User Manual

Decos XML Import Service 3.2.10

Contents

 [1](#_Toc506553754)

[1. Introduction 3](#_Toc506553755)

[2. Configuration:Crawlconfig.xml 4](#_Toc506553756)

[3. Configuration: DecosXMLCrawler.exe.config 7](#_Toc506553757)

[4. Imported notes about validation: 9](#_Toc506553758)

## Introduction

This document describes configuration required for XML import service on the installation server.

## Configuration:Crawlconfig.xml

This file is located at installed directory of service.

Sample configuration 1

|  |
| --- |
| <CrawlConfig > <Crawl SchemaFile="" Directory="E:\testcrawler\input"  BackupDirectory="E:\testcrawler\backup"  FailedDirectory="E:\testcrawler\failed"  CSV="1" CsvSplitCount="120" Delimiter=","  Time="12:30" Sub="1" DisableFileSysWatch="1" WaitInterval="5000" >  <Template>  <Default>C:\Decos\Apps\Decos Document XML Import\XSL\afs.xsl</Default> </Template> </Crawl>  <Crawl Directory="E:\testcrawler\input"  BackupDirectory="E:\testcrawler\backup"  FailedDirectory="E:\testcrawler\failed" Sub="1" > <Template>  <Default>C:\Decos\Apps\Decos Document XML Import\XSL\Test.xsl</Default> </Template> </Crawl> </CrawlConfig> |

Sample configuration 2

|  |
| --- |
|  |

**Descriptions:**

**DisableFileSysWatch –** The attribute is a non-mandatory setting. If value is set to ‘1’ the file system watcher will be disabled. If the setting is missing or set to ‘0’ than File System watcher will be enabled.

**SchemaFile** – This attribute can be used for validation of input xml file at the first step.

The full path of schema file (xsd) is required for this attribute. This is not mandatory field.

**Directory** – Directory path of the input folder, where the source xml file will be kept.

**Sub- This node is optional** .if value set ‘1’ the file watcher would keep watch on sub directory of directory specified in the Directory node. If not specified then the service will watch only the directory specified in Directory node.

**CSV**- In case of \*.CSV file this node must be set to ‘1’.Default watch on \*.XML file. In case of \*.XML file remove this node.

**Delimiter-This node is optional**. Set this node when processing \*.CSV file .Default separator is ‘,’ if not mentioned.

**CSVNOHEADERS**-**This is optional** node and recommended if watch is created on \*.CSV file and the end user don’t want to process Header then set CSVNOHEADERS=”true” if the end user want to process header also then set CSVNOHEADERS=”false”. In case of \*.xml no need to set this node or just remove it.

**CsvSplitCount**- **This is optional** node is used to specify the chunk size of xml nodes which are generated from CSV file. If the user specifies “100” for example then an xml file will be generated consisting of 100 nodes. By default CsvSplitCount is 100.

**Time-** Set this time node with appropriate time when need to import .If you want to set multiple times then enter the values of pipe separated values.

**BackupDirectory-**This is optional. This directory contains all the processed files in directory specified in the BackupDirectory node.

**FailedDirectory: -** This is optional. If the service fails to process the files, than the files are stored in ‘FailedDirectory’ instead of ‘BackupDirectory’. If this attribute is not present in crawlconfig, then the failed files are stored in the ‘BackupDirectory’ with extension”FAILED”.

**WaitInterval**: This is optional setting. If its configured the service delays the processing of each file for the configured timespan (in milliseconds).

**Template – This node is optional**. If you don’t want to transform your xml file then you don’t need to define this node in the configuration file. When you don’t define Template node in the configuration file then this service directly process the xml file and try to insert records from that xml file. You can define IF or DEFAULT node as child node of this Template node.

**IF** – If you want to process the xml file based on some condition at that time you can use this node. You can use this node when you want to transform the xml file based on some criteria and with different xsl file in this scenario you can use this node. If you want to add some criteria then you have to add this criterion in the TestKey attribute of this node and you have to define value of this criterion in TestValue attribute. You have to specify whole xsl file name with full path in this node.

E.g. you want to process the xml file based on the DocType (/scanJob/DocType) node value. When DocType node contains “Passport” value then you want to process that file with DocumentUnilease.xsl xsl file and if DocType node contains Vodafone value then you want to process that xml file with

Specfic.xsl xsl file which is present in passport directory. In this scenario you can use this IF node. You can specify any condition in the configuration for the xml file. If the condition is true then this service transform the xml file with the specified Xsl file otherwise it directly process that xml file.

**DEFAULT** – If you want to process the xml files with default xsl file then you can define xsl file name in the DEFAULT node. When you specified the DEFAULT node in the configuration file then this service use the default xsl file and use that xsl file to transform the xml file. .

For example if you want to watch on \*.CSV file which is separated by ‘&’ and want to read the Headers of these files and want to keep backup of these files after processing and the location of watch directory is share and there would not watch on sub directory then this we have to design CrawlConfig.xml like below

## Configuration: DecosXMLCrawler.exe.config

|  |
| --- |
|  |

1) **USEADMINCONNECT**: If value is set to 1, then DecosApp object in licensing library will be created using a database connection that will be derived from the decosadmin.ini file.

2) If **USEADMINCONNECT** value is set to **0**, then DecosApp object in licensing library will be created using a NameValueCollection like appsettings.

Licensing library will read and validate license data from decos database.

The following items can be present in this collection:

**DBTYPE:** Contains the database type to connect to. Can be one of the following: SQLSERVER, MYSQL, and ORACLE

**SERVER:** Contains the server name to connect to.

**DBNAME:** Contains the database name to open.

**USER:** Contains the username to connect with.

**PASSWORD:** Contains the password to connect with.

**NO\_OF\_CONNECTION\_TRIALS**: How many time (s) services will try to connect to Decos database. If not provided then service considers 12 as default.

**CONNECTION\_MINUTES\_INTERVAL:** How much interval in minute’s service will wait to reconnect to decos database. If not provided then service considers 5 as default.

**MANUAL\_CRAWL\_TIMEINTERVAL:** After how much time interval service should start manual crawl .Value should be in minute.

**TRASFORM\_IN\_IMPORTSERVICE**: If this setting is set TRUE, input xml files will be transformed in Xml Import Service itself. Otherwise by default it will be transformed in Decos Doc layer.

**MAILSERVER:** Name of the mail server (SMTP mail server).

**SENDERMAILID:** Mail id of the sender from whom the mail is going to be sent.

**RECIPIENTMAILIDS:** Recipient mail id‘s to whom the mail is to be sent .We can insert multiple mail id’s by semicolon ‘;’ separated.

**SMTPUSERNAME:** For network credentials SMTP user name is used.

**SMTPPASSWORD:** SMTP password.

## Imported notes about validation:

Validation at first stage on input xml:

1. When <Crawl SchemaFile=”” attribute is set then this schema file can be used to validate input xml file at the first stage.
2. If validation of input xml file fails then the input xml file will get copy to failed folder with suffix “INVALID”.

Validation before import stage:

1. When the service is installed a folder named “Schema” is created in the installed directory.
2. This folder contains two schema’s namely xis.xsd and xisnew.xsd.
3. The service supports two formats of xmls . The schemas are validated against the namespace specified in the xsl.
4. The validation is optional for old xsd (xis.xsd) but it is mandatory for the new (xisnew.xsd).
5. If validation fails with the old xsd it still tries to import in decos document. This provides backwards compatibility.
6. In case of xisnew.xsd if the validation fails, than the files are reported as ‘Failed files’ and hence moved to FailedDirectory.