23/03/2023

SERIES M/Adapter

Configuration and use

Stephane.durand@kwsoft.fr

Table Of Contents

I. Introduction

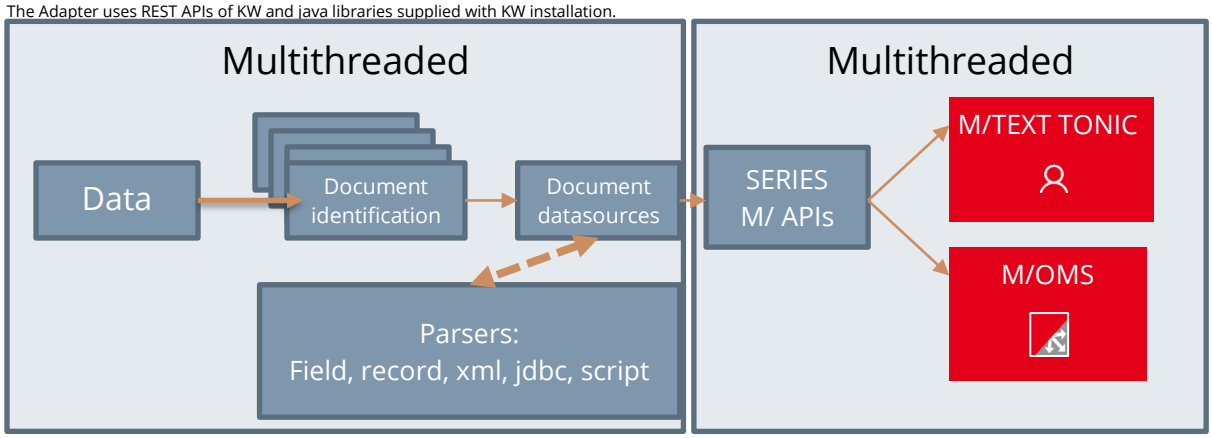
The Adapter module was created to facilitate StreamServe projects migrations.
It can reproduce the scanning directory mechanism and can process data streams such as StreamIn, FieldIn, RDI, PDF, OTF.
Beside StreamServe migration context, it can be used to process formats not natively supported by KW.

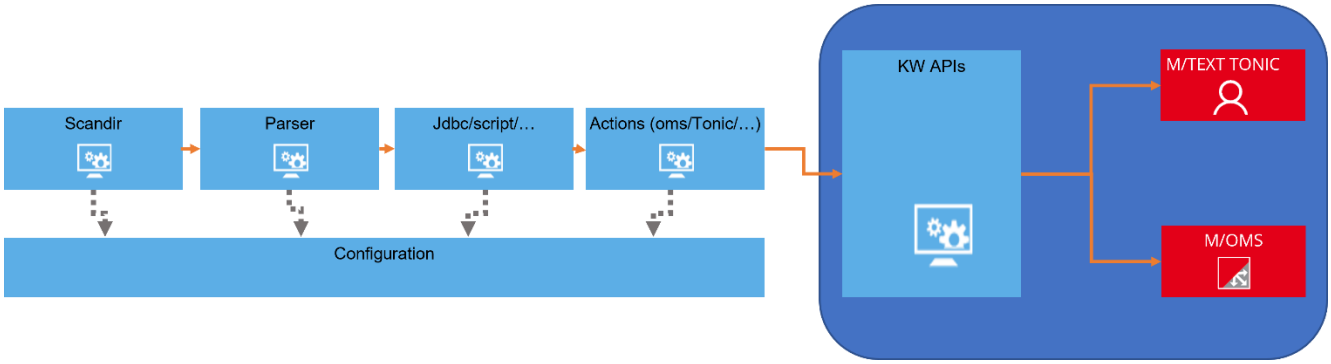
II. Principle

The Adapter is in charge to catch data streams deposited in one, or several, scanned folders.
Once taken in account, the data streams are:

- Identified
- Parsed to:
  - Split the original data stream into document unitary data streams
  - Creating a field/blocs dictionary attached to the document (this dictionary is converted into xml when submitted to KW)
- Depending on document settings in the Adapter, the documents are set to KW to :
  - Unitary PDF response
  - Print to MOMS
  - Create a Tonic document

The Adapter uses REST APIs of KW and java libraries supplied with KW installation.





