExtensionLocation

Spécifie l'emplacement du fichier d'extension du code-barres. Voir la section sur les fonctions d'extension du code-barres d'Altova pour plus d'informations. Le string fourni doit être une URL absolue qui donne l'emplacement de base du fichier à utiliser.

COM et .NET

Signature: `JavaBarcodeExtensionLocation(string url)`

Java

Signature: `public void setJavaBarcodeExtensionLocation(string url)`

6.4.1.5.2.9 JavaExtensionsEnabled

Active ou désactive des fonctions d'extension Java. Une valeur de `true` active des extensions Java ; `false` les désactive. La valeur par défaut est `true`.

COM et .NET

Signature: `JavaExtensionsEnabled(boolean enable)`

Java

Signature: `public void setJavaExtensionsEnabled(boolean enable)`

6.4.1.5.2.10 KeepFormatting

Spécifie si le formatage du document original doit être gardé (autant que possible) ou non. Une valeur `true` garde le formatage ; une valeur `false` ne garde pas le formatage. La valeur par défaut est `true`.

COM et .NET

Signature: `KeepFormatting(boolean keep)`

Java

Signature: `public void setKeepFormatting(boolean keep)`

6.4.1.5.2.11 LastErrorMessage

Récupère un string qui est le dernier message d'erreur provenant du moteur RaptorXML.

COM et .NET

Signature: `string LastErrorMessage()`

Java

Signature: `public string getLastErrorMessage()`

6.4.1.5.2.12 LoadXMLWithPSVI

Active la validation des fichiers XML d'entrée et génère des infos post-schema-validation pour les fichiers. Une valeur de `true` active la validation XML et génère des infos post-schema-validation pour les fichiers XML ; `false` désactive la validation. La valeur par défaut est `true`.

COM et .NET

Signature: `LoadXMLWithPSVI(boolean enable)`

Java

Signature: `public void setLoadXMLWithPSVI(boolean enable)`

6.4.1.5.2.13 MainOutput

Retourne la sortie principale de la dernière tâche exécutée.

COM et .NET

Signature: `string MainOutput()`

Java

Signature: `public string getMainOutput()`

6.4.1.5.2.14 OutputEncoding

Définit l'encodage pour le document de résultat. Utilise un nom d'encodage IANA officiel comme `UTF-8`, `UTF-16`, `US-ASCII`, `ISO-8859-1`, en tant que string.

COM et .NET

Signature: `OutputEncoding(string encoding)`

Java

Signature: `public void setOutputEncoding(string encoding)`

6.4.1.5.2.15 OutputIndent

Active ou désactive le retrait dans le document de sortie. Une valeur de `true` active le retrait ; une valeur `false` le désactive.

COM et .NET

Signature: `OutputIndent(boolean outputIndent)`

Java

Signature: `public void setOutputIndent(boolean outputIndent)`

6.4.1.5.2.16 OutputMethod

Spécifie la sérialisation du document de sortie. Les valeurs valides sont : `xml | xhtml | html | text`. La valeur par défaut est `xml`.

COM et .NET

Signature: `OutputMethod(string format)`

Java

Signature: `public void setOutputMethod(string format)`

6.4.1.5.2.17 OutputOmitXMLDeclaraton

Active ou désactive l'inclusion de la déclaration XML dans le document de résultat. Une valeur de `true` omet la déclaration; une valeur de `false` l'inclut. La valeur par défaut est `false`.

COM et .NET

Signature: **`OutputOmitXMLDeclaration(boolean omitDeclaration)`**

Java

Signature: **`public void setOutputOmitXMLDeclaration(boolean omitDeclaration)`**

6.4.1.5.2.18 UpdatedXMLWriteMode

Spécifie comment gérer les mises à jour du fichier file. La valeur de propriété est un littéral ENUMXQueryUpdatedXML[338].

COM et .NET

Signature: **`UpdateXMLWriteMode(`**ENUMXQueryUpdatedXML[338] **`updateMode)`**

Java

Signature: **`public void setUpdateXMLWriteMode(`**ENUMXQueryUpdatedXML[338] **`updateMode)`**

6.4.1.5.2.19 XincludeSupport

Active ou désactive l'utilisation des éléments `XInclude`. Une valeur de `true` active la prise en charge XInclude ; `false` le désactive. La valeur par défaut est `false`.

COM et .NET

Signature: **`XincludeSupport(boolean xinclude)`**

Java

Signature: **`public void setXincludeSupport(boolean xinclude)`**

6.4.1.5.2.20 XMLValidationErrorsAsWarnings

Active le traitement des erreurs de validation XML en tant qu'avertissements. Prend la booléenne `true` ou `false`.

COM et .NET

Signature: **`XMLValidationErrorsAsWarnings(boolean enable)`**

Java

Signature: **`public void setXMLValidationErrorsAsWarnings(boolean enable)`**

6.4.1.5.2.21 XMLValidationMode

Définit le mode de validation XML, qui est une énumération littérale de `ENUMXSDVersion` [340] qui détermine s'il faut contrôler la validité ou la bonne formation.

COM et .NET

Signature: **`XMLValidationMode(ENUMXMLValidationMode` [337] **`valMode)`**

Java

Signature: **`public void setXMLValidationMode(ENUMXMLValidationMode` [337] **`valMode)`**

6.4.1.5.2.22 XQueryFileName

Spécifie le fichier XQuery à utiliser. Le string fourni doit être une URL absolue qui donne l'emplacement du fichier XQuery à utiliser.

COM et .NET

Signature: **`XQueryFileName(string fileurl)`**

Java

Signature: **`public void setXQueryFileName(string fileurl)`**

6.4.1.5.2.23 XQueryFromText

Fournir, en tant que string de texte, les contenus du document XQuery à utiliser

COM et .NET

Signature: **XQueryFromText(**string **xqtext)**

Java

Signature: **public** void **setXQueryFromText(**string **xqtext)**

6.4.1.5.2.24 XQueryUpdateVersion

Spécifie la version XQuery Update à utiliser. La valeur de propriété est un littéral ENUMXQueryVersion[338].

COM et .NET

Signature: **XQueryUpdateVersion(**ENUMXQueryUpdateVersion[338] **version)**

Java

Signature: **public** void **setXQueryUpdateVersion(**ENUMXQueryUpdateVersion[338] **version)**

6.4.1.5.2.25 XSDVersion

Définit la version de schéma XML par rapport à laquelle le document XML sera validé. La valeur est une énumération littérale de ENUMXSDVersion[340].

COM et .NET

Signature: **XSDVersion(**ENUMXSDVersion[340] **version)**

Java

Signature: **public** void **setXSDVersion(**ENUMXSDVersion[340] **version)**

6.4.1.6 XSLT

L'interface/classe **IXSLT/XSLT** fournit les méthodes pour pour exécuter les transformations XSLT et valider les documents liés à XSLT. Vous pouvez aussi fournir des données pour la transformation par le biais des paramètres externes.

Veuillez noter que le nom de l'interface dans l'API COM/.NET est différent que celui de la classs dans l'API Java :

- Dans COM/.NET : **IXSLT**
- Dans Java : **XSLT**

6.4.1.6.1 Méthodes

Les méthodes de l'interface **IXSLT** (COM/.NET) et de la classe **XSLT** (Java) sont décrites dans cette section.

6.4.1.6.1.1 AddExternalParameter

Ajoute le nom et la valeur d'un nouveau paramètre externe. Chaque paramètre externe et sa valeur doit être spécifiée dans un appel séparé à la méthode. Les paramètre doivent être déclarés dans le document XSLT. Puisque les valeurs de paramètre sont des expressions XPath, les valeurs de paramètre qui sont des strings doivent être entourées dans des guillemets simples. Le paramètre `name` contient le nom de la variable, qui est un QName, en tant que string. Le paramètre `value` contient la valeur de la variable en tant que string.

COM et .NET

Signature: **AddExternalParameter(**string **name,** string **value)**

Java

Signature: **public** void **addExternalParameter(**string **name,** string **value)**

6.4.1.6.1.2 ClearExternalParameterList

Supprime la liste des paramètres externes créée par la méthode AddExternalParameter[319].

COM et .NET

Signature: **ClearExternalParameterList()**

Java

Signature: `public void clearExternalParameterList()`

6.4.1.6.1.3 Execute

Exécute la transformation XSLT conformément à la spécification XSLT nommée dans la propriété `EngineVersion`[311], et enregistre le résultat dans un fichier de sortie nommé dans le paramètre `outputFile`. Si une erreur se produit, une `RaptorXMLException`[287] est soulevée. Utiliser l'opération de la propriété `LastErrorMessage` pour accéder aux informations supplémentaires.

COM et .NET

Signature: `boolean Execute(string outputFile)`

Java

Signature: `public boolean execute(string outputFile)`

6.4.1.6.1.4 ExecuteAndGetResultAsString

Exécute la transformation XSLT conformément à la spécification XSLT nommée dans la propriété `EngineVersion`[311], et retourne le résultat en tant que string. Cette méthode ne produit pas de fichiers de résultat supplémentaires, comme des graphiques ou les résultats secondaires. De même, il ne contient pas de résultats binaires comme des fichiers OOXML `.docx` . Si des fichiers de sortie supplémentaires sont nécessaires, utiliser la méthode `Execute`[308]. Si une erreur se produit, une `RaptorXMLException`[287] est soulevée. Utiliser l'opération de la propriété `LastErrorMessage` pour accéder aux informations supplémentaires.

COM et .NET

Signature: `string ExecuteAndGetResultAsString()`

Java

Signature: `public string executeAndGetResultAsString()`

6.4.1.6.1.5 ExecuteAndGetResultAsStringWithBaseOutputURI

Exécute la transformation XSLT conformément à la spécification XSLT nommée dans la propriété `EngineVersion`[311], et retourne le résultat en tant que string à l'emplacement défini par la base URI. Le paramètre `baseURI` est un string qui fournit une URI. Cette méthode ne produit pas de fichiers de résultat supplémentaires, comme des graphiques ou les résultats secondaires. De même, il ne contient pas de résultats binaires comme des fichiers OOXML `.docx`. Si des fichiers de sortie supplémentaires sont nécessaires, utiliser la méthode `Execute`[308]. Si une erreur se produit, une `RaptorXMLException`[287] est soulevée. Utiliser l'opération de la propriété `LastErrorMessage` pour accéder aux informations supplémentaires.

COM et .NET

Signature: `string ExecuteAndGetResultAsStringWithBaseOutputURI(string baseURI)`

Java

Signature: `public string ExecuteAndGetResultAsStringWithBaseOutputURI(string baseURI)`

6.4.1.6.1.6 IsValid

Retourne le résultat de validation du document XSLT conformément à la spécification XSLT nommée dans la propriété `EngineVersion`[311]. Le résultat est `true` en cas de succès, `false` en cas d'échec. Si une erreur se produit, une `RaptorXMLException`[287] est soulevée. Utiliser la propriété `LastErrorMessage` pour accéder à l'information supplémentaire.

COM et .NET

Signature: `boolean IsValid()`

Java

Signature: `public boolean isValid()`

6.4.1.6.2 Propriétés

Les propriétés de l'interface **IXSLT** (COM/.NET) et de la classe **XSLT** (Java) sont décrites dans cette section.

6.4.1.6.2.1 AdditionalOutputs

Retourne les sorties supplémentaires de la dernière tâche exécutée.

COM et .NET

Signature: `string AdditionalOutputs()`

Java

Signature: `public string getAdditionalOutputs()`

6.4.1.6.2.2 ChartExtensionsEnabled

Active ou désactive les fonctions d'extension de graphique d'Altova. Une valeur de `true` active les extensions de graphique ; `false` les désactive. La valeur par défaut est `true`.

COM et .NET

Signature: `ChartExtensionsEnabled(boolean enable)`

Java

Signature: `public void setChartExtensionsEnabled(boolean enable)`

6.4.1.6.2.3 DotNetExtensionsEnabled

Active ou désactive les fonctions d'extension de .NET. Une valeur de `true` active les extensions .NET ; `false` les désactive. La valeur par défaut est `true`.

COM et .NET

Signature: `DotNetExtensionsEnabled(boolean enable)`

Java

Signature: `public void setDotNetExtensionsEnabled(boolean enable)`

6.4.1.6.2.4 EngineVersion

Spécifie la version XSLT à utiliser. La valeur de propriété est un littéral ENUMXSLTVersion[341].

COM et .NET

Signature: **EngineVersion(**ENUMXSLTVersion[341] **version)**

Java

Signature: **public** void **setEngineVersion(**ENUMXSLTVersion[341] **version)**

6.4.1.6.2.5 IndentCharacters

Soumet le string de caractère qui sera utilisé en tant que retrait dans la sortie.

COM et .NET

Signature: **IndentCharacters(**string **indentChars)**

Java

Signature: **public** void **setIndentCharacters(**string **indentChars)**

6.4.1.6.2.6 InitialTemplateMode

Définit le mode initial pour le traitement XSLT. Les modèles présentant une valeur de mode égale au string soumis seront traités.

COM et .NET

Signature: **InitialTemplateMode(**string **mode)**

Java

Signature: **public** void **setInitialTemplateMode(**string **mode)**

6.4.1.6.2.7 InputXMLFileName

Définit l'emplacement, en tant qu'une URL, du document XML à traiter. Le string fourni doit être une URL absolue qui donne l'emplacement exact du fichier XML.

COM et .NET

Signature: `InputXMLFileName(string url)`

Java

Signature: `public void setInputXMLFileName(string url)`

6.4.1.6.2.8 InputXMLFromText

Apporte le contenu du document XML à traiter. Le string fourni est le contenu du document XML à traiter.

COM et .NET

Signature: `InputXMLFromText(string xml)`

Java

Signature: `public void setInputXMLFromText(string xml)`

6.4.1.6.2.9 JavaBarcodeExtensionLocation

Spécifie l'emplacement du fichier d'extension du code-barres. Voir la section sur les fonctions d'extension du code-barres d'Altova pour plus d'informations. Le string fourni doit être une URL absolue qui donne l'emplacement de base du fichier à utiliser.

COM et .NET

Signature: `JavaBarcodeExtensionLocation(string url)`

Java

Signature: `public void setJavaBarcodeExtensionLocation(string url)`

6.4.1.6.2.10 JavaExtensionsEnabled

Active ou désactive des fonctions d'extension Java. Une valeur de `true` active des extensions Java ; `false` les désactive. La valeur par défaut est `true`.

COM et .NET

Signature: `JavaExtensionsEnabled(boolean enable)`

Java

Signature: `public void setJavaExtensionsEnabled(boolean enable)`

6.4.1.6.2.11 LastErrorMessage

Récupère un string qui est le dernier message d'erreur provenant du moteur RaptorXML.

COM et .NET

Signature: `string LastErrorMessage()`

Java

Signature: `public string getLastErrorMessage()`

6.4.1.6.2.12 LoadXMLWithPSVI

Active la validation des fichiers XML d'entrée et génère des infos post-schema-validation pour les fichiers. Une valeur de `true` active la validation XML et génère des infos post-schema-validation pour les fichiers XML ; `false` désactive la validation. La valeur par défaut est `true`.

COM et .NET

Signature: `LoadXMLWithPSVI(boolean enable)`

Java

Signature: `public void setLoadXMLWithPSVI(boolean enable)`

6.4.1.6.2.13 MainOutput

Retourne la sortie principale de la dernière tâche exécutée.

COM et .NET

Signature: `string MainOutput()`

Java

Signature: `public string getMainOutput()`

6.4.1.6.2.14 NamedTemplateEntryPoint

Spécifie le nom en tant que string, du modèle nommé à utiliser en tant que point d'entrée pour la transformation.

COM et .NET

Signature: `NamedTemplateEntryPoint(string template)`

Java

Signature: `public void setNamedTemplateEntryPoint(string template)`

6.4.1.6.2.15 SchemaImports

Spécifie comment traiter les importations de schéma sur la base des valeurs d'attribut des éléments `xs:import`. Le type de gestion est spécifié par le littéral `ENUMSchemaImports` qui est soumis.

COM et .NET

Signature: `SchemaImports(ENUMSchemaImports 333 importOption)`

Java

Signature: `public void setSchemaImports(ENUMSchemaImports 333 importOption)`

6.4.1.6.2.16 SchemalocationHints

Spécifie le mécanisme à utiliser pour situer le schéma. Le mécanisme est spécifié par le littéral `ENUMLoadSchemalocation` qui a été sélectionné.

COM et .NET

Signature: **SchemalocationHints(**ENUMLoadSchemalocation 331 **hint)**

Java

Signature: **public** void **setSchemalocationHints(**ENUMSchemalocation 331 **hint)**

6.4.1.6.2.17 SchemaMapping

Définit quel mappage utiliser pour localiser le schéma. Le mappage est spécifié par le littéral `ENUMSchemaMapping` qui est sélectionné.

COM et .NET

Signature: **SchemaMapping(**ENUMSchemaMapping 334 **mappingOption)**

Java

Signature: **public** void **setSchemaMapping(**ENUMSchemaMapping 334 **mappingOption)**

6.4.1.6.2.18 StreamingSerialization

Active la sérialisation de streaming. Dans le mode streaming, les données stockées dans la mémoire sont minimisées et le traitement est plus rapide. Une valeur de `true` active la sérialisation de streaming ; `false` la désactive.

COM et .NET

Signature: **StreamingSerialization(**boolean **enable)**

Java

Signature: **public** void **setStreamingSerialization(**boolean **enable)**

6.4.1.6.2.19 XincludeSupport

Active ou désactive l'utilisation des éléments `XInclude`. Une valeur de `true` active la prise en charge XInclude ; `false` le désactive. La valeur par défaut est `false`.

COM et .NET

Signature: **`XincludeSupport(boolean xinclude)`**

Java

Signature: **`public void setXincludeSupport(boolean xinclude)`**

6.4.1.6.2.20 XMLValidationErrorsAsWarnings

Active le traitement des erreurs de validation XML en tant qu'avertissements. Prend la booléenne `true` ou `false`.

COM et .NET

Signature: **`XMLValidationErrorsAsWarnings(boolean enable)`**

Java

Signature: **`public void setXMLValidationErrorsAsWarnings(boolean enable)`**

6.4.1.6.2.21 XMLValidationMode

Définit le mode de validation XML, qui est une énumération littérale de `ENUMXSDVersion` 340 qui détermine s'il faut contrôler la validité ou la bonne formation.

COM et .NET

Signature: **`XMLValidationMode(`**`ENUMXMLValidationMode` 337 **`valMode)`**

Java

Signature: **`public void setXMLValidationMode(`**`ENUMXMLValidationMode` 337 **`valMode)`**

6.4.1.6.2.22 XSDVersion

Définit la version de schéma XML par rapport à laquelle le document XML sera validé. La valeur est une énumération littérale de ENUMXSDVersion[340].

COM et .NET

Signature: **XSDVersion(**ENUMXSDVersion[340] **version)**

Java

Signature: **public** void **setXSDVersion(**ENUMXSDVersion[340] **version)**

6.4.1.6.2.23 XSLFileName

Spécifie le fichier XSLT à utiliser. Le string fourni doit être une URL absolue qui donne l'emplacement du fichier XSLT à utiliser.

COM et .NET

Signature: **XSLFileName(**string **fileurl)**

Java

Signature: **public** void **setXSLFileName(**string **fileurl)**

6.4.1.6.2.24 XSLFromText

Fournit en tant que string texte, les contenus du document XSLT à utiliser

COM et .NET

Signature: **XSLFromText(**string **xsltext)**

Java

Signature: **public** void **setXSLFromText(**string **xsltext)**

6.4.2 Énumérations

Les énumérations des API de serveur COM/.NET et Java sont décrites dans cette section. Chaque description comprend des liens vers les méthodes ou les propriétés qui utilisent l'énumération.

6.4.2.1 ENUMAssessmentMode

Définit le mode d'attribution du Valideur XML pour être strict ou lâche :

- **`eAssessmentModeStrict`**: Définit le mode d'attribution de validité de schéma sur `Strict`. Il s'agit de la valeur par défaut.
- **`eAssessmentModeLax`**: Définit le mode d'attribution de validité de schéma sur `Lax`.

COM et .NET

`eAssessmentModeStrict`	`= 0`
`eAssessmentModeLax`	`= 1`

Utilisé par

Interface	*Propriété*
IXMLValidator 295	AssessmentMode 297

Java

```
public enum ENUMAssessmentMode {
       eAssessmentModeLax
       eAssessmentModeStrict   }
```

Utilisé par

Class	*Method*

XMLValidator 295	setAssessmentMode 297

6.4.2.2 ENUMErrorFormat

Spécifie le format de la sortie d'erreur :

- **eFormatText**: Définit le format de sortie d'erreur sur `Text`. Il s'agit de la valeur par défaut.
- **eFormatShortXML**: Définit le format de sortie d'erreur sur `ShortXML`. Ce format est une forme abrégée du format `LongXML`.
- **eFormatLongXML**: Définit le format de sortie d'erreur sur `LongXML`. Des trois formats de sortie, il s'agit du format qui fournit le plus de détails.

COM et .NET

eFormatText	= 0
eFormatShortXML	= 1
eFormatLongXML	= 2

Utilisé par

Interface	*Propriété*
IServer 277	ErrorFormat 281

Java

```
public enum ENUMErrorFormat {
        eFormatText
        eFormatShortXML
        eFormatLongXML }
```

Utilisé par

Classe	*Méthode*
RaptorXMLFactory 277	setErrorFormat 281

6.4.2.3 ENUMLoadSchemalocation

Indique comment l'emplacement du schéma doit être déterminé. La sélection est basée sur l'attribut d'emplacement du schéma du document d'instance XML. Cet attribut pourrait être `xsi:schemaLocation` ou `xsi:noNamespaceSchemaLocation`.

- **eSHLoadBySchemalocation** utilise l'URL de l'attribut d'emplacement de schéma dans le document d'instance XML. Ce littéral d'énumération est la **valeur par défaut**.
- **eSHLoadByNamespace** utilise la partie de l'espace de noms de `xsi:schemaLocation` et un string vide dans le cas de `xsi:noNamespaceSchemaLocation` pour situer le schéma par le biais d'un mappage de catalogue.
- **eSHLoadCombiningBoth**: si soit l'URL d'espace de noms ou l'URL d'emplacement de schéma a un mappage de catalogue, alors le mappage de catalogue sera utilisé. Si tous deux ont des mappages de catalogue, la valeur de ENUMSchemaMapping[334] décide quel mappage est utilisé. Si ni l'espace de noms ni l'emplacement de schéma a un mappage de catalogue, l'URL d'emplacement de schéma est utilisé.
- **eSHLoadIgnore**: Les attributs `xsi:schemaLocation` et `xsi:noNamespaceSchemaLocation` sont tous les deux ignorés.

COM et .NET

eSHLoadBySchemalocation	= 0
eSHLoadByNamespace	= 1
eSHLoadCombiningBoth	= 2
eSHLoadIgnore	= 3

Utilisé par

Interface	*Propriété*
IXMLValidator[295]	SchemalocationHints[304]
IXSLT[319]	SchemalocationHints[327]

Java

```
public enum ENUMLoadSchemalocation {
       eSHLoadBySchemalocation
       eSHLoadByNamespace
       eSHLoadCombiningBoth
       eSHLoadIgnore    }
```

Utilisé par

Classe	*Méthode*
XMLValidator[295]	setSchemalocationHints[304]
XSLT[319]	setSchemalocationHints[327]

6.4.2.4 ENUMSchemaImports

Définit le comportement des éléments `xs:import` du schéma, chacun possède un attribut optionnel `namespace` et une attribut `schemaLocation` optionnel.

- **`eSILoadBySchemalocation`** uses the value of the `schemaLocation` attribute pour situer le schéma, en prenant en compte les mappages de catalogue. Si l'attribut `namespace` est présent, l'espace de noms sera importé (mis sous licence).
- **`eSILoadPreferringSchemalocation`**: Si l'attribut `schemaLocation` est présent, il est utilisé, en prenant en compte les mappages de catalogue. Si aucun attribut `schemaLocation` n'est présent, la valeur de l'attribut `namespace` est utilisée par le biais d'un mappage de catalogue. Ce littéral d'énumération est la **valeur par défaut.**
- **`eSILoadByNamespace`** utilise la valeur de l'attribut `namespace` pour situer le schéma par le biais d'un mappage de catalogue.
- **`eSILoadCombiningBoth`**: Si soit l'URL de `namespace` ou l'URL de `schemaLocation` a un mappage de catalogue, le mappage de catalogue sera utilisé. Si tous deux ont des mappages de catalogue, la valeur du paramètre `ENUMSchemaMapping` 334 décidera quel mappage sera utilisé. Si ni l'URL de `namespace` ni de `schemaLocation` n'a un mappage de catalogue, l'URL `schemaLocation` sera utilisée.
- **`eSILicenseNamespaceOnly`**: L'espace de noms est importé. Aucun document de schéma n'est importé.

COM et .NET

eSILoadBySchemalocation	= 0
eSILoadPreferringSchemalocation	= 1
eSILoadByNamespace	= 2
eSICombiningBoth	= 3
eSILicenseNamespaceOnly	= 4

Utilisé par

Interface	*Propriété*
IXMLValidator 295	SchemaImports 304
IXSLT 319	SchemaImports 326

Java

```
public enum ENUMSchemaImports {
        eSILoadBySchemalocation
        eSILoadPreferringSchemalocation
        eSILoadByNamespace
        eSILoadCombiningBoth
        eSILicenseNamespaceOnly   }
```

Utilisé par

Classe	*Méthode*
XMLValidator 295	setSchemaImports 304
XSLT 319	setSchemaImports 326

6.4.2.5 ENUMSchemaMapping

Spécifie lequel des deux mappages de catalogue est préféré : les espaces de noms ou les URL de schema-location. Cette énumération est utile pour différencier ENUMLoadSchemalocation 331 et ENUMSchemaImports 333 .

- **eSMPreferNamespace**: Sélectionne l'espace de noms.
- **eSMPreferSchemalocation**: Sélectionne l'emplacement de schéma. Il s'agit de la valeur par défaut.

COM et .NET

eSMPreferSchemalocation	= 0
eSMPreferNamespace	= 1

Utilisé par

Interface	*Propriété*
IXMLValidator 295	SchemaMapping 305
IXSLT 319	SchemaMapping 327

Java

```
public enum ENUMSchemaMapping {
       eSMPreferSchemalocation
       eSMPreferNamespace   }
```

Utilisé par

Classe	*Méthode*
IXMLValidator 295	setSchemaMapping 305
IXSLT 319	setSchemaMapping 327

6.4.2.6 ENUMValidationType

Spécifie quelle validation à effectuer et, dans le cas des documents XML, que la validation s'effectue par rapport à un DTD ou un XSD.

- **eValidateAny**: Le type de document (par exemple, XML ou XSD) est détecté, et la validation est définie automatiquement pour ce type de document.
- **eValidateXMLWithDTD**: Spécifie la validation d'un document XML par rapport à un DTD.
- **eValidateXMLWithXSD**: Spécifie la validation d'un document XML par rapport à un XSD (Schéma XML).
- **eValidateDTD**: Spécifie la validation d'un document DTD.
- **eValidateXSD**: Spécifie la validation d'un document XSD (W3C Schéma XML).
- **eValidateJSON**: Spécifie la validation d'un document d'instance JSON.
- **eValidateJSONSchema**: Spécifie la validation d'un document Schéma JSON conformément à Schéma JSON v4.
- **eValidateAvro**: Spécifie la validation d'un fichier binaire Avro. Les données Avro dans le fichier binaire sont validées par rapport au Schéma Avro contenu dans le fichier binaire.
- **eValidateAvroSchema**: Spécifie la validation d'un schéma Avro par rapport à la spécification de schéma Avro.
- **eValidateAvroJSON**: Spécifie la validation d'un fichier de données Avro sérialisé de JSON par rapport au schéma Avro.

COM et .NET

eValidateAny	= 0
eValidateXMLWithDTD	= 1
eValidateXMLWithXSD	= 2
eValidateDTD	= 3
eValidateXSD	= 4
eValidateJSON	= 5
eValidateJSONSchema	= 6
eValidateAvro	= 7
eValidateAvroSchema	= 8
eValidateAvroJSON	= 9

Utilisé par

Interface	*Méthode*
IXMLValidator 295	isValid 296

Java

```
public enum ENUMValidationType {
       eValidateAny
```

```
        eValidateXMLWithDTD
        eValidateXMLWithXSD
        eValidateDTD
        eValidateXSD
        eValidateJSON
        eValidateJSONSchema
        eValidateAvro
        eValidateAvroSchema
        eValidateAvroJSON }
```

Utilisé par

Classe	*Méthode*
XMLValidator 295	isValid 296

6.4.2.7 ENUMWellformedCheckType

Spécifie le type du contrôle de la bonne formation à effectuer (pour XML, DTD ou JSON).

- **eWellformedAny**: Le type de document est détecté, et le type de contrôle est défini automatiquement.
- **eWellformedXML**: Contrôle la bonne formation d'un document XML.
- **eWellformedDTD**: Contrôle la bonne formation d'un document DTD.
- **eWellformedJSON**: Contrôle la bonne formation d'un document JSON.

COM et .NET

eWellFormedAny	= 0
eWellFormedXML	= 1
eWellFormedDTD	= 2
eWellFormedJSON	= 3

Utilisé par

Interface	*Méthode*
IXMLValidator 295	isWellFormed 296

Java

```
public enum ENUMWellformedCheckType {
        eWellformedAny
        eWellformedXML
        eWellformedDTD
        eWellformedJSON    }
```

Utilisé par

Classe	*Méthode*
XMLValidator 295	isWellFormed 296

6.4.2.8 ENUMXMLValidationMode

Spécifie le type de validation XML à effectuer (validation ou contrôle de la bonne formation).

- **eProcessingModeWF**: Règle le mode de traitement XML sur `wellformed`. Il s'agit de la valeur par défaut.
- **eProcessingModeValid**: Règle le mode de traitement XML sur `validation`.
- **eProcessingModeID**: Interne, pas destiné à l'utilisation.

COM et .NET

eXMLValidationModeWF	= 0
eXMLValidationModeID	= 1
eXMLValidationModeValid	= 2

Utilisé par

Interface	*Propriété*
IXMLValidator 295	XMLValidationMode 306
IXQuery 307	XMLValidationMode 317
IXSLT 319	XMLValidationMode 328

Java

```
public enum ENUMXMLValidationMode {
       eProcessingModeValid
       eProcessingModeWF
       eProcessingModeID  }
```

Utilisé par

Classe	*Méthode*
XMLValidator 295	setXMLValidationMode 306
XQuery 307	setXMLValidationMode 317
XSLT 319	setXMLValidationMode 328

6.4.2.9 ENUMXQueryUpdatedXML

Spécifie comment gérer les mises à jour XQuery.

- **eUpdatedDiscard**: Les mises à jour sont éliminées et ne sont pas écrites sur le fichier.
- **eUpdatedWriteback**: Les mises à jour sont écrites dans le fichier d'entrée XML spécifié avec (set) InputXMLFileName[312].
- **eUpdatedAsMainResult**: Les mises à jour sont écrites dans l'emplacement spécifié par le paramètre outputFile de ExecuteUpdate[288].

COM et .NET

eUpdatedDiscard	= 1
eUpdatedWriteback	= 2
eUpdatedAsMainResult	= 3

Utilisé par

Interface	*Propriété*
IXQuery[307]	UpdatedXMLWriteMode[316]

Java

```
public enum ENUMXQueryUpdatedXML {
       eUpdatedDiscard
       eUpdatedWriteback
       eeUpdatedAsMainResult }
```

Utilisé par

Classe	*Méthode*
XQuery[307]	setUpdatedXMLWriteMode[316]

6.4.2.10 ENUMXQueryUpdateVersion

Définit la version XQuery Update à utiliser pour le traitement (exécution ou validation).

- **eXQUpdateVersion10**: Définit la version XQuery Update à XQuery Update 1.0.

Note: le littéral d'énumération Java est nommé différemment que le littéral COM/.NET. *Voir ci-dessous.*

COM et .NET

eXQUpdateVersion10	= 1

Utilisé par

Interface	*Propriété*
IXQuery 307	XQueryUpdateVersion 318

Java

```
public enum ENUMXQueryUpdateVersion {
       eVersion10        }
```

Utilisé par

Classe	*Méthode*
XQuery 307	setXQueryUpdateVersion 318

.

6.4.2.11 ENUMXQueryVersion

Définit la version XQuery à utiliser pour le traitement (exécution ou validation).

- **eXQVersion10**: Définit la version XQuery à XQuery 1.0.
- **eXQVersion30**: Définit la version XQuery à XQuery 3.0. The default value.
- **eXQVersion31**: Définit la version XQuery à XQuery 3.1.

Note : les littéraux d'énumération Java sont nommés différemment que les littéraux COM/.NET. *Voir ci-dessous.*

COM et .NET

eXQVersion10	= 1
eXQVersion30	= 3
eXQVersion31	= 31

Utilisé par

Interface	*Propriété*
IXQuery 307	EngineVersion 311

Java

```
public enum ENUMXQueryVersion {
       eVersion10
       eVersion30
       eVersion31 }
```

Utilisé par

Classe	*Méthode*
XQuery 307	setEngineVersion 311

6.4.2.12 ENUMXSDVersion

Spécifie la version de Schéma XML à utiliser pour la validation.

- **eXSDVersionAuto**: La version de Schéma XML est détectée automatiquement depuis l'attribut `vc:minVersion du` document XSD. Si la valeur de cet attribut est `1.1`, alors le document est considéré être XSD 1.1. Si l'attribut a une autre valeur, ou si aucune valeur n'existe, le document est considéré comme étant un XSD 1.0.
- **eXSDVersion10**: Définit la version de Schéma XML pour une validation au Schéma XML 1.0.
- **eXSDVersion11**: Définit la version de Schéma XML pour une validation au Schéma XML 1.1.

COM et .NET

eXSDVersionAuto	= 0
eXSDVersion10	= 1
eXSDVersion11	= 2

Utilisé par

Interface	*Propriété*
IXMLValidator 295	XSDVersion 306
IXQuery 307	XSDVersion 318
IXSLT 319	XSDVersion 329

Java

```
public enum ENUMXSDVersion {
       eXSDVersionAuto
       eXSDVersion10
       eXSDVersion11   }
```

Utilisé par

Classe	*Méthode*
XMLValidator 295	setXSDVersion 306
XQuery 307	setXSDVersion 318
XSLT 319	setXSDVersion 329

6.4.2.13 ENUMXSLTVersion

Définit la version XSLT à utiliser pour le traitement (validation ou la transformation XSLT).

- **eVersion10**: Définit la version XSLT sur XSLT 1.0.
- **eVersion20**: Définit la version XSLT sur XSLT 2.0.
- **eVersion30**: Définit la version XSLT sur XSLT 3.0.

COM et .NET

eVersion10	= 1
eVersion20	= 2
eVersion30	= 3

Utilisé par

Interface	*Propriété*
IXSLT 319	EngineVersion 323

Java

```
public enum ENUMXSLTVersion {
       eVersion10
       eVersion20
       eVersion30 }
```

Utilisé par

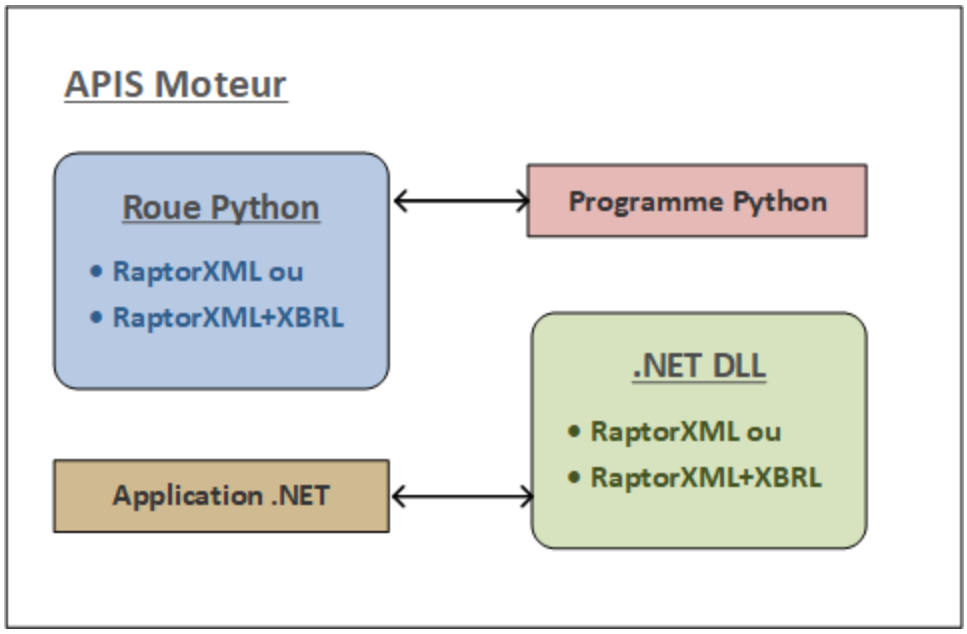
Classe	*Méthode*
XSLT 319	setEngineVersion 323

7 API de moteur : Python et .NET

RaptorXML Server fournit deux API de moteur :

- Un fichier wheel Python (`.whl`), qui est l'API de moteur Python : `raptorxml<versiondetails>.whl`
- Un fichier DLL .NET (`.dll`), qui est l'API de moteur .NET : `raptorxmlapi.dll`

Ces deux API de moteur proposent les fonctions RaptorXML Server en tant que paquets séparés indépendants de RaptorXML Server (*voir figure ci-dessous*). Chaque paquet doit être installé sur l'appareil de l'utilisateur avant de pouvoir être importé en tant que module Python ou être intégré dans une application .NET personnalisée. Étant donné que tout ce traitement est effectué localement sur l'appareil de l'utilisateur, les API de moteur Python et .NET fournissent un accès détaillé aux modèles de toute instance XML et XBRL valide, schémas XSD et taxonomies XBRL. Les API présentent un vaste ensemble de méthodes pour itérer sur le contenu des instances XBRL ou pour permettre d'extraire des fragments spécifiques d'information depuis les taxonomies XBRL avec quelques lignes de code.



Veuillez noter les points suivants concernant les API de moteur :

- Une fois avoir installé RaptorXML Server, les deux API de moteur seront situés dans le dossier `bin` du dossier d'installation de RaptorXML Server.
- Les API de moteur fournissent un traitement avancé supplémentaire par le biais d'objets plus versatiles dans leurs API.
- Afin d'utiliser un API de moteur, une version licencée de RaptorXML Server doit être installé sur l'appareil sur lequel le programme Python ou l'application .NET est exécutée (voir *Utilisation* ci-dessous).

Utilisation

Vous pouvez créer un programme Python ou une application .NET comme suit :

Programme Python
Un programme Python peut accéder aux fonctions RaptorXML en utilisant des objets API Python[346] (voir ici[346]). Lorsque programme Python est exécuté, il utilisera la bibliothèque RaptorXML qui a été installée dans votre environnement Python lorsque vous installez la roue Python. Veuillez noter que la roue Python est compatible avec la version Python 3.11.5.

Application .NET
Une application .NET peut accéder aux fonctions RaptorXML en utilisant des objets API .NET[356] (voir ici[346]). Lorsque l'application .NET est exécutée, elle utilisera le RaptorXML qui est contenu dans la DLL d'API .NET.

Licence

Afin d'utiliser un API de moteur, une version licencée de RaptorXML Server doit être installée sur l'appareil sur lequel le programme Python ou l'application .NET est exécutée. Voir la section License[344] pour plus d'information.

7.1 Licence

Pour exécuter un pack API sur un appareil client, cette machine doit être mise sous licence en tant que client RaptorXML Server. La mise sous licence consiste en deux étapes :

1. Enregistrer la machine en tant qu'un client de RaptorXML Server avec Altova LicenseServer
2. Attribuer à RaptorXML Server une licence de LicenseServer sur cet appareil.

Si vous projetez d'utiliser le pack d'API provenant d'un appareil particulier, deux situations probables apparaissent :

- Si la machine client exécute d'ores et déjà une installation sous licence de RaptorXML Server, alors le pack API peut être exécuté sans qu'il vous soit nécessaire de prendre des mesures supplémentaires. Cela est dû au fait que la machine est déjà mise sous licence pour exécuter RaptorXML Server. Par conséquent, l'utilisation du pack d'API sur cet appareil est couverte par la licence attribuée à RaptorXML Server sur cet appareil.
- Si RaptorXML Server n'est pas installé sur cet appareil client, et que vous ne souhaitez pas installer RaptorXML Server sur cet appareil pour une raison quelconque. Dans ce cas, vous pouvez toujours enregistrer l'appareil en tant qu'un client RaptorXML Server et l'attribuer à une licence RaptorXML Server. Suivre les étapes suivantes :

Pour enregistrer un appareil (sur lequel RaptorXML Server n'est pas installé) en tant qu'un client RaptorXML Server, utiliser l'application de ligne de commande **`registerlicense.exe`**, qui se trouve dans le dossier **`bin`** de l'application :

Windows	`Program Files\Altova\RaptorXMLServer2024\bin`
Linux	`/opt/Altova/RaptorXMLServer2024/bin`
Mac	`/usr/local/Altova/RaptorXMLServer2024/bin`

Dans la ligne de commande, exécuter la commande :

```
registerlicense <LicenseServer>
```

où **`<LicenseServer>`** est l'adresse IP ou le nom d'hôte de l'appareil de LicenseServer.

Cette commande enregistrera l'appareil en tant qu'un client RaptorXML Server auprès de Altova LicenseServer. Pour plus d'informations concernant comment attribuer une licence RaptorXML Server à l'appareil et pour plus d'informations concernant la mise sous licence, consulter la documentation LicenseServer Altova.