III. Configuration

The Adapter uses several configuration files ordered in a strict folder structure.
Most settings support hot changes and/or synchronization with a dedicated project managed in KW Workbench. This dedicated project is deployed to the KW server like any other KW project.

A. Setting files format

Those files use the java .properties format (https://fr.wikipedia.org/wiki/.properties).
They are ISO-8859-1 encoded and the settings are stored like "KEY=Value", special characters that cannot be represented with ISO-8859-1 must be encoded as "Unicode escapes" :

```
UnicodeInputCharacter:
UnicodeEscape:
\ UnicodeMarker HexDigit HexDigit HexDigit HexDigit
UnicodeMarker:
u {u}
HexDigit:
(one of)
0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F
```

Ex: \u2122 = "™

Some special escape sequences :

Value	Escape sequence
\ (anti-slash)	\\
Tabulation	\t
form feed	\f
New line	\n
Carriage return	\r

1. Parent configuration file

Each configuration file can refer un parent configuration. When a parameter is not present in a file, it will be searched in the parent file.
Paths are always relative to the working directory of the Adapter.
Ex :
PARENT=common\\common.properties

B. Folder structure

- <Adapter root folder>
  - bak : backup for processed files.
  - dictionaries : list the ***.event** files that describe the dictionary structure (and therefore the final xml). The syntax use the StreamServe event files.
  - dsc : description files for StreamIn, FieldIn, PDFIn, SAPGOFIn, RDI, XML. These description files are here to :
    - Recognize the data stream (invoice ?, letter, ...)
    - Extract dictionary for each document from the data stream.
    - The file extensions are:
      - ***.dsc** : StreamIn, FieldIn, RDI, XML
      - ***.pdfin** : PDFIn
      - *.sapgofin : OTFIn
  - log : folder for log files
  - scanners : This folder contains the configuration files for the directory scanners. At startup all the **properties** files in this folder are considered as scanner configuration.
  - tmp : folder for tmp files.
  - xf : This folder contains the Freemarker templates (https://freemarker.apache.org/) that will be used to create the control job files that are sent with preformatted documents to M/OMS. The templates extension is **.ftl**.
  - **<main config>.properties** : At root level a main configuration file will contain global settings such as connection information, synchronization settings, etc...

C. Series/M Connection

The Adapter has to connect Series/M to submit documents or to update its configuration.
The connection settings are located in the main configuration file at root level (Ex: main.properties)
The connection parameters are :
KW_SERVER : name or ip address of KW server
KW_SERVER_PORT : listening port of KW server
SSL_ENABLED : HTTPS=true, HTTP=false
KW_USER : the user to connect Series/M server
KW_PASSORD_PLAIN : user password.
KW_PASSWORD_CRYPTED : user password crypted using the Series/M encryption tool.
Ex:

```
KW_SERVER=localhost
KW_SERVER_PORT=8080
SSL_ENABLED=false
KW_USER=mtext
KW_PASSWORD_CRYPTED= A050FB1827A63EED
```

D. Settings synchronization

The Adapter configuration, except the main configuration file, can be managed as a dedicated project in the Workbench. This project will be deployed to the Series/M server like any other Series/M project.
In this case the adapter is setup to indicate the project name it must use for its settings updates.
We can also use a folder on disk instead of Series/M server project. In both cases the Adapter will periodically check for changes and will update its configuration files.
Files deletion is not taken in account during this synchronization.
Synchronization parameters :
KW_SYNCHRO_SETTINGS_FOLDER : Root folder with which the Adapter will synchronize its settings.
KW_SYNCHRO_SETTINGS_PROJECT : Workbench project that will be used for synchronization. **KW_SYNCHRO SETTINGS_FOLDER and KW_SYNCHRO_SETTINGS_PROJECT cannot be used at same time**.
KW_SYNCHRO_SETTINGS_SCHEDULE : schedule setup for synchronization.
Two syntaxes are possible :
1. CRON : CRON syntax with java Quartz implementation (http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/crontrigger.html)
2. A period : <integer><unit).

Possible units are:

a. Milliseconds : ms
b. Seconds : s
c. Minutes : mn
d. Hours : h
e. Days : d

Ex :

```
KW_SYNCHRO_SETTINGS_PROJECT=Adapter
KW_SYNCHRO_SETTINGS_SCHEDULE=1mn
```
(Synchronize each minute)

E. Documents configuration

Each document type has specific settings to define how i twill be processed.

1. Identification

Documents identification is made by "parsers". The parsers are the description files located in the folder "**dsc**". There are several kinds of parsers : "StreamIn" (each line is a record with one or several fields), "FieldIn" (each line has the field name and its value), RDI (specific SAP format closed to FieldIn), PDF, OTF, XML.
Each scanner configuration lists a series of parsers that will be used in sequence to identify the document.
Once a document has been identified, the parser will then extract the fields from the data stream to create a data dictionary (that will be converted into XML during submit to Series/M.
Exception : for XML format, only the identification is made, the dictionary is supposed to be the XML contents itself.
The document name will be used to find the appropriate settings in the configuration files (template to use, datasource name, routing,…)

2. Template

A template is the document model that will be used by Series/M to create the document layout.
The syntax to indicate the document template is <document>_TEMPLATE=<template path>
Ex :
```
Invoice_TEMPLATE=/Invoicing/Templates/Invoice.template
```

3. Data source

The templates are related to one or several dictionaries (=data model). The data model is linked to the document as a data source with a name.
When calling a template with a data stream, the name of the data source corresponding to the data stream must be indicated.
The syntax for this is :
<document>_DATASOURCE=<data source name>
Ex :
```
Invoice_DATASOURCE=DATA
```

a) Additional data sources

- Additional files

A template in KW can use several data sources. It is possible in the Adapter configuration for the document to indicate additional files that should be processed.
These files will be parsed like the main data stream and will therefore create new data sources that can be used for extra data sources of the document.
When parsed, the extra files will be identified as a document (like main data stream). This document will not refere any template but will just indicate its data source. This data source will be referred in the main document.

Syntax :
1- At « master » document level, indicate the additional files to pick up and analyze :
<document>_ADDITIONAL_FILES=<file1> ;<file2> ;etc…
2- Indicate the documents that should be identified by these additional files :
<(additional) document>_ROUTAGE=**NONE**
<(additional) document>_DATASOURCE=<additional data source name>
3- In the master document, indicate the additional data sources :
<document>_ADDITIONAL_DATASOURCES=<additional document1> :<additional data source 1> ;<additional document 2> :<additional data source 2> ;etc…

Exemple :
Here an extra file containing routing information is supplied with the main data file. It is created in the "routing" folder with same name than the main file, except the "rtg" extension. The additional routing file will trigger the document "ROUTING".

```
Invoice_TEMPLATE=/Invoicing/Templates/Invoice.template
Invoice_DATASOURCE=DATA
Invoice_ADDITIONAL_DATASOURCES=ROUTING:ROUTING_DATASOURCE
Invoice_ADDITIONAL_FILES=..\\routing\\${originBaseName!}.rtg
```

```
ROUTING_TEMPLATE=NONE
ROUTING_DATASOURCE=ROUTING_DATASOURCE
```

More details :
When an invoice data stream is received, we are going to the folder "routing" pick up a file with same name with extension ".rtg". The variable $originBaseName!} is replaced by Freemarker.
The routing file is parsed and raises a document named "ROUTING". This document is set up with a template name set to "NONE" that indicates the Adapter it must be sent to Series/M.
The ROUTING document has a main data source named "ROUTING_DATASOURCE".
Then in the master document we indicate an additional data source in this way ROUTING:ROUTING_DATASOURCE (= from the document "ROUTING", use the data source "ROUTING_DATASOURCE").
When the document will be sent to Series/M, it will use the main XML data model for the data source "DATA", and the additional data source for "ROUTING_DATASOURCE".