Déployer sur Linux

Pour déployer l'application **`registerlicense`** avec votre pack Python wheel, les bibliothèques partagées qui sont listées ci-dessous doivent être présentes dans un répertoire **`lib`** frère. Les librairies partagées peuvent être copiées depuis votre dossier d'installation Raptor :

```
/opt/Altova/RaptorXMLServerRaptorXMLServer2024/lib
```

- libcrypto.so.1.0.0
- libssl.so.1.0.0
- libstdc++.so.6
- libtbb.so.2

7.2 Python API

L'API Python RaptorXML permet aux données contenues dans des documents XML et des documents de Schéma XML d'être accédés et manipulés dans des scripts Python. Voici quelques exemples types d'une utilisation des API Python :

- mise en œuvre des règles de validation personnalisées et messages d'erreur
- exporter du contenu depuis des documents XML vers une base de données
- exporter du contenu depuis des documents XML vers des formats de données personnalisés
- naviguer et interroger de manière interactive le modèle de données des documents XML dans un shell Python ou un notebook Jupyter (http://jupyter.org/)

Les API de Python

Les API de Python (pour XML et XSD) fournissent un accès à la méta-information, à l'information structurelle et aux données contenues dans XML et XSD. En guise de résultat, les scripts Python peuvent être créés qui utilisent les API pour accéder et traiter l'information de document. Par exemple, un script Python peut être passé à RaptorXML Server qui écrit les données d'un document XML vers une base de données ou un fichier CSV.

Les scripts d'exemple pour les API Python de Raptor sont disponibles sous : https://github.com/altova

Les API de Python sont décrites dans leurs références API :

- Référence API Python v1
- Référence API Python v2

Note : Python API v1 **de Raptor est obsolète**. Veuillez utiliser Python API v2.

Pack RaptorXML Server pour Python

Dans votre installation de RaptorXML Server, vous trouverez aussi un Package de Python en format Roue. Vous pouvez utiliser la commande `pip` de Python pour installer ce package en tant que module de votre installation Python. Une fois que le module RaptorXML a été installé, vous pouvez utiliser les fonctions du module dans votre code. De cette manière, les fonctions de RaptorXML peuvent être utilisées aisément dans tout programme Python que vous écrivez, avec d'autres bibliothèques Python tierces comme des librairies de graphique.

Pour plus d'information concernant l'utilisation du package Python de RaptorXML Server, voir la section RaptorXML Server en tant que package Python 349.

Note : La roue Python est uniquement compatible avec la version Python 3.11.5.

Scripts Python

Un script Python utilisé par l'utilisateur est soumis avec le paramètre `--script` d#un certain nombre de commandes, y compris :

- `valxml-withxsd (xsi)` 63
- `valxsd (xsd)` 74

Ces commandes invoquant des scripts Python peuvent être utilisées aussi bien sur l'Interface de ligne de commande (CLI) 57 qu'avec l'interface HTTP 227. L'utilisation des scripts Python avec les API Python de RaptorXML Server est décrite sous : https://github.com/altova.

Rendre les scripts Python sûrs

Lorsqu'un script Python est spécifié dans une commande via HTTP pour RaptorXML Server, le script fonctionnera uniquement s'il est situé dans le répertoire de confiance 233. Le script est exécuté depuis le répertoire de confiance. La spécification d'un script Python provenant d'un autre répertoire quelconque résultera en une erreur. Le répertoire de confiance est spécifié dans le paramètre `server.script-root-dir` 232 du fichier de configuration de serveur 231, et un répertoire de confiance **doit** être spécifié si vous souhaitez utiliser des scripts Python. Veuillez vous assurer que tous les scripts Python à utiliser soient sauvegardés dans ce répertoire.

Bien que toutes les sorties générées par le serveur pour les requêtes de tâche HTTP soient écrites dans le répertoire de sortie de tâche 233 (qui est un sous-répertoire de `output-root-directory` 233), cette limitation ne s'applique pas aux scripts Python, qui peuvent écrire dans n'importe quel emplacement. L'administrateur de serveur doit revoir les scripts Python dans le répertoire de confiance 233 en ce qui concerne des problèmes de vulnérabilité potentiels.

7.2.1 Versions d'API Python

RaptorXML Server prend en charge plusieurs versions d'API Python. Toute version d'API Python précédente est aussi prise en charge par la version actuelle de RaptorXML Server. La version d'API Python est sélectionnée par le flag de ligne de commande **`--script-api-version=MAJOR_VERSION`**. Le défaut de l'argument de `MAJOR_VERSION` est toujours la version actuelle. Un nouvel API Python `MAJOR_VERSION` de RaptorXML Server est introduit lorsque des changements incompatibles ou des améliorations importantes sont introduites. Les utilisateurs de l'API ne doivent pas mettre à niveau leurs scripts existants lorsqu'une nouvelle version majeure est publiée.

Il est recommandé de :

- utiliser le flag `--script-api-version=MAJOR_VERSION` pour invoquer des scripts utilitaires depuis la ligne de commande RaptorXML Server (ou l'API Web). Cela vous garantit que les scripts fonctionnent toujours comme prévu après les mises à jour de RaptorXML Server, même si une nouvelle `MAJOR_VERSION` d'API a été publiée.
- Vous utilisez la dernière version de l'API pour de nouveaux projets, même si des versions précédentes seront prises en charge par des releases futures de RaptorXML Server.

Les versions API Python recensées ci-dessous sont disponibles actuellement. La documentation des différents API est disponible en ligne dans les emplacements indiqués ci-dessous.

Exemples de fichiers

Pour des exemples de script qui utilisent les API Python de Raptor, allez à https://github.com/altova.

API Python version 1

Introduit avec la version v2014 de RaptorXML Server

Flag de ligne de commande : `--script-api-version=1`
Documentation : Python API Version 1 Reference

Voici l'API Python de RaptorXML Server original. Il couvre la prise en charge pour accéder au modèle interne de RaptorXML Server pour :

- XML 1.0 et XML 1.1 (module API **`xml`**)
- Schéma XML 1.0 et Schéma XML 1.1 (module API **`xsd`**)
- XBRL 2.1 (module API **`xbrl`**)

L'API peut être utilisé par le biais de fonctions de callback qui sont mises en place dans un fichier de script Python.

- `on_xsi_valid`
- `on_xsd_valid`
- `on_dts_valid`
- `on_xbrl_valid`

Un script est spécifié avec l'option `--script` sur la ligne de commande. Les fonctions de callback sont invoquées uniquement si la validation réussit. Les détails concernant les fonctions de rappel et l'API sont décrits dans la référence "Python API version 1" de RaptorXML Server.

Note : Python API v1 **de Raptor est obsolète**. Veuillez utiliser Python API v2.

API Python version 2

Introduit avec la version v2015r3 de RaptorXML Server. La dernière version API est `2.9.0`.

Flag de ligne de commande	Release
`--script-api-version=2`	*v 2015r3*
`--script-api-version=2,1`	*v 2015r4*
`--script-api-version=2,2`	*v 2016*
`--script-api-version=2,3`	*v 2016r2*
`--script-api-version=2,4`	*v 2017*
`--script-api-version=2.4.1`	*v 2018*
`--script-api-version=2.5.0`	*v 2018r2*
`--script-api-version=2.6.0`	*v 2019*
`--script-api-version=2.7.0`	*v2019r3*
`--script-api-version=2.8.0`	*v2020*
`--script-api-version=2.8.1`	*v2020r2*
`--script-api-version=2.8.2`	*v2021*

`--script-api-version=2.8.3`	*v2021r2*
`--script-api-version=2.8.4`	*v2022r2*
`--script-api-version=2.8.5`	*v2023r2sp1*
`--script-api-version=2.8.6`	*v2024*
`--script-api-version=2.9.0`	*v2024r2*
Documentation : Python API Version 2 Reference	

Cette version API introduit plus de 300 nouvelles classes et réorganise les modules depuis l'API Python version 1 de RaptorXML Server de manière à ce que des informations utilisées fréquemment (par exemple des données PSVI) peuvent être accédées plus simplement et des API liés sont regroupés logiquement (par exemple, `xbrl.taxonomy`, `xbrl.formula`, `xbrl.table`). Dans cette version, les fonctions de callback sont invoquées non seulement si la validation réussit, mais aussi si la validation échoue. Pour illustrer ce comportement, le nom des fonctions de callback a été changé en :

- `on_xsi_finished`
- `on_xsd_finished`
- `on_dts_finished`
- `on_xbrl_finished`

Pour activer la modularisation, RaptorXML Server prend maintenant en charge plusieurs options `--script`. Les callbacks implémentés dans ces fichiers script Python sont exécutés dans l'ordre spécifié dans la ligne de commande.

7.2.2 RaptorXML Server en tant que package Python

À partir de RaptorXML Server 2024, l'API Python est disponible en tant que package wheel Python natif pour **Python 3.11.5**. Le package wheel Python peut être installé en tant que module d'extension avant votre distribution Python 3.11.5 préférée (par exemple, depuis python.org). Certaines distributions Python 3 (par exemple, provenant de jupyter.org, anaconda.org et SciPy.org) comprennent une palette étendue de modules d'extension pour les big data, les mathématiques, la science, l'ingénierie et les graphiques. Ces modules sont maintenant disponibles pour RaptorXML Server sans qu'il soit nécessaire de générer ces modules spécifiquement pour RaptorXML Server. Sinon, le package wheel fonctionne de la même manière que l'application **`RaptorXMLXBRL-python.exe`** qui est contenue dans RaptorXML Server.

Note : le package wheel Python est un module d'extension natif 3.11.5 et il est compatible avec la version Python 3.11.5.

Note : le package wheel Python ne contient pas le Python API v1.

Note : si vous effectuez une mise à jour de votre version de RaptorXML Server, assurez-vous de mettre à jour le package wheel Python dans votre environnement Python.

L'information nécessaire pour installer correctement le package RaptorXML Server est indiquée dans les sections ci-dessous :

Pour plus d'informations concernant l'utilisation de l'API Python de RaptorXML Server, voir référence API Python et exemples 347. Voir aussi les scripts d'exemple qui utilisent l'API Python de Raptor sous https://github.com/altova.

Nom du fichier wheel

Les fichiers wheel sont nommés conformément au motif suivant :

```
raptorxmlserver-{version}(-{build tag})?-{python tag}-{abi tag}-{platform tag}.whl
```

Exemple:

```
raptorxmlserver-2.9.0-cp35-cp35m-win_amd64.whl
```

Emplacement du fichier wheel

Un fichier wheel est contenu avec votre installation de RaptorXML Server. Il se trouve dans le dossier **bin** de l'application :

Fenêtres	`Program Files\Altova\RaptorXMLServer2024\bin`
Linux	`/opt/Altova/RaptorXMLServer2024/bin`
Mac	`/usr/local/Altova/RaptorXMLServer2024/bin`

Installer un wheel avec pip

Pour installer le package RaptorXML Server en tant que module Python, utiliser la commande **pip** :

```
pip install <wheel-file>.whl
python -m pip install <wheel-file>.whl
```

Quand vous avez installé Python 3.11.5 ou la version plus élevée depuis python.org, alors `pip` sera déjà installée. Sinon, vous devrez tout d'abord installer **pip**. Pour plus d'informations, voir https://docs.python.org/3/installing/.

Dépannage de l'installation

Dans le cas où vous utilisez des versions plus anciennes de l'interpréteur Python, vous allez éventuellement devoir ajuster votre installation pour utiliser les dernières bibliothèques **vcruntime** sur Windows ou des bibliothèques standard C++ sur Unix. Ces bibliothèques sont distribuées avec RaptorXML Serveret peuvent être utilisées tel que décrit ci-dessous.

Windows

Si **vcruntime140_1.dll** manque, copiez-le depuis le dossier `Program Files\Altova\RaptorXMLServer2024\bin` vers le dossier d'installation Python (le dossier contenant

python.exe). (De manière plus générale, l'interpréteur Python doit savoir où il trouve les DLL ou les bibliothèques partagées.)

Linux
Si votre bibliothèque C++ de système est obsolète, alors votre interpréteur Python ne saura pas où trouver la bibliothèque C++ la plus récente qui est utilisée par le package Python RaptorXML Server et distribuée avec RaptorXML Server. Ceci peut être réglé en utilisant **$LD_LIBRARY_PATH** pour pointer vers la plus récente bibliothèque dans le dossier RaptorXML Server, tel que : `$ export LD_LIBRARY_PATH=/opt/Altova/RaptorXMLServer2024/lib`.

macOS
Si votre bibliothèque C++ de système est obsolète, alors votre interpréteur Python ne saura pas où trouver la bibliothèque C++ la plus récente qui est utilisée par le package Python RaptorXML Server et distribuée avec RaptorXML Server. Ceci peut être réglé en utilisant **$DYLD_LIBRARY_PATH** pour pointer vers la plus récente bibliothèque dans le dossier RaptorXML Server, tel que : `$ export DYLD_LIBRARY_PATH=/usr/local/Altova/RaptorXMLServer2024/lib`.

Le fichier root catalog

Le module RaptorXML pour Python doit pouvoir situer **RootCatalog.xml**, le fichier de catalogue root qui est stocké dans votre dossier d'installation RaptorXML Server. La raison est que le module RaptorXML peut utiliser le catalogue pour situer correctement les différentes ressources, comme des schémas et d'autres spécifications, que le module référence pour pouvoir effectuer des fonctions comme des validations et des transformations. Le module RaptorXML situera automatiquement **RootCatalog.xml** si l'emplacement du catalogue n'a pas été modifié après l'installation de RaptorXML Server.

Au cas où vous déplacez ou modifiez votre environnement RaptorXML Server, ou si vous déplacez **RootCatalog.xml** depuis son emplacement d'origine, vous pouvez spécifier l'emplacement du catalogue par le moyen des variables d'environnement et le fichier config JSON du module RaptorXML 352. Voir la liste ci-dessous pour consulter les moyens se trouvant à votre disposition. Le module RaptorXML détermine l'emplacement de **RootCatalog.xml** en consultant les ressources suivantes dans l'ordre donné.

1	Variable d'environnement `ALTOVA_RAPTORXML_PYTHON_CATALOGPATH`	Créer avec une valeur qui est le chemin vers **RootCatalog.xml**
2	Registre HKLM : `SOFTWARE\Altova\RaptorXMLServer\Installation_v2024_x64\Setup\CatalogPath`	La clé de registre est ajoutée par l'installateur RaptorXML Server. La valeur est le chemin vers **RootCatalog.xml**. *Uniquement Windows*
3	Emplacement : `/opt/Altova/RaptorXMLServer2024/etc/RootCatalog.xml`	*Uniquement Linux*
4	Emplacement : `/usr/local/Altova/RaptorXMLServer2024/etc/RootCatalog.xml`	*Uniquement Mac*
5	Variable d'environnement `ALTOVA_RAPTORXML_PYTHON_CONFIG`	Créer avec une valeur qui est le chemin vers le fichier de config JSON 352.

6	Emplacement : `.altova/raptorxml-python.config`	Le fichier de config JSON[352] dans le répertoire de travail actuel
7	Emplacement : `~/.config/altova/raptorxml-python.config`	Le fichier de config JSON[352] dans le répertoire home de l'utilisateur
8	Emplacement : `/etc/altova/altova/raptorxml-python.config`	Le fichier de config JSON[352]. *Uniquement Linux et Mac*

Le fichier de config JSON

Vous pouvez créer un fichier de config JSON pour le module RaptorXMLServer. Ce fichier sera utilisé par les options 5 à 8 dans la table ci-dessus pour situer le fichier de catalogue root[351]. Le fichier de config JSON doit contenir un map avec une clé `"CatalogPath"` qui a une valeur qui est le chemin vers le fichier de catalogue root[351].

Liste du fichier config JSON

```
{
    "CatalogPath": "/path/to/RootCatalog.xml"
}
```

7.2.3 Déboguer Scripts Phython côté Serveur

La plupart des fonctionnalités de débogage—à l'exception des callbacks spécifiques au serveur—peuvent être utilisées dans un interpréteur Python standard ou dans un environnement (virtuel) après avoir installé le module RaptorXML Server en utilisant `pip`:

```
pip install –upgrade "/path/to/RaptorXML/application-folder/bin/raptorxml-version-cp37-cp37m-winversion.whl"
```

Après avoir installé la roue, vous devriez pouvoir utiliser toute IDE Python pour déboguer un script. Vous pourriez essayer d'extraire la fonctionnalité principale vers une fonction séparée qui prend un objet instance. Ceci peut ensuite être appelé (i) par les callbacks de RaptorXML Server, ou (ii) en exécutant directement le script avec un interpréteur Python.

```
from altova_api.v2 import xml, xsd, xbrl

def main(instance):
    # Here goes the application specific logic

# Main entry point, will be called by RaptorXML after the XML instance validation job has finished
def on_xsi_finished(job, instance):
    # instance object will be None if XML Schema validation was not successful
    if instance:
        main(instance)
```



```
# Main entry point, will be called by RaptorXML after the XBRL instance validation job has
finished
def on_xbrl_finished(job, instance):
    # instance object will be None if XBRL 2.1 validation was not successful
    if instance:
        main(instance)

if __name__ == '__main__':
    # parse arguments and create an instance
    instance = …
    main(instance)
```

7.2.4 Déboguer Scripts Python dans Visual Studio Code

Nous assumons l'installation à jour de Visual Studio Code (VS Code) avec l'extension **ms-python.python** installée. Veuillez lire les configurations Python debug dans le guide Visual Studio Code officiel pour un aperçu général.

Veuillez noter les points suivants :

- Ce guide utilise **raptorxml-python** en tant que commande pour exécuter RaptorXML Server en tant qu'interpréteur Python.
- L'exécutable **raptorxml-python** est disponible dans le dossier **bin** de votre dossier d'application RaptorXML Server.

Aperçu

Nous présentons deux méthodes pour utiliser VS Code afin de déboguer les scripts Python dans RaptorXML Server.

- La Méthode 1 fonctionne également pour les serveurs et les rappels de RaptorXML Python (option de **--script**).
- La Méthode 2 ne requiert aucune modification de code source. Il s'agit d'une invocation modifiée de RaptorXML. La Méthode 2 ne fonctionne pas pour les serveurs ni pour les rappels de RaptorXML Python (option de **--script**).
- Les deux méthodes fonctionnent avec un interpréteur Python standard et le module RaptorXML Python intégré (**'import altova_api.v2 as altova'**).

Méthode 1 : changer votre code source

Exécutez les points suivants :

1. Exécutez : **raptorxml-python -m pip install --upgrade debugpy**
2. Ajoutez les lignes suivantes à votre code source Python :

```
python
import debugpy
debugpy.listen(5678)
debugpy.wait_for_client()
debugpy.breakpoint()
```

3. Copiez cette configuration de lancement vers VS Code **launch.json** (les valeurs par défaut permettront les valeurs ci-dessus) et sélectionnez-la pour **Run**.

```
json5
  {
    "name": "Python: Remote Attach",
    "type": "python",
    "request": "attach",
    "connect": {
    "host": "localhost",
    "port": 5678
    },
    "pathMappings": [
      {
        "localRoot": "${workspaceFolder}",
        "remoteRoot": "."
      }
    ]
  }
```

   Vous pouvez également l'exécuter en utilisant la commande de menu **Run->Add Configuration...->Python->Remote Attach** avec les valeurs par défaut acceptées.

4. Exécutez votre script Python (ou RaptorXML avec les rappels **--script**) comme d'habitude.
5. Commencez à déboguer (normalement avec le raccourci **F5**).

Méthode 2 : Utiliser une ligne de commande modifiée

Réalisez les étapes suivantes :

1. Ajoutez la configuration de lancement (comme pour la Méthode 1 ci-dessus), et sélectionnez-la pour **Run**.
2. Définissez un point d'arrêt dans votre script Python.
3. Exécutez la commande: **raptorxml-python -m debugpy --listen 0.0.0.0:5678 --wait-for-client your-script-.py**
4. Commencez à déboguer (normalement avec le raccourci **F5**).

Note : Le débogage fonctionne également avec les conteneurs et serveurs distants. Vous devez changer la clé **host** de l'entrée **connect** dans la configuration de lancement. Vous pouvez aussi utiliser d'autres ports à condition que les lignes de code ou de commande et **launch.json** ont des valeurs consistantes.

Définir raptorxml-python.exe en tant qu'interprète par défaut de VS Code

Il est possible de configurer **raptorxml-python.exe** en tant qu'interprète Python par défaut de VS Code. Configurez en ajoutant ce qui suit à votre code VS **settigs.json** file:

```
json
  "python.defaultInterpreterPath": "/path/to/raptorxml-python.exe"
   ...
```

Dans ce cas, il est possible d'utiliser une configuration de lancement dite "Current File" qui lance le scripte pour déboguer. Consultez la documentation VS Code officielle pour tout détail.

7.2.5 Questions fréquemment posées

Q: *Je veux écrire un script Python qui crée une nouvelle instance XML, un élément à la fois, pendant qu'il s'exécute à l'intérieur du serveur raptor. Ceux-ci doivent être sérialisés vers la sortie avec différents encodages et formatages dépendant des paramètres. Est-ce possible dans %APPNAME%> ?*

A: Non, ceci n'est actuellement pas possible parce que nous ne disposons pas d'API pour créer des instances XML arbitraires. Toutefois, quand il s'agit de générer des instances XBRL, nous avons un API qui gère de nombreux détails techniques (tels qu'éviter d'écrire des doublons de contextes/unités, et bien plus). Voir https://www.altova.com/manual/en/raptorapi/pyapiv2/2.9.0/html/xbrl.InstanceDocumentBuilder.html pour plus d'information.

Q: *J'aimerais utiliser lxml. Est-ce que je peux installer les bibliothèques lxml dans le dossier Python sous « RaptorXMLXBRLServer2024/lib/ » ?*

A: Vous pouvez installer directement la plupart des modules Python en exécutant la commande suivante dans un terminal qui a des droits d'administrateur :

```
"/path/to/RaptorXML/application-folder/bin/RaptorXMLXBRL-python.exe" -m pip install lxml
```

Q: *Est-ce que je peux créer un grand string qui contient l'instance XML, pour parser ensuite le tout et le resérialiser ?*

A: C'est une possibilité. Vous pouvez parser et valider des instances XML et XBRL depuis un tampon de chaînes utilisant un API Python comme suit :

```
from altova_api.v2 import xml
txt = '''<?xml version="1.0" encoding="utf-8"?>
<doc>
    <elem attr="foo">bar</elem>
</doc>'''
inst = xml.Instance.create_from_buffer(txt.encode('utf-8')).result
print(inst.serialize())
```

7.3 .NET Framework API

L'**API .NET Framework** de RaptorXML Server vous permet d'intégrer **le moteur RaptorXML** dans des applications écrites en C# et d'autres langages .NET.

Elle est mise en place en tant qu'un assembly .NET et place **le moteur RaptorXML** directement à l'intérieur d'une application ou d'un mécanisme d'extension basé sur un .NET-framework comme VSTO (Visual Studio Tools for Office). L'API propose un accès granulaire pour valider les documents et pour requêter leur modèle de données interne depuis RaptorXML Server.

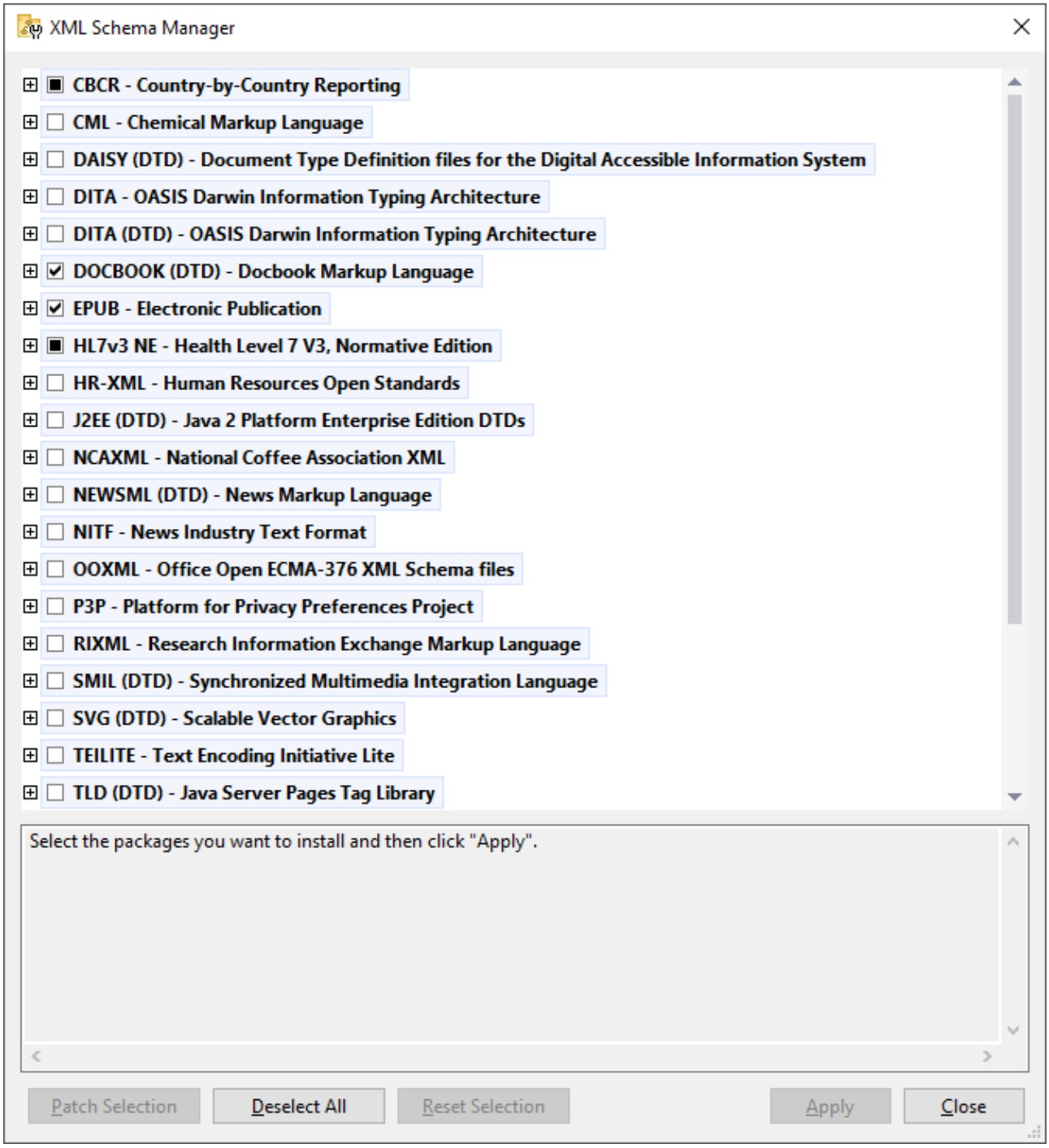
Références et ressources

- *Documentation API :* vous trouverez la documentation API .NET Framework de RaptorXML Server la plus récente sous https://www.altova.com/manual/en/raptorapi/dotnetapiv2/2.9.0/html/index.html.
- *Code d'exemple :* le code d'exemple est hébergé sous https://github.com/altova/RaptorXML-Examples.

8 Gestionnaire de schéma

XML Gestionnaire de schéma est un outil qui propose un moyen centralisé d'installer et de gérer des schémas XML (DTD pour XML et Schémas XML) pour une utilisation sur toutes les applications « XML-Schema-aware » d'Altova, y compris RaptorXML Server

- Sur Windows, Gestionnaire de schéma a une interface utilisateur graphique *(voir la capture d'écran ci-dessous*) et est aussi disponible dans la ligne de commande. (Les applications desktop d'Altova sont disponibles sur Windows uniquement ; *voir la liste ci-dessous*.)
- Sur Linux et Mac Gestionnaire de schéma, l'outil est disponible uniquement dans la ligne de commande. (Les applications serveur d'Altova sont disponibles sur Windows, Linux et macOS ; *voir la liste ci-dessous*.)



Application d'Altova qui fonctionnent avec Schema Manager

Applications desktop (Windows uniquement)	Applications de serveur (Windows, Linux, macOS)
XMLSpy (toutes éditions)	RaptorXML Server, RaptorXML+XBRL Server

MapForce (toutes éditions)	StyleVision Server
StyleVision (toutes éditions)	
Authentic Desktop Enterprise Edition	

Installation et désinstallation de Gestionnaire de schéma

Gestionnaire de schéma est installé automatiquement quand vous installez d'abord une nouvelle version de l'Altova Mission Kit ou toute application « XML-schema-aware » d'Altova (*voir la table ci-dessus*).

De même, il est supprimé automatiquement lorsque vous désinstallez la dernière application XML-schema-aware d'Altova depuis votre ordinateur.

Fonctions <% SCHEMA-MANAGER%>

Gestionnaire de schéma propose les fonctions suivantes :

- Affiche les schémas XML installés sur votre ordinateur et contrôle si de nouvelles versions sont disponibles pour le téléchargement.
- Télécharge des versions plus récentes des schémas XML indépendamment du cycle de release des produits Altova. (Altova stocke des schémas en ligne et vous pouvez les télécharger via Gestionnaire de schéma.)
- Installer ou désinstaller une des multiples versions d'un schéma donné (ou toutes les versions, si nécessaire).
- Un schéma XML peut avoir des dépendances sur d'autres schémas. Lorsque vous installez ou désinstallez un schéma particulier, Gestionnaire de schéma vous informe sur d'autres schémas dépendants et les installera ou désinstallera également automatiquement.
- Gestionnaire de schéma utilise le mécanisme du catalogue XML pour mapper les références de schéma aux fichiers locaux. Dans le cas de larges schémas XML, le traitement sera plus rapide que si les schémas étaient à un emplacement à distance.
- Tous les schémas majeurs sont disponibles via Gestionnaire de schéma et sont régulièrement mis à jour pour les dernières versions. Ceci vous fournit une ressource unique pour gérer tous vos schémas et les mettre à disposition de toutes les applications « XML-schema-aware » d'Altova.
- Les changements réalisés dans Gestionnaire de schéma prennent effet pour tous les produits d'Altova sur cet appareil.

Comment cela fonctionne ?

Altova stocke tous les schémas XML utilisés dans les produits Altova en ligne. Ce référentiel est mis à jour lorsque de nouvelles versions de schémas sont publiées. Gestionnaire de schéma affiche des informations sur les derniers schémas disponibles lorsqu'ils sont appelés dans son formulaire GUI de même que sur CLI. Vous pouvez ensuite installer, mettre à jour ou désinstaller les schémas via Gestionnaire de schéma.

Gestionnaire de schéma installe également les schémas d'une autre manière. Sur le site web d'Altova (https://www.altova.com/schema-manager), vous pouvez sélectionner un schéma et ses Schémas dépendants que vous souhaitez installer. Le site web préparera un fichier de type `.altova_xmlschemas` pour le téléchargement qui contient des informations sur la sélection de schéma. Lorsque vous double-cliquez sur ce fichier ou le passez à Gestionnaire de schéma via CLI comme argument de la commande `installer` 371, Gestionnaire de schéma installera les schémas que vous avez sélectionnés.

Cache local : suivre vos schémas
Toutes les informations sur les schémas installés sont suivies dans un répertoire cache centralisé sur votre ordinateur, situé ici :

Windows	`C:\ProgramData\Altova\pkgs\.cache`
Linux	`/var/opt/Altova/pkgs\.cache`
macOS	`/var/Altova/pkgs`

Ce répertoire cache est mis à jour régulièrement avec le dernier statut des schémas dans l'emplacement de stockage en ligne d'Altova. Ces mises à jour sont réalisées aux moments suivants :

- À chaque fois que vous lancez Gestionnaire de schéma.
- Lorsque vous exécutez RaptorXML Server pour la première fois dans un jour donné du calendrier.
- Si RaptorXML Server est ouvert plus de 24 heures, le cache est mis à jour toutes les 24 heures.
- Vous pouvez aussi mettre à jour le cache en exécutant la commande de `mise à jour`[374] dans l'interface de ligne de commande.

Pour cette raison, le cache permet à Gestionnaire de schéma de suivre continuellement vos schémas installés par rapport aux schémas disponibles en ligne sur le site web d'Altova.

Ne modifiez pas le cache manuellement !

Le répertoire de cache local est entretenu automatiquement sur la base des schémas que vous installez ou désinstallez. Il ne devrait pas être altéré ou supprimé manuellement. Si vous êtes amené à réinitialiser Gestionnaire de schéma à son état original "intact", alors, sur l'interface de la ligne de commande (CLI) : (i) exécutez la commande `reset`[372], et (ii) exécutez la commande `initialize`[370]. (En alternative, exécutez la commande **`reset`** avec l'option **`--i`**.)

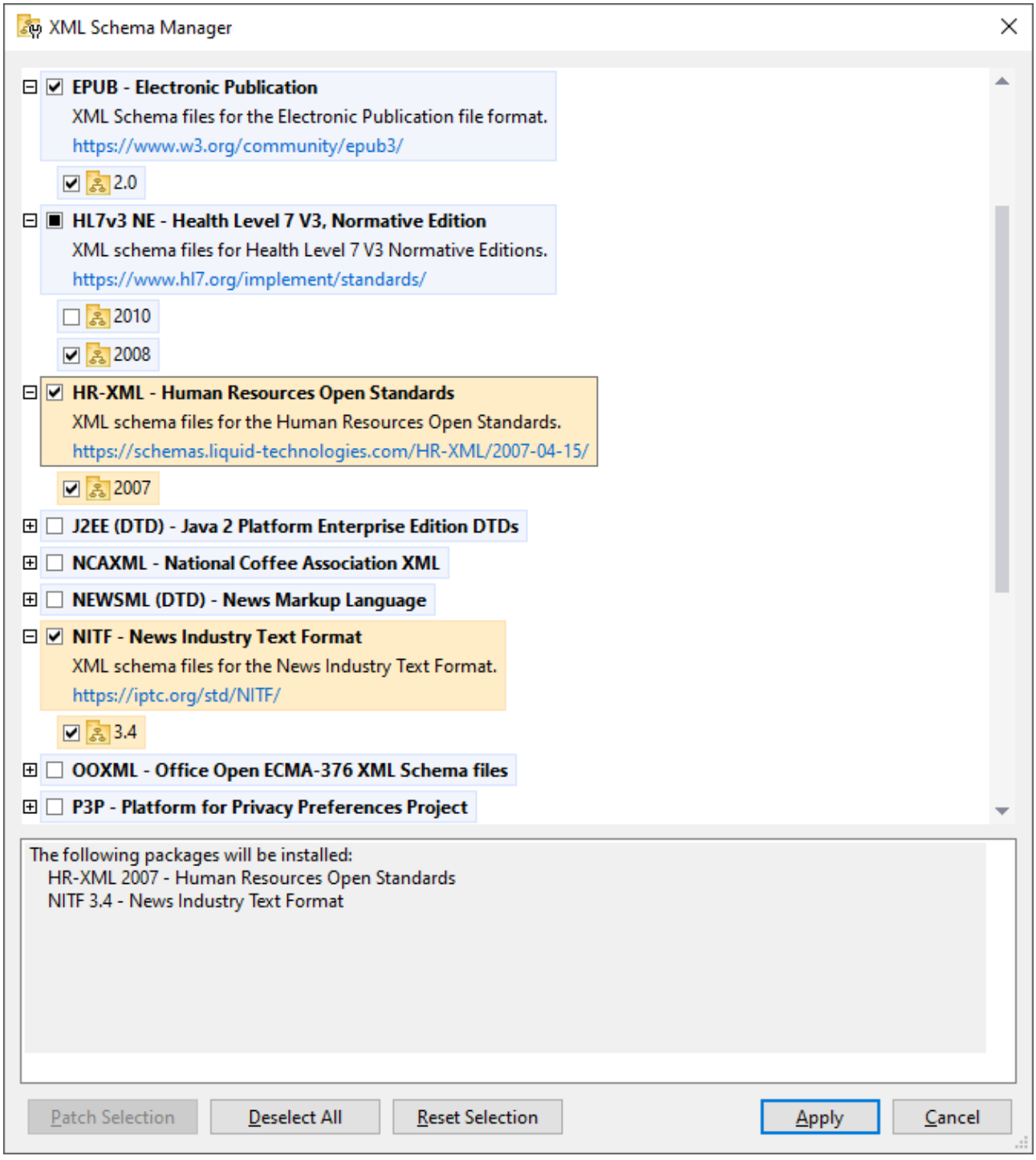
8.1 Exécuter Schema Manager

Interface utilisateur graphique

Vous pouvez accéder à la GUI de Gestionnaire de schéma des manières suivantes :

- *Durant l'installation de RaptorXML Server:* Vers la fin de la procédure d'installation, sélectionnez la case à cocher *Invoke Altova XML-Schema Manager* pour accéder directement à la GUI de Gestionnaire de schéma. Ceci vous permettra d'installer les schémas au cours de la procédure d'installation de votre application Altova.
- Via le fichier `.altova_taxonomies` téléchargé de Altova website: Double-cliquez sur le fichier téléchargé pour exécuter la GUI de Gestionnaire de schéma, qui sera configurée pour installer les schémas que vous avez sélectionnés (le site web) pour installation.

Une fois que la GUI de Gestionnaire de schéma (*capture d'écran ci-dessous*) a été ouverte, les schémas déjà installés seront affichés tels sélectionnés. Si vous voulez installer un schéma additionnel, sélectionnez-le. Si vous voulez désinstaller un schéma déjà installé, désélectionnez-le. Une fois que vous avez faits vos sélections et/ou désélections, vous êtes prêts pour appliquer vos changements. Les schémas qui seront installés ou désinstallés seront mis en surbrillance et un message sur les modifications à venir sera posté dans le volet Messages au niveau inférieur de la fenêtre Gestionnaire de schéma (*voir la capture d'écran*).



Interface de ligne de commande

Vous pouvez exécuter Gestionnaire de schéma depuis une interface de ligne de commande en sélectionnant son fichier exécutable, **`xmlschemamanager.exe`**.

Le fichier **`xmlschemamanager.exe`** est situé dans le dossier suivant :

- *Sur Windows :* `C:\ProgramData\Altova\SharedBetweenVersions`
- *Sur Linux ou macOS (application serveur uniquement) :* `%INSTALLDIR%/bin`, où `%INSTALLDIR%` est le répertoire d'installation du programme.

Vous pouvez alors utiliser toute commande dans la section de référence de la commande CLI 369.

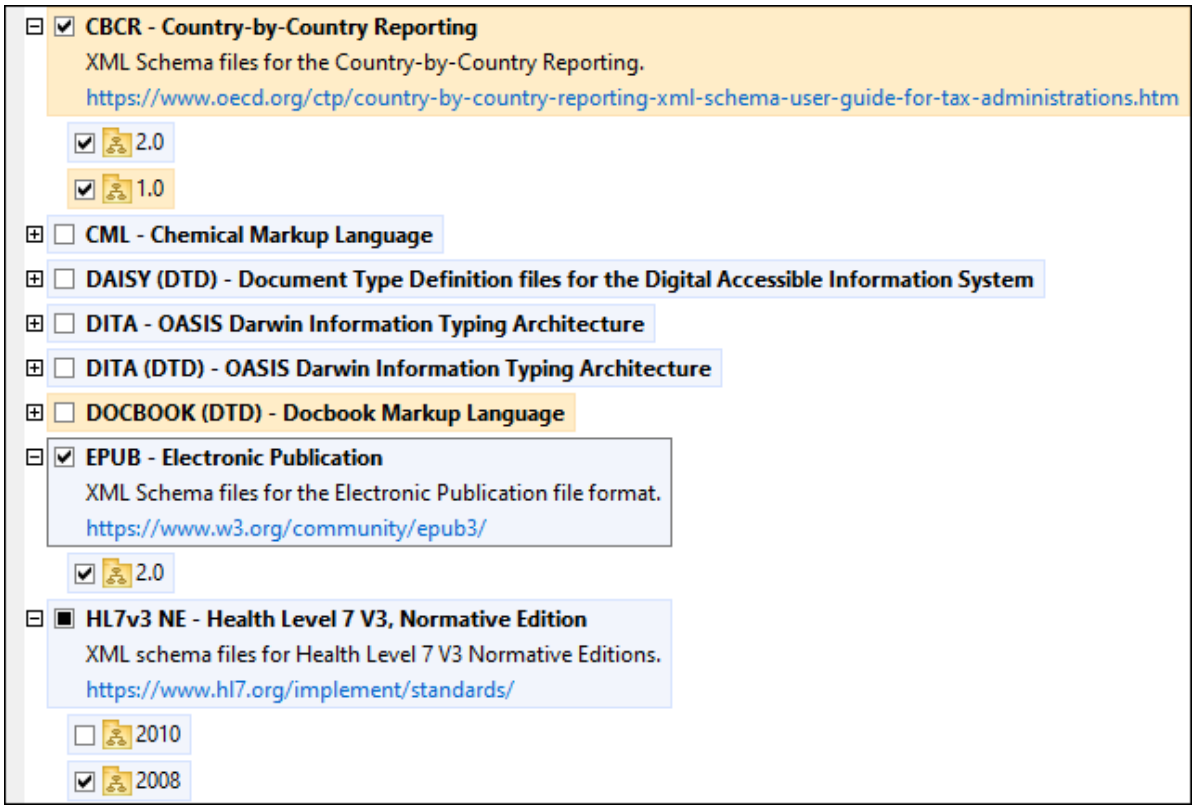
Pour afficher l'aide pour la commande, exécutez l'étape suivante :

- *Sur Windows :* `xmlschemamanager.exe --help`
- *Sur Linux ou macOS (application serveur uniquement) :* `sudo ./xmlschemamanager --help`

8.2 Catégories de statut

Gestionnaire de schéma catégorise les schémas sous sa gestion comme suit :

- *Schémas installés.* Ceux-ci sont affichés dans la GUI avec leurs cases à cocher sélectionnées (*dans la capture d'écran ci-dessous, les versions cochées et bleues des schémas EPUB et HL7v3 NE sont des schémas installés*). Si toutes les versions de schéma sont sélectionnées, alors la marque de sélection est une coche. Si au moins une version de schéma est décochée, alors la coche de sélection est un carré coloré plein. Vous pouvez décocher un schéma installé pour le **désinstaller** ; (*dans la capture d'écran ci-dessous, le DocBook DTD est installé et a été désélectionné, le préparant ainsi pour la désinstallation*).
- *Schémas désinstallés disponibles.* Ils sont affichés dans la GUI avec leurs cases à cocher non sélectionnées. Vous pouvez sélectionner les schémas que vous souhaitez **installer**.



- Les *Schémas pouvant être mises à niveau* sont ceux qui ont été revus par leurs émetteurs depuis qu'ils ont été installés. Ils sont indiqués dans la GUI par une icône . Vous pouvez **retoucher** le schéma installé avec la révision disponible.