- JDBC Data source

Not described in this document.

- Script Data source

Not described in this document.

4. Control job file (M/OMS)

For preformatted documents (pdf, otf), a control job file must be built and sent to M/OMS with the document. The control job files are located in the folder « xf" with an extension ".ftl".
By default, the Adapter will search for <document name>.ftl, but we can indicate another file with the parameter <document>_OMS_CONTROL_JOB_TEMPLATE.
The control job files are Freemarker templates and can contain dynamic values from the document context (See K Freemarker)
Ex :

```
Invoice_OMS_CONTROL_JOB_TEMPLATE=commonControlJob.ftl
```

5. Destination

1- M/OMS
To print the document to M/OMS, indicate the property <document>_PRINTER=OMS.
Ex :

```
Invoice_PRINTER=OMS
```

2- TONIC
To create a Tonic document, set the parameter <document>_TONIC_PATH=<full document path in Tonic>
The document path can use Freemarker synax to make it dynamic and document context dependant (cf. Freemarker).
Ex :

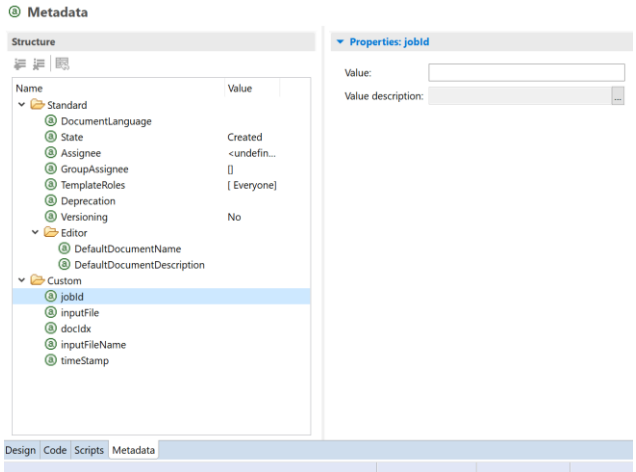
```
Invoice_TONIC_PATH=/Tonic/Invoicing/${docType}/${date}/${originBaseName}
```

3- PDF Response
The PDF responses are unitary and cannot concatenate several documents.
Set a value for the parameter <document>_RESPONSE_PATH.
Ex :

```
Invoice_RESPONSE_PATH=./Invoicing/${docType}/${originBaseName}.pdf
```

6. Metadata

When submitted to Series/M, the document metadata can also be set. The document context can be used, following the Freemarker syntax.



To indicate the metadata and their values, use the paramete<document>_METADATA=<metadata name> :<metadata value> ;<metadata name> :<metadata value> ;etc...
Several metadata can be set using ";" as separator. The metadata name and its value are separated by ":".
This setting is parsed by Freemarker and can use dynamic values from document context.
Ex :

```
Invoice_METADATA=jobId:${jobId!};inputFile:$P{origin!};inputFileName:${originBaseName!};docIdx:${docIdx
!};timestamp:${timestamp!}
```

7. M/OMS parameters

Once all the documents have been submitted, if they were sent to M/OMS, it's possible to update M/OMS parameter. It's specially used to update a status allowing M/OMS to process the document.
This parameter is parsed by Freemarker and can use dynamic values from document context.
The update is launched only if all the documents have been processed without error detection.
The syntax is <document>_UPDATE_OMS_PARAMETERS=<parameter> :<value> ;<parameter> :<value> ;etc...
Ex :

```
ZAPP_APPNOTE_UPDATE_OMS_PARAMETERS=JOB_ID:${jobId!};JOB_SUBMIT_STATUS:DONE
```

In this example, the stack will process the document only if the parameter JOB_SUBMIT_STATUS is set to "DONE". If an error occurs, the update won't be launched and the documents will have their default status, preventing the stack to process them.

8. Filters

Once the data dictionary has been made for a document, it can be viewed as an XML. For Series/M, this XML is the Data Model that will be used to create the document.
The Adapter allows to transform this XML before the sending to Series/M by indicating a filter to apply. This filter can be an XSLT style sheet, or an external program working in stdin/stdout mode.

a) Xslt

Indicate an XSLT style sheet in this way : <document>_FILTER=<path to style sheet>
The path **must** terminate by "**.xsl**" or "**.xslt**" extension.
Ex :

```
Invoice_FILTER=../xslt/Invoice.xslt
```

b) Command

Un external program can be called to transform the XML. This program has to read the original XML file from its standard input, and must return the transformation result to its standard output.
<document>_FILTER=<program name>
Ex :

```
Invoice_FILTER=../filters/InvoiceFilter.exe
```

F. Scanners

The scanner configurations files must be located in the "scanners" folder, wih **.properties** extension.
Each scanner must contains at least the following settings :

FOLDER=<folder to scan>
FILTER=<file name filter (MSDOS syntax)>
SCHEDULE=<scan schedule>

Ex:

```
FOLDER=scanners/scan
FILTER=*.*
SCHEDULE=5s
```

Other settings :
PARSERS=<parser1>;<parser2>;etc...
This is the list of parsers that will be used to analyse input streams. The order is important as the parsers are used sequentially, it stops once a document was identified.
Ex :

```
PARSERS=dsc/Invoice.dsc;dsc/routing.dsc
```

ENCODING=<encoding name, UTF-8, ISO8859-1,...>
The encoding that should be used to read the files received. It will be ignored for XML or PDF.
Ex:

```
ENCODING=ISO8859-1
```

PARENT=<Fichier de configuration parent>
The "PARENT » option can be used in any .properties file. It indicates a configuration file that will be used when a parameter can't be found.
Ex :

```
PARENT=../main.properties
```

DELETE_FILES=true/false
Controls if the scanner must remove or not the files found by the scanner (should be yes most times).
Ex :

```
DELETE_FILES=true
```

1. Filters

The files found by the scanner can be transformed by a filter. Either the JDE filter (specific to JDE tagged PDFs), or via an external command running in stdin/stdout. If the scanner receives XML streams, an XSLT type filter can also be specified.

To save the file before its transformation by the filter, indicate a value for the backup folder.
Indicate the type of data that the filter is supposed to produce.

a) Command

Ex :

```
FILTER_TYPE=COMAND
FILTER_ARGUMENTS=<programme à lancer>
FILTER_BACKUP_FOLDER=bak
FILTER_RESULT_EXTENSION=txt
```

b) JDE

Ex :

```
FILTER_TYPE=JDE
FILTER_ARGUMENTS=-arg dsc/jde.txt -field
FILTER_BACKUP_FOLDER=bak
FILTER_RESULT_EXTENSION=txt
```

c) XSLT

Ex :

```
FILTER_TYPE=XSLT
FILTER_ARGUMENTS=<chemin de la feuille de style XSLT>
FILTER_BACKUP_FOLDER=bak
FILTER_RESULT_EXTENSION=txt
```

G. Parsers

Parsers are of different types depending on the flows to be analysed. Some work in line-by-line mode (fieldin, streamin, rdi), others require specific tools (PDF, OTF, XML).
The extensions of the configuration files for the parsers are :
.dsc : rdi, fieldin, streamin, xmlin
.pdfin : PDF
.sapgonfin : OTF

1. RDI

The RDI format is supported natively in the Adapter. Its description file just contains a keyword to indicate whether you are doing "standard" RDI or "simplified" (more compact) RDI.
The name of the document that will be used as a reference for the parameterisation is the name of the SAP form found in the header of the RDI (see image below).

Ex 1 :

r3rdi_simple.dsc
```
1  R3RDI "SIMPLE"
2  MODE SIMPLE
```

The first line "R3RDI "SIMPLE"" simply gives the parser a name. This name will be used in log messages.
The mode then indicates that a "simplified" RDI is being parsed.