Points à noter

- Dans la capture d'écran ci-dessus, les deux schémas CBCR sont cochés. Celui avec un arrière-plan bleu est déjà installé. Celui avec un arrière-plan jaune est désinstallé et a été sélectionné pour l'installation. Notez que le schéma HL7v3 NE 2010 n'est pas installé et n'a pas été sélectionné pour l'installation.
- Un arrière-plan jaune signifie que le schéma sera modifié d'une manière ou d'une autre quand le bouton **Appliquer** est cliqué. Si un schéma est décoché et a un arrière-plan jaune, cela signifie qu'il sera

désinstallé quand le bouton **Appliquer** est cliqué. Dans la capture d'écran ci-dessus, le DocBook DTD a un tel statut.

- Lorsque vous exécutez Gestionnaire de schéma depuis la ligne de commande, la commande list[371] est utilisée avec différentes options pour pour recenser les différentes catégories de schémas :

xmlschemamanager.exe list	Recense tous les schémas installés et disponibles ; ceux pouvant être mis à niveau sont également indiqués.
xmlschemamanager.exe list -i	Recense les schémas installés uniquement ; ceux pouvant être mis à niveau sont également indiqués
xmlschemamanager.exe list -u	Recense les schémas pouvant être mis à niveau

Note : Sur Linux et macOS, **use sudo ./ xmlschemamanager list**

8.3 Retoucher ou Installer un schéma

Retoucher un schéma installé

Occasionnellement, des schémas XML peuvent recevoir des patchs (mises à niveau ou révisions) de leurs émetteurs. Lorsque Gestionnaire de schéma détecte que des patchs sont disponibles, ceux-ci sont indiqués dans les listes de schéma de Gestionnaire de schéma et vous pouvez installer les patchs rapidement.

Dans la GUI

Les patchs sont indiqués par l'icône . (*Voir aussi la rubrique précédente sur les catégories de statut*[364].) Si les patchs sont disponibles, le bouton **Patch Selection** sera activé. Cliquez dessus pour sélectionner et préparer tous les patchs pour installation. Dans la GUI, l'icône de chaque schéma sera patchée de à , et le volet des Messages en bas du dialogue recense les patchs qui doivent être appliqués. Lorsque vous êtes prêt pour installer des patchs sélectionnés, cliquez sur **Appliquer**. Tous les correctifs seront appliqués ensemble. Notez que si vous décochez un schéma marqué pour une correction, vous désinstallerez de fait ce schéma.

Sur le CLI

Pour appliquer un patch dans l'interface de ligne de commande :

1. Exécuter la commande `list -u`[371] . Cela liste tout schéma lorsque des mises à niveau sont disponibles.
2. Exécutez la commande `upgrade`[374] pour installer les patchs.

Installer un schéma disponible

Vous pouvez installer des schémas en utilisant soit la GUI Gestionnaire de schéma ou en envoyant à Gestionnaire de schéma les instructions d'install via la ligne de commande.

Note : Si le schéma actuel référence d'autres schémas, les schémas référencées sont aussi installés.

Dans la GUI

Pour installer des schémas utilisant la GUI Gestionnaire de schéma GUI, sélectionnez les schémas que vous voulez installer et cliquez sur **Appliquer**.

Vous pouvez aussi sélectionner les schémas que vous voulez installer sur le site web d'Altova et générer un fichier téléchargeable `.altova_xmlschemas`. Lorsque vous double-cliquez sur ce fichier, il ouvrira Gestionnaire de schéma avec les schémas que vous vouliez présélectionner. La seule chose qui vous reste à faire, c'est cliquer sur **Appliquer**.

Sur le CLI

Pour installer des schémas via la ligne de commande, exécutez la commande `install`[371] :

```
xmlschemamanager.exe install [options] Schema+
```

où `Schéma` est le schéma (ou les schémas) que vous voulez installer ou un fichier `.altova_xmlschemas`. Un schéma est référencé par un identifiant de format `<name>-<version>`. (Les identifiants de schémas sont affichés quand vous exécutez la commande `list`[371].) Vous pouvez saisir autant de schémas que vous le souhaitez. Pour plus de détails, voir la description de la commande `install`[371].

Note : sur Linux ou macOS, utilisez la commande **`sudo ./xmlschemamanager`**.

Installer un schéma requis
Lorsque vous exécutez une commande activée par XML dans RaptorXML Server, et que RaptorXML Server découvre qu'un schéma dont il a besoin pour exécuter la commande n'est pas présente ou est incomplète, Gestionnaire de schéma affichera l'information sur les schémas manquants. Vous pouvez ensuite installer directement tout schéma manquant via Gestionnaire de schéma.

Dans la GUI de Schema Manager, vous pouvez consulter tous les schémas précédemment installés à tout moment en exécutant Gestionnaire de schéma depuis **Outils | Gestionnaire de schéma**.

8.4 Désinstaller un schéma, Réinitialiser

Désinstaller un schéma

Vous pouvez désinstaller des schémas en utilisant soit la GUI Gestionnaire de schéma ou en envoyant à Gestionnaire de schéma les instructions d'installation via la ligne de commande.

Note : si la Police que vous voulez désinstaller référence d'autres Schémas, alors les Schémas référencées sont également désinstallées.

Dans la GUI

Pour désinstaller les schémas utilisant la GUI Gestionnaire de schéma, effacez leurs cases à cocher et cliquez sur **Appliquer**. Les schémas sélectionnés et leurs schémas référencées seront désinstallés.

Pour désinstaller tous les schémas, cliquez sur **Désélectionner tout** et cliquez sur **Appliquer**.

Sur le CLI

Pour désinstaller des schémas via la ligne de commande, exécutez la commande `désinstaller` 373 :

```
xmlschemamanager.exe uninstall [options] Schema+
```

où chaque argument **`Schéma`** est le schéma que vous voulez désinstaller ou un fichier **`.altova_xmlschemas`**. Un schéma est spécifié par un identifiant qui a un format **`<name>-<version>`**. (Les identifiants de schémas sont affichés quand vous exécutez la commande `list` 371.) Vous pouvez saisir autant de schémas que vous le souhaitez. Pour plus de détails, voir la description de la commande `désinstaller` 373.

Note : sur Linux ou macOS, utilisez la commande **`sudo ./xmlschemamanager`**.

Réinitialiser Gestionnaire de schéma

Vous pouvez réinitialiser Gestionnaire de schéma. Ceci supprime toutes les schémas installés et le répertoire de mise sous cache.

- Dans la GUI, cliquez sur **Reset Selection**.
- Dans la CLI, exécutez la commande `reset` 372.

Une fois avoir exécuté cette commande, vous devrez exécuter la commande `initialize` 370, pour pouvoir recréer le répertoire de mise sous cache. En alternative, exécutez la commande `reset` 372 avec l'option `-i`.

Notez que `reset -i` 372 restaure l'installation originale du produit, il est recommandé d'exécuter la commande `update` 374 après avoir réalisé la réinitialisation. En alternative, exécutez la commande `reset` 372 avec les options `-i` and `-u`.

8.5 Interface de ligne de commande (CLI)

Pour appeler Gestionnaire de schéma dans la ligne de commande, vous devez connaître le chemin de l'exécutable. Par défaut, l'exécutable Gestionnaire de schéma est installé dans le chemin suivant :

Windows	`C:\ProgramData\Altova\SharedBetweenVersions\XMLSchemaManager.exe`
Linux	`/opt/Altova/RaptorXMLServer2024/bin/xmlschemamanager`
macOS	`/usr/local/Altova/RaptorXMLServer2024/bin/xmlschemamanager`

Note : sur les systèmes Linux et macOS, une fois que vous avez changé le répertoire à celui contenant l'exécutable, vous pouvez appeler l'exécutable avec **`sudo ./xmlschemamanager`**. Le préfixe **`./`** indique que l'exécutable est le répertoire actuel. Le préfixe **`sudo`** indique que la commande que la commande doit être exécutée avec des privilèges root.

Syntaxe de ligne de commande

La syntaxe générale pour utiliser la ligne de commande est la suivante :

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

Dans l'extrait ci-dessus, la barre verticale **`|`** sépare un ensemble d'items mutuellement exclusifs. Les crochets **`[]`** indiquent des items optionnels. De manière générale, vous pouvez saisir le chemin d'exécutable suivi soit par les options `-h`, `--help`, ou `--version` ou par une commande. Chaque commande peut contenir des options et des arguments. La liste des commandes est décrite dans les sections suivantes.

8.5.1 help

Cette commande propose une aide contextuelle pour les commandes liées à l'exécutable Gestionnaire de schéma.

Syntaxe

```
<exec> help [command]
```

Où `[command]` est un argument optionnel qui spécifie un nom de commande valide.

Veuillez noter les points suivants :

- Vous pouvez invoquer de l'aide en saisissant la commande suivie par **`-h`** ou **`--help`**, par exemple : **`<exec> list-h`**
- Si vous tapez **`-h`** or **`--help`** directement après la commande exécutable et avant une commande, vous recevrez une aide générale (pas d'aide pour la commande), par exemple : **`<exec> -h list`**

Exemple

La commande suivante affiche une aide concernant la commande `list` :

```
xmlschemamanager help list
```

8.5.2 info

Cette commande affiche des informations détaillées pour chacun des schémas fournis en tant qu'argument de `Schéma`. Cette information inclut le titre, la version, description, l'éditeur et tout schéma soumis et tout schéma référencé, et informe si le schéma a été installé ou non.

Syntaxe

```
<exec> info [options] Schema+
```

- L'argument **`Schéma`** est le nom d'un schéma ou une partie du nom de schéma. (Pour afficher une ID de pack de schéma et des informations détaillées sur son statut d'installation, vous devriez utiliser la commande `list` 371.)
- Utiliser **`<exec> info -h`** pour afficher l'aide de la commande.

Exemple

La commande suivante affiche l'information sur les derniers schémas **`DocBook-DTD`** et **`NITF`** :

```
xmlschemamanager info doc nitf
```

8.5.3 initialize

Cette commande initialise l'environnement Gestionnaire de schéma. Elle crée un répertoire de cache où les informations concernant tous les schémas sont stockées localement. L'initialisation est réalisée automatiquement la première fois qu'une application schema-cognizant d'Altova est installée. Vous n'aurez pas besoin d'exécuter cette commande dans des circonstances normales, mais vous devrez l'exécuter généralement après la commande **`reset`**.

Syntaxe

```
<exec> initialize | init [options]
```

Options

La commande **`initialize`** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **`faux`**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **`faux`**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante initialise Gestionnaire de schéma:

```
xmlschemamanager initialize
```

8.5.4 install

Cette commande installe un ou plusieurs schémas.

Syntaxe

```
<exec> install [options] Schema+
```

Pour installer de multiples schémas, ajoutez l'argument **Schéma** de nombreuses fois.

L'argument **Schéma** est l'un des suivants :

- Un identifiant de schéma (avoir un format de **<name>-<version>**, par exemple : **cbcr-2.0**). Pour trouver les identifiants de schémas que vous voulez, exécutez la commande list[371]. Vous pouvez aussi utiliser des identifiants abrégés s'ils sont uniques, par exemple **docbook**. Si vous utilisez un identifiant abrégé, alors la dernière version de ce schéma sera installée.
- Le chemin vers un fichier **.altova_xmlschemas** téléchargé depuis le site web d'Altova. Pour information sur ces fichiers, voir *Introduction à SchemaManager : Comment cela fonctionne-t-il ?*[357].

Options

La commande **install** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante installe le schéma CBCR 2.0 (Country-By-Country Reporting) et le dernier DocBook DTD:

```
xmlschemamanager install cbcr-2.0 docbook
```

8.5.5 list

Cette commande recense les schémas sous la gestion de Gestionnaire de schéma. La liste affiche comme suit

- Tous les schémas disponibles
- Les schémas contenant le string dans leur nom soumis comme argument de **Schéma**
- Seuls les schémas installés
- Seuls les schémas qui peuvent être mis à niveau

Syntaxe

```
<exec> list | ls [options] Schema?
```

Si aucun argument de **Schéma** n'est soumis, alors toutes les schémas disponibles sont recensés. Autrement, les schémas sont recensés par des options soumises (*voir l'exemple ci-dessous*). Notez que vous pouvez soumettre l'argument de **Schéma** de nombreuses fois.

Options
La commande **list** accepte les options suivantes :

`--installed, --i`	Recenser uniquement la liste des schémas installés. Le réglage par défaut est **faux**.
`--upgradeable, --u`	Recenser uniquement les schémas lorsque des mises à niveau (patchs) sont disponibles. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemples

- Pour exécuter tous les schémas disponibles, exécutez : **xmlschemamanager list**
- Pour recenser les schémas installés, exécutez : **xmlschemamanager list -i**
- Pour recenser tous les schémas qui contiennent soit "doc", soit "nitf" dans leur nom, exécutez **xmlschemamanager list doc** :

8.5.6 reset

Cette commande supprime tous les schémas installés et le répertoire de mise sous cache. Vous réinitialiserez complètement votre environnement de schéma. Une fois avoir exécuté cette commande, vous devrez exécuter la commande initialize 370, pour pouvoir recréer le répertoire de mise sous cache. En alternative, exécuter la commande **reset** avec l'option **-i**. Pusique **reset -i** restaure l'installation originale du produit, nous vous recommandons que vous exécutiez la commande update 374 après avoir réalisé la réinitialisation et l'initialisation. En alternative, exécutez la commande **reset** avec les options **-i** and **-u**

Syntaxe

```
<exec> reset [options]
```

Options
La commande **reset** accepte les options suivantes :

`--init, --i`	Initialiser Gestionnaire de schéma après le reset. Le réglage par défaut est **faux**.
`--update, --u`	Mettre à jour la liste de schémas disponibles dans le cache. Le réglage par défaut est **faux**.
`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemples

- Pour réinitialiser Gestionnaire de schéma, exécuter : **xmlschemamanager reset**
- Pour réinitialiser Gestionnaire de schéma et l'initialiser, exécutez : **xmlschemamanager reset -i**
- Pour réinitialiser Gestionnaire de schéma, initialiser-le et mettez à jour sa liste de schéma, exécutez : **xmlschemamanager reset -i-u**

8.5.7 uninstall

Cette commande désinstalle un ou plusieurs schémas. Par défaut, tout schéma référencé par la taxonomie actuelle sera également désinstallé. Pour désinstaller uniquement le schéma actuel et garder les schémas référencés, définir l'option `--k`.

Syntaxe

```
<exec> uninstall [options] Schema+
```

Pour désinstaller de multiples schémas, ajoutez l'argument **Schéma** de nombreuses fois.

L'argument **Schéma** est l'un des suivants :

- Un identifiant de schéma (avoir un format de **<name>-<version>**, par exemple : **cbcr-2.0**). Pour trouver les identifiants de schéma qui sont installés, exécutez la commande `list -i` 371 . Vous pouvez aussi utiliser un nom de schéma abrégé s'il est unique, par exemple **docbook**. Si vous utilisez un nom abrégé, alors tous les schémas qui contiennent une abréviation dans leur nom seront désinstallés.
- Le chemin vers un fichier **.altova_xmlschemas** téléchargé depuis le site web d'Altova. Pour information sur ces fichiers, voir *Introduction à SchemaManager : Comment cela fonctionne-t-il ?* 357.

Options
La commande **désinstaller** accepte les options suivantes :

`--keep-references, --k`	Définir cette option pour garder les schémas référencés. Le réglage par défaut est **faux**.

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **`faux`**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **`faux`**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante désinstalle les schémas CBCR 2.0 et EPUB 2.0 et leurs dépendances :

```
xmlschemamanager uninstall cbcr-2.0 epub-2.0
```

La commande suivante désinstalle la taxonomie **`eba-2.10`** mais pas les schémas qu'elle référence :

```
xmlschemamanager uninstall --k cbcr-2.0
```

8.5.8 update

Cette commande requête la liste des schémas disponibles depuis l'emplacement de stockage en ligne et met à jour le répertoire de mise sous cache local. Vous devriez exécuter cette commande sauf si vous avez réalisé un `reset` 372 et `initialize` 370.

Syntaxe

```
<exec> update [options]
```

Options

La commande **`mise à jour`** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **`faux`**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **`faux`**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante met à jour le cache local avec la liste des derniers schémas :

```
xmlschemamanager update
```

8.5.9 upgrade

Cette commande met à niveau toutes les schémas installés qui peuvent être mis à niveau à la dernière version *patchée* disponible. Vous pouvez identifier des schémas à mettre à niveau en exécutant la commande `list -u` 371.

Note : La commande **`mettre à niveau`** supprime une Police dépréciée si aucune version plus récente n'est disponible.

Syntaxe

```
<exec> upgrade [options]
```

Options

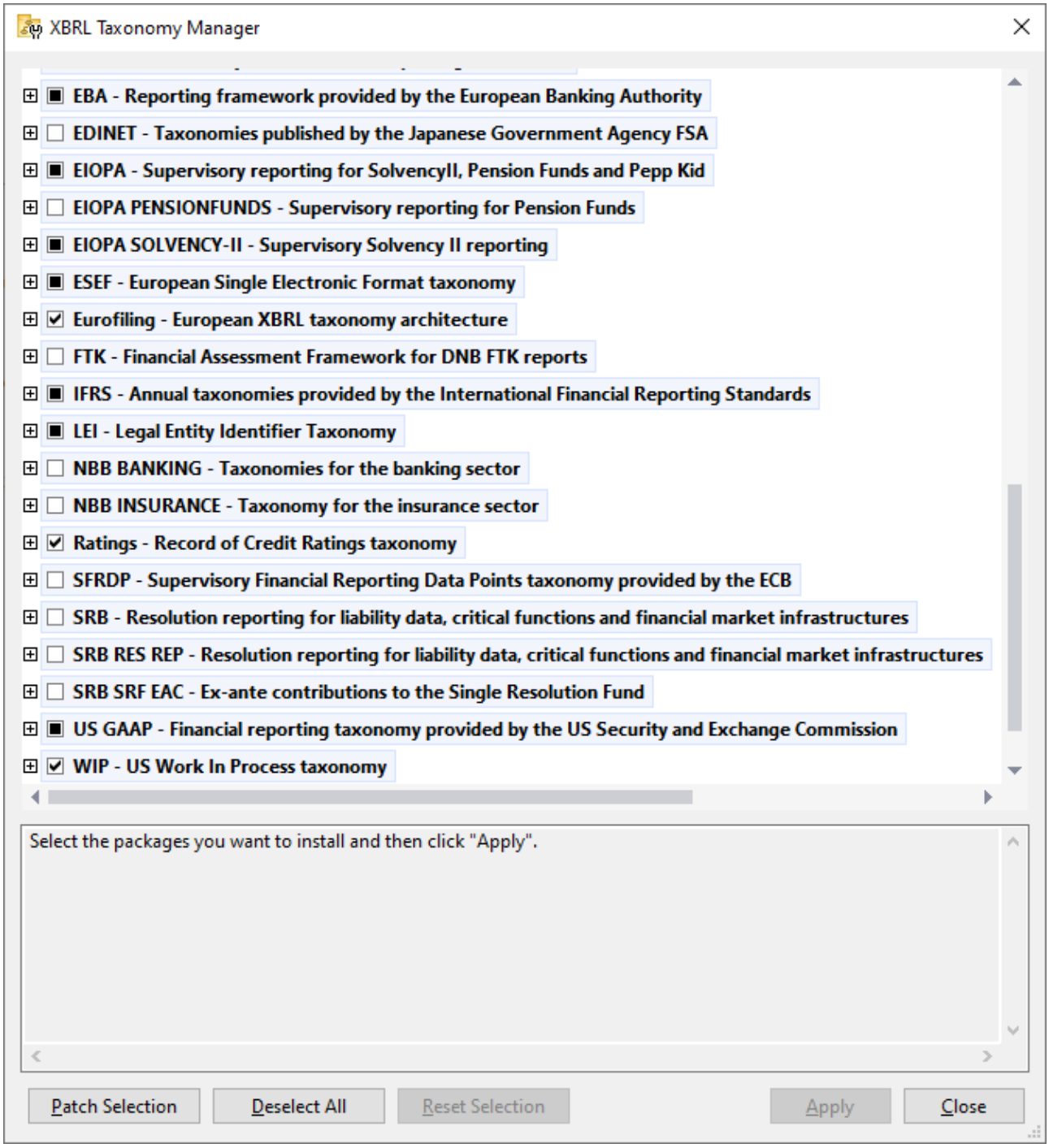
La commande **`mise à niveau`** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **`faux`**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **`faux`**.
`--help, --h`	Afficher l'aide pour la commande.

9 Gestionnaire de taxonomie

XBRL Gestionnaire de taxonomie est un outil qui propose un moyen centralisé d'installer et de gérer des taxonomies XBRL pour une utilisation sur toutes les applications XBRL Altova, y compris RaptorXML Server

- Sur Windows, Gestionnaire de taxonomie a une interface utilisateur graphique *(voir la capture d'écran ci-dessous*) et est aussi disponible dans la ligne de commande. (Les applications desktop d'Altova sont disponibles sur Windows uniquement ; *voir la liste ci-dessous*.)
- Sur Linux et Mac Gestionnaire de taxonomie, l'outil est disponible uniquement dans la ligne de commande. (Les applications serveur d'Altova sont disponibles sur Windows, Linux et macOS ; *voir la liste ci-dessous*.)



Les applications activées par XBRL d'Altova

Applications desktop (Windows uniquement)	Applications de serveur (Windows, Linux, macOS)

Altova XBRL Add-ins pour Excel (EBA, Solvency II, II)	MapForce Server (Standard et Advanced Edition).
MapForce Enterprise Edition	RaptorXML+XBRL Server
StyleVision Enterprise Edition	StyleVision Server
XMLSpy Enterprise Edition	

Installation et désinstallation de Gestionnaire de taxonomie

Gestionnaire de taxonomie est installé automatiquement quand vous installez d'abord une nouvelle version de l'Altova Mission Kit Enterprise Edition ou toute application activée par XBRL d'Altova (*voir la table ci-dessus*).

De même, il est supprimé automatiquement lorsque vous désinstallez la dernière application activée pour XBRL Altova depuis votre ordinateur.

Fonctions <% TAXON-MANAGER%>

Gestionnaire de taxonomie propose les fonctions suivantes :

- Consulter les taxonomies XBRL installées sur votre ordinateur, et contrôlez si de nouvelles versions sont disponibles pour le téléchargement.
- Télécharger des versions plus récentes des taxonomies XBRL indépendamment du cycle de release des produits Altova. (Altova stocke des taxonomies en ligne et vous pouvez les télécharger via Gestionnaire de taxonomie.)
- Installer ou désinstaller une des versions d'une taxonomie donnée (ou toutes les versions si nécessaire).
- Une taxonomie XBRL peut avoir des dépendances sur d'autres taxonomies. Lorsque vous installez ou désinstallez une taxonomie particulière, Gestionnaire de taxonomie vous informe sur d'autres taxonomies dépendantes et les installera ou désinstallera également automatiquement.
- Gestionnaire de taxonomie utilise le mécanisme du catalogue XML pour mapper les références de schéma aux fichiers locaux. Dans le cas de taxonomies XBRL larges, le traitement sera plus rapide que si les taxonomies étaient à un emplacement à distance.
- Toutes les taxonomies majeures sont disponibles via Gestionnaire de taxonomie et sont régulièrement mises à jour pour les dernières versions. Ceci vous fournit une ressource unique pour gérer toutes vos taxonomies et les mettre à disposition de toutes les applications activées par XBRL d'Altova.
- Les changements réalisés dans Gestionnaire de taxonomie prennent effet pour tous les produits d'Altova sur cet appareil.

Comment cela fonctionne ?

Altova stocke toutes les taxonomies XBRL utilisées dans les produits Altova en ligne. Ce référentiel est mis à jour lorsque de nouvelles versions des taxonomies sont publiées. Gestionnaire de taxonomie affiche des informations sur les dernières taxonomies disponibles lorsqu'elles sont appelées dans son formulaire GUI de même que sur CLI. Vous pouvez ensuite installer, mettre à jour eou désinstaller les taxonomies via Gestionnaire de taxonomie.

Gestionnaire de taxonomie installe également les taxonomies d'une autre manière. Sur le site web d'Altova (https://www.altova.com/taxonomy-manager), vous pouvez sélectionner une taxonomie et ses taxonomies dépendantes que vous souhaitez installer. Le site web préparera un fichier de type `.altova_taxonomies` pour le téléchargement qui contient des informations sur la sélection de taxonomie. Lorsque vous double-cliquez sur

ce fichier ou le passez à Gestionnaire de taxonomie via CLI comme argument de la commande **installer**[390], Gestionnaire de taxonomie installera les taxonomies que vous avez sélectionnés.

Cache local : suivre vos taxonomies

Toutes les informations sur les taxonomies installées sont suivies dans un répertoire cache centralisé sur votre ordinateur, situé ici :

Fenêtres	`C:\ProgramData\Altova\pkgs\.cache`
Linux	`/var/opt/Altova/pkgs\.cache`
macOS	`/var/Altova/pkgs`

Ce répertoire cache est mis à jour régulièrement avec le dernier statut des taxonomies dans l'emplacement de stockage en ligne d'Altova. Ces mises à jour sont réalisées aux moments suivants :

- À chaque fois que vous lancez Gestionnaire de taxonomie.
- Lorsque vous exécutez RaptorXML Server pour la première fois dans un jour donné du calendrier.
- Si RaptorXML Server est ouvert plus de 24 heures, le cache est mis à jour toutes les 24 heures.
- Vous pouvez aussi mettre à jour le cache en exécutant la commande de `mise à jour`[393] dans l'interface de ligne de commande.

Pour cette raison, le cache permet à Gestionnaire de taxonomie de suivre continuellement vos taxonomies installées par rapport aux taxonomies disponibles en ligne sur le site web d'Altova.

Ne modifiez pas le cache manuellement !

Le répertoire de cache local est entretenu automatiquement sur la base des taxonomies que vous installez ou désinstallez ; il ne doit pas être modifié ou supprimé manuellement. Il ne devrait pas être altéré ou supprimé manuellement. Si vous êtes amené à réinitialiser Gestionnaire de taxonomie à son état original "intact", alors, sur l'interface de la ligne de commande (CLI) : (i) exécutez la commande `reset`[391], et (ii) exécutez la commande `initialize`[389]. (En alternative, exécutez la commande **`reset`** avec l'option **`--i`**.)

Proxy HTTP

Vous pouvez utiliser un proxy HTTP pour les connexions Gestionnaire de taxonomie.

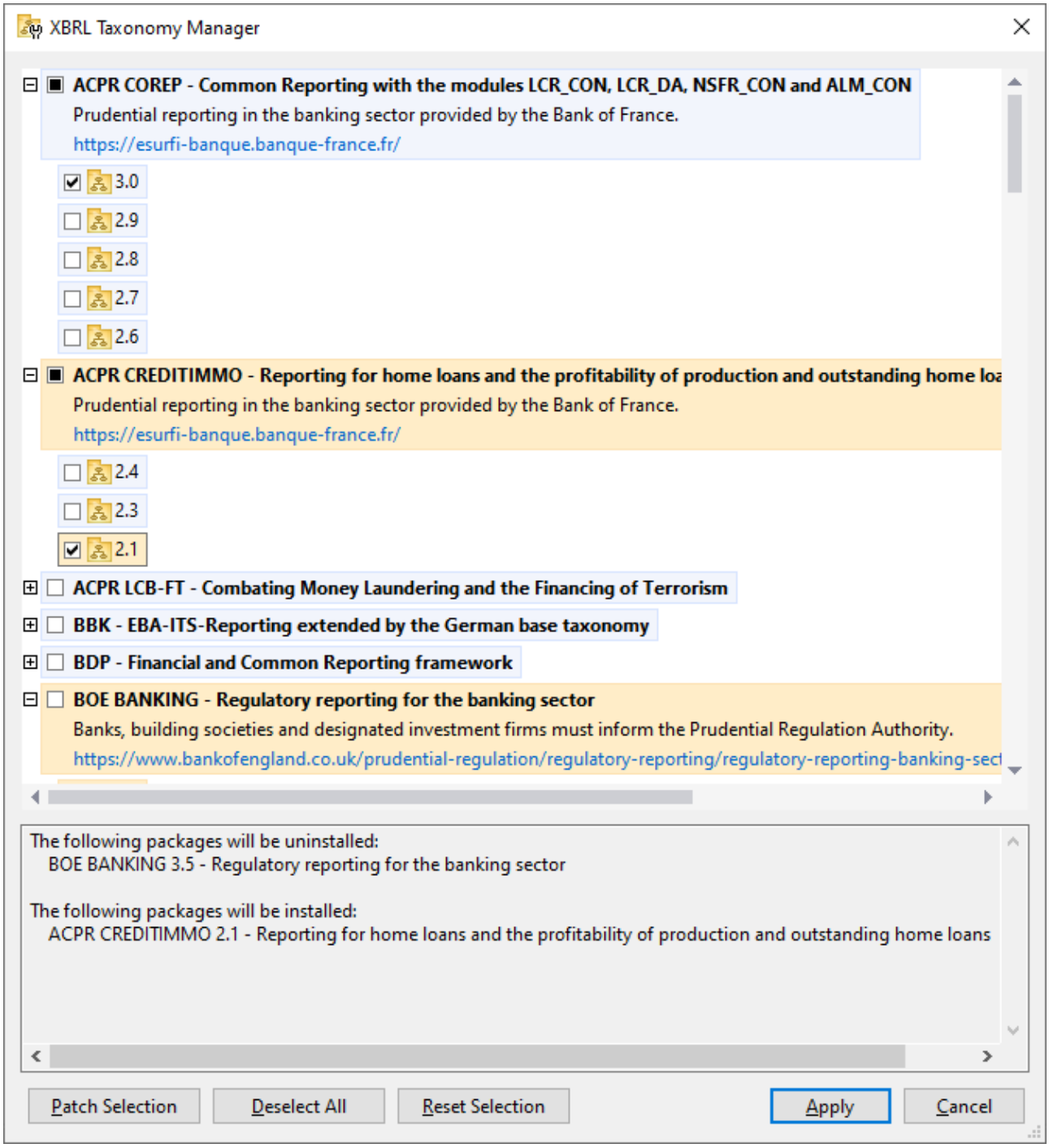
9.1 Exécuter le Gestionnaire de taxonomie

Interface utilisateur graphique

Vous pouvez accéder à la GUI de Gestionnaire de taxonomie des manières suivantes :

- *Durant l'installation de RaptorXML Server:* Vers la fin de la procédure d'installation, sélectionnez la case à cocher *Invoke Altova Taxonomy Manager* pour accéder au Gestionnaire de taxonomie XBRL directement. Ceci vous permettra d'installer les taxonomies au cours de la procédure d'installation de votre application Altova.
- *Après l'installation de RaptorXML Server:* Une fois que votre installation a été installée, vous pouvez accéder à la GUI de Gestionnaire de taxonomie à tout moment,.
- Via le fichier `.altova_taxonomies` téléchargé du Centre de téléchargement de la taxonomie XBRL d'Altova: Double-cliquez sur le fichier téléchargé pour exécuter la GUI de Gestionnaire de taxonomie, qui sera configurée pour installer les taxonomies que vous avez sélectionnées (le site web) pour installation.

Une fois que Gestionnaire de taxonomie GUI (*capture d'écran ci-dessous*) a été ouvert, les taxonomies déjà installées seront affichées telles sélectionnées. Si vous voulez installer une taxonomie supplémentaire, sélectionnez-la. Si vous voulez désinstaller une taxonomie déjà installée, désélectionnez-la. Une fois que vous avez faits vos sélections et/ou désélections, vous êtes prêts pour appliquer vos changements. Les taxonomies qui seront installées ou désinstallées seront mises en surbrillance et un message sur les modifications à venir sera posté dans le volet Messages au niveau inférieur de la fenêtre Gestionnaire de taxonomie (*voir la capture d'écran*).



Interface de ligne de commande

Vous pouvez exécuter Gestionnaire de taxonomie depuis une interface de ligne de commande en sélectionnant son fichier exécutable, `taxonomymanager.exe`.

Le fichier **taxonomymanager.exe** est situé dans le dossier suivant :

- *Sur Windows :* `C:\ProgramData\Altova\SharedBetweenVersions`
- *Sur Linux ou macOS (application serveur uniquement) :* `%INSTALLDIR%/bin`, où `%INSTALLDIR%` est le répertoire d'installation du programme.

Vous pouvez alors utiliser toute commande de la section de référence de la commande CLI.

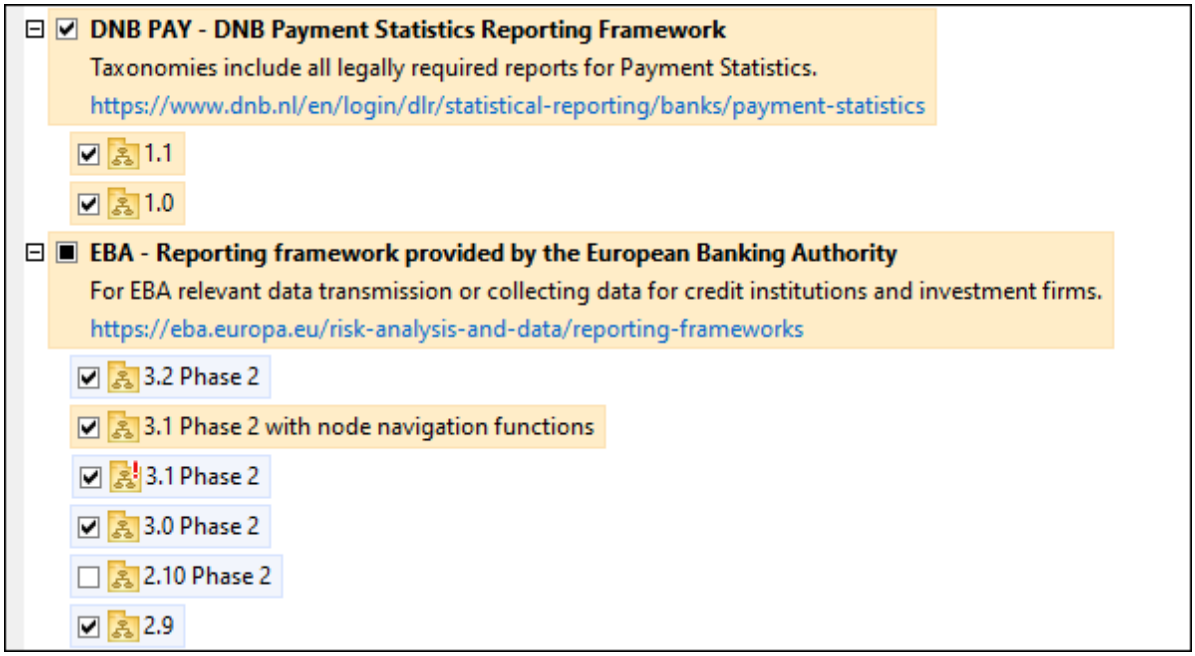
Pour afficher l'aide pour la commande, exécutez l'étape suivante :

- *Sur Windows :* `taxonomymanager.exe --help`
- *Sur Linux ou macOS (application serveur uniquement) :* `sudo ./taxonomymanager --help`

9.2 Catégories de statut

Gestionnaire de taxonomie catégorise les taxonomies sous sa gestion comme suit :

- *Installer taxonomies.* Celles-ci sont affichées dans la GUI avec leurs cases à cocher sélectionnées (*dans la capture d'écran ci-dessous, les versions cochées ci-de la taxonomie DNB et EBA sont des taxonomies installées*). Si toutes les versions de taxonomie sont sélectionnées, alors la marque de sélection est une coche. Si au moins une version est décochée, alors la coche de sélection est un carré coloré plein. Vous pouvez décocher une taxonomie installée pour la **désinstaller**.
- *Désinstaller taxonomies disponibles.* Elles sont affichées dans la GUI avec leurs cases à cocher non sélectionnées. Vous pouvez sélectionner les taxonomies que vous souhaitez **installer**.



- Les *taxonomies pouvant être mises à niveau* sont celles qui ont été revues par leurs émetteurs depuis qu'elles ont été installées. Elles sont indiquées par leur GUI par une icône  (*voir la capture d'écran ci-dessus*). Vous pouvez **retoucher** la taxonomie installée avec la révision disponible.

Points à noter

- Dans la capture d'écran ci-dessus, les taxonomies DNB et quelques unes des taxonomies EBA sont cochées. Celles avec un arrière-plan bleu sont déjà installées. Celles avec un arrière-plan jaune sont désinstallées et ont été sélectionnées pour l'installation. Notez que (i) la taxonomie EBA 2.10 Phase 2 n'est pas installée et n'a pas été sélectionnée pour l'installation, (ii) la taxonomie EBA 3.1 Phase 2 a été installée, mais elle a été retouchée par son émetteur depuis qu'elle a été installée et le correctif n'a pas encore été installé.
- Lorsque vous exécutez Gestionnaire de taxonomie depuis la ligne de commande, la commande `list` 390 est utilisée avec différentes options pour pour recenser les différentes catégories de taxonomies :

`taxonomymanager.exe list`	Recense toutes les taxonomies installées et disponibles ; celles

	pouvant être mises à niveau sont également indiquées.
`taxonomymanager.exe list -i`	Recense les taxonomies installées uniquement ; celles pouvant être mises à niveau sont également indiquées
`taxonomymanager.exe list -u`	Recense les taxonomies pouvant être mises à niveau

Note : Sur Linux et macOS, **`use sudo ./taxonomymanager list`**

9.3 Retoucher ou Installer une taxonomie

Retoucher une taxonomie installée

Occasionnellement, des taxonomies XBRL peuvent recevoir des patchs (mises à niveau ou révisions) depuis leurs émetteurs. Lorsque Gestionnaire de taxonomie détecte que des patchs sont disponibles, ceux-ci sont indiqués dans les listes de taxonomie Gestionnaire de taxonomie et vous pouvez installer des patchs rapidement.

Dans la GUI

Les patchs sont indiqués par l'icône . (*Voir aussi la rubrique précédente sur les catégories de statut*[383].) Si les patchs sont disponibles, le bouton **Patch Selection** sera activé. Cliquez dessus pour sélectionner et préparer tous les patchs pour installation. Dans la GUI, l'icône de chaque taxonomie sera patchée de à , et le volet des Messages en bas du dialogue recense les patchs qui doivent être appliqués. Lorsque vous êtes prêt pour installer des patchs sélectionnés, cliquez sur **Appliquer**. Tous les correctifs seront appliqués ensemble. Notez que si vous décochez une taxonomie marquée pour une correction, réellement vous désinstallerez cette taxonomie.

Sur le CLI

Pour appliquer un patch dans l'interface de ligne de commande :

1. Exécuter la commande `list -u`[390] . Cela liste des taxonomies lorsque des mises à niveau de patch sont disponibles.
2. Exécuter la commande `upgrade`[393] pour installer les patches.

Installer une taxonomie disponible

Vous pouvez installer des taxonomies en utilisant soit la GUI Gestionnaire de taxonomie ou en envoyant Gestionnaire de taxonomie les instructions d'installation via la ligne de commande.

Note : Si la taxonomie actuelle référence d'autres taxonomies, les taxonomies référencées sont aussi installées.

Dans la GUI

Pour installer une taxonomie utilisant la GUI Gestionnaire de taxonomie GUI, sélectionnez les taxonomies que vous voulez installer et cliquez sur **Appliquer**.

Vous pouvez aussi sélectionner les taxonomies que vous voulez installer sur le site web d'Altova et générer un fichier téléchargeable `.altova_taxonomies`. Lorsque vous double-cliquez sur ce fichier, il ouvrira Gestionnaire de taxonomie avec les taxonomies que vous vouliez présélectionner. La seule chose qui vous reste à faire, c'est cliquer sur **Appliquer**.

Sur le CLI

Pour installer des taxonomies via la ligne de commande, exécutez la commande `installer`[390] :

```
taxonomymanager.exe install [options] Taxonomie+
```

où `Taxonomie` est la taxonomie (ou les taxonomies) que vous voulez installer ou un fichier `.altova_taxonomies`. Une taxonomie est référencée par un identifiant de format `<name>-<version>`. (Les

identifiants de taxonomies sont affichés quand vous exécutez la commande **list**[390].) Vous pouvez saisir autant de taxonomies que vous le souhaitez. Pour plus de détails, voir la description de la commande install[390].

Note : sur Linux ou macOS, utilisez la commande **sudo ./taxonomymanager**.

Installer une taxonomie requise

Lorsque vous exécutez une commande activée par XBRL dans RaptorXML Server, et que RaptorXML Server découvre qu'une taxonomie dont elle a besoin pour exécuter la commande n'est pas présente ou est incomplète, Gestionnaire de taxonomie sera affiché l'information sur la taxonomie manquante. Vous pouvez ensuite installer directement toute taxonomie manquante via Gestionnaire de taxonomie.

Dans le Gestionnaire de taxonomie GUI, vous pouvez consulter toutes les taxonomies précédemment installées á tout moment en exécutant Gestionnaire de taxonomie de **Outils | Gestionnaire de taxonomie**.

9.4 Désinstaller une taxonomie, Réinitialiser

Désinstaller une taxonomie

Vous pouvez désinstaller des taxonomies en utilisant soit la GUI Gestionnaire de taxonomie ou en envoyant Gestionnaire de taxonomie les instructions d'installation via la ligne de commande.

Note : si la taxonomie que vous voulez désinstaller référence d'autres taxonomies, alors les taxonomies référencées sont également désinstallées.

Dans la GUI

Pour désinstaller les taxonomies utilisant la GUI Gestionnaire de taxonomie, effacez leurs cases à cocher et cliquez sur **Appliquer**. Les taxonomies sélectionnées et leurs taxonomies référencées seront désinstallées.

Pour désinstaller les taxonomies, cliquez sur **Désélectionner tout** et cliquez sur **Appliquer**.

Sur le CLI

Pour désinstaller des taxonomies via la ligne de commande, exécutez la commande **`désinstaller`** :

```
taxonomymanager.exe uninstall [options] Taxonomy+
```

où chaque argument **`Taxonomie`** est la taxonomie que vous voulez désinstaller ou un fichier **`.altova_taxonomies`**. Une taxonomie est spécifiée par un identifiant qui a un format **`<name>-<version>`**. (Les identifiants de taxonomies sont affichés quand vous exécutez la commande **`list`**[390].) Vous pouvez saisir autant de taxonomies que vous le souhaitez. Pour plus de détails, voir la description de la commande `désinstaller`[392].