Ex 2 :

r3rdi_std.dsc
```
1  R3RDI "STANDARD"
2  MODE STANDARD
```

2. StreamIn

StreamIn parsers use the StreamServe descriptor file syntax. Scripts can be used but with javascript syntax. The line being parsed is available via the "line" variable. The current document is available via "currentEvent" (can be null).
Example of a StreamIn descriptor with javascript for identification. In this example we parse identical records but each document must consist of two records. The lines are counted and a modulo function triggers the first record for each odd line. The first record is identical to the second (same fields) mails uses the "NewEvent" keyword which triggers a new document.

```
StreamIn "InFile"
Begin
    Record "AVIS" 1 ChrSep ";"
        Match SCRIPT {
            if (lineCounter == null || lineCounter == undefined){
                    var lineCounter = 0;
                    //var callCounter = 0;
                }
            function check(aLine){
                var matchAvis = aLine.startsWith('AVIS');
                if (! matchAvis){
                    return false;
                }
                if ((lineCounter % 2) == 0){
                    lineCounter = 1;
                    return true;
                }else{
                    return false;
                }
            }
            check(line);
        }
        NewEvent "AvisCoupure";
        Fields
            "ID";
            "Adr_Coupure";
            "Adr_Client_1";
            "Adr_Client_2";
            Etc…(liste de champs)
        End
    End
    Record "AVIS" 1 ChrSep ";"
        Match SCRIPT{
            function check(aLine){
                var matchAvis = aLine.startsWith('AVIS');
                if (! matchAvis){
                    return false;
                }
                lineCounter++;
                return true;
            }
            check(line);
        }
        InEvent "AvisCoupure";
        Fields
            "ID";
            "Adr_Coupure";
            "Adr_Client_1";
            "Adr_Client_2";
```

```
            Etc…(liste de champs)
        End
    End
End
```

3. FieldIn

FieldIn parsers use the syntax of StreamServe FieldIn descriptor files, but scripts are not supported.
Ex :

```
FIELDIN  "fieldin"
    PosLabelStartEvent 1
     LabelStartEvent "BEGIN"
     ChrSepEvent ":"
     LabelPrefix "prefix"
     ChrSepPrefix     ":"
     LabelFieldCont   "+"
     FieldContString " "
    IGNORELEVEL
    POSFIELD 1
    ChrSepField ";"
 END
```

←→

```
BEGIN:INVOICE
TITLE;+Invoice
TITLE;+Suite1 invoice
TITLE;+Suite2 invoice
DATE;25/03/1971
AMOUNT;1234567.25
prefix:
invoiceNumber;12345678901
VAT;15%
VAT_AMOUNT;1234.56
ITEM_NO;1
ITEM_ID;1
ITEM_DESC;un machin
ITEM_UNIT_PRICE;12.00
ITEM_QTY;10
ITEM_PRICE;120.00

BEGIN:INVOICE
TITLE;+Invoice
TITLE;+Suite1 invoice
```

document name
new document trigger
field value
field name
separator

4. Formatted documents

Parsers for formatted documents allow documents to be identified according to the data contained on the page. The syntax is similar to StreamServe's PreformatIn event type but is not strictly the same for indicating page types.
Scripts on fields are supported.

- ✓ Text deletion is only supported for OTF.
- ✓ Deleting images is not supported.
- ✓ Parsers for PDFs must have the extension ".pdfin".
- ✓ Parsers for OTF must have the extension ".sapgofin".