Note : sur Linux ou macOS, utilisez la commande **`sudo ./taxonomymanager`**.

Réinitialiser Gestionnaire de taxonomie

Vous pouvez réinitialiser Gestionnaire de taxonomie.

- Dans la GUI, cliquez sur **Reset Selection**. Ceci réinitialise la GUI pour montrer quelles taxonomies sont actuellement installées. Toute sélection ou désélection que l'utilisateur a effectué dans la session actuelle sera annulée.
- Dans la CLI, exécutez la commande `reset`[391]. Ceci supprime toutes les taxonomies installées et le répertoire de mise sous cache.

Une fois avoir exécuté cette commande, vous devrez exécuter la commande `initialize`[389], pour pouvoir recréer le répertoire de mise sous cache. En alternative, exécutez la commande `reset`[391] avec l'option `-i`.

Notez que `reset -i`[391] restaure l'installation originale du produit, il est recommandé d'exécuter la commande `update`[393] après avoir réalisé la réinitialisation. En alternative, exécutez la commande `reset`[391] avec les options `-i` and `-u`.

9.5 Interface de ligne de commande (CLI)

Pour appeler Gestionnaire de taxonomie dans la ligne de commande, vous devez connaître le chemin de l'exécutable. Par défaut, l'exécutable Gestionnaire de taxonomie est installé dans le chemin suivant :

```
C:\ProgramData\Altova\SharedBetweenVersions\TaxonomyManager.exe
```

Note : sur les systèmes Linux et macOS, une fois que vous avez changé le répertoire à celui contenant l'exécutable, vous pouvez appeler l'exécutable avec **`sudo ./taxonomymanager`**. Le préfixe **`./`** indique que l'exécutable est le répertoire actuel. Le préfixe **`sudo`** indique que la commande que la commande doit être exécutée avec des privilèges root.

Syntaxe de ligne de commande

La syntaxe générale pour utiliser la ligne de commande est la suivante :

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

Dans l'extrait ci-dessus, la barre verticale **`|`** sépare un ensemble d'items mutuellement exclusifs. Les crochets **`[]`** indiquent des items optionnels. De manière générale, vous pouvez saisir le chemin d'exécutable suivi soit par les options `-h`, `--help`, ou `--version` ou par une commande. Chaque commande peut contenir des options et des arguments. La liste des commandes est décrite dans les sections suivantes.

9.5.1 help

Cette commande propose une aide contextuelle pour les commandes liées à l'exécutable Gestionnaire de taxonomie.

Syntaxe

```
<exec> help [command]
```

Où `[command]` est un argument optionnel qui spécifie un nom de commande valide.

Veuillez noter les points suivants :

- Vous pouvez invoquer de l'aide une commande en saisissant La section commande suivie par **`-h`** ou **`--help`**, par exemple : **`<exec> list-h`**
- Si vous tapez **`-h`** or **`--help`** directement après la commande exécutable et avant une commande, vous recevrez une aide générale (pas d'aide pour la commande), par exemple : **`<exec> -h list`**

Exemple

La commande suivante affiche une aide concernant la commande `list` :

```
./taxonomymanager help list
```


9.5.2 info

Cette commande affiche des informations détaillées pour chacune des taxonomies fournies en tant qu'argument de `Taxonomie`. Cette information inclut le titre, la version, description, l'éditeur et toute taxonomie soumise et toute taxonomie dépendante, et mentionne si la taxonomie a été installée ou non.

Syntaxe

```
<exec> info [options] Taxonomy+
```

- L'argument **`Taxonomie`** est le nom de la taxonomie ou une partie du nom de la taxonomie. (Pour afficher une ID de pack de taxonomie et des informations détaillées sur son statut d'installation, vous devriez utiliser la commande `list`[390].)
- Utiliser **`<exec> info -h`** pour afficher l'aide de la commande.

Exemple

La commande suivante affiche des informations détaillées concernant les taxonomies **`eba-2.10`** et **`us-gaap-2020.0`** :

```
taxonomymanager info eba-2.1.0 us-gaap-2020.0
```

9.5.3 initialize

Cette commande initialise l'environnement Gestionnaire de taxonomie. Elle crée un répertoire de cache où les informations concernant toutes les taxonomies sont stockées localement. L'initialisation est réalisée automatiquement la première fois qu'une application activée par XBRL est installée. Vous n'aurez pas besoin d'exécuter cette commande dans des circonstances normales, mais vous devrez l'exécuter généralement après la commande **`reset`**.

Syntaxe

```
<exec> initialize | init [options]
```

Options

La commande **`initialize`** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **`faux`**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **`faux`**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante initialise Gestionnaire de taxonomie:

```
taxonomymanager initialize
```

9.5.4 install

Cette commande installe une ou plusieurs taxonomies.

Syntaxe

```
<exec> install [options] Taxonomy+
```

Pour installer de multiples taxonomies, ajoutez l'argument **Taxonomy** de nombreuses fois.

L'argument **Taxonomy** est l'un des suivants :

- Un identifiant de taxonomie (avoir un format de **<name>-<version>**, par exemple : **eba-2.10**). Pour trouver les identifiants de taxonomie que vous voulez, exécutez la commande list[390]. **Vous pouvez aussi utiliser des identifiants abrégés s'ils sont uniques, par exemple** eba. Si vous utilisez un identifiant abrégé, alors la dernière version de cette taxonomie sera installée.
- Le chemin vers un fichier **.altova_taxonomies** téléchargé depuis le site web d'Altova. Pour information sur ces fichiers, voir *Introduction à TaxonomyManager : Comment cela fonctionne-t-il ?*[376].

Options
La commande **install** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante installe les dernières taxonomies **eba** (European Banking Authority) et **us-gaap** (US Generally Accepted Accounting Principles) :

```
taxonomymanager install eba us-gaap
```

9.5.5 list

Cette commande recense les taxonomies sous la gestion de Gestionnaire de taxonomie. La liste affiche comme suit

- Toutes les taxonomies disponibles
- Les taxonomies contenant le string dans leur nom soumis comme argument **Taxonomy**
- Seules les taxonomies installées
- Seules les taxonomies qui peuvent être mises à niveau

Syntaxe

```
<exec> list | ls [options] Taxonomy?
```

Si aucun argument **Taxonomy** n'est soumis, alors toutes les taxonomies disponibles sont recensées. Autrrement, les taxonomies sont recensées par des options soumises (*voir l'exemple ci-dessous*). Notez que vous pouvez soumettre l'argument **Taxonomy** de nombreuses fois.

Options
La commande **list** accepte les options suivantes :

`--installed, --i`	Uniquement la liste des taxonomies installées. Le réglage par défaut est **faux**.
`--upgradeable, --u`	Recenser uniquement les taxonomies lorsque des mises à niveau (patches) sont disponibles. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemples

- Pour exécuter toutes les taxonomies disponibles, exécutez : **taxonomymanager list**
- Pour recenser les taxonomies installées, exécutez : **taxonomymanager list -i**
- Pour recenser tous les taxonomies qui contiennent soit "eba", soit "us-gaap" dans leur nom, exécutez **taxonomymanager list eba us-gaap** :

9.5.6 reset

Cette commande supprime toutes les taxonomies installées et le répertoire de mise sous cache. Vous réinitialiserez complètement votre environnement de taxonomie. Une fois avoir exécuté cette commande, vous devrez exécuter la commande `initialize`[389], pour pouvoir recréer le répertoire de mise sous cache. En alternative, exécuter la commande `reset` avec l'option `-i`. Pusique `reset -i` restaure l'installation originale du produit, nous vous recommandons que vous exécutiez la commande `update`[393] après avoir réalisé la réinitialisation et l'initialisation. En alternative, exécutez la commande reset avec les options `-i`[389] et `-u`[389].

Syntaxe

```
<exec> reset [options]
```

Options
La commande **reset** accepte les options suivantes :

`--init, --i`	Initialiser Gestionnaire de taxonomie après le reset. Le réglage par défaut est **faux**.
`--update, --u`	Mettre à jour la liste de taxonomies disponibles dans le cache. Le réglage par défaut est **faux**.

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemples

- Pour réinitialiser Gestionnaire de taxonomie, exécuter : **taxonomymanager reset**
- Pour réinitialiser Gestionnaire de taxonomie et l'initialiser, exécutez : **taxonomymanager reset -i**
- Pour réinitialiser Gestionnaire de taxonomie, initialiser-la et mettez à jour sa liste de taxonomie, exécutez : **taxonomymanager reset -i-u**

9.5.7 uninstall

Cette commande désinstalle une ou plusieurs taxonomies. Par défaut, toute taxonomie référencée par la taxonomie actuelle sera également désinstallée. Pour désinstaller uniquement la taxonomie actuelle et garder les taxonomies référencées, définir l'option `--k`.

Syntaxe

```
<exec> désinstaller [options] Taxonomy+
```

Pour désinstaller de multiples taxonomies, ajoutez l'argument **Taxonomy** de nombreuses fois.

L'argument **Taxonomy** est l'un des suivants :

- Un identifiant de taxonomie (avoir un format de **<name>-<version>**, par exemple : **eba-2.10**). Pour trouver les identifiants de taxonomie qui sont installées, exécutez la commande `list -i` 390 . **Vous pouvez aussi utiliser un nom de taxonomie abrégé s'il est unique, par exemple** eba. Si vous utilisez un nom abrégé, alors toutes les taxonomies qui contiennent une abréviation dans leur nom seront désinstallées.
- Le chemin vers un fichier **.altova_taxonomies** téléchargé depuis le site web d'Altova. Pour information sur ces fichiers, voir *Introduction à TaxonomyManager : Comment cela fonctionne-t-il ?* 376.

Options
La commande **désinstaller** accepte les options suivantes :

`--keep-references, --k`	Définir cette option pour garder les taxonomies référencées. Le réglage par défaut est **faux**.
`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante désinstalle les taxonomies **eba-2.10** et **us-gaap-2020.0** et leurs dépendances :

```
taxonomymanager uninstall eba-2.10 us-gaap-2020.0
```

La commande suivante désinstalle la taxonomie **eba-2.10** mais pas les taxonomies qu'elle référence :

```
taxonomymanager uninstall --k eba-2.10
```

9.5.8 update

Cette commande requête la listes des taxonomies disponibles depuis le magasin en ligne et met à jour le répertoire de mise sous cache local. Vous devriez exécuter cette commande sauf si vous avez réalisé un reset[391] et initialize[389].

Syntaxe

```
<exec> update [options]
```

Options

La commande **mise à jour** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

Exemple

La commande suivante met à jour le cache local avec la liste des dernières taxonomies :

```
taxonomymanager update
```

9.5.9 upgrade

Cette commande met à niveau toutes les taxonomies éligibles qui peuvent être mises à niveau à la dernière version *patch* disponible. Vous pouvez identifier des taxonomies à mettre à niveau en exécutant la commande list -u[390].

Note : La commande **mettre à niveau** supprime une taxonomie dépréciée si aucune version plus récente n'est disponible.

Syntaxe

```
<exec> upgrade [options]
```

Options
La commande **mise à niveau** accepte les options suivantes :

`--silent, --s`	Afficher uniquement des messages d'erreur. Le réglage par défaut est **faux**.
`--verbose, --v`	Afficher des informations détaillées lors de l'exécution. Le réglage par défaut est **faux**.
`--help, --h`	Afficher l'aide pour la commande.

10 Information supplémentaire

Cette section contient les informations supplémentaires suivantes :

10.1 Codes de sortie

Les codes de sortie suivant sont disponibles :

0	Validation réussie
1	Validation échouée avec erreur / Processus interrompu par Ctrl+C/Break/terminal fermé / Licence expirée pendant l'exécution
11	RaptorXML n'a pas pu démarrer ; la raison est indiquée dans le fichier log
22	Impossible de charger le catalogue root / Impossible de charger le fichier de liste
64	Commande/options invalide
77	Impossible d'acquérir une licence pendant le démarrage
128+**n**	RaptorXML a été terminé en raison du nombre de signal **n**. Tous les codes de sortie au-dessus de **128** indiquent l'arrêt en résultat d'un signal externe reçu ou un signal déclenché internement. Par exemple, si le code de sortie est **134**, le numéro de signal est **134-128=6** (le numéro de **SIGABRT**).

10.2 Indices d'emplacement de schéma

Les documents d'instance peuvent utiliser des indices pour indiquer l'emplacement de schéma. Deux attributs sont utilisés pour les indices :

- `xsi:schemaLocation` pour les documents de schéma avec des espaces de noms cible. La valeur d'attribut est une paire d'items, le premier d'entre eux est un espace de noms, le second est une URL qui situe un document de schéma. Le nom d'espace de noms doivent correspondre à l'espace de noms cible du document de schéma.

```
<document xmlns="http://www.altova.com/schemas/test03"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.altova.com/schemas/test03 Test.xsd">
```

- `xsi:noNamespaceSchemaLocation` pour les documents de schéma sans espaces de noms cible. La valeur de l'attribut est l'URL du document de schéma. Le document de schéma référencé ne doit avoir aucun espace de noms cible.

```
<document xmlns="http://www.altova.com/schemas/test03"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="Test.xsd">
```

L'option `--schemalocation-hints` spécifie comment ces deux attributs doivent être utilisés en tant qu'indices, en particulier comment l'information d'attribut `schemaLocation` doit être gérée (*voir la description d'option ci-dessus*). Veuillez noter que RaptorXML Server considère que la partie espace de noms de la valeur `xsi:noNamespaceSchemaLocation` est la string vide.

Les indices d'emplacement de schéma peuvent aussi être donnés dans une instruction `import` d'un document de Schéma XML.

```
<import namespace="someNS" schemaLocation="someURL">
```

Dans l'instruction `import`, aussi, des indices peuvent être donnés par le biais d'un espace de noms qui peut être mappé vers un schéma dans un fichier de catalogue, ou directement en tant qu'une URL dans l'attribut `schemaLocation`. L'option `--schema-imports` [213] (pour XBRL et XSD/XML) spécifie comment l'emplacement de schéma doit être sélectionné.

11 Informations moteur

Cette section contient des informations concernant les processeurs XSLT et XQuery contenus dans RaptorXML Server. Cette information concerne principalement le comportement du processeur dans des situations où les specifications permettent un choix en ce qui concerne le comportement jusqu'à la mise en place. Cette section contient aussi des informations concernant les fonctions d'extension Altova pour XPath/XQuery.

11.1 Informations concernant le moteur XSLT et XQuery

Les moteurs XSLT et XQuery de RaptorXML Server suivent de près les spécifications W3C et sont donc plus strictes que les moteurs Altova précédents, comme dans les versions précédentes de XMLSpy et d'AltovaXML, le prédécesseur de RaptorXML. Ainsi, de petites erreurs qui étaient ignorées par les moteurs précédents sont maintenant marquées en tant qu'erreurs par RaptorXML Server.

Par exemple :

- Il s'agit d'une erreur de type (`err:XPTY0018`) si le résultat d'un opérateur de chemin contient aussi bien les nœuds que les non-nœuds.
- Il s'agit d'une erreur de type (`err:XPTY0019`) si `E1` dans une expression de chemin `E1/E2` n'évalue pas à une séquence de nœuds.

Si vous rencontrez ce type d'erreur, modifiez soit le document XSLT/XQuery, soit le document d'instance selon vos besoins.

Cette section décrit les fonctions spécifiques à la mise en place des moteurs, organisée par spécification :

11.1.1 XSLT 1.0

Le moteur XSLT 1.0 de RaptorXML Server est conforme aux Recommandations XSLT 1.0 du 16 novembre 1999 et aux Recommandations XPath 1.0 du 16 novembre 1999 du World Wide Web Consortium (W3C's). Veuillez noter les informations suivantes concernant l'implémentation.

Notes concernant l'implémentation

Lorsque l'attribut `method` de `xsl:output` est défini sur HTML, ou si la sortie HTML est sélectionnée par défaut, les caractères spéciaux dans le fichier XML ou XSLT sont insérés dans le document HTML en tant que références de caractère HTML dans la sortie. Par exemple, le caractère U+00A0 (la référence de caractère hexadécimale pour un espace insécable) est inséré dans le code HTML soit en tant que référence de caractère (` ` ou ` `) soit en tant que référence d'entité, ` ` .

11.1.2 XSLT 2.0

Cette section :

Conformité

Le moteur XSLT 2.0 de RaptorXML Server est conforme aux Recommandations XSLT 2.0 du 23 janvier 2007 et aux Recommandations XPath 2.0 du 14 décembre 2010 du World Wide Web Consortium (W3C's).

Rétrocompatibilité

Le moteur XSLT 2.0 est rétrocompatible. Généralement, la compatibilité rétroactive du moteur XSLT 2.0 entre en jeu si vous utilisez le moteur XSLT 2.0 (paramètre CLI `--xslt=2` 218) pour traiter une feuille de style XSLT 1.0 ou une instruction. Veuillez noter qu'il peut y avoir des différences dans les sorties produites par le moteur XSLT 1.0 et la rétrocompatibilité du moteur XSLT 2.0.

Espaces de nom

Votre feuille de style XSLT 2.0 devrait déclarer les espaces de noms suivants afin que vous puissiez utiliser les constructeurs de type et les fonctions disponibles dans XSLT 2.0. Les préfixes indiqués ci-dessous sont utilisés de manière conventionnelle ; vous pourriez utiliser les préfixes alternatifs si vous le souhaitez.

Nom d'espace de nom	Préfixe	Espace de nom URI
Types de schéma XML	`xs:`	`http://www.w3.org/2001/XMLSchema`
Fonctions XPath 2.0	`fn:`	`http://www.w3.org/2005/xpath-functions`

Généralement ces espaces de nom seront déclarés sur l'élément `xsl:stylesheet` ou `xsl:transform`, tel que montré dans la liste suivante :

```
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    ...
</xsl:stylesheet>
```

Veuillez noter les points suivants :

- Le moteur XSLT 2.0 utilise l'espace de nom XPath 2.0 et Fonctions XQuery 1.0 (recensées dans la table ci-dessus) en tant que son **espace de noms de fonctions par défaut**. Ainsi, vous pouvez utiliser les fonctions XPath 2.0 et XSLT 2.0 dans votre feuille de style sans aucun préfixe. Si vous

déclarez l'espace de nom des fonctions XPath 2.0 dans votre feuille de style avec un préfixe, vous pourrez utiliser en plus le préfixe attribué dans la déclaration.
- Lors de l'utilisation des constructeurs de types et des types provenant de l'espace de nom du Schéma XML, le préfixe utilisé dans la déclaration d'espace de nom doit être utilisé lors de l'appel du constructeur de type (par exemple, `xs:date`).
- Certaines fonctions XPath 2.0 portent le même nom que les types de données du schéma XML. Par exemple, pour les fonctions XPath `fn:string` et `fn:boolean`, il existe des types de données du schéma XML portant le même nom local : `xs:string` et `xs:boolean`. Donc si vous décidez d'utiliser l'expression XPath `string('Hello')`, l'expression évalue en tant que `fn:string('Hello')` et non pas en tant que `xs:string('Hello')`.

Compatibilité avec le schéma

Le moteur XSLT 2.0 est compatible avec le schéma. Vous pouvez ainsi utiliser des types de schéma définis par l'utilisateur et l'instruction `xsl:validate`.

Comportement spécifique à l'implémentation

Ci-dessous, vous trouverez une description de la gestion du moteur XSLT 2.0 des aspects spécifiques à l'implémentation du comportement de certaines fonctions XSLT 2.0.

`xsl:result-document`
Les encodages pris en charge en supplément sont (les codes spécifiques à Altova) : `x-base16tobinary` et `x-base64tobinary`.

`function-available`
La fonction teste la disponibilité des fonctions in-scope (XSLT, XPath, et fonctions d'extension).

`unparsed-text`
L'argument `href` accepte (i) les chemins relatifs pour les fichiers dans le dossier base-uri et (ii) les chemins absolus avec ou sans `file://` protocol. Les encodages pris en charge de manière supplémentaire sont (spécifiques à Altova): `x-binarytobase16` et `x-binarytobase64`. Exemple : `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`
L'argument `href` accepte (i) les chemins relatifs pour les fichiers dans le dossier base-uri et (ii) les chemins absolus avec ou sans `file://` protocol. Les encodages pris en charge de manière supplémentaire sont (spécifiques à Altova): `x-binarytobase16` et `x-binarytobase64`.

Note : les valeurs d'encodage suivantes, qui sont mises en œuvre dans des versions antérieures du produit prédécesseur de RaptorXML, AltovaXML, sont dépcréciées à présent : `base16tobinary`, `base64tobinary`, `binarytobase16` and `binarytobase64`.

11.1.3 XSLT 3.0

Le moteur XSLT 3.0 de RaptorXML Server est conforme aux Recommandation XSLT 3.0 du 8 juin 2017 et aux Recommandations XQuery 3.1 du 21 mars 2017 du World Wide Web Consortium (W3C's).

Le moteur XSLT 3.0 présente les mêmes caractéristiques spécifiques à l'implémentation que le moteur XSLT 2.0 399. De plus, il contient une prise en charge pour un certain nombre de nouvelles fonctions XSLT 3.0 : les fonctions et opérateurs XPath/XQuery 3.1, et la spécification XPath 3.1.

Note : la fonction de streaming optionnelle n'est pas prise en charge actuellement. Le document complet sera chargé dans la mémoire quelle que soit la valeur de l'attribut `streamable`. S'il y a assez de mémoire disponible, alors : (i) le document entier sera traité - sans streaming, (ii) les constructions garanties streamables seront traitées correctement, comme si l'exécution utilisait le streaming, et (iii) les erreurs de streaming ne seront pas détectées. Dans les applis 64-bit, l'exécution non-streaming ne devrait pas représenter de problème. S'il y a tout de même un souci avec la mémoire, une solution serait d'ajouter plus de mémoire au système.

Espaces de noms

Votre feuille de style XSLT 3.0 devrait déclarer les espaces de noms pour que vous puissiez utiliser tous les types de constructeurs et fonctions disponibles dans XSLT 3.0. Les préfixes ci-dessous sont utilisés de manière conventionnelle ; vous pourriez utiliser des préfixes alternatifs si vous le souhaitez.

Nom Espaces de noms	Préfixe	Espace de noms URI
Types de schéma XML	`xs:`	`http://www.w3.org/2001/XMLSchema`
Fonctions XPath/XQuery 3.1	`fn:`	`http://www.w3.org/2005/xpath-functions`
Fonctions Math	`math:`	`http://www.w3.org/2005/xpath-functions/math`
Fonctions Map	`map:`	`http://www.w3.org/2005/xpath-functions/map`
Fonctions Array	`array:`	`http://www.w3.org/2005/xpath-functions/array`
Codes d'erreur XQuery, XSLT et XPath	`err:`	`http://www.w3.org/2005/xpath-functions/xqt-errors`
Fonctions de sérialisation	`output`	`http://www.w3.org/2010/xslt-xquery-serialization`

Généralement, ces espaces de noms seront déclarés dans l'élément `xsl:stylesheet` ou `xsl:transform`, tel que recensé dans la liste suivante :

```
<xsl:stylesheet version="3.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   ...
</xsl:stylesheet>
```

Les points suivants doivent être notés :

- Le moteur XSLT 3.0 utilise les Fonctions XPath et XQuery et l'espace de noms de Operators 3.1 (recensé dans la table ci-dessus) comme son **espace de noms des fonctions par défaut**. Donc vous pouvez utiliser les fonctions de cet espace de noms dans votre feuille de style sans préfixe, alors vous pouvez utiliser le préfixe en plus assigné à la déclaration.
- Quand vous utilisetz les constructeurs de type et les types depuis l'espace de noms du schéma XML, le préfixe utilisé dans la déclaration de l'espace de noms doit être utilisé lorsque le constructeur de type est appelé (par exemple, `xs:date`).
- Certaines fonctions XPath/XQuery ont le même nom que les types de base de données de schéma XML. Par exemple, pour les fonctions XPath `fn:string` et `fn:boolean`, il existe des types de données de schéma XML avec les mêmes noms locaux : `xs:string` et `xs:boolean`. Donc si vous deviez utiliser le string de l'expression XPath (`'Hello'`), l'expression évalue comme `fn:string('Hello') - et non comme xs:string('Hello')`.

11.1.4 XQuery 1.0

Cette section :

Conformité

Le moteur XQuery 1.0 de RaptorXML Server est conforme à la Recommandation XQuery 1.0 du 14 décembre 2010 du World Wide Web Consortium (W3C's). Le standard XQuery accorde un pouvoir discrétionnaire concernant la mise en place de nombreuses fonctions. Ci-dessous, vous trouverez une liste expliquant comment le moteur XQuery 1.0 implémente ces fonctions.

Compatibilité avec le schéma

Le moteur XQuery 1.0 est **schema-aware**.

Encodage

Les encodages de caractères UTF-8 et UTF-16 sont pris en charge.

Espaces de nom

Les URI d'espace de nom suivant et leurs liaisons associées sont prédéfinies.

Nom d'espaces de nom	Préfixe	URI Espace de noms
Types de schéma XML	`xs:`	`http://www.w3.org/2001/XMLSchema`
Instance de schéma	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Fonctions intégrées	`fn:`	`http://www.w3.org/2005/xpath-functions`
Fonctions locales	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Veuillez noter les points suivants :

- Le moteur XQuery 1.0 Engine reconnaît les préfixes recensés ci-dessus comme étant liés aux espaces de noms correspondants.
- Étant donné que l'espace de noms des fonctions intégrées recensé ci-dessus est l'espace de noms des fonctions par défaut dans XQuery, le préfixe `fn:` ne doit pas nécessairement être utilisé lorsque des fonctions intégrées sont invoquées (par exemple, `string("Hello")` appellera la fonction `fn:string`). Néanmoins, le préfixe `fn:` peut être utilisé pour appeler une fonction intégrée sans avoir à déclarer l'espace de noms dans le prologue query (par exemple : `fn:string("Hello")`).
- Vous pouvez changer l'espace de noms des fonctions par défaut en déclarant l'expression `default function namespace` dans le prologue de de requête.
- En cas d'utilisation des types depuis l'espace de noms du Schéma XML, le préfixe `xs:` peut être utilisé sans devoir déclarer explicitement les espace de noms et lier ces préfixes dans le prologue de requête. (Exemple : `xs:date` and `xs:yearMonthDuration`.) Si vous souhaitez utiliser d'autres préfixes pour l'espace de noms du schéma XML, cela doit être déclaré explicitement dans le prologue de requête. (Exemple: `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)
- Veuillez noter que les types de données `untypedAtomic`, `dayTimeDuration`, et `yearMonthDuration` ont été déplacés, avec les CR de 23 January 2007, depuis l'espace de noms des Types de données XPath vers l'espace de noms du schéma XML, donc : `xs:yearMonthDuration`.

Si des espaces de noms pour les fonctions, les constructeurs de type, les tests de nœud, etc. sont mal attribués, une erreur est rapportée. Veuillez noter, néanmoins, que certaines fonctions portent le même nom que les types de données de schéma, par ex. `fn:string` et `fn:boolean`. (Les deux `xs:string` et `xs:boolean` sont définis.) Le préfixe d'espace de noms détermine si la fonction ou le constructeur de type est utilisé.

XML document de source et validation

Les documents XML utilisés dans l'exécution d'un document XQuery avec le moteur XQuery 1.0 doit être bien formé. Néanmoins, ils ne doivent pas être valides conformément à un schéma XML. Si le fichier n'est pas valide, le fichier invalide est chargé sans information de schéma. Si le fichier XML est associé avec un schéma interne et est valide conformément à ce schéma, l'information de validation post-schéma sera générée pour les données XML et sera utilisée pour l'évaluation de requête.

Contrôle de type statique et dynamique

La phase d'analyse statique contrôle les aspects de la requête comme la syntaxe, si des références externes existent (par ex. pour les modules), si des fonctions et des variables invoquées sont définies, etc. Si une erreur est détectée dans la phase de l'analyse statique, elle sera rapportée et l'exécution sera stoppée.

Le contrôle de type dynamique est effectué lors de l'exécution, lorsque la requête est réellement exécutée. Si un type est incompatible avec les exigences d'une opération, une erreur sera rapportée. Par exemple, l'expression `xs:string("1") + 1` retourne une erreur parce que l'opération d'édition ne peut pas être effectuée sur un opérande de type `xs:string`.

Modules de bibliothèque

Les modules de Bibliothèque stockent les fonctions et les variables de manière à ce qu'elles puissent être réutilisées. Le moteur XQuery 1.0 prend en charge des modules qui sont stockés **dans un seul fichier XQuery externe**. Un tel fichier de module doit contenir une déclaration `module` dans son prologue, qui associe un espace de noms cible. Voici un module d'exemple :

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

Toutes les fonctions et les variables déclarées dans le module font partie de l'espace de noms associé au module. Celui-ci est utilisé en l'important dans un fichier XQuery avec l'instruction `import module` se trouvant dans le prologue de requête. L'instruction `import module` importe uniquement les fonctions et les variables déclarées directement dans le fichier de module de bibliothèque. Comme suit :

```
import module namespace modlib = "urn:module-library" at "modulefilename.xq";
if         ($modlib:company = "Altova")
then        modlib:webaddress()
else        error("No match found.")
```

Fonctions externes

Les fonctions externes ne sont pas prises en charge, c.à.d. dans les expressions utilisant le mot-clé `external`, comme dans :

```
declare function hoo($param as xs:integer) as xs:string external;
```

Collations

La collation par défaut est la collation de point de code Unicode, qui compare les chaînes sur la base de leur point de code Unicode. Les autres collations prises en charge sont les collations ICU recensées ici[408]. Pour utiliser une collation spécifique, fournir son URI tel que donné dans la liste des collations prises en charge[408]. Toute comparaison de chaîne, y compris pour les fonctions `fn:max` et `fn:min` seront effectuées conformément à la collation spécifiée. Si l'option de collation n'est pas spécifiée, la collation de point de code Unicode par défaut est utilisée.

Précision des types numériques

- Le type de données `xs:integer` est une précision arbitraire, c.à.d. il peut représenter n'importe quel chiffre.
- Le type de données `xs:decimal` a une limite de 20 chiffres après la virgule.
- Les types de données `xs:float` et `xs:double` ont une précision limitée de 15 chiffres.

Prise en charge des instructions XQuery

L'instruction `Pragma` n'est pas prise en charge. Si elle survient, elle sera ignorée et l'expression de fallback sera évaluée.

Comportement spécifique à la mise en œuvre

CI-dessous, vous trouverez une description pour savoir comment les moteurs XQuery et XQuery Update 1.0 gèrent les aspects de certaines fonctions spécifiques à la mise en œuvre.

`unparsed-text`

L'argument `href` accepte (i) les chemins relatifs pour les fichiers dans le dossier base-uri et (ii) les chemins absolus avec ou sans `file://` protocol. Les encodages pris en charge de manière supplémentaire sont (spécifiques à Altova): `x-binarytobase16` et `x-binarytobase64`. Exemple : `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`

L'argument `href` accepte (i) les chemins relatifs pour les fichiers dans le dossier base-uri et (ii) les chemins absolus avec ou sans `file://` protocol. Les encodages pris en charge de manière supplémentaire sont (spécifiques à Altova): `x-binarytobase16` et `x-binarytobase64`.

Note : les valeurs d'encodage suivantes, qui sont mises en œuvre dans des versions antérieures du produit prédécesseur de RaptorXML, AltovaXML, sont dépcréciées à présent : `base16tobinary`, `base64tobinary`, `binarytobase16` and `binarytobase64`.

11.1.5 XQuery 3.1

Le moteur XQuery 3.1 de RaptorXML Server est conforme à la Recommandation XQuery 3.1 du 21 mars 2017 du Consortium du World Wide Web (W3C's) et comprend la prise en charge des Fonctions 3.1 XPath et XQuery. La spécification XQuery 3.1 est un sur ensemble de la spécification 3.0. C'est pourquoi le moteur XQuery 3.1 prend en charge les fonctions XQuery 3.0.

Les caractéristiques spécifiques à l'implémentation sont les mêmes que pour XQuery 1.0[403].

11.2 Fonctions XSLT et XPath/XQuery

Cette section réunit les fonctions d'extension Altova et d'autres fonctions d'extension qui peuvent être utilisées dans les expressions XPath et/ou XQuery. Les fonctions d'extension Altova peuvent être utilisées avec les moteurs XSLT et XQuery d'Altova, et elles offrent des fonctions supplémentaires à celles disponibles dans les bibliothèques de fonctions définies dans les standards W3C.

Cette section décrit principalement les fonctions d'extension XPath/XQuery qui ont été créées par Altova pour fournir des opérations supplémentaires. Ces fonctions 409 peuvent être calculées par les moteurs XSLT et XQuery d'Altova selon les règles décrites dans cette section. Pour information sur les fonctions XPath/XQuery régulières, voir la Fonction de référence XPath/XQuery d'Altova.

Points généraux

Les points généraux suivants devraient être notés :

- Les fonctions des bibliothèques de fonction core définies dans les spécifications W3C speuvent être appelées sans préfixe. La raison étant que les moteurs XSLT et XQuery d'Altova lisent les fonctions sans préfixe comme appartenant à l'espace de nom `http://www.w3.org/2005/xpath-functions`, qui est l'espace de nom des fonctions par défaut spécifiées dans les spécifications des fonctions XPath/XQuery. Si cet espace de nom est déclaré explicitement dans un document XSLT ou XQuery, le préfixe utilisé dans la déclaration d'espace de nom peut aussi être utilisé en option sur les noms de fonction.
- En général, si une fonction escompte une séquence d'un item en tant qu'argument, et qu'une séquence de plus d'un item est soumise, une erreur sera retournée.
- Toutes les comparaisons de strings sont réalisées en utilisant la collation de point de code Unicode.
- Les résultats qui sont des QNames sont sérialisés sous la forme `[prefix:]localname`.

Précision de la décimale xs:
La précision se réfère au nombre de chiffres dans le nombre et la spécification requiert un minimum de 18 chiffres. Pour les opérations de division qui produisent un résultat de type `xs:decimal`, la précision est de 19 chiffres après le point décimal sans arrondissement.

Fuseau horaire implicite
Lorsque deux valeurs `date`, `time`, ou `dateTime` doivent être comparées, le fuseau horaire des valeurs comparées doit être connu. Si le fuseau n'est explicitement donné dans une telle valeur, le fuseau horaire implicite est utilisé. Le fuseau horaire implicite est prélevé de l'horloge du système et sa valeur peut être contrôlée avec la fonction `implicit-timezone()`.

Collations
La collation par défaut est la collation de point de code Unicode qui compare les chaînes sur la base de leur point de code Unicode. Le processeur utilise l'Unicode Collation Algorithm. D'autres collations prises en charge sont les collations ICU recensées ci-dessous ; pour en utiliser une, fournissez son URI tel qu'énnoncé dans la table ci-dessous. Toute comparaison de chaîne, y compris en ce qui concerne les fonctions `max` et `min`, sera effectuée conformément à la collation spécifiée. Si l'option de collation n'est pas spécifiée, la collation de point de code Unicode par défaut sera utilisée.

Langage	URI

da: Danois	da_DK
de: Allemand	de_AT, de_BE, de_CH, de_DE, de_LI, de_LU
en: Anglais	en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW
es: Espagnol	es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE
fr: Français	fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG
it: Italien	it_CH, it_IT
ja: Japonais	ja_JP
nb: Norvégien Bokmål	nb_NO
nl: Néerlandais	nl_AW, nl_BE, nl_NL
nn: Nynorsk	nn_NO
pt: Portugais	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: Russe	ru_MD, ru_RU, ru_UA
sv: Suédois	sv_FI, sv_SE

Axe du nom d'espace
L'axe du nom d'espace est devenu obsolète dans XPath 2.0. Néanmoins, l'utilisation de l'axe du nom d'espace est prise en charge. Pour accéder aux information de l'espace de nom, avec des mécanismes XPath 2.0, utilisez les fonctions `in-scope-prefixes()`, `namespace-uri()` et `namespace-uri-for-prefix()`.

11.2.1 Fonctions d'extension Altova

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova,** `http://www.altova.com/xslt-extensions`, et sont indiquées dans cette section par le préfixe `altova:`, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Les fonctions définies dans les spécifications de Fonctions XPath/XQuery de W3C peuvent être utilisées dans : (i) les expressions XPath dans un contexte XSLT, et (ii) dans les expressions XQuery dans un document

XQuery. Dans cette documentation, nous indiquons les fonctions à utiliser dans le contexte précédent (XPath dans XSLT) avec un symbole XP et les appelons fonctions XPath ; les fonctions qui peuvent être utilisées dans le contexte à venir (XQuery) sont indiquées avec un symbole XQ ; elles fonctionnent en tant que fonctions XQuery. Les spécifications XSLT de W3C —pas les spécifications de Fonctions XPath/XQuery —définissent également les fonctions qui peuvent être utilisées dans des expressions XPath dans des documents XSLT. Ces fonctions sont marquées avec un symbole XSLT et sont appelées fonctions XSLT. Les versions XPath/XQuery et XSLT dans lesquelles une fonction peut être utilisée sont indiquées dans la description de la fonction (*voir symboles ci-dessous*). Les fonctions provenant des librairies de fonction XPath/XQuery et XSLT sont recensées dans un préfixe. Les fonctions d'extension provenant d'autres librairies, comme les fonctions d'extension Altova, sont regroupés avec un préfixe.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

Usage de Fonctions d'extension Altova

Pour pouvoir utiliser les fonctions d'extension d'Altova, vous devez déclarer l'espace de nom des fonctions d'extension d'Altova (*d'abord mettre en surbrillance dans la liste de codes ci-dessous*) puis utiliser les fonctions d'extension pou qu'elles soient résolues comme appartenant à cet espace de noms (*voir deuxième mise en surbrillance*). L'exemple ci-dessous utilise la fonction d'extension d'Altova appelée âge.

```
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    xmlns:altova="http://www.altova.com/xslt-extensions">
    <xsl:output method="text" encoding="ISO-8859-1"/>
    <xsl:template match="Persons">
        <xsl:for-each select="Person">
            <xsl:value-of select="concat(Name, ': ')"/>
            <xsl:value-of select="altova:age(xs:date(BirthDate))"/>
            <xsl:value-of select="' years&#x0A;'"/>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

Fonctions XSLT[411]

Les fonctions XSLT peuvent uniquement être utilisées dans des expressions XPath dans un contexte XSLT (comme les fonctions `current-group()` ou `key()` de XSLT 2.0). Ces fonctions ne sont pas prévues pour, et ne fonctionneront pas dans un contexte non-XSLT (par exemple, dans un contexte XQuery). Les fonctions XBRL Altova peuvent uniquement être utilisées avec des éditions des produits Altova qui présentent une prise en charge XBRL.

Fonctions XPath/XQuery

Les fonctions XPath/XQuery peuvent être utilisées tous les deux dans les expressions XPath dans les contextes XSLT et dans les expressions XQuery :

- Date/Heure 414
- Géolocalisation 431
- Liée à l'image 441
- Numérique 446
- Séquence 467
- String 476
- Divers 482

Fonctions Graphiques (Éditions Enterprise et Server uniquement seulement)

Les fonctions d'extension Altova pour les graphiques 484 sont prises en charge uniquement par les édition Enterprise et Server des produits Altova et permettent de générer des graphiques d'être généré à partir de données XML.

Fonctions de code-barres

Les fonctions d'extension à code-barres d'Altova 497 permettent la génération de code-barres et leur placement dans une sortie générée par le biais des feuilles de styles XSLT.

11.2.1.1 Fonctions XSLT

Les fonctions d'extension XSLT peuvent être utilisées dans les expressions XPath dans un contexte XSLT. Elles ne fonctionneront pas dans un contexte non-XSLT (par exemple dans un contexte XQuery).

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova,** `http://www.altova.com/xslt-extensions`, et sont indiquées dans cette section par le préfixe `altova:`, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

Fonctions générales

distinct-nodes [altova:]

altova:distinct-nodes(*node()**) **asnode()*** XSLT1 XSLT2 XSLT3

Prend un ensemble d'un ou de plusieurs nœuds en tant que son entrée et retourne le même ensemble moins les nœuds avec des valeurs dupliquées. La comparaison s'effectue en utilisant la fonction XPath/XQuery `fn:deep-equal`.

Exemples

- **altova:distinct-nodes**(country) retourne tous les nœuds `country` enfant moins ceux possédant des valeurs dupliquées.

evaluate [altova:]

altova:evaluate(XPathExpression *as xs:string***[, ValueOf$p1, ... ValueOf$pN])** XSLT1 XSLT2 XSLT3

Prend une expression XPath, passée en tant que chaîne, en tant que son argument obligatoire. Elle retourne la sortie de l'expression évaluée. Par exemple : **altova:evaluate('//Name[1]')** retourne les contenues du premier élément `Name` dans le document. Veuillez noter que l'expression `//Name[1]` est passée en tant que chaîne en l'enfermant dans des guillemets simples.

La fonction `altova:evaluate` peut prendre des arguments supplémentaires en option. Ces arguments sont les valeurs des variables in-scope qui portent les noms `p1`, `p2`, `p3`... `pN`. Veuillez noter les points suivants concernant l'utilisation : (i) Les variables doivent être définies avec les noms de la formule `pX`, lorsque `X` est un entier ; (ii) les arguments de la fonction `altova:evaluate` (*voir signature ci-dessus*), à partir du deuxième argument, fournissent les valeurs de la variables, avec la séquence des arguments correspondant à la séquence des variables classées numériquement : `p1` à `pN`: le deuxième argument sera la valeur de la variable `p1`, le troisième argument celui de la variable `p2`, etc. ; (iii) Les valeurs de variable doivent être de type `item*`.

Exemple

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

outputs `"hi 20 10"`

Dans les listes ci-dessus, veuillez noter les points suivants :

- Le deuxième argument de l'expression `altova:evaluate` est la valeur attribuée à la variable `$p1`, le troisième argument est celui attribué à la variable `$p2`, etc.
- Veuillez noter que le quatrième argument de la fonction est une valeur de chaîne, ce qui est indiqué par le fait qu'elle est contenue dans des guillemets.
- L'attribut `select` de l'élément `xs:variable` fournit l'expression XPath. Puisque cette expression doit être de type `xs:string`, elle est contenue dans des guillemets simples.

Exemples pour mieux illustrer l'utilisation des variables

-
```
<xsl:variable name="xpath" select="'$p1'" />
<xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
```
Sort la valeur du premier élément `Name`.

- `<xsl:variable name="xpath" select="'$p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />`
  Sort "//Name[1]"

La fonction d'extension `altova:evaluate()` est utile lorsqu'une expression XPath dans la feuille de style XSLT contient une ou plusieurs parties qui doivent être évaluées dynamiquement. Par exemple, prenez comme exemple une situation dans laquelle un utilisateur saisit sa requête pour le critère de tri et le critère est stocké dans l'attribut `UserReq/@sortkey`. Dans la feuille de style, vous pouvez ensuite avoir l'expression : `<xsl:sort select="altova:evaluate(../UserReq/@sortkey)" order="ascending"/>`. La fonction `altova:evaluate()` lit l'attribut `sortkey` de l'élément enfant `UserReq` du parent du nœud contextuel. Si, par exemple, la valeur de l'attribut `sortkey` est `Price`, alors `Price` est retourné par la fonction `altova:evaluate()` et devient la valeur de l'attribut `select` : `<xsl:sort select="Price" order="ascending"/>`. Si cette instruction `sort` apparaît dans le contexte d'un élément appelé `Order`, alors les éléments `Order` seront triés conformément aux valeurs de leurs enfants `Price`. En alternative, si la valeur de `@sortkey` était, par exemple, `Date`, alors les éléments `Order` seraient triés selon les valeurs de leurs enfants `Date`. Donc le critère de triage pour `Order` est choisi à partir de l'attribut `sortkey` lors de l'exécution. Cela n'aurait pas pu se réaliser avec une expression telle que : `<xsl:sort select="../UserReq/@sortkey" order="ascending"/>`. Dans le cas montré ci-dessus, le critère de tri aurait été l'attribut `sortkey` lui-même, et non pas `Price` ou `Date` (ou tout autre contenu actuel de `sortkey`).

Note : Le contexte statique inclut des espaces de nom, des types et des fonctions, mais pas des variables, depuis l'environnement d'appel. L'URI de base et l'espace de nom par défaut sont hérités.

Plus d'exemples

- Variables statiques : `<xsl:value-of select="$i3, $i2, $i1" />`
  Sort les valeurs des trois variables.

- Expression XPath dynamique avec des variables dynamiques :
  `<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />`
  Sortie "30 20 10"

- Expression XPath dynamique sans variable dynamique :
  `<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath)" />`
  Sortie erreur : Aucune variable définie pour $p3.

encode-for-rtf [altova:]

altova:encode-for-rtf(input *as xs:string*, **preserveallwhitespace** *as xs:boolean*, **preservenewlines** *as xs:boolean*) **asxs:string** XSLT2 XSLT3

Convertit la chaîne d'entrée en tant que code pour RTF. Les espaces blancs et les nouvelles lignes seront préservés selon la valeur booléenne spécifiée pour leurs arguments respectifs.

[Haut 411]

Fonctions XBRL

Les fonctions XBRL Altova peuvent uniquement être utilisées avec des éditions des produits Altova qui présentent une prise en charge XBRL.

▼ xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***) asnode()*** XSLT2 XSLT3
Prend un nœud en tant que son argument d'entrée et retourne l'ensemble des nœuds de notes de pieds XBRL référencées par le nœud d'entrée.

▼ xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName***, *xs:string***) asnode()*** XSLT2 XSLT3
Prend deux arguments d'entrée : un nom de nœud et l'emplacement de fichier de taxonomie contenant le nœud. La fonction retourne les nœuds de libellés XBRL associés avec le nœud d'entrée.

[Haut 411]

11.2.1.2 Fonctions XPath/XQuery : Date et heure

Les fonctions d'extension date/heure d'Altova peuvent être utilisées dans les expressions XPath et XQuery et fournissent des fonctions supplémentaires pour le traitement des données contenues en tant que les types de données de date et d'heures variés de XML Schema. Les fonctions dans cette section peuvent être utilisées avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova. Ils sont disponibles dans des contextes XPath/XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova,** `http://www.altova.com/xslt-extensions`, et sont indiquées dans cette section par le préfixe `altova:`, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

▼ Regroupées selon les fonctionnalités

[Haut 414]

Ajouter une durée à xs:dateTime XP3.1 XQ3.1

Ces fonctions ajoutent une durée à **xs:dateTime** et retournent **xs:dateTime**. Le type xs:dateTime a un format de CCYY-MM-DDThh:mm:ss.sss. Il s'agit d'une concaténation des formats xs:date et xs:time séparés par la lettre T. Un suffixe de fuseau horaire (+01:00, par exemple) est optionnel.

add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(DateTime *as xs:dateTime*, **Years** *as xs:integer*)
asxs:dateTime XP3.1 XQ3.1

Ajoute une durée en années à xs:dateTime (*voir exemples ci-dessous*). Le deuxième argument est le nombre d'années à être ajouté à xs:dateTime fourni en tant que le premier argument. Le résultat est de type xs:dateTime.

Exemples

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) retourne 2024-01-15T14:00:00
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) retourne 2010-01-15T14:00:00

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *as xs:dateTime*, **Months** *as xs:integer*)
asxs:dateTime XP3.1 XQ3.1

Ajoute une durée en mois à xs:dateTime (*voir exemples ci-dessous*). Le deuxième argument est le nombre de mois à être ajouté à xs:dateTime fourni en tant que le premier argument. Le résultat est de type xs:dateTime.

Exemples

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) retourne 2014-11-15T14:00:00
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) retourne 2013-11-15T14:00:00

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *as xs:dateTime*, **Days** *as xs:integer*) **asxs:dateTime** XP3.1 XQ3.1

Ajoute une durée en jours à xs:dateTime (*voir exemples ci-dessous*). Le deuxième argument est le nombre de jours à être ajouté à xs:dateTime fourni en tant que le premier argument. Le résultat est de type xs:dateTime.

Exemples

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) retourne 2014-01-25T14:00:00
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) retourne 2014-01-07T14:00:00

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *as xs:dateTime*, **Hours** *as xs:integer***)**
asxs:dateTime **XP3.1** **XQ3.1**
Ajoute une durée en heures à `xs:dateTime` (*voir exemples ci-dessous*). Le deuxième argument est le nombre d'heures à être ajouté à `xs:dateTime` fourni en tant que le premier argument. Le résultat est de type `xs:dateTime`.

Exemples

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) retourne 2014-01-15T23:00:00
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) retourne 2014-01-15T05:00:00

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *as xs:dateTime*, **Minutes** *as xs:integer***)**
asxs:dateTime **XP3.1** **XQ3.1**
Ajoute une durée en minutes à `xs:dateTime` (*voir exemples ci-dessous*). Le deuxième argument est le nombre of minutes à être ajouté à `xs:dateTime` fourni en tant que le premier argument. Le résultat est de type `xs:dateTime`.

Exemples

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) retourne 2014-01-15T14:55:00
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) retourne 2014-01-15T14:05:00

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *as xs:dateTime*, **Seconds** *as xs:integer***)**
asxs:dateTime **XP3.1** **XQ3.1**
Ajoute une durée en secondes à `xs:dateTime` (*voir exemples ci-dessous*). Le deuxième argument est le nombre de secondes à être ajouté à `xs:dateTime` fourni en tant que le premier argument. Le résultat est de type `xs:dateTime`.

Exemples

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) retourne 2014-01-15T14:00:30
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) retourne 2014-01-15T14:00:05

[Haut 414]

Ajouter une durée à xs: date **XP3.1** **XQ3.1**

Ces fonctions ajoutent une durée à **xs: date** et retournent **xs: date**. Le type `xs:date` a un format `CCYY-MM-DD`.

add-years-to-date [altova:]

altova:add-years-to-date(Date *as xs:date*, **Years** *as xs:integer*) **asxs:date** XP3.1 XQ3.1
Ajoute une durée en années à une date. Le deuxième argument est le nombre d'années à être ajouté à `xs:date` fourni en tant que le premier argument. Le résultat est de type `xs:date`.

Exemples

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) retourne 2024-01-15
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) retourne 2010-01-15

add-months-to-date [altova:]

altova:add-months-to-date(Date *as xs:date*, **Months** *as xs:integer*) **asxs:date** XP3.1 XQ3.1
Ajoute une durée en mois à une date. Le deuxième argument est le nombre de mois à être ajouté à `xs:date` fourni en tant que le premier argument. Le résultat est de type `xs:date`.

Exemples

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) retourne 2014-11-15
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) retourne 2013-11-15

add-days-to-date [altova:]

altova:add-days-to-date(Date *as xs:date*, **Days** *as xs:integer*) **asxs:date** XP3.1 XQ3.1
Ajoute une durée en jours à une date. Le deuxième argument est le nombre de jours à être ajouté à `xs:date` fourni en tant que le premier argument. Le résultat est de type `xs:date`.

Exemples

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) retourne 2014-01-25
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) retourne 2014-01-07

[Haut 414]

Formater et récupérer des durées XP3.1 XQ3.1

Ces fonctions parsent une entrée **xs:duration** ou **xs:string** et retournent respectivement un **xs:string** ou **xs:duration**.

format-duration [altova:]

altova:format-duration(Duration *as xs:duration*, **Picture** *as xs:string*) **asxs:string** XP3.1 XQ3.1
Formate une durée qui est soumise en tant que le premier argument, selon une chaîne d'image soumise en tant que le second argument. La sortie est une chaîne de texte formatée conformément à la chaîne d'image.

Exemples

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") retourne "Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"
- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") retourne "Months:03 Days:02 Hours:02 Minutes:53"

parse-duration [altova:]

altova:parse-duration(InputString *as xs:string*, **Picture** *as xs:string*) **asxs:duration** XP3.1 XQ3.1

Prend un patterned string en tant que le premier argument et une chaîne image en tant que le second argument. La chaîne d'entrée est parsée sur la base de la chaîne d'image et une `xs:duration` est retournée.

Exemples

- **altova:parse-duration**("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") retourne "P2DT2H53M11.7S"
- **altova:parse-duration**("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") retourne "P3M2DT2H53M"

[Haut 414]

Ajouter une durée à xs:time XP3.1 XQ3.1

Ces fonctions ajoutent une durée à **xs:time** et retournent **xs:time**. Le type `xs:time` a une forme lexicale de `hh:mm:ss.sss`. Un fuseau horaire en option peut être suffixé. La lettre `Z` indique le Temps universel coordonné (UTC). Tous les autres fuseaux horaires sont représentés par leur différence de l'UTC dans le format `+hh:mm`, ou `-hh:mm`. Si aucune valeur de fuseau horaire n'est présente, elle est considérée inconnue ; elle n'est pas considérée être UTC.

add-hours-to-time [altova:]

altova:add-hours-to-time(Time *as xs:time*, **Hours** *as xs:integer*) **asxs:time** XP3.1 XQ3.1

Ajoute une durée en heures à une heure de temps. Le deuxième argument est le nombre d'heures à être ajouté à `xs:time` fourni en tant que le premier argument. Le résultat est de type `xs:time`.

Exemples

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) retourne 21:00:00
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) retourne 04:00:00

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) **asxs:time** XP3.1 XQ3.1

Ajoute une durée en minutes à une heure. Le deuxième argument est le nombre de minutes à être ajouté à `xs:time` fourni en tant que le premier argument. Le résultat est de type `xs:time`.

Exemples

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) retourne 14:55:00
- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) retourne 14:05:00

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) **asxs:time** XP3.1 XQ3.1

Ajoute une durée en secondes à une heure. Le deuxième argument est le nombre de secondes à être ajouté à `xs:time` fourni en tant que le premier argument. Le résultat est de type `xs:time`. Le composant Secondes peut être contenu dans une plage de `0` à `59.999`.

Exemples

- **`altova:add-seconds-to-time`**`(xs:time("14:00:00"), 20)` retourne `14:00:20`
- **`altova:add-seconds-to-time`**`(xs:time("14:00:00"), 20.895)` retourne `14:00:20.895`

[Haut 414]

Supprimer la partie du fuseau horaire des types de données date/heures XP3.1 XQ3.1

Ces fonctions permettent de supprimer le fuseau horaire des valeurs **`xs:dateTime`**, **`xs:date`** ou **`xs:time`** actuelles, respectivement. Veuillez noter que la différence entre `xs:dateTime` et `xs:dateTimeStamp` est que dans le cas de ce dernier, la partie fuseau horaire est requise (alors qu'elle est optionnelle dans le premier des deux cas). Donc, le format d'une valeur `xs:dateTimeStamp` est : `CCYY-MM-DDThh:mm:ss.sss±hh:mm`. ou `CCYY-MM-DDThh:mm:ss.sssZ`. Si la date et l'heure sont lues depuis l'horloge du système, en tant que `xs:dateTimeStamp`, la fonction `current-dateTime-no-TZ()` peut être utilisée pour supprimer le fuseau horaire s'il est requis.

current-date-no-TZ [altova:]

`altova:current-date-no-TZ()` **`as xs:date`** XP3.1 XQ3.1

Cette fonction ne prend aucun argument. Elle supprime la partie fuseau horaire de `current-date()` (qui est la date actuelle selon l'horloge système) et retourne une valeur `xs:date`.

Exemples

Si la date actuelle est **`2014-01-15+01:00`**:

- **`altova:current-date-no-TZ`**`()` retourne `2014-01-15`

current-dateTime-no-TZ [altova:]

`altova:current-dateTime-no-TZ()` **`as xs:dateTime`** XP3.1 XQ3.1

Cette fonction ne prend aucun argument. Elle supprime la partie fuseau horaire de `current-dateTime()` (qui est la date-heure actuelle selon l'horloge système) et retourne une valeur`xs:dateTime`.

Exemples

Si la date-heure actuelle est **`2014-01-15T14:00:00+01:00`**:

- **`altova:current-dateTime-no-TZ`**`()` retourne `2014-01-15T14:00:00`

current-time-no-TZ [altova:]

`altova:current-time-no-TZ()` **`as xs:time`** XP3.1 XQ3.1

Cette fonction ne prend aucun argument. Elle supprime la partie fuseau horaire de `current-time()` (qui est l'heure actuelle selon l'horloge système) et retourne une valeur `xs:time`.

Exemples

Si l'heure actuelle est **`14:00:00+01:00`**:

- **`altova:current-time-no-TZ`**`()` retourne `14:00:00`

date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date***) asxs:date XP3.1 XQ3.1**
Cette fonction prend un argument `xs:date`, en supprime la partie fuseau horaire et retourne une valeur `xs:date`. Veuillez noter que la date n'est pas modifiée.

Exemples

- **altova:date-no-TZ**(xs:date("2014-01-15+01:00")) retourne 2014-01-15

dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *as xs:dateTime***) asxs:dateTime XP3.1 XQ3.1**
Cette fonction prend un argument `xs:dateTime`, en supprime la partie fuseau horaire, et retourne une valeur `xs:dateTime`. Veuillez noter que ni la date ni l'heure n'est modifiée.

Exemples

- **altova:dateTime-no-TZ**(xs:date("2014-01-15T14:00:00+01:00")) retourne 2014-01-15T14:00:00

time-no-TZ [altova:]

altova:time-no-TZ(InputTime *as xs:time***) asxs:time XP3.1 XQ3.1**
Cette fonction prend un argument `xs:time`, en supprime la partie de fuseau horaire, et retourne une valeur `xs:time`. Veuillez noter que l'heure n'est pas modifiée.

Exemples

- **altova:time-no-TZ**(xs:time("14:00:00+01:00")) retourne 14:00:00

[Haut 414]

Retourne le nombre de jours, d'heures, de minutes, de secondes des durées XP3.1 XQ3.1

Ces fonctions retournent le nombre de jours dans un mois, et le nombre d'heures, de minutes et de secondes, respectivement depuis les durées.

days-in-month [altova:]

altova:days-in-month(Year *as xs:integer*, **Month** *as xs:integer***) asxs:integer XP3.1 XQ3.1**
Retourne le nombre de jours dans le mois spécifié. Le mois est spécifié avec les arguments `Year` et `Month`.

Exemples

- **altova:days-in-month**(2018, 10) retourne 31
- **altova:days-in-month**(2018, 2) retourne 28
- **altova:days-in-month**(2020, 2) retourne 29

hours-from-dayTimeDuration-accumulated

altova:hours-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*) **asxs:integer**
XP3.1 XQ3.1
Retourne le nombre total d'heures dans la durée soumise par l'argument `DayAndTime` (qui est de type `xs:duration`). Les heures dans les composants `Day` et `Time` sont additionnés pour donner un résultat qui est un entier. Le décompte d'une nouvelle heure dure uniquement 60 minutes complètes. Des durées négatives entraînent une valeur d'heures négative.

Exemples

- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5D")) retourne 120, qui est le nombre total d'heures dans 5 jours.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H")) retourne 122, qui est le nombre total d'heures dans 5 jours plus 2 heures.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H60M")) retourne 123, qui est le nombre total d'heures dans 5 jours plus 2 heures et 60 mins.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H119M")) retourne 123, qui est le nombre total d'heures dans 5 jours plus 2 heures et 119 mins.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H120M")) retourne 124,qui est le nombre total d'heures dans 5 jours plus 2 heures et 120 mins.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("-P5DT2H")) retourne -122

minutes-from-dayTimeDuration-accumulated

altova:minutes-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*) **asxs:integer**
XP3.1 XQ3.1
Retourne le nombre total de minutes dans la durée soumise par l'argument `DayAndTime` (qui est de type `xs:duration`). Les minutes dans les composants `Day` et `Time` sont additionnés pour donner un résultat qui est un entier. Des durées négatives entraînent une valeur de minute négative.

Exemples

- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT60M")) retourne 60
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H")) retourne 60, qui est le nombre total de minutes dans une heure.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H40M")) retourne 100
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("P1D")) retourne 1440, qui est le nombre total de minutes dans un jour.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("-P1DT60M")) retourne -1500

seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*) **asxs:integer**
XP3.1 XQ3.1
Retourne le nombre total de minutes dans la durée soumise par l'argument `DayAndTime` (qui est de type `xs:duration`). Les minutes dans les composants `Day` et `Time` sont additionnés pour donner un résultat qui est un entier. Des durées négatives entraînent une valeur de minute négative .

Exemples

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1M")) retourne 60,

qui est le nombre total de secondes dans une minute.
- `altova:seconds-from-dayTimeDuration-accumulated(xs:duration("PT1H"))` retourne `3600`, qui est le nombre total de secondes dans une heure.
- `altova:seconds-from-dayTimeDuration-accumulated(xs:duration("PT1H2M"))` retourne `3720`
- `altova:seconds-from-dayTimeDuration-accumulated(xs:duration("P1D"))` retourne `86400`, qui est le nombre total de secondes dans un jour.
- `altova:seconds-from-dayTimeDuration-accumulated(xs:duration("-P1DT1M"))` retourne `-86460`

Retourne le jour de la semaine à partir de xs:dateTime ou xs:date XP3.1 XQ3.1

Ces fonctions retournent le jour de la semaine (en tant qu'entier) depuis `xs:dateTime` ou `xs:date`. Les jours de la semaine sont numérotés (format américain) de `1` à `7`, avec `Sunday=1`. Dans le format européen, la semaine commence par Lundi (`=1`). Dans le format américain elle commence par `Sunday=1`. Configurer en utilisant l'entier `0` et où un entier est accepté pour indiquer le format.

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *as xs:dateTime*) **asxs:integer** XP3.1 XQ3.1

Prend une date-avec-heure en tant que son seul argument et retourne le jour de la semaine de cette date sous forme d'un entier. Les jours de la semaine sont numérotés en commençant avec `Sunday=1`. Si le format européen est requis (où `Monday=1`), utiliser l'autre signature de cette fonction (*voir signature suivante ci-dessous*).

Exemples

- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"))` retourne `2`, ce qui indique un lundi.

altova:weekday-from-dateTime(DateTime *as xs:dateTime*, **Format** *as xs:integer*) **asxs:integer** XP3.1 XQ3.1

Prend une date-avec-heure en tant que son premier argument et retourne le jour de la semaine de cette date sous forme d'un entier. Si le second argument (entier) est `0`, les jours de la semaine sont numérotés de `1` à `7` en commençant avec `Sunday=1`. Si le second argument est un entier différent de `0`, alors `Monday=1`. S'il n'y a pas de second argument, la fonction est lue comme possédant l'autre signature de cette fonction (*voir signature précédente*).

Exemples

- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 1)` retourne `1`, ce qui indique un lundi
- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 4)` retourne `1`, ce qui indique un lundi
- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 0)` retourne `2`, ce qui indique un lundi.

weekday-from-date [altova:]

altova:weekday-from-date(Date *as xs:date*) **asxs:integer** XP3.1 XQ3.1

Prend une date en tant que son seul argument et retourne le jour de la semaine de cette date sous forme

d'un entier. Les jours de la semaine sont numérotés en commençant avec `Sunday=1`. Si le format européen est requis (où `Monday=1`), utiliser l'autre signature de cette fonction (*voir signature suivante ci-dessous*).

Exemples

- **`altova:weekday-from-date`**`(xs:date("2014-02-03+01:00"))` retourne `2`, ce qui indique un lundi.

`altova:weekday-from-date(Date` *`as xs:date`*`, `**`Format`** *`as xs:integer`*`)` **`asxs:integer`** XP3.1 XQ3.1
Prend une date en tant que son premier argument et retourne le jour de la semaine de cette date sous forme d'un entier. Si le second argument (`Format`) est `0`, les jours de la semaine sont numérotés de `1` à `7` en commençant avec `Sunday=1`. Si le second argument est un entier différent de `0`, alors `Monday=1`. S'il n'y a pas de second argument, la fonction est lue comme possédant l'autre signature de cette fonction (*voir signature précédente*).

Exemples

- **`altova:weekday-from-date`**`(xs:date("2014-02-03"), 1)` retourne `1`, ce qui indique un lundi
- **`altova:weekday-from-date`**`(xs:date("2014-02-03"), 4)` retourne `1`, ce qui indique un lundi
- **`altova:weekday-from-date`**`(xs:date("2014-02-03"), 0)` retourne `2`, ce qui indique un lundi.

[Haut 414]

Retourne le nombre de la semaine à partir de xs:dateTime ou xs:date XP2 XQ1 XP3.1 XQ3.1

Ces fonctions retournent le nombre de la semaine (en tant qu'entier) depuis `xs:dateTime` ou `xs:date`. La numérotation des semaines est disponible dans les formats de calendrier US, ISO/European et Islamiques. La numérotation des semaines est différente dans ces formats de calendrier parce que la semaine est considérée démarrer avec un jour différent selon le format (dimanche pour le format US, lundi pour le format ISO/Européen, et samedi dans le format islamique).

weeknumber-from-date [altova:]

`altova:weeknumber-from-date(Date` *`as xs:date`*`, `**`Calendar`** *`as xs:integer`*`)` **`asxs:integer`** XP2 XQ1 XP3.1 XQ3.1
Retourne le numéro de la semaine de l'argument **`Date`** soumis en tant qu'entier. Le deuxième argument (**`Calendar`**) spécifie le système de calendrier à suivre.
Les valeurs de **`Calendar`** prises en charge sont :

- **`0 = US calendar`** (*semaine commence dimanche*)
- **`1 = ISO standard, European calendar`** (*semaine commence lundi*)
- **`2 = Islamic calendar`** (*semaine commence samedi*)

Le réglage par défaut est `0`.

Exemples

- **`altova:weeknumber-from-date`**`(xs:date("2014-03-23"), 0)` retourne `13`
- **`altova:weeknumber-from-date`**`(xs:date("2014-03-23"), 1)` retourne `12`
- **`altova:weeknumber-from-date`**`(xs:date("2014-03-23"), 2)` retourne `13`
- **`altova:weeknumber-from-date`**`(xs:date("2014-03-23") )` retourne `13`

Le jour de la date dans les exemples ci-dessus (2014-03-23) est dimanche. Les calendriers US et musulmans sont donc une semaine en avant par rapport au calendrier européen à ce jour.

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *as xs:dateTime*, **Calendar** *as xs:integer*)
asxs:integer XP2 XQ1 XP3.1 XQ3.1

Retourne le numéro de la semaine de l'argument **DateTime** soumis en tant qu'entier. Le deuxième argument (**Calendar**) spécifie le système de calendrier à suivre.
Les valeurs de **Calendar** prises en charge sont :

- **0 = US calendar** (*semaine commence dimanche*)
- **1 = ISO standard, European calendar** (*semaine commence lundi*)
- **2 = Islamic calendar** (*semaine commence samedi*)

Le réglage par défaut est 0.

Exemples

- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 0) retourne 13
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 1) retourne 12
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 2) retourne 13
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00")) retourne 13

Le jour du dateTime dans les exemples ci-dessus (2014-03-23T00:00:00) est dimanche. Les calendriers US et musulmans sont donc une semaine en avant par rapport au calendrier européen à ce jour.

[Haut 414]

Construire le type de date, d'heure ou de durée à partir de leurs composants lexicaux XP3.1 XQ3.1

Les fonctions prennent les composants lexicaux du type de données xs:date, xs:time ou xs:duration en tant qu'arguments d'entrée et les combinent pour construire le type de données respectif.

build-date [altova:]

altova:build-date(Year *as xs:integer*, **Month** *as xs:integer*, **Date** *as xs:integer*)
asxs:date XP3.1 XQ3.1

Les premier, second et troisième arguments sont respectivement l'année, le mois et la date. Ils sont combinés pour construire une valeur de type xs:date. Les valeurs de l'entier doivent se situer dans le cadre de la plage correcte de cette partie de la date. Par exemple, le deuxième argument (pour la partie du mois) ne devrait pas être supérieur à 12.

Exemples

- **altova:build-date**(2014, 2, 03) retourne 2014-02-03

▼ build-time [altova:]

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) **asxs:time** **XP3.1** **XQ3.1**

Les premiers, seconds et troisièmes arguments sont, respectivement, les valeurs d'heure (`0` to `23`), de minutes (`0` to `59`) et de secondes (`0` to `59`). Ils sont combinés pour construire une valeur de type `xs:time`. Les valeurs des entiers doivent se trouver dans le cadre de la plage correcte de cette partir de temps en particulier. Par exemple, le deuxième argument (`Minutes`) ne devrait pas être supérieur à `59`. Pour ajouter une partie fuseau horaire à la valeur, utiliser l'autre signature de cette fonction (*voir signature suivante*).

Exemples

- `altova:build-time(23, 4, 57)` retourne `23:04:57`

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*, **TimeZone** *as xs:string*) **asxs:time** **XP3.1** **XQ3.1**

Les premiers, seconds et troisièmes arguments sont, respectivement, les valeurs d'heure (`0` to `23`), de minutes (`0` to `59`) et de secondes (`0` to `59`). Le quatrième argument est une chaîne qui fournit la partie fuseau horaire de la valeur. Les quatre arguments sont combinés pour construire une valeur de type `xs:time`. Les valeurs des entiers doivent se trouver dans le cadre de la plage correcte de cette partie de temps en particulier. Par exemple, le deuxième argument (`Minutes`) ne doit pas être supérieur à `59`.

Exemples

- `altova:build-time(23, 4, 57, '+1')` retourne `23:04:57+01:00`

▼ build-duration [altova:]

altova:build-duration(Years *as xs:integer*, **Months** *as xs:integer*) **asxs:yearMonthDuration** **XP3.1** **XQ3.1**

Prend deux arguments pour construire une valeur de type `xs:yearMonthDuration`. Les premiers arguments fournissent la partie `Years` de la valeur de durée, alors que le deuxième argument fournie la partie `Months`. Si le deuxième argument (`Months`) est supérieur ou égale à `12`, alors l'entier est divisé par `12`; le quotient est ajouté au premier argument pour fournir la partie `Years` de la valeur de durée alors que le reste (de la division) fournit la partie `Months`. Pour construire une durée de type `xs:dayTimeDuration`., voir la signature suivante.

Exemples

- `altova:build-duration(2, 10)` retourne `P2Y10M`
- `altova:build-duration(14, 27)` retourne `P16Y3M`
- `altova:build-duration(2, 24)` retourne `P4Y`

altova:build-duration(Days *as xs:integer*, **Hours** *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) as **xs:dayTimeDuration** **XP3.1** **XQ3.1**

Prend quatre arguments et les combine pour construire une valeur de type `xs:dayTimeDuration`. Le premier argument fournit la partie `Days` de la valeur de durée, le deuxième, troisième et quatrième argument fournit respectivement les parties `Hours`, `Minutes` et `Seconds` de la valeur de durée. Chacun des trois arguments Time est converti en une valeur équivalente en termes de l'unité suivante plus élevée et le résultat est utilisé pour le calcul d'une valeur de durée totale. Par exemple, 72 secondes est converti en `1M+12S` (1 minute et 12 secondes), et cette valeur est utilisée pour le calcul de la valeur de durée totale. Pour construire une durée de type `xs:yearMonthDuration`., voir la signature précédente.

Exemples

- `altova:build-duration(2, 10, 3, 56)` retourne `P2DT10H3M56S`
- `altova:build-duration(1, 0, 100, 0)` retourne `P1DT1H40M`

- `altova:build-duration(1, 0, 0, 3600)` retourne `P1DT1H`

[Haut 414]

Construire le type de date, dateHeure ou heure à partir de l'entrée de chaîne XP2 XQ1 XP3.1 XQ3.1

Ces fonctions prennent des chaînes en tant qu'arguments et construisent des types de données `xs:date`, `xs:dateTime`, ou `xs:time`. La chaîne est analysée pour les composants du type de données basé sur un argument de modèle soumis.

parse-date [altova:]

altova:parse-date(Date *as xs:string*, **DatePattern** *as xs:string*) **as xs:date** XP2 XQ1 XP3.1 XQ3.1

Retourne la chaîne d'entrée **Date** en tant qu'une valeur **xs:date**. Le deuxième argument **DatePattern** spécifie le modèle (séquence des composants) de la chaîne d'entrée. **DatePattern** est décrit avec les spécificateurs de composants regroupés ci-dessous et avec les séparateurs de composant qui peuvent être n'importe quel caractère. Voir les exemples ci-dessous.

D	Jour
M	Mois
Y	Année

Le modèle dans **DatePattern** doit correspondre au modèle dans **Date**. Puisque la sortie est de type **xs:date**, la sortie aura toujours le format lexical **YYYY-MM-DD**.

Exemples

- **altova:parse-date**(xs:string("09-12-2014"), "[D]-[M]-[Y]") retourne `2014-12-09`
- **altova:parse-date**(xs:string("09-12-2014"), "[M]-[D]-[Y]") retourne `2014-09-12`
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") retourne `2014-06-03`
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") retourne `2014-06-03`
- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") retourne `2014-06-03`

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *as xs:string*, **DateTimePattern** *as xs:string*) **asxs:dateTime** XP2 XQ1 XP3.1 XQ3.1

Retourne la chaîne d'entrée **DateTime** en tant que valeur **xs:dateTime**. Le deuxième argument **DateTimePattern** spécife le modèle (séquence des composants) de la chaîne d'entrée. **DateTimePattern** est décrit avec les spécificateurs de composants regroupés ci-dessous et avec les séparateurs de composant qui peuvent être n'importe quel caractère. Voir les exemples ci-dessous.

D	Date
M	Mois
Y	Année
H	Heure
m	Minutes
s	Secondes

Le modèle dans **DateTimePattern** doit correspondre au modèle dans **DateTime**. Puisque la sortie est de type **xs:dateTime**, la sortie aura toujours le format lexical **YYYY-MM-DDTHH:mm:ss**.

Exemples

- **altova:parse-dateTime**(xs:string("09-12-2014 13:56:24"), "[M]-[D]-[Y] [H]:[m]:[s]") retourne 2014-09-12T13:56:24
- **altova:parse-dateTime**("time=13:56:24; date=09-12-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") retourne 2014-12-09T13:56:24

parse-time [altova:]

altova:parse-time(Time *as xs:string*, **TimePattern** *as xs:string*) **asxs:time** XP2 XQ1 XP3.1 XQ3.1

Retourne la chaîne d'entrée **Time** en tant qu'une valeur **xs:time**.Le deuxième argument **TimePattern** spécifie le modèle (séquence des composants) de la chaîne d'entrée. **TimePattern** est décrit avec les spécificateurs de composants regroupés ci-dessous et avec les séparateurs de composant qui peuvent être n'importe quel caractère. Voir les exemples ci-dessous.

H	Heure
m	minutes
s	secondes

Le modèle dans **TimePattern** doit correspondre au modèle dans **Time**. Puisque la sortie est de type **xs:time**, la sortie aura toujours le format lexical **HH:mm:ss**.

Exemples

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") retourne 13:56:24
- **altova:parse-time**("13-56-24", "[H]-[m]") retourne 13:56:00
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") retourne 13:56:24
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") retourne 13:56:24

[Haut 414]

Fonctions liées à l'âge XP3.1 XQ3.1

Ces fonctions retournent l'âge tel que calculé (i) entre une date d'argument d'entrée et la date actuelle, ou (ii) entre deux dates d'argument d'entrée. La fonction **altova:age** retourne l'âge en termes d'années, la fonction **altova:age-details** retourne l'âge en tant qu'une séquence de trois entiers indiquant les années, mois et jours de l'âge.

age [altova:]

altova:age(StartDate *as xs:date*) **asxs:integer** XP3.1 XQ3.1

Retourne un entier représentant l'âge *en années* d'un objet, en comptant depuis une date de départ soumise en tant que l'argument et se terminant avec la date actuelle (prise depuis l'horloge système). Si l'argument d'entrée est une date supérieure ou égale à une année dans le futur, la valeur de retour sera négative.

Exemples

Si la date actuelle est **2014-01-15** :

- **altova:age**(xs:date("2013-01-15")) retourne 1
- **altova:age**(xs:date("2013-01-16")) retourne 0
- **altova:age**(xs:date("2015-01-15")) retourne -1
- **altova:age**(xs:date("2015-01-14")) retourne 0

altova:age(StartDate *as xs:date*, **EndDate** *as xs:date*) **asxs:integer** XP3.1 XQ3.1

Retourne un entier représentant l'âge *en années* d'un objet, en comptant depuis une date de départ soumise en tant que l'argument jusqu'à une date de fin qui est de deuxième argument. La valeur de retour sera négative si le premier argument est tardif d'une année ou plus que le deuxième argument.

Exemples

Si la date actuelle est **2014-01-15**:

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) retourne 10
- **altova:age**(xs:date("2000-01-15"), current-date()) retourne 14 si la date actuelle est 2014-01-15
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) retourne -4

age-details [altova:]

altova:age-details(InputDate *as xs:date*) as (**xs:integer**)* XP3.1 XQ3.1

Retourne trois entiers qui sont respectivement les années, les mois et les jours entre la date soumise en tant que l'argument et la date actuelle (prise depuis l'horloge système). Le résultat de la somme de years+months+days donne le total de la différence de temps entre les deux dates (la date d'entrée et la date actuelle). La date d'entrée peut avoir une valeur précédant ou succédant à la date actuelle mais que la date d'entrée soit précédente ou succédant n'est pas indiqué par le signe des valeurs de retour ; les valeurs de retour sont toujours positives.

Exemples

Si la date actuelle est **2014-01-15**:

- **altova:age-details**(xs:date("2014-01-16")) retourne (0 0 1)
- **altova:age-details**(xs:date("2014-01-14")) retourne (0 0 1)
- **altova:age-details**(xs:date("2013-01-16")) retourne (1 0 1)
- **altova:age-details**(current-date()) retourne (0 0 0)

altova:age-details(Date-1 *as xs:date*, **Date-2** *as xs:date*) as (**xs:integer**)* XP3.1 XQ3.1

Retourne trois entiers qui sont respectivement les années, les mois et les jours entre les deux dates d'argument. Le résultat de la somme de years+months+days donne le total de la différence de temps entre les deux dates d'entrée ; peu importe que la date soit la précédente ou la subséquente des deux dates, elle est soumise en tant que le premier argument. Les valeurs de retour n'indiquent pas si la date d'entrée se produit avant ou après la date actuelle. Les valeurs de retour sont toujours positives.

Exemples

- **altova:age-details**(xs:date("2014-01-16"), xs:date("2014-01-15")) retourne (0 0 1)
- **altova:age-details**(xs:date("2014-01-15"), xs:date("2014-01-16")) retourne (0 0 1)

[Haut 414]

Fonctions Epoch time (heure Unix) XP3.1 XQ3.1

Epoch time est un système horaire utilisé dans les systèmes Unix. Il définit tout moment donné comme étant le nombre de secondes écoulées depuis 00:00:00 UTC le 1er janvier 1970. Ces fonctions Epoch time convertissent les valeurs **xs:dateTime** en valeurs Epoch time et vice versa.

dateTime-from-epoch [altova:]

altova:dateTime-from-epoch(Epoch *as xs:decimal* **as xs:dateTime** XP3.1 XQ3.1
Epoch time est un système horaire utilisé sur les systèmes Unix. Il définit tout moment donné comme étant l nombre de secondes écoulées depuis 00:00:00 UTC le 1er janvier 1970. La fonction **dateTime-from-epoch** retourne l'équivalent **xs:dateTime** d'un Epoch time, l'ajuste pour son fuseau horaire local et inclut l'information du fuseau horaire dans le résultat.

La fonction prend un argument `xs:decimal` et retourne une valeur `xs:dateTime` qui inclut une partie (fuseau horaire) **TZ**. Le résultat est obtenu en calculant l'équivalent UTC **dateTime** de Epoch time, et en l'ajoutant à son fuseau horaire local (pris de l'horloge système). Par exemple, si la fonction est exécutée sur un appareil qui a été défini pour être dans un fuseau horaire +01:00 (relatif à UTC), après avoir calculé l'équivalent UTC **dateTime**, une heure sera ajoutée au résultat. L'information du fuseau horaire, qui est une partie lexicale optionnelle du résultat **xs:dateTime**, est également rapportée dans le résultat **dateTime**. Comparez ce résultat avec celui de **dateTime-from-epoch-no-TZ**, et consultez également la fonction **epoch-from-dateTime**.

Exemples

Les exemples ci-dessous supposent un fuseau horaire local UTC +01:00. En conséquence, l'équivalent UTC **dateTime** de l'Epoch time soumis sera incrémenté d'une heure. Le fuseau horaire est rapporté dans le résultat.

- **altova:dateTime-from-epoch**(34) retourne `1970-01-01T01:00:34+01:00`
- **altova:dateTime-from-epoch**(62) retourne `1970-01-01T01:01:02+01:00`

dateTime-from-epoch-no-TZ [altova:]

altova:dateTime-from-epoch-no-TZ(Epoch *as xs:decimal* **as xs:dateTime** XP3.1 XQ3.1
Epoch time est un système horaire utilisé sur les systèmes Unix. Il définit tout moment donné comme étant l nombre de secondes écoulées depuis 00:00:00 UTC le 1er janvier 1970. La fonction **dateTime-from-epoch-no-TZ** retourne l'équivalent **xs:dateTime** d'un Epoch time, l'ajuste pour son fuseau horaire local, mais n'inclut pas l'information du fuseau horaire dans le résultat.

La fonction prend un `xs:decimal` argument et retourne une valeur `xs:dateTime` qui n'inclut pas de partie (fuseau horaire) **TZ**. Le résultat est obtenu en calculant l'équivalent UTC **dateTime** de Epoch time, et en l'ajoutant à son fuseau horaire local (pris de l'horloge système). Par exemple, si la fonction est exécutée sur un appareil qui a été défini pour être dans un fuseau horaire +01:00 (relatif à UTC), après avoir calculé l'équivalent, une heure sera ajoutée au résultat. L'information du fuseau horaire, qui est une partie lexicale optionnelle du résultat **xs:dateTime**, n'est pas rapportée dans le résultat **dateTime**. Comparez ce résultat avec celui de **dateTime-from-epoch**, et consultez également la fonction **epoch-from-dateTime**.

Exemples

Les exemples ci-dessous supposent un fuseau horaire local UTC +01:00. En conséquence,

l'équivalent UTC **dateTime** de l'Epoch time soumis sera incrémenté d'une heure. Le fuseau horaire n'est pas rapporté dans le résultat.

- **altova:dateTime-from-epoch**(34) returns 1970-01-01T01:00:34
- **altova:dateTime-from-epoch**(62) returns 1970-01-01T01:01:02

▼ epoch-from-dateTime [altova:]

altova:epoch-from-dateTime(dateTimeValue *as xs:dateTime*) **as xs:decimal** XP3.1 XQ3.1
Epoch time est un système horaire utilisé sur les systèmes Unix. Il définit tout moment donné comme étant l nombre de secondes écoulées depuis 00:00:00 UTC le 1er janvier 1970. La fonction **epoch-from-dateTime** retourne l'équivalent **xs:dateTime** d'un Epoch time, l'ajuste pour son fuseau horaire local et inclut l'information du fuseau horaire dans le résultat. Veuillez noter que vous devrez explicitement construire la valeur **xs:dateTime**. La valeur soumise **xs:dateTime** peut ou ne peut pas contenir la partie optionnelle **TZ** (fuseau horaire).

Que la partie du fuseau horaire soit soumise en tant que partie de l'argument ou non, le décalage du fuseau horaire local (pris de l'horloge système) est retiré de l'argument soumis **dateTimeValue**. Ceci porduit l'heure UTC équivalente de laquelle l'Epoch time équivalent est calculé. Par exemple, si la fonction est exécutée sur un appareil qui a été défini pour être dans un fuseau horaire +01:00 (relatif à UTC), une heure sera retirée de la valeur soumise **dateTimeValue** avant de calculer la valeur Epoch. Consultez également la fonction **dateTime-from-epoch**.

▣ *Exemples*

Les exemples ci-dessous supposent un fuseau horaire local UTC +01:00. En conséquence, une heure sera soustraite à **dateTime** avant de calculer l'Epoch time.

- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34+01:00")) returns 34
- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34")) returns 34
- **altova:epoch-from-dateTime**(xs:dateTime("2021-04-01T11:22:33")) returns 1617272553

[Haut 414]

11.2.1.3 Fonctions XPath/XQuery : Géolocalisation

Les fonctions d'extension de géolocalisation XPath/XQuery suivantes sont prises en charge dans la version actuelle de RaptorXML Server et peuvent être utilisées dans (i) des expressions XPath dans un contexte XSLT, ou (ii) des expressions XQuery dans un document XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova,** **http://www.altova.com/xslt-extensions**, et sont indiquées dans cette section par le préfixe **altova:**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce

qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

format-geolocation [altova:]

altova:format-geolocation(Latitude *as xs:decimal*, **Longitude** *as xs:decimal*, **GeolocationOutputStringFormat** *as xs:integer*) **asxs:string**

XP3.1 XQ3.1

Prend la latitude et la longitude en tant que les deux premiers arguments, et sort la géolocalisation en tant que chaîne. Le troisième argument, **GeolocationOutputStringFormat**, est le format de la chaîne de sortie de géolocalisation ; il utilise des valeurs d'entier allant de `1` à `4` pour identifier le format de chaîne de sortie (*voir 'Formats de chaîne de sortie de géolocalisation' ci-dessous*). Les valeurs de latitude vont de `+90` à `-90` (`N` à `S`). Les valeurs de longitude vont de `+180` à `-180` (`E` à `O`).

Note : La fonction image-exif-data 441 et les attributs de métadonnées Exif peuvent être utilisés pour fournie les chaînes d'entrée.

Exemples

- **altova:format-geolocation**(33.33, -22.22, 4) retourne xs:string "33.33 -22.22"
- **altova:format-geolocation**(33.33, -22.22, 2) retourne xs:string "33.33N 22.22W"
- **altova:format-geolocation**(-33.33, 22.22, 2) retourne xs:string "33.33S 22.22E"
- **altova:format-geolocation**(33.33, -22.22, 1) retourne xs:string "33°19'48.00"S 22° 13'12.00"E"

Formats de chaîne de sortie de géolocalisation:

La latitude et longitude fournies sont formatées dans un des formats de sortie indiqués ci-dessous. Le format désiré est défini par son ID d'entier (`1 à 4`). Les valeurs de latitude vont de `+90` à `-90` (`N` à `S`). Les valeurs de longitude vont de `+180` à `-180` (`E` à `O`).

1
Degrés, minutes, secondes décimales, avec orientation suffixée (`N/S, E/O`) **D°M'S.SS"N/S D°M'S.SS"E/W** *Exemple :* **33°55'11.11"N 22°44'66.66"W**

2
Degrés décimaux, avec orientation suffixée (`N/S, E/O`) **D.DDN/S D.DDE/W** *Exemple :* **33.33N 22.22W**

3
Degrés, minutes, secondes décimales, avec signe préfixé (+/-); le signe plus (N/E) est optionnel +/-D°M'S.SS" +/-D°M'S.SS" *Exemple :* 33°55'11.11" -22°44'66.66"

4
Degrés décimaux, avec signe préfixé (+/-); le signe plus (N/E) est optionnel +/-D.DD +/-D.DD *Exemple :* 33.33 -22.22

- *Attribut Altova Exif : Géolocalisation*

  La machine Altova XPath/XQuery génère l'attribut **Geolocation** personnalisable depuis les onglets standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

parse-geolocation [altova:]

altova:parse-geolocation(GeolocationInputString *as xs:string*) **asxs:decimal+** XP3.1 XQ3.1

Parse l'argument GeolocationInputString fourni et retourne la latitude et longitude de géolocalisation (dans cet ordre) en séquence deux items décimaux xs:decimal. Les formats dans lesquels la chaîne d'entrée de géolocalisation peut être fournie sont recensés ci-dessous.

Note : La fonction image-exif-data 441 et l'attribut @Geolocation 441 de métadonnées Exif peuvent être utilisés pour fournir la chaîne d'entrée de géolocalisation (*voir exemple ci-dessous*).

- *Exemples*

  - **altova:parse-geolocation**("33.33 -22.22") retourne la séquence de deux xs:decimals (33.33, 22.22)
  - **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") retourne la séquence de deux xs:decimals (48.8582222222222, 24.2945)
  - **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') retourne la séquence de deux xs:decimals (48.8582222222222, 24.2945)
  - **altova:parse-geolocation**(**image-exif-data**(//MyImages/Image20141130.01)**/@Geolocation**)retourne une séquence de deux xs:decimals

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de `+90` à `-90` (`N` à `S`). Les valeurs de longitude varient de `+180` à `-180` (`E` à `W`).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (`"`) alors que les indicateurs d'unité échappés sont marqués en bleu (`""`).

- Degrés, minutes, secondes décimales, avec orientation suffixée (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  Exemple : **`33°55'11.11"N  22°44'55.25"W`**

- Degrés, minutes, secondes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/E`**) est optionnel
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  Exemple : **`33°55'11.11"  -22°44'55.25"`**

- Degrés, minutes décimales, avec orientation suffixée (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  Exemple : **`33°55.55'N  22°44.44'W`**

- Degrés, minutes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/E`**) est optionnel
  `+/-D°M.MM'  +/-D°M.MM'`
  Exemple : **`+33°55.55'  -22°44.44'`**

- Degrés décimaux, avec orientation suffixée (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  Exemple : **`33.33N  22.22W`**

- Degrés décimaux, avec signe préfixé (`+/-`) ; le signe plus pour (**`N/S E/W`**) est optionnel
  `+/-D.DD  +/-D.DD`
  Exemple : **`33.33  -22.22`**

Exemples de combinaisons de format :

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **`Geolocation`** personnalisable depuis les onglets standard de métadonnées Exif. **`Geolocation`** est une concaténation de quatre onglets Exif : `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-km [altova:]

altova:geolocation-distance-km(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) **asxs:decimal** **XP3.1** **XQ3.1**

Calcule la distance entre deux géolocalisations en kilomètres. Les formats dans lesquels une chaîne d'entrée de géolocalisation peut être fournie sont recensés ci-dessous. Les valeurs de latitude vont de +90 à -90 (N à S). Les valeurs de longitude vont de +180 à -180 (E à O).

Note : La fonction image-exif-data 441 et l'attribut @Geolocation 441 des métadonnées d'Exif peuvent être utilisés pour fournir les chaînes d'entrée de géolocalisation.

Exemples

- **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") retourne xs:decimal 4183.08132372392

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de +90 à -90 (N à S). Les valeurs de longitude varient de +180 à -180 (E à W).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (") alors que les indicateurs d'unité échappés sont marqués en bleu ("").

- Degrés, minutes, secondes décimales, avec orientation suffixée (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Exemple : **33°55'11.11"N 22°44'55.25"W**

- Degrés, minutes, secondes décimales, avec signe préfixé (+/-) ; le signe plus pour (**N/E**) est optionnel
  +/-D°M'S.SS" +/-D°M'S.SS"
  Exemple : **33°55'11.11" -22°44'55.25"**

- Degrés, minutes décimales, avec orientation suffixée (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  Exemple : **33°55.55'N 22°44.44'W**

- Degrés, minutes décimales, avec signe préfixé (+/-) ; le signe plus pour (N/E) est optionnel
  +/-D°M.MM' +/-D°M.MM'
  Exemple : **+33°55.55' -22°44.44'**

- Degrés décimaux, avec orientation suffixée (N/S, E/W)
  D.DDN/S D.DDW/E
  Exemple : **33.33N 22.22W**

- Degrés décimaux, avec signe préfixé (+/-) ; le signe plus pour (N/S E/W) est optionnel
  +/-D.DD +/-D.DD
  Exemple : **33.33 -22.22**

Exemples de combinaisons de format :
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **Geolocation** personnalisable depuis les onglets standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-mi [altova:]

altova:geolocation-distance-mi(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) **asxs:decimal** **XP3.1** **XQ3.1**
Calcule la distance entre deux géolocalisations en miles. Les formats dans lesquels une chaîne d'entrée de géolocalisation peut être fournie sont recensés ci-dessous. Les valeurs de latitude vont de +90 à -90 (N à S). Les valeurs de longitude vont de +180 à -180 (E à O).

Note : La fonction image-exif-data[441] et l'attribut @Geolocation[441] des métadonnées d'Exif peuvent être utilisés pour fournir les chaînes d'entrée de géolocalisation.

Exemples

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""")
  retourne xs:decimal 2599.40652340653

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont

permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de +90 à -90 (N à S). Les valeurs de longitude varient de +180 à -180 (E à W).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (") alors que les indicateurs d'unité échappés sont marqués en bleu ("").

- Degrés, minutes, secondes décimales, avec orientation suffixée (N/S, E/W)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  Exemple : `33°55'11.11"N  22°44'55.25"W`

- Degrés, minutes, secondes décimales, avec signe préfixé (+/-) ; le signe plus pour (**N/E**) est optionnel
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  Exemple : `33°55'11.11"  -22°44'55.25"`

- Degrés, minutes décimales, avec orientation suffixée (N/S, E/W)
  `D°M.MM'N/S  D°M.MM'W/E`
  Exemple : `33°55.55'N  22°44.44'W`

- Degrés, minutes décimales, avec signe préfixé (+/-) ; le signe plus pour (**N/E**) est optionnel
  `+/-D°M.MM'  +/-D°M.MM'`
  Exemple : `+33°55.55'  -22°44.44'`

- Degrés décimaux, avec orientation suffixée (N/S, E/W)
  `D.DDN/S  D.DDW/E`
  Exemple : `33.33N  22.22W`

- Degrés décimaux, avec signe préfixé (+/-) ; le signe plus pour (**N/S E/W**) est optionnel
  `+/-D.DD  +/-D.DD`
  Exemple : `33.33  -22.22`

Exemples de combinaisons de format :

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **Geolocation** personnalisable depuis les onglets standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-within-polygon [altova:]

altova:geolocation-within-polygon(Geolocation *as xs:string*, **((PolygonPoint** *as xs:string*)+)) **as xs:boolean** **XP3.1** **XQ3.1**

Détermine si **Geolocation** (le premier argument) se trouve dans l'espace polygonal décrit par les arguments **PolygonPoint**. Si les arguments PolygonPoint ne forment pas une figure fermée (formée lorsque le premier point et le dernier point sont identiques), alors le premier point est implicitement ajouté en tant que le dernier point afin de pouvoir clore la figure. Tous les arguments (Geolocation et PolygonPoint+) sont donnés par chaînes d'entrées de géolocalisation (*formats recensés ci-dessous*). Si l'argument Geolocation se trouve dans l'espace polygonal, la fonction retourne true(); sinon, elle retourne false(). Les valeurs de latitude vont de +90 à -90 (N à S). Les valeurs de longitude vont de +180 à -180 (E à O).

Note : La fonction image-exif-data 441 et l'attribut @Geolocation 441 de métadonnées d'Exif peut être utilisée pour fournir les chaînes d'entrée de géolocalisation.

Exemples

- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24", "58 -32"))** retourne true()
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24"))** retourne true()
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2""))** retourne true()

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de +90 à -90 (N à S). Les valeurs de longitude varient de +180 à -180 (E à W).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (") alors que les indicateurs d'unité échappés sont marqués en bleu ("").

- Degrés, minutes, secondes décimales, avec orientation suffixée (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Exemple : **33°55'11.11"N 22°44'55.25"W**

- Degrés, minutes, secondes décimales, avec signe préfixé (+/-) ; le signe plus pour (**N/E**) est optionnel
  +/-D°M'S.SS" +/-D°M'S.SS"
  Exemple : **33°55'11.11" -22°44'55.25"**

- Degrés, minutes décimales, avec orientation suffixée (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  Exemple : **33°55.55'N 22°44.44'W**

- Degrés, minutes décimales, avec signe préfixé (+/-) ; le signe plus pour (**N/E**) est optionnel
  +/-D°M.MM' +/-D°M.MM'
  Exemple : **+33°55.55' -22°44.44'**

- Degrés décimaux, avec orientation suffixée (N/S, E/W)
  D.DDN/S D.DDW/E
  Exemple : **33.33N 22.22W**

- Degrés décimaux, avec signe préfixé (+/-) ; le signe plus pour (**N/S E/W**) est optionnel
  +/-D.DD +/-D.DD
  Exemple : **33.33 -22.22**

Exemples de combinaisons de format :
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **Geolocation** personnalisable depuis les onglets standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-within-rectangle [altova:]

altova:geolocation-within-rectangle(Geolocation *as xs:string*, **RectCorner-1** *as xs:string*, **RectCorner-2** *as xs:string*) **asxs:boolean** **XP3.1** **XQ3.1**
Détermine si **Geolocation** (le premier argument) se trouve dans le rectangle défini par le second et le troisième argument, **RectCorner-1** et **RectCorner-2**, qui spécifient les coins opposés du rectangle. Tous les arguments (Geolocation, **RectCorner-1** et **RectCorner-2**) sont indiqués par des chaînes d'entrées de géolocalisation (*formats recensés ci-dessous*). Si l'argument Geolocation se trouve dans le rectangle, la fonction retourne true(); sinon, elle retourne false(). Les valeurs de latitude vont de +90 à -90 (N à S). Les valeurs de longitude vont de +180 à -180 (E à O).

Note : La fonction image-exif-data 441 et l'attribut @Geolocation 441 de métadonnées Exif peuvent être utilisés pour fournir les chaînes d'entrée de géolocalisation.

Exemples

- **altova:geolocation-within-rectangle("33 -22", "58 -32", "-48 24")** retourne `true()`
- **altova:geolocation-within-rectangle("33 -22", "58 -32", "48 24")** retourne `false()`
- **altova:geolocation-within-rectangle("33 -22", "58 -32", "48°51'29.6""S 24° 17'40.2""")** retourne `true()`

Formats de string d'entrée de géolocalisation :

Le string d'entrée de géolocalisation doit contenir la latitude et la longitude (dans cet ordre) séparées par un espace. Les strings peuvent tous présenter les formats suivants. Les combinaisons sont permises. La latitude peut donc être dans un format et la longitude dans un autre. Les valeurs de latitude varient de `+90` à `-90` (`N` à `S`). Les valeurs de longitude varient de `+180` à `-180` (`E` à `W`).

Note : L'utilisation de guillemets simples ou doubles pour la délimitation des arguments de string entraînera une non-concordance avec l'utilisation de guillemets simples ou doubles pour indiquer, respectivement les valeurs de minutes et de secondes. Dans ces cas, les guillemets utilisés pour indiquer les minutes et les secondes doivent être échappés en les doublant. Dans les exemples présentés dans cette section, les guillemets utilisés pour délimiter les strings d'entrée sont marqués en jaune (") alors que les indicateurs d'unité échappés sont marqués en bleu ("").

- Degrés, minutes, secondes décimales, avec orientation suffixée (`N/S`, `E/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Exemple : **33°55'11.11"N 22°44'55.25"W**

- Degrés, minutes, secondes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**N/E**) est optionnel
  +/-D°M'S.SS" +/-D°M'S.SS"
  Exemple : **33°55'11.11" -22°44'55.25"**

- Degrés, minutes décimales, avec orientation suffixée (`N/S`, `E/W`)
  D°M.MM'N/S D°M.MM'W/E
  Exemple : **33°55.55'N 22°44.44'W**

- Degrés, minutes décimales, avec signe préfixé (`+/-`) ; le signe plus pour (**N/E**) est optionnel
  +/-D°M.MM' +/-D°M.MM'
  Exemple : **+33°55.55' -22°44.44'**

- Degrés décimaux, avec orientation suffixée (`N/S`, `E/W`)
  D.DDN/S D.DDW/E
  Exemple : **33.33N 22.22W**

- Degrés décimaux, avec signe préfixé (`+/-`) ; le signe plus pour (**N/S E/W**) est optionnel
  +/-D.DD +/-D.DD
  Exemple : **33.33 -22.22**

Exemples de combinaisons de format :

33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Attribut Altova Exif : Géolocalisation

La machine Altova XPath/XQuery génère l'attribut **Geolocation** personnalisable depuis les onglets standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

[Top 431]

11.2.1.4 Fonctions XPath/XQuery : Relatives aux images

Les fonctions d'extension XPath/XQuery relatives à l'image suivantes sont prises en charge dans la version actuelle de RaptorXML Server et peuvent être utilisées dans (i) des expressions XPath dans un contexte XSLT, ou dans (ii) des expressions XQuery dans un document XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova,** **http://www.altova.com/xslt-extensions**, et sont indiquées dans cette section par le préfixe **altova:**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(Base64String *as string***) asstring?** XP3.1 XQ3.1

Prend le code Base64 d'un fichier d'image en tant que son argument et retourne l'extension de fichier de l'image comme enregistré dans le codage Base64 de l'image. La valeur retournée est une suggestion basée sur l'information du type d'image disponible dans le codage. Si cette information n'est pas

disponible, une chaîne vide est retournée. Cette fonction est utile si vous souhaitez enregistrer une image Base64 en tant que fichier et que vous souhaitez extraire dynamiquement une extension de fichier appropriée.

Exemples

- `altova:suggested-image-file-extension(/MyImages/MobilePhone/Image20141130.01)` retourne `'jpg'`
- `altova:suggested-image-file-extension($XML1/Staff/Person/@photo)` retourne `''`

Dans les exemples ci-dessus, les nœuds fournis en tant qu'arguments de la fonction sont assumés contenir une image codée Base64. Le premier exemple extrait `jpg` en tant que type et extension de fichier. Dans le second exemple, le codage Base64 soumis ne fournit pas une information de fichier d'extension utile.

image-exif-data [altova:]

`altova:image-exif-data(Base64BinaryString` *`as string`*`)` **`aselement?`** **XP3.1** **XQ3.1**

Prend une image codée Base64 en tant que son argument et retourne un élément appelé **`Exif`** qui contient les métadonnées Exif de l'image. Celles-ci sont créées en tant que paires attribute-value pairs de l'élément `Exif`. Les noms d'attribut sont les onglets de données Exif trouvés dans le codage Base64. La liste des onglets des spécifications Exif est indiquée ci-dessous. Si un onglet spécifique à un distributeur est présent dans les données Exif, cet onglet et sa valeur seront aussi retournés en tant que paire attribute-value. Outre les onglets de métadonnées Exif standard (*voir la liste ci-dessous*), des paires attribute-value spécifiques à Altova sont également générées. Ces attributs Exif Altova sont recensés ci-dessous.

Exemples

- Pour accéder à n'importe quel attribut, utiliser la fonction comme suit :
  `image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude`
  `image-exif-data(//MyImages/Image20141130.01)/@Geolocation`
- Pour accéder à tous les attributs, utiliser la fonction comme suit:
  `image-exif-data(//MyImages/Image20141130.01)/@*`
- Pour accéder au nom de tous les attributs, utiliser l'expression suivante :
  `for $i in image-exif-data(//MyImages/Image20141130.01)/@* return name($i)`
  Cela est utile pour trouver les noms des attributs retournés par la fonction.

Attribut Altova Exif : Géolocalisation

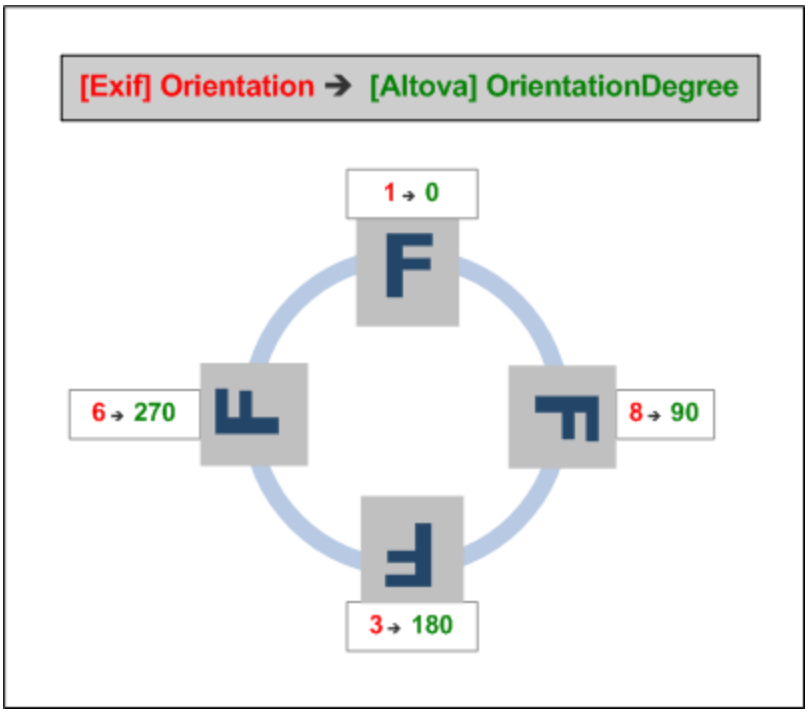
La machine Altova XPath/XQuery génère l'attribut **`Geolocation`** personnalisable depuis les onglets standard de métadonnées Exif. **`Geolocation`** est une concaténation de quatre onglets Exif : `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`, avec des unités ajoutées (voir table ci-dessous).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

Altova Exif Attribute: OrientationDegree

La machine Altova XPath/XQuery génère l'attribut personnalisé **`OrientationDegree`** à partir de l'onglet de métadonnées Exif **`Orientation`**.

`OrientationDegree` traduit l'onglet standard Exif `Orientation` à partir d'une valeur d'entier (`1`, `8`, `3`, ou `6`) aux valeurs de degrés respectives de chacun (`0`, `90`, `180`, `270`), tel que montré dans la figure ci-dessous. Veuillez noter qu'il n'y a pas de traductions de la valeur `Orientation` de `2`, `4`, `5`, `7`. (Ces orientations sont obtenus en basculant l'image `1` à travers son centre axial vertical pour obtenir l'image avec une valeur de `2`, puis en pivotant cette image par sauts de 90° dans le sens des aiguilles d'une montre pour obtenir les valeurs de `7`, `4`, et `5`, respectivement).



Listing of standard Exif meta tags

- `ImageWidth`
- `ImageLength`
- `BitsPerSample`
- `Compression`
- `PhotometricInterpretation`
- `Orientation`
- `SamplesPerPixel`
- `PlanarConfiguration`
- `YCbCrSubSampling`
- `YCbCrPositioning`
- `XResolution`
- `YResolution`
- `ResolutionUnit`

- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription
- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit

- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio
- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

[Top 441]

11.2.1.5 Fonctions XPath/XQuery : Numérique

Les fonctions d'extension numériques d'Altova peuvent être utilisées dans des expressions XPath et XQuery et proposent des fonctions supplémentaires pour le traitement des données. Les fonctions dans cette section peuvent être utilisée avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova. Ils sont disponibles dans des contextes XPath/XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova,** `http://www.altova.com/xslt-extensions`, et sont indiquées dans cette section par le préfixe `altova:`, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

Fonctions de numérotation automatique

▼ generate-auto-number [altova:]

altova:generate-auto-number(ID *as xs:string*, **StartsWith** *as xs:double*, **Increment** *as xs:double*, **ResetOnChange** *as xs:string*) as `xs:integer` XP1 XP2 XQ1 XP3.1 XQ3.1

Génère un numéro à chaque fois que la fonction est appelée. Le premier numéro, qui est généré la première fois que la fonction est appelée, est spécifié par l'argument `StartsWith`. Chaque appel subséquent vers la fonction génère un nouveau numéro, ce numéro est augmenté au-dessus du numéro précédemment généré par la valeur spécifiée dans l'argument `Increment`. En effet, la fonction `altova:generate-auto-number` crée un compteur comportant un nom spécifié par l'argument `ID`, et dont le compteur est augmenté à chaque fois que la fonction est appelée. Si la valeur de l'argument `ResetOnChange` change de celle de l'appel de fonction précédent, la valeur du numéro à générer est réinitialisée à la valeur `StartsWith`. La numérotation automatique peut être réinitialisée en utilisant la fonction `altova:reset-auto-number`.

Exemples

- `altova:generate-auto-number("ChapterNumber", 1, 1, "SomeString")` retournera un nombre à chaque fois que la fonction est appelée, en commençant avec `1`, et en augmentant de `1` avec chaque appel de la fonction. Tant que le quatrième argument demeure `"SomeString"` dans chaque appel subséquent, l'augmentation se poursuivra. Lorsque la valeur du quatrième argument change, le compteur (appelé `ChapterNumber`) sera réinitialisé à 1. La valeur de `ChapterNumber` peut aussi être réinitialisée par un appel de la fonction `altova:reset-auto-number` comme ceci :