Ex :

```
Event "Mail_in"
PageIn
// MessageID "2942224"

    Field "Invoice_No" 51 229 111 241 $Invoice_No;
 Field "Client_No" 50 249 114 260 $Client_No;
 Field "Echeance" 51 266 115 279 $Echeance Format "Ddd.mm.yyyy";
 Field "Compte" 52 286 120 299 $Compte;
 Field "Montant" 51 305 118 318 $Montant Format "Nk=,d=.";
 Field "From" 52 324 449 337 $From;
 Field "To" 52 343 451 356 $To;
 Field "Niveau" 52 362 69 376 $Niveau;
    LxfSelection
       Begin
    <?xml version="1.0"?>
    <!DOCTYPE s-lxfin SYSTEM "s-lxfin.dtd">

    <s-lxfin>
    <page size="595.3,841.9" order="1" xoffset="15" type="FIRST">
    <pattern name="Mail" position="17.0,2.0" size="21.0,24.0" options="276" match="E"/>
    <field name="MailPattern" position="17.0,2.0" size="21.0,24.0" options="20" />
    <field name="Invoice_No" description="Numéro de Invoice" position="50.0,228.0" size="60.0,12.0" variable="Invoice_No" options="20"/>
    <field name="Client_No" description="Numéro de client" position="49.0,248.0" size="64.0,11.0" variable="Client_No" options="20"/>
    <field name="Echeance" description="Echéance " position="50.0,265.0" size="64.0,13.0" variable="Echeance" format="Ddd.mm.yyyy" options="20"/>
    <field name="Compte" description="Compte de contrat" position="51.0,285.0" size="68.0,13.0" variable="Compte" options="20"/>
    <field name="Montant" description="Montant" position="50.0,304.0" size="67.0,13.0" variable="Montant" format="Nk=,d=." options="20"/>
    <field name="From" description="Emetteur du mail" position="51.0,323.0" size="397.0,13.0" variable="From" options="20"/>
    <field name="To" description="Destinataire du mail" position="51.0,342.0" size="399.0,13.0" variable="To" options="20"/>
    <field name="Niveau" description="Type de mail de relance" position="51.0,361.0" size="17.0,14.0" variable="Niveau" options="20"/>
    </page>
    </s-lxfin>
       End
    End

End
```

Only the XML part is used.
All dimensions are in points, starting from the upper left corner of the page.
<**page**>: represents a page type.
Attribute **type**: the type of page must be indicated: FIRST, NEXT, LAST.
<**pattern**>: identification of the page.
Attribute **match**: identification pattern of the page. If wildcards are enabled, the syntax is that of java regular expressions (http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/crontrigger.html).
Attribute **options** :
20 = no use of wildcards
Other values = use of wildcards in the "match" attribute expression
<**field**> : definition of the fields of the page
Attribute options:

- 20 = keep spaces, do not delete text
- 21=keep spaces, delete text
- 18=delete spaces, do not delete text
- 19=delete spaces, delete text

Javascript can be used to calculate the field:
Ex :
Here the javascript uses java libraries (Apache commons Lang) to make replacements.

To note: the CDATA section which allows to use special characters to avoid using xml entities instead of special characters.

```
<field name="QR" position="180.0,590.0" size="160.0,150.0" variable="QR" options="21"><![CDATA[
QR = org.apache.commons.lang3.StringUtils.replace(QR,"\n","");
QR = org.apache.commons.lang3.StringUtils.replace(QR,"\r","");
QR = org.apache.commons.lang3.StringUtils.replace(QR,"\\c013\\","/r");
QR = org.apache.commons.lang3.StringUtils.replace(QR,"\\c010\\","/n");
    return QR;
]]></field>
```

a) Control JobFichier de contrôle

Pre-formatted documents (except for JDE tagged PDFs) are sent to MOMS. This requires a control file that will tell MOMS the target (KW_MEDIUM) and various parameters of the documents.
The templates for these control files should be placed in the <root folder>/xf, in the form <document>.ftl.
The templates for the control files will be evaluated by Freemarker, which will replace the variable data according to the context of each document. Fields mapped to documents can be used in the control file.
Ex :

```
<#ftl output_format="XML">
<?xml version="1.0"?>
<document>
    <parameter name="DOCTYPE" value="${_docType}"/>
        <parameter name="JOB_ID" value="${jobId}"/>
        <parameter name="DOC_POSITION" value="${_docNum!}"/>
        <parameter name="OUT_FILE_NAME" value="-"/>
        <parameter name="JOB_DATE" value="${_date!}"/>
        <parameter name="JOB_TIME" value="${_time!}"/>
        <parameter name="PRINTER_NAME" value=""/>
        <parameter name="DOCUMENT_NO" value="${No_Matricule}"/>
        <parameter name="EMAIL_TO" value="${Email}"/>
        <parameter name="CLIENT_NO" value="${No_Matricule}"/>
        <parameter name="PERIODE_SALAIRE" value="${Periode!}"/>
        <target medium="SALAIRE_ARCHIVE" pages="1-" number="01" type="orig">
        </target>
        <#if Email?has_content>
            <target medium="SALAIRE_MAIL" pages="1-" number="01" type="orig">
            </target>
        <#else>
            <target medium="SALAIRE_PRINT" pages="1-" number="01" type="orig">
            </target>
        </#if>
    <pagedefinitions>
    <page number="1">
    <ocl value="PAGE_CONTROL DUPLEX_MODE=DUPLEX" /> </page>
    </pagedefinitions>
</document>
```