```
altova:reset-auto-number("ChapterNumber").
```

reset-auto-number [altova:]

altova:reset-auto-number(ID *as xs:string***)** XP1 XP2 XQ1 XP3.1 XQ3.1

Cette fonction réinitialise le numéro du compteur de numérotation automatique nommé dans l'argument `ID`. Le numéro est réinitialisé au numéro spécifié par l'argument `StartsWith` de la fonction `altova:generate-auto-number` qui a créé le compteur nommé dans l'argument `ID`.

Exemples

- **altova:reset-auto-number**("ChapterNumber") réinitialise le numéro du compteur de numérotation automatique nommé `ChapterNumber` qui a été créé par la fonction `altova:generate-auto-number`. Le numéro est réinitialisé à la valeur de l'argument `StartsWith` de la fonction `altova:generate-auto-number` qui a créé `ChapterNumber`.

[Top 446]

Fonctions numériques

hex-string-to-integer [altova:]

altova:hex-string-to-integer(HexString *as xs:string***) asxs:integer** XP3.1 XQ3.1

Prend un argument de chaîne qui est l'équivalent Base-16 d'un entier dans le système décimal (Base-10), et retourne l'entier décimal.

Exemples

- **altova:hex-string-to-integer**('1') retourne 1
- **altova:hex-string-to-integer**('9') retourne 9
- **altova:hex-string-to-integer**('A') retourne 10
- **altova:hex-string-to-integer**('B') retourne 11
- **altova:hex-string-to-integer**('F') retourne 15
- **altova:hex-string-to-integer**('G') retourne une erreur
- **altova:hex-string-to-integer**('10') retourne 16
- **altova:hex-string-to-integer**('01') retourne 1
- **altova:hex-string-to-integer**('20') retourne 32
- **altova:hex-string-to-integer**('21') retourne 33
- **altova:hex-string-to-integer**('5A') retourne 90
- **altova:hex-string-to-integer**('USA') retourne une erreur

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Integer *as xs:integer***) asxs:string** XP3.1 XQ3.1

Prend un argument d'entier et retourne son équivalent de Base-16 en tant que chaîne.

Exemples

- **altova:integer-to-hex-string**(1) retourne '1'
- **altova:integer-to-hex-string**(9) retourne '9'
- **altova:integer-to-hex-string**(10) retourne 'A'
- **altova:integer-to-hex-string**(11) retourne 'B'

- `altova:integer-to-hex-string(15)` retourne `'F'`
- `altova:integer-to-hex-string(16)` retourne `'10'`
- `altova:integer-to-hex-string(32)` retourne `'20'`
- `altova:integer-to-hex-string(33)` retourne `'21'`
- `altova:integer-to-hex-string(90)` retourne `'5A'`

[Top 446]

Fonctions de formatage de numéro

[Top 446]

11.2.1.6 Fonctions XPath/XQuery : Schéma

Les fonctions d'extension Altova recensées ci-dessous retournent l'information de schéma. Ci-dessous, vous trouverez les descriptions des fonctions, ainsi que des (i) exemples et (ii) une liste des composants de schéma et de leurs propriétés respectives. Elles peuvent être utilisées avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova et sont disponibles dans des contextes XPath/XQuery.

Information de schéma depuis les documents de schéma
La fonction **altova:schema** détient deux arguments : un avec zéro arguments et l'autre avec deux arguments. La fonction à zéro argument retourne l'ensemble du schéma. Ensuite, à partir de là, vous pouvez naviguer dans le schéma pour localiser les composants de schéma que vous souhaitez. La fonction à deux arguments retourne un type de composant spécifique qui est identifié par son QName. Dans les deux cas, la valeur de retour est un fonction. Pour naviguer dans le composant retourné, vous devez sélectionner une propriété de ce composant spécifique. Si la propriété est un item non atomique (c'est à dire, s'il s'agit d'un composant), vous pouvez aller plus loin en choisissant une propriété de ce composant. Si la propriété sélectionnée est un item atomique, la valeur de l'item est retournée et vous ne pouvez pas aller plus loin.

Note : Dans des expressions XPath, le schéma doit avoir été importé dans l'environnement de traitement, par exemple, dans XSLT avec l'instruction **xslt:import-schema**. Dans des expressions XQuery, le schéma doit être importé explicitement utilisant un schema import.

Information de schéma depuis les nœuds XML
La fonction **altova:type** soumet le nœud à un document XML et retourne l'information de type du nœud depuis le PSVI.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, http://www.altova.com/xslt-extensions**, et sont indiquées dans cette section par le préfixe **altova:**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des

publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

▼ Schéma (zéro arguments)

altova:schema() as (function(xs:string) as item()*)? XP3.1 XQ3.1
Retourne le composant de **schema** en entier. Vous pouvez donc aller plus loin dans le composant de **schema** en sélectionnant une des propriétés du composant de **schema**.

- Si cette propriété est un composant, vous pouvez aller encore plus en loin en sélectionnant une des propriétés de ce composant. Vous pouvez renouveler cette étape en allant plus loin dans le schéma.
- Si le composant est une valeur atomique, celle-ci sera retournée et vous ne pourrez pas aller plus loin.

Les propriétés du composant de **schema** sont :

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

Les propriétés de tous les types de composant (à part **schema**) sont regroupées ci-dessous.

Note: Dans des expressions XQuery, le schéma doit être importé explicitement. Dans des expressions XPath, le schéma doit avoir été importé dans l'environnement de traitement, par exemple dans XSLT avec l'instruction **xslt:import**.

⊟ *Exemples*

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema() ("type definitions") return $typedef ("name")` retourne les noms de tous les Types simples ou Types complexes dans le schéma

- `import schema "" at "C:\Test\ExpReport.xsd"; altova:schema() ("type definitions")[1]("name")` retourne le nom du premier de tous les Types simples ou Types complexes dans le schéma

Composants et leurs propriétés

- Assertion

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Assertion"
test	XPath Property Record	

- Déclaration d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Declaration"
name	string	Local name of the attribute
target namespace	string	Namespace URI of the attribute
type definition	Type simple or Type complexe	
scope	A function with properties ("class":"Scope", "variety": "global" or "local", "parent": the containing Type complexe or Attribute Group)	
value constraint	If present, a function with properties ("class": "Value Constraint", "variety": "fixed" or "default", "value": atomic value, "lexical form": string. Note that the "value" property is not available for namespace-sensitive types	
inheritable	boolean	

- Déclaration de groupe d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Group Definition"
name	string	Local name of the attribute group
target namespace	string	Namespace URI of the attribute group
attribute uses	Sequence of (Attribute Use)	
attribute wildcard	Optional Attribute Wildcard	

- Utilisation d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Use"

required	boolean	true if the attribute is required, false if optional
value constraint	See Attribute Declaration	
inheritable	boolean	

- Caractère générique

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

- Type complexe

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
base type definition	Complex Type Definition	
final	Sequence of strings ("restriction"|"extension")	
context	Empty sequence (not implemented)	
derivation method	string ("restriction"|"extension")	
abstract	boolean	
attribute uses	Sequence of Attribute Use	
attribute wildcard	Optional Attribute Wildcard	
content type	function with properties: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Type	

	simple)	
prohibited substitutions	Sequence of strings ("restriction"|"extension")	
assertions	Sequence of Assertion	

Déclaration d'élément

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
type definition	Type simple or Type complexe	
type table	function with properties ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Type simple or Type complexe)	
scope	function with properties ("class":"Scope", "variety": ("global"|"local"), "parent": optional Type complexe)	
value constraint	see Attribute Declaration	
nillable	boolean	
identity-constraint definitions	Sequence of Identity Constraint	
substitution group affiliations	Sequence of Element Declaration	
substitution group exclusions	Sequence of strings ("restriction"|"extension")	
disallowed substitutions	Sequence of strings ("restriction"|"extension"|"substitution")	
abstract	boolean	

Caractère générique d'élément

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined"	

	and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

- Facette

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	The name of the facet, for example "minLength" or "enumeration"
value	depends on facet	The value of the facet
fixed	boolean	
typed-value	For the enumeration facet only, array(xs:anyAtomicType*)	An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the "value" property is a sequence of strings, regardless of the actual type)

- Contrainte d'identité

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Identity-Constraint Definition"
name	string	Local name of the constraint
target namespace	string	Namespace URI of the constraint
identity-constraint category	string ("key"|"unique"|"keyRef")	
selector	XPath Property Record	
fields	Sequence of XPath Property Record	
referenced key	(For keyRef only): Identity Constraint	The corresponding key constraint

- Groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Groupe de modèle"
compositor	string ("sequence"|"choice"|"all")	
particles	Séquence de particule	

- Définition de groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de groupe de modèle"

name	string	Nom local du groupe de modèle
target namespace	string	URI d'espace du groupe de modèle
model group	Groupe de modèle	

Notation

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Déclaration de notation"
name	string	Nom local de la notation
target namespace	string	URI d'espace de nom de la notation
system identifier	anyURI	
public identifier	string	

Particule

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Particule"
min occurs	entier	
max occurs	entier ou string("unbounded")	
term	Déclaration d'élément, Caractère générique d'élément ou ModelGroup	

Type simple

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de type simple"
name	string	Nom local du type (vide si anonyme)
target namespace	string	URI d'espace de noms du type (vide si anonyme)
final	Séquence de string("restriction"|"extension"|"list"|"union")	
context	composant contenant	
base type definition	Type simple	
facets	Séquence de Facette	
fundamental facets	Séquence vide (pas implémentée)	
variety	string ("atomic"|"list"|"union")	
primitive type definition	Type simple	

item type definition	uniquement pour les types de liste) Type simple	
member type definitions	(uniquement pour les types d'union) Séquence de Type simple	

Alternative de type

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type Alternative"
test	XPath Property Record	
type definition	Type simple ou Type complexe	

XPath Property Record

Nom de propriété	Type de propriété	Valeur de propriété
namespace bindings	Séquence des fonctions avec les propriétés ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	L'URI de base statique de l'expression XPath
expression	string	L'expression XPath en tant que string

Schéma (deux arguments)

altova:schema(ComponentKind *as xs:string*, **Name** *as xs:QName*) **as (function(xs:string) as item()*)? XP3.1 XQ3.1**

Retourne le type du composant qui est spécifié dans le premier argument qui a un nom identique à celui fourni dans le second argument. Vous pouvez donc aller plus loin en sélectionnant une des propriétés du composant.

- Si cette propriété est un composant, vous pouvez aller encore plus en profondeur en sélectionnant une des propriétés de ce composant. Vous pouvez renouveler cette étape en allant plus loin dans le schéma.
- Si le composant est une valeur atomique, celle-ci sera retournée et vous ne pourrez pas aller plus loin.

Note: Dans des expressions XQuery, le schéma doit être importé explicitement. Dans des expressions XPath, le schéma doit avoir été importé dans l'environnement de traitement, par exemple dans XSLT avec l'instruction **xslt:import**.

Exemples

```
import schema "" at "C:\Test\ExpReport.xsd";
```

```
altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
("content type")("particles")[3]!.("term")("kind")
```
retourne la propriété `kind` du terme du troisième composant de `particles`. Ce composant de particules est un descendant de la déclaration d'élément ayant un `QName` de `OrgChart`

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  let $typedef := altova:schema("type definition", xs:QName("emailType"))
  for $facet in $typedef ("facets")
  return [$facet ("kind"), $facet("value")]
  ```
  retourne, pour chaque **`facet`** de chaque composant **`emailType`**, un array contenant le type et la valeur de cette facette

Composants et leurs propriétés

Assertion

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Assertion"
test	XPath Property Record	

Déclaration d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Declaration"
name	string	Local name of the attribute
target namespace	string	Namespace URI of the attribute
type definition	Type simple or Type complexe	
scope	A function with properties ("class":"Scope", "variety": "global" or "local", "parent": the containing Type complexe or Attribute Group)	
value constraint	If present, a function with properties ("class": "Value Constraint", "variety": "fixed" or "default", "value": atomic value, "lexical form": string. Note that the "value" property is not available for namespace-sensitive types	
inheritable	boolean	

Déclaration de groupe d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Group Definition"
name	string	Local name of the attribute group
target namespace	string	Namespace URI of the attribute
  ```

		group
attribute uses	Sequence of (Attribute Use)	
attribute wildcard	Optional Attribute Wildcard	

- Utilisation d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Use"
required	boolean	true if the attribute is required, false if optional
value constraint	See Attribute Declaration	
inheritable	boolean	

- Caractère générique

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

- Type complexe

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
base type definition	Complex Type Definition	
final	Sequence of strings ("restriction"|"extension")	
context	Empty sequence (not implemented)	
derivation method	string ("restriction"|"extension")	
abstract	boolean	
attribute uses	Sequence of Attribute Use	
attribute wildcard	Optional Attribute Wildcard	

content type	function with properties: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Type simple)	
prohibited substitutions	Sequence of strings ("restriction"|"extension")	
assertions	Sequence of Assertion	

Déclaration d'élément

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type complexe"
name	string	Local name of the type (empty if anonymous)
target namespace	string	Namespace URI of the type (empty if anonymous)
type definition	Type simple or Type complexe	
type table	function with properties ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Type simple or Type complexe)	
scope	function with properties ("class":"Scope", "variety": ("global"|"local"), "parent": optional Type complexe)	
value constraint	see Attribute Declaration	
nillable	boolean	
identity-constraint definitions	Sequence of Identity Constraint	
substitution group affiliations	Sequence of Element Declaration	
substitution group exclusions	Sequence of strings ("restriction"|"extension")	
disallowed substitutions	Sequence of strings ("restriction"|"extension"|"substitution")	
abstract	boolean	

Caractère générique d'élément

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Wildcard"
namespace constraint	function with properties ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	string ("strict"|"lax"|"skip")	

- Facette

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	The name of the facet, for example "minLength" or "enumeration"
value	depends on facet	The value of the facet
fixed	boolean	
typed-value	For the enumeration facet only, array(xs:anyAtomicType*)	An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the "value" property is a sequence of strings, regardless of the actual type)

- Contrainte d'identité

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Identity-Constraint Definition"
name	string	Local name of the constraint
target namespace	string	Namespace URI of the constraint
identity-constraint category	string ("key"|"unique"|"keyRef")	
selector	XPath Property Record	
fields	Sequence of XPath Property Record	
referenced key	(For keyRef only): Identity Constraint	The corresponding key constraint

- Groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Groupe de modèle"

compositor	string ("sequence"|"choice"|"all")	
particles	Séquence de particule	

Définition de groupe de modèle

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de groupe de modèle"
name	string	Nom local du groupe de modèle
target namespace	string	URI d'espace du groupe de modèle
model group	Groupe de modèle	

Notation

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Déclaration de notation"
name	string	Nom local de la notation
target namespace	string	URI d'espace de nom de la notation
system identifier	anyURI	
public identifier	string	

Particule

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Particule"
min occurs	entier	
max occurs	entier ou string("unbounded")	
term	Déclaration d'élément, Caractère générique d'élément ou ModelGroup	

Type simple

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Définition de type simple"
name	string	Nom local du type (vide si anonyme)
target namespace	string	URI d'espace de noms du type (vide si anonyme)
final	Séquence de string("restriction"|"extension"|"list"|"union")	

context	composant contenant	
base type definition	Type simple	
facets	Séquence de Facette	
fundamental facets	Séquence vide (pas implémentée)	
variety	string ("atomic"|"list"|"union")	
primitive type definition	Type simple	
item type definition	uniquement pour les types de liste) Type simple	
member type definitions	(uniquement pour les types d'union) Séquence de Type simple	

Alternative de type

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Type Alternative"
test	XPath Property Record	
type definition	Type simple ou Type complexe	

XPath Property Record

Nom de propriété	Type de propriété	Valeur de propriété
namespace bindings	Séquence des fonctions avec les propriétés ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	L'URI de base statique de l'expression XPath
expression	string	L'expression XPath en tant que string

Type

altova:type(Node *as item?***) as (function(xs:string) as item()*)?** XP3.1 XQ3.1
La fonction **altova:type** soumet un nœud d'élément ou d'attribut d'un document XML et document et retourne l'information du type du nœud depuis le PSVI.

Note: Le document XML doit avoir une déclaration de schéma afin que ce schéma puisse être référencé.

Exemples

```
for $element in //Email
let $type := altova:type($element)
return $type
```

retourne une fonction qui contient l'information du type du nœud

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type ("kind")
  ```
  prend le composant du type du nœud (Type simple ou Type complexe) et retourne la valeur de la propriété **kind** du composant

Composants et leurs propriétés

Assertion

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Assertion"
test	XPath Property Record	

Déclaration d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Declaration"
name	string	Local name of the attribute
target namespace	string	Namespace URI of the attribute
type definition	Type simple or Type complexe	
scope	A function with properties ("class":"Scope", "variety": "global" or "local", "parent": the containing Type complexe or Attribute Group)	
value constraint	If present, a function with properties ("class": "Value Constraint", "variety": "fixed" or "default", "value": atomic value, "lexical form": string. Note that the "value" property is not available for namespace-sensitive types	
inheritable	boolean	

Déclaration de groupe d'attribut

Nom de propriété	Type de propriété	Valeur de propriété
kind	string	"Attribute Group Definition"
name	string	Local name of the attribute group
target namespace	string	Namespace URI of the attribute group
attribute uses	Sequence of (Attribute Use)	


  ```

| | | |
|---|---|---|
| attribute wildcard | Optional Attribute Wildcard | |

Utilisation d'attribut

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Attribute Use" |
| required | boolean | true if the attribute is required, false if optional |
| value constraint | See Attribute Declaration | |
| inheritable | boolean | |

Caractère générique

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Wildcard" |
| namespace constraint | function with properties ("class": "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings" | |
| process contents | string ("strict"\|"lax"\|"skip") | |

Type complexe

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Type complexe" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| base type definition | Complex Type Definition | |
| final | Sequence of strings ("restriction"\|"extension") | |
| context | Empty sequence (not implemented) | |
| derivation method | string ("restriction"\|"extension") | |
| abstract | boolean | |
| attribute uses | Sequence of Attribute Use | |
| attribute wildcard | Optional Attribute Wildcard | |
| content type | function with properties: ("class":"Content Type", "variety":string | |

| | | |
|---|---|---|
| | ("element-only"\|"empty"\|"mixed"\|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"\|"suffix"), "wildcard": Wildcard), "simple type definition": Type simple) | |
| prohibited substitutions | Sequence of strings ("restriction"\|"extension") | |
| assertions | Sequence of Assertion | |

- Déclaration d'élément

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Type complexe" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| type definition | Type simple or Type complexe | |
| type table | function with properties ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Type simple or Type complexe) | |
| scope | function with properties ("class":"Scope", "variety": ("global"\|"local"), "parent": optional Type complexe) | |
| value constraint | see Attribute Declaration | |
| nillable | boolean | |
| identity-constraint definitions | Sequence of Identity Constraint | |
| substitution group affiliations | Sequence of Element Declaration | |
| substitution group exclusions | Sequence of strings ("restriction"\|"extension") | |
| disallowed substitutions | Sequence of strings ("restriction"\|"extension"\|"substitution") | |
| abstract | boolean | |

- Caractère générique d'élément

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|

| | | |
|---|---|---|
| kind | string | "Wildcard" |
| namespace constraint | function with properties ("class": "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings" | |
| process contents | string ("strict"\|"lax"\|"skip") | |

Facette

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | The name of the facet, for example "minLength" or "enumeration" |
| value | depends on facet | The value of the facet |
| fixed | boolean | |
| typed-value | For the enumeration facet only, array(xs:anyAtomicType*) | An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the "value" property is a sequence of strings, regardless of the actual type) |

Contrainte d'identité

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Identity-Constraint Definition" |
| name | string | Local name of the constraint |
| target namespace | string | Namespace URI of the constraint |
| identity-constraint category | string ("key"\|"unique"\|"keyRef") | |
| selector | XPath Property Record | |
| fields | Sequence of XPath Property Record | |
| referenced key | (For keyRef only): Identity Constraint | The corresponding key constraint |

Groupe de modèle

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Groupe de modèle" |
| compositor | string ("sequence"\|"choice"\|"all") | |

| | | |
|---|---|---|
| particles | Séquence de particule | |

Définition de groupe de modèle

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Définition de groupe de modèle" |
| name | string | Nom local du groupe de modèle |
| target namespace | string | URI d'espace du groupe de modèle |
| model group | Groupe de modèle | |

Notation

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Déclaration de notation" |
| name | string | Nom local de la notation |
| target namespace | string | URI d'espace de nom de la notation |
| system identifier | anyURI | |
| public identifier | string | |

Particule

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Particule" |
| min occurs | entier | |
| max occurs | entier ou string("unbounded") | |
| term | Déclaration d'élément, Caractère générique d'élément ou ModelGroup | |

Type simple

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Définition de type simple" |
| name | string | Nom local du type (vide si anonyme) |
| target namespace | string | URI d'espace de noms du type (vide si anonyme) |
| final | Séquence de string("restriction"\|"extension"\|"list"\|"union") | |
| context | composant contenant | |

| | | |
|---|---|---|
| base type definition | Type simple | |
| facets | Séquence de Facette | |
| fundamental facets | Séquence vide (pas implémentée) | |
| variety | string ("atomic"\|"list"\|"union") | |
| primitive type definition | Type simple | |
| item type definition | uniquement pour les types de liste) Type simple | |
| member type definitions | (uniquement pour les types d'union) Séquence de Type simple | |

- Alternative de type

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| kind | string | "Type Alternative" |
| test | XPath Property Record | |
| type definition | Type simple ou Type complexe | |

- XPath Property Record

| Nom de propriété | Type de propriété | Valeur de propriété |
|---|---|---|
| namespace bindings | Séquence des fonctions avec les propriétés ("prefix": string, "namespace": anyURI) | |
| default namespace | anyURI | |
| base URI | anyURI | L'URI de base statique de l'expression XPath |
| expression | string | L'expression XPath en tant que string |

## 11.2.1.7 Fonctions XPath/XQuery : Séquence

Les fonctions d'extension de la séquence d'Altova peuvent être utilisées dans les expressions XPath et XQuery et proposent des fonctions supplémentaires pour le traitement des données. Les fonctions dans cette section peuvent être utilisées avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova. Ils sont disponibles dans des contextes XPath/XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova,** `http://www.altova.com/xslt-extensions`, et sont indiquées dans cette section par le préfixe `altova:`, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

| | |
|---|---|
| *Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :* | XP1 XP2 XP3.1 |
| *Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :* | XSLT1 XSLT2 XSLT3 |
| *Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :* | XQ1 XQ3.1 |

## attributes [altova:]

**altova:attributes(AttributeName** *as xs:string*) **asattribute()*** XP3.1 XQ3.1

Retourne tous les attributs possédant un nom local qui est le même que le nom fourni dans l'argument d'entrée, `AttributeName`. La recherche est sensible à la casse et est conduite le long de l'axe `attribute::` . Cela signifie que le nœud contextuel doit être le nœud d'élément parent.

*Exemples*

- `altova:attributes("MyAttribute")` retourne `MyAttribute()*`

**altova:attributes(AttributeName** *as xs:string*, **SearchOptions** *as xs:string*) **asattribute()*** XP3.1 XQ3.1

Retourne tous les attribut possédant un nom local qui est le même que le nom fourni dans l'argument d'entrée, `AttributeName`. La recherche est sensible à la casse et est conduite le long de l'axe `attribute::` . Le nœud contextuel doit être le nœud d'élément parent. Le deuxième argument est une chaîne contenant des flags optionnels. Les flags disponibles sont :
`r` = passe à une recherche d'expression régulière ; `AttributeName` doit alors être une chaîne de recherche d'expression régulière ;
`f` = si cette option est spécifiée, alors `AttributeName` fournit une concordance complète ; dans le cas contraire, `AttributeName` ne nécessite qu'une concordance partielle d'un nom d'attribut pour retourner cet attribut. Par exemple : si `f` n'est pas spécifié, `MyAtt` retournera `MyAttribute`;
`i` = passe à une recherche insensible à la casse ;
`p` = comprend le préfixe d'espace de nom dans la recherche ; `AttributeName` devrait ensuite contenir le préfixe d'espace de nom, par exemple : `altova:MyAttribute`.
Les flags peuvent être écrits dans n'importe quel ordre. Les flags invalides généreront des erreurs. Un ou plusieurs flags peuvent être omis. La chaîne vide est permise et produire le même effet que la fonction n'ayant qu'un seul argument (*signature précédente*). Néanmoins, une séquence vide n'est pas permise en tant que le deuxième argument.

*Exemples*

- `altova:attributes("MyAttribute", "rfip")` retourne `MyAttribute()*`
- `altova:attributes("MyAttribute", "pri")` retourne `MyAttribute()*`
- `altova:attributes("MyAtt", "rip")` retourne `MyAttribute()*`

- `altova:attributes("MyAttributes", "rfip")` ne retourne aucune correspondance.
- `altova:attributes("MyAttribute", "")` retourne `MyAttribute()*`
- `altova:attributes("MyAttribute", "Rip")` retourne une erreur de flag non reconnu.
- `altova:attributes("MyAttribute", )` retourne une erreur d'argument manquant.

### elements [altova:]

**`altova:elements(ElementName` *`as xs:string`*`) aselement()* XP3.1 XQ3.1`**

Retourne tous les éléments qui ont un nom local identique au nom fourni dans l'argument d'entrée, `ElementName`. La recherche est sensible à la casse et est conduite le long de l'axe `child::`. Le nœud contextuel doit être le nœud parent de/s l'élément/s recherché.

*Exemples*

- `altova:elements("MyElement")` retourne `MyElement()*`

**`altova:elements(ElementName` *`as xs:string`*`, SearchOptions` *`as xs:string`*`) aselement()* XP3.1 XQ3.1`**

Retourne tous les éléments qui ont un nom local identique au nom fourni dans l'argument d'entrée, `ElementName`. La recherche est sensible à la casse et est conduite le long de l'axe `child::`. Le nœud contextuel doit être le nœud parent de/s l'élément/s recherché. Le second argument est une chaîne contenant des flags optionnels. Les flags disponibles sont :

`r` = passe à une recherche d'expression régulière ; `ElementName` doit alors être une chaîne de recherche d'expression régulière ;

`f` = si cette option est spécifiée, alors `ElementName` fournit une concordance complète ; dans le cas contraire, `ElementName` ne nécessite qu'une concordance partielle d'un nom d'élément pour retourner cet élément. Par exemple : si `f` n'est pas spécifié, `MyElem` retournera `MyElement` ;

`i` = passe à une recherche insensible à la casse ;

`p` = comprend le préfixe d'espace de nom dans la recherche ; `ElementName` devrait ensuite contenir le préfixe d'espace de nom, par exemple : `altova:MyElement`.

Les flags peuvent être écrits dans n'importe quel ordre. Les flags invalides généreront des erreurs. Un ou plusieurs flags peuvent être omis. La chaîne vide est autorisée et produira le même effet que la fonction n'ayant qu'un argument (*signature précédente*). Néanmoins, une séquence vide n'est pas autorisée.

*Exemples*

- `altova:elements("MyElement", "rip")` retourne `MyElement()*`
- `altova:elements("MyElement", "pri")` retourne `MyElement()*`
- `altova:elements("MyElement", "")` retourne `MyElement()*`
- `altova:elements("MyElem", "rip")` retourne `MyElement()*`
- `altova:elements("MyElements", "rfip")` retourne aucune correspondance
- `altova:elements("MyElement", "Rip")` retourne une erreur flag-non reconnu.
- `altova:elements("MyElement", )` retourne une erreur second-argument-manquant.

### find-first [altova:]

**`altova:find-first((Sequence` *`as item()*`*`), (Condition( Sequence-Item` *`as xs:boolean`*`))`**
**`asitem()? XP3.1 XQ3.1`**

Cette fonction prend deux arguments. Le premier argument est une séquence d'un ou de plusieurs items de tout type de données. Le second argument, `Condition`, est une référence à une fonction XPath qui prend un argument (possède une arité de `1`) et retourne un booléen. Chaque item de **`Sequence`** est soumis

à son tour à la fonction référencée dans `Condition`. (*Rappel :* cette fonction prend un seul argument.) Le premier item **`Sequence`** qui cause la fonction dans **`Condition`** à évaluer à **`true()`** est retourné en tant que le résultat de **`altova:find-first`**, et l'itération s'arrête.

### *Exemples*

- **`altova:find-first`**`(5 to 10, function($a) {$a mod 2 = 0})` retourne `xs:integer 6`

L'argument **`Condition`** référence la fonction en ligne XPath 3.0, **`function()`**, qui déclare une fonction en ligne nommée **`$a`** puis la définit. Chaque item dans l'argument **`Sequence`** de **`altova:find-first`** est passé à son tour à **`$a`** en tant que sa valeur d'entrée. La valeur d'entrée est testée sur la condition dans la définition de la fonction (`$a mod 2 = 0`). La première valeur d'entrée pour satisfaire cette condition est retournée en tant que le résultat de **`altova:find-first`** (dans ce cas **`6`**).

- **`altova:find-first`**`((1 to 10), (function($a) {$a+3=7}))` retourne `xs:integer 4`

#### *Autres exemples*

Si le fichier **`C:\Temp\Customers.xml`** existe :

- **`altova:find-first`**`( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` retourne `xs:string C:\Temp\Customers.xml`

Si le fichier **`C:\Temp\Customers.xml`** n'existe pas et que **`http://www.altova.com/index.html`** existe :

- **`altova:find-first`**`( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` retourne `xs:string http://www.altova.com/index.html`

Si le fichier **`C:\Temp\Customers.xml`** n'existe pas, et que **`http://www.altova.com/index.html`** n'existe pas non plus :

- **`altova:find-first`**`( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` ne retourne aucun résultat

#### *Notes à propos des exemples indiqués ci-dessus*

- La fonction XPath 3.0, `doc-available`, prend un seul argument de chaîne, qui est utilisé en tant qu'URI, et retourne `true` si un nœud de document est trouvé à l'URI soumis. (Le document à l'URI soumis doit donc être un document XML.)
- La fonction `doc-available` peut être utilisée pour **`Condition`**, le second argument de `altova:find-first`, parce qu'il ne prend qu'un seul argument (arité=1), parce qu'il prend un `item()` en tant qu'entrée (une chaîne qui est utilisée en tant qu'URI), et retourne une valeur booléenne.
- Veuillez noter que la fonction `doc-available` est uniquement référencée, elle n'est pas appelée. Le suffixe `#1` qui y est attaché indique une fonction avec une arité de 1. Sous sa forme complète, `doc-available#1` signifie simplement : *Utiliser la fonction doc-availabe() à l'arité=1, en l'y passant en tant que son seul argument, chacun à son tour, chacun des items dans la première séquence*. En résultat, chacun des deux chaînes sera passée à **`doc-available()`**, qui utilise la chaîne en tant qu'URI et teste si un nœud de document existe à l'URI. S'il en existe un, **`doc-available()`** évalue à `true()` et cette chaîne est retournée en tant que le résultat de la fonction **`altova:find-first`**. *Note à propos de la fonction doc-available() : les chemins relatifs sont résolus relativement à l'URI de base actuel, qui est par défaut l'URI du document XML à partir duquel la fonction est chargée.*

### find-first-combination [altova:]

**altova:find-first-combination((Seq-01** *as item()**)**, **(Seq-02** *as item()**)**, **(Condition( Seq-01-Item, Seq-02-Item** *as xs:boolean***)) asitem()* XP3.1 XQ3.1**

Cette fonction prend trois arguments :

- Les deux premiers arguments, **Seq-01** et **Seq-02**, sont des séquences d'un ou de plusieurs items de tout type de données.
- Le troisième argument, **Condition**, est une référence à une fonction XPath qui prend deux arguments (a une arité de 2) et retourne un booléen.

Les items de **Seq-01** et **Seq-02** sont passés dans des paires ordonnées (un item de chaque séquence faisant une paire) en tant que les arguments de la fonction dans **Condition**. Les paires sont classées comme suit :

```
If   Seq-01 = X1, X2, X3 ... Xn
And Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

La première paire ordonnée qui entraîne la fonction **Condition** à évaluer à **true()** est retournée en tant que le résultat de **altova:find-first-combination**. Veuillez noter que : (i) si la fonction **Condition** itère par le biais des paires d'argument soumises et n'évalue pas une fois à **true()**, alors **altova:find-first-combination** retournera *Aucun résultat* ; (ii) Le résultat de **altova:find-first-combination** sera toujours une paire d'items (de tout type de données) ou aucun item.

Exemples

- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) retourne la séquence de xs:integers (11, 21)
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) retourne la séquence de f xs:integers (11, 22)
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) retourne la séquence de xs:integers (11, 23)

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *as item()**)**, **(Seq-02** *as item()**)**, **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean***)) asitem()* XP3.1 XQ3.1**

Cette fonction prend trois arguments :

- Les deux premiers arguments, **Seq-01** et **Seq-02**, sont des séquences d'un ou de plusieurs items de tout type de données.
- Le troisième argument, **Condition**, est une référence à une fonction XPath qui prend deux arguments (a une arité de 2) et retourne un booléen.

Les items de **Seq-01** et **Seq-02** sont passés dans des paires ordonnées en tant que les arguments de la fonction dans **Condition**. Les paires sont classées comme suit :

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

La première paire ordonnée qui cause la fonction **Condition** à évaluer à **true()** est retournée en tant que le résultat de **altova:find-first-pair**. Veuillez noter que : (i) Si la fonction **Condition** itère par le biais

des paires d'arguments soumis et n'évalue pas une seule fois à **true()**, alors **altova:find-first-pair** retournera *Aucun résultat*; (ii) Le résultat de **altova:find-first-pair** sera toujours une paire d'items (de tout type de données) ou aucun item.

Exemples

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) retourne la séquence de xs:integers (11, 21)
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) retourne *Aucun résultat*

Veuillez noter à partir des deux exemples ci-dessus que l'ordonnance des paires est : (11, 21) (12, 22) (13, 23)...(20, 30). C'est pourquoi le second exemple retourne *Aucun résultat* (parce qu'aucune paire ordonnée de donne une somme de 33).

find-first-pair-pos [altova:]

altova:find-first-pair-pos((Seq-01 *as item()**)**,* **(Seq-02** *as item()**)**,* **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean***)) asxs:integer XP3.1 XQ3.1**
Cette fonction prend trois arguments :

- Les deux premiers arguments, **Seq-01** and **Seq-02**, sont des séquences d'un ou de plusieurs items de tout type de données.
- Le troisième argument, **Condition**, est une référence à une fonction XPath qui prend deux arguments (a une arité de 2) et retourne un booléen.

Les items de **Seq-01** et **Seq-02** sont passés dans des paires ordonnées en tant que les arguments de la fonction dans **Condition**. Les paires sont classées comme suit :

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

La position d'index de la première paire ordonnée qui entraîne la fonction **Condition** à évaluer à **true()** est retournée en tant que le résultat de **altova:find-first-pair-pos**. Veuillez noter que si la fonction **Condition** itère par le biais des paires d'arguments soumises et n'évalue pas une seule fois à **true()**, alors **altova:find-first-pair-pos** retournera *Aucun résultat*.

Exemples

- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) retourne 1
- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) retourne *Aucun résultat*

Veuillez noter à partir des deux exemples ci-dessus que l'ordonnance des paires est : (11, 21) (12, 22) (13, 23)...(20, 30). dans le premier exemple, la première paire entraîne la fonction **Condition** à évaluer à **true()**, et donc sa position d'index dans la séquence, **1**, est retournée. Le second exemple retourne *Aucun résultat* parce qu'aucune paire ne totalise pas une somme de 33.

find-first-pos [altova:]

altova:find-first-pos((Sequence *as item()**)**, **(Condition(Sequence-Item** *as xs:boolean***))**
asxs:integer **XP3.1** **XQ3.1**
Cette fonction prend deux arguments. Le premier argument est une séquence d'un ou de plusieurs items de tout type de données. Le second argument, `Condition`, est une référence à une fonction XPath qui prend un argument (a une arité de **1**) et retourne une booléenne. Chaque item de **Sequence** est soumis à son tour à la fonction référencée dans `Condition`. (*Rappel :* cette fonction prend un seul argument.) Le premier item **Sequence** qui cause la fonction dans **Condition** à évaluer à **true()** voit sa position index dans **Sequence** retournée en tant que résultat de **altova:find-first-pos**, et l'itération stoppe.

- *Exemples*

  - **altova:find-first-pos**`(5 to 10, function($a) {$a mod 2 = 0})` retourne `xs:integer 2`
  L'argument **Condition** référence la fonction en ligne XPath 3.0, **function()**, qui déclare une fonction en ligne nommée **$a** et puis la définit. Chaque item dans l'argument de **Sequence** de **altova:find-first-pos** est passé à son tour à **$a** en tant que sa valeur d'entrée. La valeur d'entrée est testée à la condition dans la définition de la fonction (`$a mod 2 = 0`). La position d'index dans la séquence de la première valeur d'entrée pour satisfaire à cette condition est retournée en tant que le résultat de **altova:find-first-pos** (dans ce cas **2**, puisque **6**, la première valeur (dans la séquence) afin de satisfaire à la condition, est à la position d'index **2** dans la séquence).

  - **altova:find-first-pos**`((2 to 10), (function($a) {$a+3=7}))` retourne `xs:integer 3`

  Autres exemples
  Si le fichier **C:\Temp\Customers.xml** existe :
  - **altova:find-first-pos**`( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` retourne `1`

  Si le fichier **C:\Temp\Customers.xml** n'existe pas, et que **http://www.altova.com/index.html** existe :
  - **altova:find-first-pos**`( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` retourne `2`

  Si le fichier **C:\Temp\Customers.xml** n'existe pas, et que **http://www.altova.com/index.html** n'existe pas non plus :
  - **altova:find-first-pos**`( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` ne retourne aucun résultat

  Notes à propos des exemples donnés ci-dessus

  - La fonction XPath 3.0, `doc-available`, prend un seul argument de chaîne qui est utilisé en tant qu'un URI, et retourne `true` si un nœud de document est trouvé à l'URI soumis. (Le document à l'URI soumis doit donc être un document XML.)
  - La fonction `doc-available` peut être utilisée pour **Condition**, le second argument de `altova:find-first-pos`, parce qu'il ne prend qu'un seul argument (arité=1), parce qu'il prend un `item()` en tant qu'entrée (une chaîne qui est utilisée en tant qu'un URI), et retourne une valeur booléenne.
  - Veuillez noter que la fonction `doc-available` est uniquement référencée, elle n'est pas appelée. Le suffixe `#1` qui y est attaché indique une fonction avec une arité de 1. dans sa totalité, `doc-available#1` signifie simplement : *Utiliser la fonction doc-availabe() qui a arité=1, y passant, en tant que son argument simple, chacun à son tour, chaque item dans la première séquence*. En

tant que résultat. chacune des deux chaînes sera passée à **doc-available()**, qui utilise la chaîne en tant qu'un URI et teste si un nœud de document existe à l'URI. Si c'est le cas, la fonction **doc-available()** évalue à `true()` et la position de l'index de cette chaîne dans la séquence est retournée en tant que le résultat de la fonction **altova:find-first-pos**. *Note à propos de la fonction doc-available() : les chemins relatifs sont résolus relativement à l'URI de base actuel, qui par défaut est l'URI du document XML à partir duquel la fonction est chargée.*

for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *as element()?*, **Seq2** *as element()?*, **Function** *as function()*) **asitem()*** **XP3.1** **XQ3.1**

Les premiers deux arguments identifient deux éléments, dont les attributs sont utilisés pour générer des paires d'attribut, dans laquelle un attribut d'une paire est obtenu depuis le premier élément et l'autre attribut est obtenu depuis le second élément. Les paires d'attribut sont sélectionnées sur le fait qu'ils présentent le même nom, et les paires sont classées par ordre alphabétique (sur leur nom) dans un ensemble. Si, pour un attribut, aucun attribut correspondant n'existe dans l'autre élément, la paire sera "disjointe", ce qui signifie qu'elle consiste en un seul membre. L'item de fonction (troisième argument `Function`) est appliqué séparément à chaque paire dans la séquence des paires (joints et disjointe), résultant en une sortie qui est une séquence d'items.

Exemples

- **altova:for-each-attribute-pair**`(/Example/Test-A, /Example/Test-B, function($a, $b) {$a+b})` retourne ...

  (2, 4, 6) si
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (2, 4, 6) si
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  (2, 6) si
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  Note : *Le résultat* **(2 6)** *est obtenu par le biais de l'action suivante :* **(1+1 ()+2 3+3 4+())**. *Si un des opérandes est la séquence vide, comme c'est le cas des items 2 et 4, le résultat de l'addition est une séquence vide.*

- **altova:for-each-attribute-pair**`(/Example/Test-A, /Example/Test-B, concat#2)` retourne ...

  (11 22 33) si
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (11 2 33 4) si
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

for-each-combination [altova:]

altova:for-each-combination(FirstSequence *as item()**, **SecondSequence** *as item()**, **Function($i,$j){$i || $j}) asitem()* XP3.1 XQ3.1**

Les items des deux séquences dans les deux premiers arguments sont combinés de manière à ce que chaque item de la première séquence est combiné, dans l'ordre, une fois avec chaque item de la seconde séquence. La fonction donnée en tant que le troisième argument est appliquée à chaque combinaison dans la séquence résultante, entraînant une sortie qui est une séquence d'items (*voir exemple*).

Exemples

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) retourne ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3')

for-each-matching-attribute-pair [altova:]

altova:for-each-matching-attribute-pair(Seq1 *as element()?*, **Seq2** *as element()?*, **Function** *as function()*) **asitem()* XP3.1 XQ3.1**

Les premiers deux arguments identifient deux éléments, dont les attributs sont utilisés pour générer des paires d'attribut, dans laquelle un attribut d'une paire est obtenu depuis le premier élément et l'autre attribut est obtenu depuis le second élément. Les paires d'attribut sont sélectionnées sur le fait qu'ils présentent le même nom, et les paires sont classées par ordre alphabétique (sur leur nom) dans un ensemble. Si, pour un attribut, aucun attribut correspondant n'existe dans l'autre élément, aucune paire ne sera générée. L'item de fonction (troisième argument `Function`) est appliqué séparément à chaque paire dans la séquence des paires, résultant en une sortie qui est une séquence d'items.

Exemples

- **altova:for-each-matching-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b}) retourne ...

  (2, 4, 6) si
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (2, 4, 6) si
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  (2, 6) si
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att3="1" />

- **altova:for-each-matching-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) retourne ...

  (11, 22, 33) si
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (11, 33) si
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

substitute-empty [altova:]

altova:substitute-empty(FirstSequence *as item()**, **SecondSequence** *as item()*) **asitem()***
XP3.1 XQ3.1
Si `FirstSequence` est vide, retourne `SecondSequence`. Si `FirstSequence` n'est pas vide, retourne `FirstSequence`.

Exemples

- **altova:substitute-empty**((1,2,3), (4,5,6)) retourne (1,2,3)
- **altova:substitute-empty**((), (4,5,6)) retourne (4,5,6)

11.2.1.8 Fonctions XPath/XQuery : Chaîne

Les fonctions d'extension de chaîne d'Altova peuvent être utilisées dans les expressions XPath et XQuery et proposent des fonctions supplémentaires pour le traitement des données. Les fonctions dans cette section peuvent être utilisées avec les moteurs **XPath 3.0** et **XQuery 3.0** d'Altova. Ils sont disponibles dans des contextes XPath/XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova, `http://www.altova.com/xslt-extensions`**, et sont indiquées dans cette section par le préfixe **`altova:`**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

camel-case [altova:]

altova:camel-case(InputString *as xs:string*) **asxs:string** XP3.1 XQ3.1
Retourne la chaîne d'entrée **`InputString`** en CamelCase. La chaîne est analysée en utilisant l'expression régulière **`'\s'`** (qui est un raccourci pour le caractère d'espace blanc). Le premier caractère non-espace blanc après un espace blanc ou une séquence de plusieurs espaces blancs est mis en majuscule. Le premier caractère dans la chaîne de sortie est mis en majuscule.

Exemples

- `altova:camel-case("max")` retourne `Max`
- `altova:camel-case("max max")` retourne `Max Max`
- `altova:camel-case("file01.xml")` retourne `File01.xml`
- `altova:camel-case("file01.xml file02.xml")` retourne `File01.xml File02.xml`
- `altova:camel-case("file01.xml  file02.xml")` retourne `File01.xml  File02.xml`
- `altova:camel-case("file01.xml -file02.xml")` retourne `File01.xml -file02.xml`

altova:camel-case(InputString *as xs:string*, **SplitChars** *as xs:string*, **IsRegex** *as xs:boolean*) **asxs:string** XP3.1 XQ3.1

Convertit la chaîne d'entrée **InputString** en camel case en utilisant **SplitChars** pour déterminer le/s caractère/s qui déclenche/nt la prochaine mise en majuscule. **SplitChars** est utilisé en tant qu'expression régulière quand **IsRegex = true()**, ou en tant que caractères normaux quand **IsRegex = false()**. Le premier caractère dans la chaîne de sortie est mis en majuscule.

Exemples

- `altova:camel-case("setname getname", "set|get", true())` retourne `setName getName`
- `altova:camel-case("altova\documents\testcases", "\", false())` retourne `Altova\Documents\Testcases`

char [altova:]

altova:char(Position *as xs:integer*) **asxs:string** XP3.1 XQ3.1

Retourne une chaîne contenant le caractère à la position spécifiée par l'argument `Position`, dans la chaîne obtenue en convertissant la valeur de l'item contextuel en `xs:string`. La chaîne de résultat sera vide si aucun caractère n'existe à l'index soumis par l'argument `Position`.

Exemples

Si l'item contextuel est **1234ABCD**:

- `altova:char(2)` retourne `2`
- `altova:char(5)` retourne `A`
- `altova:char(9)` retourne la chaîne vide.
- `altova:char(-2)` retourne la chaîne vide.

altova:char(InputString *as xs:string*, **Position** *as xs:integer*) **asxs:string** XP3.1 XQ3.1

Retourne une chaîne contenant la caractère à la position spécifiée par l'argument `Position`, dans la chaîne soumise en tant que l'argument `InputString`. La chaîne de résultat sera vide si aucun caractère n'existe à l'index soumis par l'argument `Position`.

Exemples

- `altova:char("2014-01-15", 5)` retourne `-`
- `altova:char("USA", 1)` retourne `U`
- `altova:char("USA", 10)` retourne la chaîne vide.
- `altova:char("USA", -2)` retourne la chaîne vide.

create-hash-from-string[altova:]

altova:create-hash-from-string(InputString *as xs:string*) **asxs:string** XP2 XQ1 XP3.1 XQ3.1
altova:create-hash-from-string(InputString *as xs:string*, **HashAlgo** *as xs:string*)

asxs:string XP2 XQ1 XP3.1 XQ3.1
Génère un string de hashage depuis `InputString` en utilisant l'algorithme de hashage spécifié par l'argument `HashAlgo`. Les algorithmes de hashage suivants peuvent être spécifiés (en majuscule ou en minuscule) : **MD5**, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**. Si le deuxième argument n'est pas spécifié (*voir la première signature ci-dessus*), l'algorithme de hashage **SHA-256** sera utilisé.

Exemples

- **altova:create-hash-from-string**('abc') retourne un string de hashage généré en utilisant l'algorithme de hachage **SHA-256**.
- **altova:create-hash-from-string**('abc', 'md5') retourne un string de hashage généré en utilisant l'algorithme de hachage **MD5**.
- **altova:create-hash-from-string**('abc', 'MD5') retourne un string de hashage généré en utilisant l'algorithme de hachage **MD5**.

first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) **asxs:string** XP3.1 XQ3.1
Retourne une chaîne contenant le premier `X-Number` des caractères de la chaîne obtenue en convertissant la valeur de l'item de contexte en `xs:string`.

Exemples

Si l'item de contexte est **1234ABCD** :

- **altova:first-chars**(2) retourne 12
- **altova:first-chars**(5) retourne 1234A
- **altova:first-chars**(9) retourne 1234ABCD

altova:first-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) **asxs:string** XP3.1 XQ3.1
Retourne une chaîne contenant le premier `X-Number` des caractères de la chaîne soumise en tant que l'argument `InputString`.

Exemples

- **altova:first-chars**("2014-01-15", 5) retourne 2014-
- **altova:first-chars**("USA", 1) retourne U

format-string [altova:]

altova:format-string(InputString *as xs:string*, **FormatSequence** *as item()**) **asxs:string** XP3.1 XQ3.1
Le string d'entrée (premier argument) contient des paramètres de position (`%1`, `%2`, etc). Chaque paramètre est remplacé par l'item de string qui est situé dans la position correspondante dans la séquence de format (soumise en tant que le second argument). Donc le premier item dans la séquence de format remplace de paramètre de positionnement `%1`, le second item remplace `%2`, etc. La fonction retourne ce string formaté qui contient les remplacements. Si aucun string n'existe pour un paramètre de positionnement, alors le paramètre de positionnement lui-même est retourné. Cela se produit lorsque l'index d'un paramètre de positionnement est supérieur au nombre d'items dans la séquence de format.

Exemples

- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe')) retourne "Hello Jane, John, Joe"
- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom')) retourne

"Hello Jane, John, Joe"
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom')) retourne "Hello Jane, John, Tom"
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe')) retourne "Hello Jane, John, %4"

▼ last-chars [altova:]

altova:last-chars(X-Number *as xs:integer*) **asxs:string** XP3.1 XQ3.1
Retourne une chaîne contenant le dernier X-Number de caractères de la chaîne obtenue en convertissant la valeur de l'item contextuel en xs:string.

Exemples

Si l'item contextuel est **1234ABCD** :
- **altova:last-chars**(2) retourne CD
- **altova:last-chars**(5) retourne 4ABCD
- **altova:last-chars**(9) retourne 1234ABCD

altova:last-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) **asxs:string** XP3.1 XQ3.1
Retourne une chaîne contenant le dernier X-Number de caractères de la chaîne soumise en tant que l'argument InputString.

Exemples

- **altova:last-chars**("2014-01-15", 5) retourne 01-15
- **altova:last-chars**("USA", 10) retourne USA

▼ pad-string-left [altova:]

altova:pad-string-left(StringToPad *as xs:string*, **StringLength** *as xs:integer*, **PadCharacter** *as xs:string*) **asxs:string** XP3.1 XQ3.1
L'argument PadCharacter est un caractère unique. Il est bourré à la gauche de la chaîne pour augmenter le nombre de caractères dans StringToPad de manière à ce que ce nombre soit équivalent à la valeur d'entier de l'argument StringLength. L'argument StringLength peut avoir toute valeur d'entier (positive ou négative), mais le padding n'aura lieu que si la valeur de StringLength est supérieure au nombre de caractères dans StringToPad. Si StringToPad. comporte plus de caractères que la valeur de StringLength, alors StringToPad ne sera pas modifié.

Exemples

- **altova:pad-string-left**('AP', 1, 'Z') retourne 'AP'
- **altova:pad-string-left**('AP', 2, 'Z') retourne 'AP'
- **altova:pad-string-left**('AP', 3, 'Z') retourne 'ZAP'
- **altova:pad-string-left**('AP', 4, 'Z') retourne 'ZZAP'
- **altova:pad-string-left**('AP', -3, 'Z') retourne 'AP'
- **altova:pad-string-left**('AP', 3, 'YZ') retourne une erreur pad-character-too-long

▼ pad-string-right [altova:]

altova:pad-string-right(StringToPad *as xs:string*, **StringLength** *as xs:integer*,

PadCharacter *as xs:string*) **asxs:string** XP3.1 XQ3.1
L'argument PadCharacter est un caractère unique. Il est bourré à la droite de la chaîne pour augmenter le nombre de caractères dans StringToPad de manière à ce que ce nombre soit équivalent à la valeur d'entier de l'argument StringLength. L'argument StringLength peut avoir toute valeur d'entier (positive ou négative), mais le padding n'aura lieu que si la valeur de StringLength est supérieure au nombre de caractères dans StringToPad. Si StringToPad. comporte plus de caractères que la valeur de StringLength, alors StringToPad ne sera pas modifié.

Exemples

- **altova:pad-string-right**('AP', 1, 'Z') retourne 'AP'
- **altova:pad-string-right**('AP', 2, 'Z') retourne 'AP'
- **altova:pad-string-right**('AP', 3, 'Z') retourne 'APZ'
- **altova:pad-string-right**('AP', 4, 'Z') retourne 'APZZ'
- **altova:pad-string-right**('AP', -3, 'Z') retourne 'AP'
- **altova:pad-string-right**('AP', 3, 'YZ') retourne une erreur pad-character-too-long

repeat-string [altova:]

altova:repeat-string(**InputString** *as xs:string*, **Repeats** *as xs:integer*) **asxs:string** XP2 XQ1 XP3.1 XQ3.1
Génère une chaîne qui est composée du premier argument InputString répété Repeats nombre de fois.

Exemples

- **altova:repeat-string**("Altova #", 3) retourne "Altova #Altova #Altova #"

substring-after-last [altova:]

altova:substring-after-last(**MainString** *as xs:string*, **CheckString** *as xs:string*)
asxs:string XP3.1 XQ3.1
Si CheckString est trouvé dans MainString, alors la sous-chaîne qui se produit après CheckString dans MainString est retournée. Si CheckString n'est pas trouvé dans MainString, la chaîne vide est retournée. Si CheckString est une chaîne vide, alors MainString est retourné dans sa totalité. S'il y a plus d'une survenance de CheckString dans MainString, alors la sous-chaîne après la dernière survenance de CheckString est retournée.

Exemples

- **altova:substring-after-last**('ABCDEFGH', 'B') retourne 'CDEFGH'
- **altova:substring-after-last**('ABCDEFGH', 'BC') retourne 'DEFGH'
- **altova:substring-after-last**('ABCDEFGH', 'BD') retourne ''
- **altova:substring-after-last**('ABCDEFGH', 'Z') retourne ''
- **altova:substring-after-last**('ABCDEFGH', '') retourne 'ABCDEFGH'
- **altova:substring-after-last**('ABCD-ABCD', 'B') retourne 'CD'
- **altova:substring-after-last**('ABCD-ABCD-ABCD', 'BCD') retourne ''

substring-before-last [altova:]

altova:substring-before-last(**MainString** *as xs:string*, **CheckString** *as xs:string*)
asxs:string XP3.1 XQ3.1
Si CheckString est trouvé dans MainString, alors la sous-chaîne qui se produit avant CheckString dans MainString est retournée. Si CheckString n'est pas trouvé dans MainString, ou si CheckString est

une chaîne vide, la chaîne vide est retournée. S'il y a plus d'une survenance de `CheckString` dans `MainString`, alors la sous-chaîne avant la dernière survenance de `CheckString` est retournée.

Exemples

- **altova:substring-before-last**('ABCDEFGH', 'B') retourne 'A'
- **altova:substring-before-last**('ABCDEFGH', 'BC') retourne 'A'
- **altova:substring-before-last**('ABCDEFGH', 'BD') retourne ''
- **altova:substring-before-last**('ABCDEFGH', 'Z') retourne ''
- **altova:substring-before-last**('ABCDEFGH', '') retourne ''
- **altova:substring-before-last**('ABCD-ABCD', 'B') retourne 'ABCD-A'
- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') retourne 'ABCD-ABCD-'

substring-pos [altova:]

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*)
asxs:integer XP3.1 XQ3.1

Retourne la position de caractère de la première occurrence de `StringToFind` dans la chaîne `StringToCheck`. La position du caractère est retournée en tant qu'un entier. Le premier caractère de `StringToCheck` a la position `1`. Si `StringToFind` ne se produit pas dans le cadre de `StringToCheck`, l'entier `0` est retourné. Pour contrôler la deuxième occurrence ou une occurrence ultérieure de `StringToCheck`, utiliser la signature suivante de cette fonction.

Exemples

- **altova:substring-pos**('Altova', 'to') retourne 3
- **altova:substring-pos**('Altova', 'tov') retourne 3
- **altova:substring-pos**('Altova', 'tv') retourne 0
- **altova:substring-pos**('AltovaAltova', 'to') retourne 3

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*, **Integer** *as xs:integer*) **asxs:integer** XP3.1 XQ3.1

Retourne la position de caractère de `StringToFind` dans la chaîne, `StringToCheck`. La recherche de `StringToFind` commence à partir de la position de caractère indiquée par l'argument `Integer` ; la sous-chaîne du caractère avant cette position n'est pas recherchée. Néanmoins, l'entier retourné, est la position de la chaîne trouvée dans le cadre de la chaîne *entière*, `StringToCheck`. Cette signature est utile pour trouver la deuxième position ou une position ultérieure d'une chaîne qui se produit plusieurs fois avec `StringToCheck`. Si `StringToFind` ne se produit pas dans le cadre de `StringToCheck`, l'entier `0` est retourné.

Exemples

- **altova:substring-pos**('Altova', 'to', 1) retourne 3
- **altova:substring-pos**('Altova', 'to', 3) retourne 3
- **altova:substring-pos**('Altova', 'to', 4) retourne 0
- **altova:substring-pos**('Altova-Altova', 'to', 0) retourne 3
- **altova:substring-pos**('Altova-Altova', 'to', 4) retourne 10

trim-string [altova:]

altova:trim-string(InputString *as xs:string*) **asxs:string** XP3.1 XQ3.1

Cette fonction prend un argument `xs:string`, supprime tout espace blanc de tête et de fin et retourne une `xs:string` « nettoyée ».

Exemples

- `altova:trim-string("  Hello World  ")` retourne `"Hello World"`
- `altova:trim-string("Hello World  ")` retourne `"Hello World"`
- `altova:trim-string("  Hello World")` retourne `"Hello World"`
- `altova:trim-string("Hello World")` retourne `"Hello World"`
- `altova:trim-string("Hello  World")` retourne `"Hello  World"`

trim-string-left [altova:]

`altova:trim-string-left(InputString` *`as xs:string`***`) asxs:string`** **XP3.1** **XQ3.1**
Cette fonction prend un argument `xs:string`, supprime tout espace blanc de tête, et retourne une `xs:string` nettoyée à gauche.

Exemples

- `altova:trim-string-left("  Hello World  ")` retourne `"Hello World  "`
- `altova:trim-string-left("Hello World  ")` retourne `"Hello World  "`
- `altova:trim-string-left("  Hello World")` retourne `"Hello World"`
- `altova:trim-string-left("Hello World")` retourne `"Hello World"`
- `altova:trim-string-left("Hello  World")` retourne `"Hello World"`

trim-string-right [altova:]

`altova:trim-string-right(InputString` *`as xs:string`***`) asxs:string`** **XP3.1** **XQ3.1**
Cette fonction prend un argument `xs:string`, supprime tout espace blanc de fin de ligne, et retourne une `xs:string` nettoyée à droite.

Exemples

- `altova:trim-string-right("  Hello World  ")` retourne `"  Hello World"`
- `altova:trim-string-right("Hello World  ")` retourne `"Hello World"`
- `altova:trim-string-right("  Hello World")` retourne `"  Hello World"`
- `altova:trim-string-right("Hello World")` retourne `"Hello World"`
- `altova:trim-string-right("Hello  World")` retourne `"Hello  World"`

11.2.1.9 Fonctions XPath/XQuery : Divers

L'objectif général suivant des fonctions d'extension XPath/XQuery sont prises en charge dans la version actuelle de RaptorXML Server et celles-ci peuvent être utilisées dans (i) des expressions XPath dans un contexte XSLT, ou dans (ii) des expressions XQuery dans un document XQuery.

Note concernant le nommage de fonctions et de l'applicabilité de la langue

Les fonctions d'extension Altova peuvent être utilisées dans les expressions XPath/XQuery. Elles fournissent des fonctions supplémentaires aux fonctions d'ores et déjà disponibles dans la librairie standard des fonctions XPath, XQuery et XSLT. Les fonctions d'extension Altova se trouvent dans **l'espace de nom des fonctions d'extension Altova,** **`http://www.altova.com/xslt-extensions`**, et sont indiquées dans cette

section par le préfixe **altova:**, qui est présupposé être lié à cet espace de nom. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Fonctions XPath (utilisées dans les expressions XPath dans XSLT) :	XP1 XP2 XP3.1
Fonctions XSLT (utilisées dans les expressions XPath dans XSLT) :	XSLT1 XSLT2 XSLT3
Fonctions XQuery (utilisées dans les expressions XQuery dans XQuery) :	XQ1 XQ3.1

decode-string [altova:]

altova:decode-string(Input *as xs:base64Binary*) **as xs:string** XP3.1 XQ3.1
altova:decode-string(Input *as xs:base64Binary*, **Encoding** *as xs:string*) **as xs:string** XP3.1 XQ3.1
Décode l'entrée base64Binary soumise à un string en utilisant l'encodage spécifié. Si aucun codage n'est spécifié, l'encodage UTF-8 est utilisé. Les encodages suivants sont pris en charge: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Exemples

- **altova:decode-string**(`$XML1/MailData/Meta/b64B`) retourne l'entrée base64Binary en tant qu'un string encodé en UTF-8
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "UTF-8"`) retourne l'entrée base64Binary en tant qu'un string encodé en UTF-8
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "ISO-8859-1"`) retourne l'entrée base64Binary en tant qu'un string encodé en ISO-8859-1

.

get-temp-folder [altova:]

altova:get-temp-folder() as `xs:string` XP2 XQ1 XP3.1 XQ3.1
Cette fonction ne prend aucun argument. Elle retourne le chemin vers le dossier temporaire de l'utilisateur actuel.

Exemples

- **altova:get-temp-folder**() retournerait, sur une machine Windows, quelque chose du genre `C:\Users\<UserName>\AppData\Local\Temp\` en tant que `xs:string`.

generate-guid [altova:]

altova:generate-guid() asxs:string XP2 XQ1 XP3.1 XQ3.1
Génère un string GUID unique.

Exemples

- **altova:generate-guid**() retourne (par exemple) `85F971DA-17F3-4E4E-994E-99137873ACCD`

▼ high-res-timer [altova:]

altova:high-res-timer() as xs:double XP3.1 XQ3.1
Retourne une valeur de minuteur haute résolution en secondes. Un minuteur de haute résolution, lorsqu'il est présent dans un système, permet des mesures temporelles de haute précision lorsque celles-ci sont requises (par exemples pour des animations et pour déterminer précisément l'heure d'exécution du code). Cette fonction fournit la résolution du minuteur haute résolution du système.

Exemples

- `altova:high-res-timer()` retourne quelque chose comme `'1.16766146154566E6'`

▼ parse-html [altova:]

altova:parse-html(HTMLText *as xs:string*) **asnode() XP3.1 XQ3.1**
L'argument `HTMLText` est un string qui contient le texte d'un document HTML. La fonction crée une arborescence HTML depuis le string. Le string soumis peut contenir l'élément HTML ou pas. Dans tous les cas, l'élément racine de l'arborescence est un élément nommé **`HTML`**. Il est préférable de s'assurer que le code HTML dans le string soumis est un HTML valide.

Exemples

- `altova:parse-html("<html><head/><body><h1>Header</h1></body></html>")` crée une arborescence HTML depuis le string soumis

▼ sleep[altova:]

altova:sleep(Millisecs *as xs:integer*) **asempty-sequence() XP2 XQ1 XP3.1 XQ3.1**
Suspend l'exécution de l'opération actuelle pour le nombre de millisecondes donné par l'argument `Millisecs`.

Exemples

- `altova:sleep(1000)` suspend l'exécution de l'opération actuelle pour 1000 millisecondes.

[Top 482]

11.2.1.10 Fonctions de graphique

Les fonctions de graphique recensées ci-dessous vous permettent de créer, générer et enregistrer des graphiques comme images. Elles sont prises en charge dans la version actuelle de votre produit Altova comme suit. Veuillez noter que, en ce qui concerne les versions futures de votre produit, la prise en charge d'une fonction peut être interrompue et le comportement de certaines fonctions peut changer. Veuillez consulter la documentation lors des publications à venir pour plus d'informations concernant la prise en charge des fonctions d'extension Altova de cette version.

Note : Les fonctions de graphique sont supportées uniquement dans les **produits de serveur d'Altova** et les **produits d'édition Enterprise de Altova**.

Note : Les formats d'image pris en charge pour les graphiques dans les éditions de serveur sont `jpg`, `png` et `bmp`. La meilleure option est png car elle est sans perte et compressée. Dans les éditions Enterprise, les formats pris en charge sont `jpg`. `png`, `bmp` et `gif`.

Fonctions de génération et d'enregistrement des graphiques

Ces fonctions prennent l'objet de graphique (obtenu avec les fonctions de création de graphique) et soit génèrent une image soit enregistrent une image dans le fichier

`altova:generate-chart-image` `($chart, $width, $height, $encoding) en tant que atomic`

alors que

- `$chart` est l'item d'extension de graphique obtenu avec la fonction `altova:create-chart`
- `$width` et `$height` doivent être spécifiés avec une unité de longueur
- `$encoding` peut être `binarytobase64` ou `binarytobase16`

La fonction retourne l'image de graphique dans le codage spécifié.

`altova:generate-chart-image` `($chart, $width, $height, $encoding, $imagetype) as atomic`

alors que

- `$chart` est l'item d'extension de graphique obtenu avec la fonction `altova:create-chart`
- `$width` et `$height` doivent être spécifiés avec une unité de longueur
- `$encoding` peut être `binarytobase64` ou `binarytobase16`
- `$imagetype` peut être un des formats d'image suivants : `png`, `gif`, `bmp`, `jpg`, `jpeg`. Veuillez noter que `gif` n'est pas pris en charge pour les produits de serveur. *Voir également la note en haut de la page*.

La fonction retourne l'image de graphique dans le format d'encodage et d'image spécifié.

`altova:save-chart-image` `($chart, $filename, $width, $height) en tant que empty()` ***(Windows uniquement)***

alors que

- `$chart` est l'item d'extension de graphique obtenu avec la fonction `altova:create-chart`
- `$filename` est le nom du fichier et le chemin sous lequel l'image de graphique doit être enregistré
- `$width` et `$height` doivent être spécifiés avec une unité de longueur

La fonction enregistre l'image du graphique au fichier spécifique dans `$filename`. En alternative à cette fonction, vous pourriez aussi utiliser la fonction `xsl:result-document` avec `encoding="x-base64tobinary"`, où le contenu image-data est obtenu soit via la fonction `generate-chart-image()` ou la fonction `chart()`.

`altova:save-chart-image` `($chart, $filename, $width, $height, $imagetype) as empty()` ***(Windows only)***

alors que

- `$chart` est l'item d'extension de graphique obtenu avec la fonction `altova:create-chart`
- `$filename` est le nom du fichier et le chemin sous lequel l'image de graphique doit être enregistré
- `$width` et `$height` doivent être spécifiés avec une unité de longueur
- `$imagetype` peut être un des formats d'image suivants : `png`, `gif`, `bmp`, `jpg`, `jpeg`. Veuillez noter que `gif` n'est pas pris en charge pour les produits de serveur. *Voir également la note en haut de la page*.

La fonction enregistre l'image de graphique sous le fichier spécifié dans `$filename` dans le format d'image spécifié. En alternative à cette fonction, vous pourriez aussi utiliser la fonction `xsl:result-document` avec `encoding="x-base64tobinary"`, où le contenu image-data est obtenu soit via la fonction `generate-chart-image()` ou la fonction `chart()`.

Fonctions de création de graphiques

Les fonctions suivantes sont utilisées pour créer des graphiques.

`altova:create-chart``($chart-config, $chart-data-series*) en tant que chart extension item`

alors que

- `$chart-config` est l'item d'extension chart-config obtenu avec la fonction `altova:create-chart-config` ou via la fonction `altova:create-chart-config-from-xml`
- `$chart-data-series` est l'item d'extension chart-data-series obtenu avec la fonction `altova:create-chart-data-series` ou la fonction `altova:create-chart-data-series-from-rows`

La fonction retourne un item d'extension chart, qui est créé depuis les données fournies par le biais des arguments.

`altova:chart``($chart-config, $chart-data-series*) en tant que chart extension item`

where

- `$chart-config` est l'item d'extension chart-config. Il s'agit d'une série non ordonnée de quatre clés : paires de valeur, où les quatre clés sont `"width"`, `"height"`, "title", et "kind". Les valeurs de `width` et `height` sont des entiers et spécifient la largeur et la hauteur du graphique en pixels. La valeur de `kind` est une de : `Pie, Pie3d, BarChart, BarChart3d, BarChart3dGrouped, LineChart, ValueLineChart, RoundGauge, BarGauge`.
- `$chart-data-series` forme un array de taille 3, où chaque array définit une série de données de graphique. Chaque array est composé du : (i) nom de la série de données, (ii) des valeurs de l'axe X, (iii) des valeurs de l'axe Y. Plusieurs séries de données peuvent être soumises ; dans l'exemple ci-dessous les deux arrays donnent respectivement des données pour les températures mensuelles minimum et maximum.

La fonction retourne un item type `xs:base64Binary` qui contient une image graphique. L'image est créée depuis des données fournies par le biais des arguments de la fonction. Puisque la fonction utilise les arrays et cartes, veuillez noter que qu'elle peut être utilisée uniquement dans XPath 3.1, XQuery 3.1 ou XSLT 3.0.

Exemple :
```
altova:chart( map{'width':800, 'height':600, "kind":"LineChart", "title":"Monthly Temperatures"}, (['Min', $temps/Month, $temps/Month/@min], ['Max', $temps/Month, $temps/Month/@max]) )
```

altova:create-chart-config($type-name, $title) en tant qu'item d'extension chart-config

alors que

- $type-name spécifie le type de graphique à créer : Pie, Pie3d, BarChart, BarChart3d, BarChart3dGrouped, LineChart, ValueLineChart, RoundGauge, BarGauge
- $title est le nom du graphique

La fonction retourne un item d'extension chart-config contenant les informations de configuration du graphique.

altova:create-chart-config-from-xml($xml-struct) en tant qu'item d'extension chart-config

alors que

- $xml-struct est la structure XML contenant l'information de la configuration du graphique

La fonction retourne un item d'extension chart-config contenant les informations de configuration du graphique. Cette information est forunie dans un fragment de données XML 489.

altova:create-chart-data-series($series-name?, $x-values*, $y-values*) en tant qu'item d'extension chart-data-series

alors que

- $series-name spécifie le nom de la série
- $x-values donne la liste des valeurs de l'axe X
- $y-values donne la liste des valeurs de l'axe Y

La fonction retourne un item d'extension chart-data-series contenant les données pour la construction du graphique : c'est à dire, les noms de la série et les données des Axes.

altova:create-chart-data-row(x, y1, y2, y3, ...) en tant qu'item d'extension chart-data-x-*N*y-row

alors que

- x est la valeur de la colonne de l'axe X de la rangée des données de graphique
- yN sont les valeurs des colonnes de l'axe Y

La fonction retourne un item d'extension chart-data-x-Ny-row qui contient les données pour la colonne de l'axe X et les colonnes de l'axe Y d'une seule série.

altova:create-chart-data-series-from-rows($series-names as xs:string*, $row*) en tant qu'item d'extension chart-data-series

alors que

- `$series-name` est le nom de la série à créer
- `$row` est l'item d'extension chart-data-x-Ny-row qui doit être créé en tant qu'une série

La fonction retourne un item d'extension chart-data-series qui contient les données pour l'axe X et l'axe Y de la série.

altova:create-chart-layer(`$chart-config, $chart-data-series*`) en tant qu'item d'extension `chart-layer`

alors que

- `$chart-config` est l'item d'extension chart-config obtenu avec la fonction `altova:create-chart-config` ou via la fonction `altova:create-chart-config-from-xml`
- `$chart-data-series` est l'item d'extension chart-data-series obtenu avec la fonction `altova:create-chart-data-series` ou la fonction `altova:create-chart-data-series-from-rows`

La fonction retourne un item d'extension chart-layer qui contient des données chart-layer.

altova:create-multi-layer-chart(`$chart-config, $chart-data-series*, $chart-layer*`)

alors que

- `$chart-config` est l'item d'extension chart-config obtenu avec la fonction `altova:create-chart-config` or ou via la fonction `altova:create-chart-config-from-xml`
- `$chart-data-series` est l'item d'extension chart-data-series obtenu avec la fonction `altova:create-chart-data-series` ou la fonction `altova:create-chart-data-series-from-rows`
- `$chart-layer` est l'item d'extension chart-layer obtenu avec la fonction `altova:create-chart-layer`

La fonction retourne un item multi-layer-chart.

altova:create-multi-layer-chart(`$chart-config, $chart-data-series*, $chart-layer*, xs:boolean $mergecategoryvalues`)

alors que

- `$chart-config` est l'item d'extension chart-config obtenu avec la fonction `altova:create-chart-config` ou via la fonction `altova:create-chart-config-from-xml`
- `$chart-data-series` est l'item d'extension chart-data-series obtenu avec la fonction `altova:create-chart-data-series` ou la fonction `altova:create-chart-data-series-from-rows`
- `$chart-layer` est l'item d'extension chart-layer obtenu avec la fonction `altova:create-chart-layer`
- `$mergecategoryvalues` fusionne les valeurs de plusieurs séries de données si `true`, ne fusionne pas si `false`

La fonction retourne un item multi-layer-chart.

11.2.1.10.1 Structure XML des données de graphique

Ci-dessous, vous trouverez la structure XML des données de graphique, telle qu'elle pourrait s'afficher pour les Fonctions d'extension Altova pour les graphiques 484. Cela affecte l'apparence du graphique spécifique. Tous les éléments ne sont pas utilisés pour tous les types de graphiques, par ex. l'élément `<Pie>` est ignoré pour les graphiques à barres.

Note : Les fonctions de graphique sont uniquement prises en charge dans les **Édition Enterprise et Server** des produits Altova.

```
<chart-config>
        <General
        SettingsVersion="1"   must be provided
        ChartKind="BarChart"   Pie, Pie3d, BarChart, StackedBarChart, BarChart3d, BarChart3dGrouped,
LineChart, ValueLineChart, AreaChart, StackedAreaChart, RoundGauge, BarGauge, CandleStick
        BKColor="#ffffff"   Color
        BKColorGradientEnd="#ffffff"   Color. In case of a gradient, BKColor and BKColorGradientEnd
define the gradient's colors
        BKMode="#ffffff"   Solid, HorzGradient, VertGradient
        BKFile="Path+Filename"   String. If file exists, its content is drawn over the background.
        BKFileMode="Stretch"   Stretch, ZoomToFit, Center, Tile
        ShowBorder="1"   Bool
        PlotBorderColor="#000000"   Color
        PlotBKColor="#ffffff"   Color
        Title=""   String
        ShowLegend="1"   Bool
        OutsideMargin="3.%"   PercentOrPixel
        TitleToPlotMargin="3.%"   PercentOrPixel
        LegendToPlotMargin="3.%"   PercentOrPixel
        Orientation="vert"   Enumeration: possible values are: vert, horz
        >
        <TitleFont
           Color="#000000"   Color
           Name="Tahoma"   String
           Bold="1"   Bool
           Italic="0"   Bool
           Underline="0"   Bool
           MinFontHeight="10.pt"   FontSize (only pt values)
           Size="8.%"   FontSize />
        <LegendFont
           Color="#000000"
           Name="Tahoma"
           Bold="0"
           Italic="0"
           Underline="0"
           MinFontHeight="10.pt"
           Size="3.5%" />
        <AxisLabelFont
           Color="#000000"
```

```
        Name="Tahoma"
        Bold="1"
        Italic="0"
        Underline="0"
        MinFontHeight="10.pt"
        Size="5.%" />
      </General>

      <Line
    ConnectionShapeSize="1.%"  PercentOrPixel
    DrawFilledConnectionShapes="1"  Bool
    DrawOutlineConnectionShapes="0"  Bool
    DrawSlashConnectionShapes="0"  Bool
    DrawBackslashConnectionShapes="0"  Bool
      />

      <Bar
    ShowShadow="1"  Bool
    ShadowColor="#a0a0a0"  Color
    OutlineColor="#000000"  Color
    ShowOutline="1"  Bool
      />

      <Area
    Transparency="0"  UINT ( 0-255 ) 255 is fully transparent, 0 is opaque
    OutlineColor="#000000"  Color
    ShowOutline="1"  Bool
      />

      <CandleStick
    FillHighClose="0"  Bool. If 0, the body is left empty. If 1, FillColorHighClose is used for the candle
body
    FillColorHighClose="#ffffff"  Color. For the candle body when close > open
    FillHighOpenWithSeriesColor="1"  Bool. If true, the series color is used to fill the candlebody when
open > close
    FillColorHighOpen="#000000"  Color. For the candle body when open > close and
FillHighOpenWithSeriesColor is false
      />

  <Colors  User-defined color scheme: By default this element is empty except for the style and has no
Color attributes
    UseSubsequentColors ="1"  Boolean. If 0, then color in overlay is used. If 1, then subsequent colors
from previous chart layer is used
    Style="User"  Possible values are: "Default", "Grayscale", "Colorful", "Pastel", "User"
    Colors="#52aca0"  Color: only added for user defined color set
    Colors1="#d3c15d"  Color: only added for user defined color set
    Colors2="#8971d8"  Color: only added for user defined color set
    ...
    ColorsN=""  Up to ten colors are allowed in a set: from Colors to Colors9
      </Colors>

      <Pie
    ShowLabels="1"  Bool
```



```
OutlineColor="#404040"  Color
ShowOutline="1"  Bool
StartAngle="0."  Double
Clockwise="1"  Bool
Draw2dHighlights="1"  Bool
Transparency="0"  Int (0 to 255: 0 is opaque, 255 is fully transparent)
DropShadowColor="#c0c0c0"  Color
DropShadowSize="5.%"  PercentOrPixel
PieHeight="10.%"  PercentOrPixel. Pixel values might be different in the result because of 3d tilting
Tilt="40.0"  Double (10 to 90: The 3d tilt in degrees of a 3d pie)
ShowDropShadow="1"  Bool
ChartToLabelMargin="10.%"  PercentOrPixel
AddValueToLabel="0"  Bool
AddPercentToLabel="0"  Bool
AddPercentToLabels_DecimalDigits="0"  UINT ( 0 – 2 )
>
<LabelFont
   Color="#000000"
   Name="Arial"
   Bold="0"
   Italic="0"
   Underline="0"
   MinFontHeight="10.pt"
   Size="4.%" />
 </Pie>

 <XY>
<XAxis  Axis
   AutoRange="1"  Bool
   AutoRangeIncludesZero="1"  Bool
   RangeFrom="0."  Double: manual range
   RangeTill="1."  Double : manual range
   LabelToAxisMargin="3.%"  PercentOrPixel
   AxisLabel=""  String
   AxisColor="#000000"  Color
   AxisGridColor="#e6e6e6"  Color
   ShowGrid="1"  Bool
   UseAutoTick="1"  Bool
   ManualTickInterval="1."  Double
   AxisToChartMargin="0.px"  PercentOrPixel
   TickSize="3.px"  PercentOrPixel
   ShowTicks="1"  Bool
   ShowValues="1"  Bool
   AxisPosition="LeftOrBottom"  Enums: "LeftOrBottom", "RightOrTop", "AtValue"
   AxisPositionAtValue = "0"  Double
   >
   <ValueFont
      Color="#000000"
      Name="Tahoma"
      Bold="0"
      Italic="0"
      Underline="0"
      MinFontHeight="10.pt"
      Size="3.%"      />
```

```
</XAxis>
<YAxis  Axis (same as for XAxis)
   AutoRange="1"
   AutoRangeIncludesZero="1"
   RangeFrom="0."
   RangeTill="1."
   LabelToAxisMargin="3.%"
   AxisLabel=""
   AxisColor="#000000"
   AxisGridColor="#e6e6e6"
   ShowGrid="1"
   UseAutoTick="1"
   ManualTickInterval="1."
   AxisToChartMargin="0.px"
   TickSize="3.px"
   ShowTicks="1"  Bool
   ShowValues="1"  Bool
   AxisPosition="LeftOrBottom"  Enums: "LeftOrBottom", "RightOrTop", "AtValue"
   AxisPositionAtValue = "0"  Double
   >
   <ValueFont
      Color="#000000"
      Name="Tahoma"
      Bold="0"
      Italic="0"
      Underline="0"
      MinFontHeight="10.pt"
      Size="3.%"/>
</YAxis>
 </XY>

 <XY3d
AxisAutoSize="1"  Bool: If false, XSize and YSize define the aspect ration of x and y axis. If true,
aspect ratio is equal to chart window
XSize="100.%"  PercentOrPixel. Pixel values might be different in the result because of 3d tilting and
zooming to fit chart
YSize="100.%"  PercentOrPixel. Pixel values might be different in the result because of 3d tilting and
zooming to fit chart
SeriesMargin="30.%"  PercentOrPixel. Pixel values might be different in the result because of 3d
tilting and zooming to fit chart
Tilt="20."  Double. -90 to +90 degrees
Rot="20."  Double. -359 to +359 degrees
FoV="50.">  Double. Field of view: 1-120 degree
>
<ZAxis
   AutoRange="1"
   AutoRangeIncludesZero="1"
   RangeFrom="0."
   RangeTill="1."
   LabelToAxisMargin="3.%"
   AxisLabel=""
   AxisColor="#000000"
   AxisGridColor="#e6e6e6"
   ShowGrid="1"
   UseAutoTick="1"
```



```
   ManualTickInterval="1."
   AxisToChartMargin="0.px"
   TickSize="3.px" >
   <ValueFont
      Color="#000000"
      Name="Tahoma"
      Bold="0"
      Italic="0"
      Underline="0"
      MinFontHeight="10.pt"
      Size="3.%"/>
</ZAxis>
 </XY3d>

 <Gauge
MinVal="0."  Double
MaxVal="100."  Double
MinAngle="225"  UINT: -359-359
SweepAngle="270"  UINT: 1-359
BorderToTick="1.%"  PercentOrPixel
MajorTickWidth="3.px"  PercentOrPixel
MajorTickLength="4.%"  PercentOrPixel
MinorTickWidth="1.px"  PercentOrPixel
MinorTickLength="3.%"  PercentOrPixel
BorderColor="#a0a0a0"  Color
FillColor="#303535"  Color
MajorTickColor="#a0c0b0"  Color
MinorTickColor="#a0c0b0"  Color
BorderWidth="2.%"  PercentOrPixel
NeedleBaseWidth="1.5%"  PercentOrPixel
NeedleBaseRadius="5.%"  PercentOrPixel
NeedleColor="#f00000"  Color
NeedleBaseColor="#141414"  Color
TickToTickValueMargin="5.%"  PercentOrPixel
MajorTickStep="10."  Double
MinorTickStep="5."  Double
RoundGaugeBorderToColorRange="0.%"  PercentOrPixel
RoundGaugeColorRangeWidth ="6.%"  PercentOrPixel
BarGaugeRadius="5.%"  PercentOrPixel
BarGaugeMaxHeight="20.%"  PercentOrPixel
RoundGaugeNeedleLength="45.%"  PercentOrPixel
BarGaugeNeedleLength="3.%"  PercentOrPixel
>
<TicksFont
   Color="#a0c0b0"
   Name="Tahoma"
   Bold="0"
   Italic="0"
   Underline="0"
   MinFontHeight="10.pt"
   Size="4.%"
/>
<ColorRanges>  User-defined color ranges. By default empty with no child element entries
   <Entry
```

```
            From="50. "   Double
            FillWithColor="1"   Bool
            Color="#00ff00"   Color
         />
         <Entry
            From="50.0"
            FillWithColor="1"
            Color="#ff0000"
         />
         ...
      </ColorRanges>
       </Gauge>
</chart-config>
```

11.2.1.10.2 Exemple : fonctions de graphique

Le document XSLT présenté ci-dessous à titre d'exemple montre comment les fonctions d'extension Altova pour les graphiques 484 peuvent être utilisées. Ci-dessous, vous trouverez un document XML et une capture d'écran de l'image de sortie générée lorsque le document XML est traité avec le document XSLT en utilisant XSLT 2.0 ou 3.0 Engine.

Note: Les fonctions de graphique sont uniquement prises en charge dans les **Éditions Enterprise et Server** des produits Altova.

Note: Pour plus d'informations concernant la création des tables de données de graphique, voir la documentation des produits Altova XMLSpy et StyleVision.

Document XSLT

Ce document XSLT (*liste ci-dessous*) utilise fonctions d'extension de graphique Altova pour générer un camembert. Elles peuvent être utilisées pour traiter le document XML recensé ci-dessous.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:altovaext="http://www.altova.com/xslt-extensions"
    exclude-result-prefixes="#all">
    <xsl:output version="4.0" method="html" indent="yes" encoding="UTF-8"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>
                    <xsl:text>HTML Page with Embedded Chart</xsl:text>
                </title>
            </head>
            <body>
                <xsl:for-each select="/Data/Region[1]">
                    <xsl:variable name="extChartConfig" as="item()*">
                        <xsl:variable name="ext-chart-settings" as="item()*">
```

```
                        <chart-config>
                            <General
                                SettingsVersion="1"
                                ChartKind="Pie3d"
                                BKColor="#ffffff"
                                ShowBorder="1"
                                PlotBorderColor="#000000"
                                PlotBKColor="#ffffff"
                                Title="{@id}"
                                ShowLegend="1"
                                OutsideMargin="3.2%"
                                TitleToPlotMargin="3.%"
                                LegendToPlotMargin="6.%"
                                >
                                <TitleFont
                                    Color="#023d7d"
                                    Name="Tahoma"
                                    Bold="1"
                                    Italic="0"
                                    Underline="0"
                                    MinFontHeight="10.pt"
                                    Size="8.%" />
                            </General>
                        </chart-config>
                    </xsl:variable>
                    <xsl:sequence select="altovaext:create-chart-config-from-xml( $ext-
chart-settings )"/>
                </xsl:variable>
                <xsl:variable name="chartDataSeries" as="item()*">
                    <xsl:variable name="chartDataRows" as="item()*">
                        <xsl:for-each select="(Year)">
                            <xsl:sequence select="altovaext:create-chart-data-row( (@id),
( .) )"/>
                        </xsl:for-each>
                    </xsl:variable>
                    <xsl:variable name="chartDataSeriesNames" as="xs:string*"
select=" ( (&quot;Series 1&quot;), &apos;&apos; )[1]"/>
                    <xsl:sequence
                        select="altovaext:create-chart-data-series-from-
rows( $chartDataSeriesNames, $chartDataRows)"/>
                </xsl:variable>
                <xsl:variable name="ChartObj" select="altovaext:create-
chart( $extChartConfig, ( $chartDataSeries), false() )"/>
                <xsl:variable name="sChartFileName" select="'mychart1.png'"/>
                <img src="{$sChartFileName, altovaext:save-chart-image( $ChartObj,
$sChartFileName, 400, 400 ) }"/>
            </xsl:for-each>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

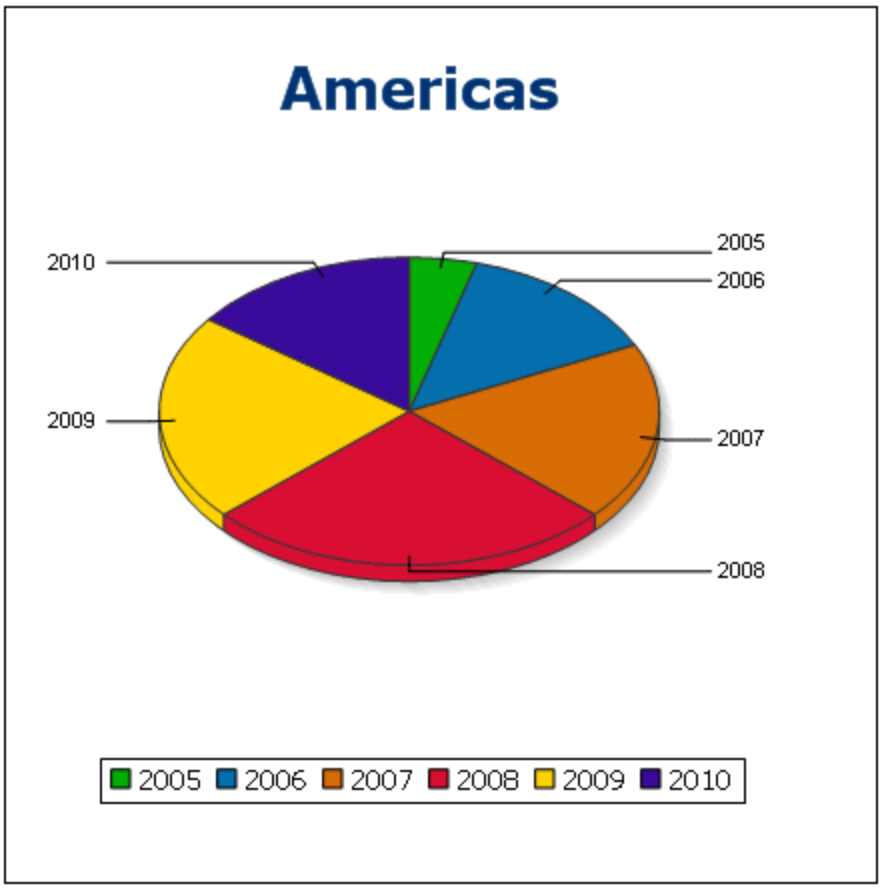
Document XML

Ce document XML peut être traité avec le document XSLT ci-dessus. Les données se trouvant dans le document XML sont utilisées pour générer le camembert affiché dans la capture d'écran ci-dessous.

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
   <ChartType>Pie Chart 2D</ChartType>
   <Region id="Americas">
      <Year id="2005">30000</Year>
      <Year id="2006">90000</Year>
      <Year id="2007">120000</Year>
      <Year id="2008">180000</Year>
      <Year id="2009">140000</Year>
      <Year id="2010">100000</Year>
   </Region>
   <Region id="Europe">
      <Year id="2005">50000</Year>
      <Year id="2006">60000</Year>
      <Year id="2007">80000</Year>
      <Year id="2008">100000</Year>
      <Year id="2009">95000</Year>
      <Year id="2010">80000</Year>
   </Region>
   <Region id="Asia">
      <Year id="2005">10000</Year>
      <Year id="2006">25000</Year>
      <Year id="2007">70000</Year>
      <Year id="2008">110000</Year>
      <Year id="2009">125000</Year>
      <Year id="2010">150000</Year>
   </Region>
</Data>
```

Image de sortie

Le camembert affiché ci-dessous est généré lorsque le document XML recensé ci-dessus est traité avec le document XSLT.



11.2.1.11 Fonctions de code-barres

Le moteur XSLT utilise des librairies Java tierces pour créer des codes-barres. Ci-dessous, vous trouverez les classes et les méthodes publiques utilisées. Les classes sont emballées dans `AltovaBarcodeExtension.jar`, qui se trouve dans le dossier *<ProgramFilesFolder>*`\Altova\Common2024\jar`.

Les librairies Java utilisées se trouvent dans les sous-dossiers du dossier *<ProgramFilesFolder>*`\Altova\Common2024\jar`:

- `barcode4j\barcode4j.jar` (Site web : http://barcode4j.sourceforge.net/)
- `zxing\core.jar` (Site web : http://code.google.com/p/zxing/)

Les fichiers de licence sont également situés dans les dossiers respectifs.

Machine virtuelle Java

Afin de pouvoir utiliser les fonctions de code-barres, une machine virtuelle Java doit être disponible sur votre appareil. Pour trouver le chemin vers la machine voir l'encadré ci-dessous.

- Si vous utilisez un produit de desktop Altova, les tentatives d'application Altova pour détecter automatiquement le chemin vers la machine virtuelle Java, en lisant (dans cet ordre) : (i) le registre Windows, et (ii) la variable d'environnement **JAVA_HOME**. Vous pouvez aussi ajouter un chemin personnalisé dans le dialogue Options de l'application ; cette entrée prendra la priorité sur tout autre chemin Java VM détecté automatiquement.
- Si vous exécutez un produit serveur Altova sur un appareil Windows, le chemin vers la machine virtuelle sera lu tout d'abord depuis le registre Windows ; si cela échoue, la variable d'environnement **JAVA_HOME** sera utilisée.
- Si vous exécutez un produit de serveur Altova sur un appareil Linux ou macOS, veuillez vous assurer que la variable d'environnement **JAVA_HOME** est définie correctement et que la bibliothèque des appareils virtuels Java (sur Windows, le fichier **jvm.dll)** puisse être située dans le répertoire **\bin\server** ou **\bin\client**.

Le package **com.altova.extensions.barcode**

Le package, `com.altova.extensions.barcode`, est utilisé pour générer la plupart des types de code-barres.

Les classes suivantes sont utilisées :

```
public class BarcodeWrapper
   static BarcodeWrapper newInstance( String name, String msg, int dpi, int orientation,
BarcodePropertyWrapper[] arrProperties )
   double getHeightPlusQuiet()
   double getWidthPlusQuiet()
   org.w3c.dom.Document generateBarcodeSVG()
   byte[] generateBarcodePNG()
   String generateBarcodePngAsHexString()
```

`public class` **BarcodePropertyWrapper** *Used to store the barcode properties that will be dynamically set later*

```
    BarcodePropertyWrapper( String methodName, String propertyValue )
   BarcodePropertyWrapper( String methodName, Integer propertyValue )
   BarcodePropertyWrapper( String methodName, Double propertyValue )
   BarcodePropertyWrapper( String methodName, Boolean propertyValue )
   BarcodePropertyWrapper( String methodName, Character propertyValue )
   String getMethodName()
   Object getPropertyValue()
```

`public class` **AltovaBarcodeClassResolver** *Enregistre la classe* `com.altova.extensions.barcode.proxy.zxing.QRCodeBean` *pour le* `qrcode` *bean, en plus des classes enregistrées par le* `org.krysalis.barcode4j.DefaultBarcodeClassResolver`.

Le package **com.altova.extensions.barcode.proxy.zxing**

Le package, `com.altova.extensions.barcode.proxy.zxing`, est utilisé pour générer le code-barres de type QRCode.

Les classes suivantes sont utilisées :

`classe` **QRCodeBean**

- *Élargit* `org.krysalis.barcode4j.impl.AbstractBarcodeBean`
- *Crée une* interface `AbstractBarcodeBean pour com.google.zxing.qrcode.encoder`

```
   void generateBarcode(CanvasProvider canvasImp, String msg)
   void setQRErrorCorrectionLevel(QRCodeErrorCorrectionLevel level)
   BarcodeDimension calcDimensions(String msg)
   double getVerticalQuietZone()
   double getBarWidth()
```

classe **`QRCodeErrorCorrectionLevel`** *Erreur niveau de correction pour le* `QRCode`

```
   static QRCodeErrorCorrectionLevel byName(String name)
   "L" = ~7% correction
   "M" = ~15% correction
   "H" = ~25% correction
   "Q" = ~30% correction
```

Exemple XSLT

Ci-dessous, vous trouverez un exemple XSLT pour démontrer comment les fonctions de code-barres sont utilisées dans une feuille de style XSLT.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:altova="http://www.altova.com"
   xmlns:altovaext="http://www.altova.com/xslt-extensions"
   xmlns:altovaext-barcode="java:com.altova.extensions.barcode.BarcodeWrapper"
   xmlns:altovaext-barcode-
property="java:com.altova.extensions.barcode.BarcodePropertyWrapper">
   <xsl:output method="html" encoding="UTF-8" indent="yes"/>
   <xsl:template match="/">
      <html>
         <head><title/></head>
         <body>
            <img alt="barcode" src="{altovaext:get-temp-folder()}barcode.png"/>
         </body>
      </html>
      <xsl:result-document
         href="{altovaext:get-temp-folder()}barcode.png"
         method="text" encoding="base64tobinary" >
         <xsl:variable name="barcodeObject"
            select="altovaext-barcode:newInstance(&apos;Code39&apos;,string(&apos;some
value&apos;),
            96,0, (altovaext-barcode-property:new( &apos;setModuleWidth&apos;, 25.4 div 96
* 2 ) ) )"/>
         <xsl:value-of select="xs:base64Binary(xs:hexBinary(string(altovaext-
barcode:generateBarcodePngAsHexString($barcodeObject)) ))"/>
      </xsl:result-document>
   </xsl:template>
</xsl:stylesheet>
```

11.2.2 Fonctions d'extension diverses

Il existe plusieurs types de fonctions prêtes à l'utilisation dans les langages de programmation comme Java et C# qui ne sont pas disponibles en tant que fonctions XQuery/XPath ou en tant que fonctions XSLT. Un bon exemple sont les fonctions mathématiques disponibles en Java, comme `sin()` et `cos()`. Si ces fonctions étaient disponibles aux designers des feuilles de style XSLT et des requêtes XQuery, cela augmenterait le champ d'application des feuilles de style et des requêtes et faciliterait considérablement les tâches des créateurs de feuilles de style. Les moteurs XSLT et XQuery utilisés dans un grand nombre de produits Altova prennent en charge l'utilisation des fonctions d'extension dans Java et .NET, et pour les scripts MSXSL pour XSLT[500]. Cette section décrit comment utiliser les fonctions d'extension et les scripts MSXSL dans vos feuilles de scripts XSLT et les documents XQuery. Les fonctions d'extension disponibles sont organisées dans les sections suivantes :

- Fonctions d'extension Java
- Fonctions d'extension .NET
- Scripts MSXSL pour XSLT[500]

Les deux problèmes principaux considérés pour les descriptions sont : (i) comment sont appelées les fonctions dans les librairies respectives ; et (ii) quelles sont les règles à suivre pour la conversion d'arguments dans un appel de fonction pour obtenir le format d'entrée requis de la fonction, et quelles sont les règles à suivre pour la conversion de retour (résultat de la fonction à l'objet de données XSLT/XQuery).

Exigences

Pour une prise en charge des fonctions d'extension, un Java Runtime Environment (pour l'accès aux fonctions Java) et le cadre de travail .NET Framework 2.0 (minimum, pour l'accès aux fonctions .NET) doit être installé sur la machine qui effectue la transformation XSLT ou l'exécution XQuery, ou doit être accessible pour les transformations.

11.2.2.1 Scripts MSXSL pour XSLT

L'élément `<msxsl:script>` contient des fonctions définies par l'utilisateur et des variables qui peuvent être appelées depuis des expressions XPath dans la feuille de style XSLT. Le `<msxsl:script>` et un élément de niveau supérieur, c'est à dire, qu'il doit être un élément enfant de `<xsl:stylesheet>` ou `<xsl:transform>`.

L'élément `<msxsl:script>` doit être dans l'espace de nom `urn:schemas-microsoft-com:xslt` (*voir exemple ci-dessus*).

Langage de script et espace de nom

Le langage de script utilisé dans le bloc est spécifié dans l'attribut `language` de l'élément `<msxsl:script>` et l'espace de nom à utiliser pour les appels de fonction depuis les expressions XPath est identifié avec l'attribut `implements-prefix` (*voir ci-dessous*).

```
<msxsl:script language="scripting-language" implements-prefix="user-namespace-prefix">

   function-1 or variable-1
   ...
```

```
   function-n or variable-n

</msxsl:script>
```

L'élément `<msxsl:script>` interagit avec le Windows Scripting Runtime, donc seuls des langages installés sur votre machine peuvent être utilisés dans l'élément `<msxsl:script>`. **La plate-forme .NET Framework 2.0 ou plus récente doit être installée pour pouvoir utiliser les scripts MSXSL.** Par conséquent, les langages de script .NET peuvent être utilisés dans l'élément `<msxsl:script>`.

L'attribut de langage doit accepter les mêmes valeurs que l'attribut `language` dans l'élément HTML `<script>`. Si l'attribut de langage n'est pas spécifié, alors Microsoft JScript est assumé par défaut.

L'attribut `implements-prefix` prend une valeur qui est un préfixe d'un nom d'espace in-scope déclaré. Cet espace de nom sera généralement un espace de nom d'utilisateur qui a été réservé pour une librairie de fonction. Toutes les fonctions et les variables définies dans l'élément `<msxsl:script>` se trouveront dans l'espace de nom identifié par le préfixe spécifié dans l'attribut `implements-prefix`. Lorsqu'une fonction est appelée depuis une expression XPath, le nom de la fonction entièrement qualifié doit se trouver dans le même espace de nom que la définition de la fonction.

Exemple

Voici un exemple d'une feuille de stype XSLT complète qui utilise une fonction définie dans un élément `<msxsl:script>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:msxsl="urn:schemas-microsoft-com:xslt"
   xmlns:user="http://mycompany.com/mynamespace">

  <msxsl:script language="VBScript" implements-prefix="user">
    <![CDATA[
    ' Input: A currency value: the wholesale price
    ' Returns: The retail price: the input value plus 20% margin,
    ' rounded to the nearest cent
    dim a as integer  = 13
    Function AddMargin(WholesalePrice) as integer
      AddMargin = WholesalePrice * 1.2 + a
    End Function
  ]]>
  </msxsl:script>

  <xsl:template match="/">
    <html>
      <body>
        <p>
          <b>Total Retail Price =
            $<xsl:value-of select="user:AddMargin(50)"/>
          </b>
          <br/>
          <b>Total Wholesale Price =
            $<xsl:value-of select="50"/>
```

```
          </b>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Types de données

Les valeurs des paramètres passés dans et hors du bloc de script sont limitées aux types de données XPath. Cette restriction ne s'applique pas aux données passées parmi les fonctions et les variables dans le bloc du script.

Assemblages

Un assemblage peut être importé dans le script en utilisant l'élément **msxsl:assembly**. L'assemblage est identifié par le biais d'un nom ou d'un URI. L'assemblage est importé lors de la compilation de la feuille de style. Voici une simple représentation de la manière d'utiliser l'élément `msxsl:assembly`.

```
<msxsl:script>
   <msxsl:assembly name="myAssembly.assemblyName" />
   <msxsl:assembly href="pathToAssembly" />

   ...

</msxsl:script>
```

Le nom d'assemblage peut être un nom complet comme :

```
"system.Math, Version=3.1.4500.1 Culture=neutral PublicKeyToken=a46b3f648229c514"
```

ou un nom court comme `"myAssembly.Draw"`.

Espaces de nom

Les espaces de nom peuvent être déclarés avec l'élément **msxsl:using**. Cela permet aux classes d'assemblage d'être écrits dans le script sans leurs espaces de nom, ce qui vous épargne un gros travail de saisie. Voici comment l'élément `msxsl:using` est utilisé de manière à déclarer des espaces de nom.

```
<msxsl:script>
   <msxsl:using namespace="myAssemblyNS.NamespaceName" />

   ...

</msxsl:script>
```

La valeur de l'attribut `namespace` est le nom de l'espace de nom.

Index

.

<

A

B

C

D

E

F

G

H

I

L

M

P

Q

R

S

T

V

W

X