5. XML

XML streams can normally be handled directly by KW. However, if the XML needs to be divided into individual documents, the Adapter can be set up to recognise the XML stream and use KW's APIs to split the overall XML stream into individual documents.
The overall operation is as follows:

1- Identification of the XML via an XPATH pattern.
2- If the recognised document has a split option:
   a. The Adapter uses a Series M/ API to split the original XML
   b. Each XML unit constitutes a data model which is sent to Series M/ to constitute a document
   c. The original XML is ignored and not sent to Series M/

a) Pattern

The patterns to identify the XML are placed in a "description" file.
The extension of the descriptor file is .dsc, it must be placed in the "dsc" folder.
The first line contains the name of the descriptor. This name is used in the logs.
XMLIN=<name>
The following lines are of the form:
    <document>=<XPATH expression>
If the XPATH expression applied to the received XML returns a result, the document is considered identified.
Example of an XML descriptor:

```
XMLIN=xmlin

#Si un des xpath "matche", la clé indique le nom du document et est utilisé dans
configuration.properties pour lire les settings correspondants au document.
#La syntaxe est en XPATH

R584302C=/R584302C/Vue_triple_sur_F0411_F4311_F4101_S1_Group/LBH_On_Ship_To_S3
R580301=/R580301/Vue_sur_F03B21_et_F03B11_S1_Group/LBH_On_Address_Number_S23
R5519002=/R5519002/Vue_logique_sur_F4211___F4074_S1_Group/On_N__référence_S2
```

b) Split

If the XML is to be split, the <document>_XML_SPLITTER=<splitter> parameter must be filled in.
The value for <splitter> corresponds to the full path of a Split configuration deployed on the KW server. Splitters are defined in the Workbench and are deployed with the KW project like other project resources (templates, data models, images,...).
For the definition of a splitter, please refer to : *MWorkbench_En.pdf, 12.9 The Data Splitting*

Ex :

```
R584302C_XML_SPLITTER=//Configuration/splitters/R584302C.splitting
```

H. Events

The .event files, stored in the "dictionaries" folder, use the StreamServe syntax.
Formats are ignored, only block and field names are taken into account.
This is to define the structure of the data dictionary (blocks/sub-blocks) which will ultimately provide the structure of the XML sent to KW.
Field and block names cannot start with a number. If necessary the Adapter will add the letter "F" as a prefix.
The name of the file must correspond to the name of the document, its call is implicit: when a document is identified (in RDI, FieldIn or StreamIn), the Adapter will search for a corresponding event file.
If there is no file, the XML produced will respect the initial block structure of the rdi or streamin. In the case of fieldIn, all fields will be at root level.
When a field created by the parser is not present in the event file, it will be created anyway, either in its original block (in the case of rdi or StreamIn), or at the root of the document (in the last case, if the field is found several times, only one instance will be retained in the final xml).
Ex :

```
StreamIn
// Generated with StreamIN
  Root Name "Message"
    Field "HEADER_mode" $HEADER_mode
    Field "HEADER_dest" $HEADER_dest
    Field "HEADER_MastCctr" $HEADER_MastCctr Format "Nk= d=,"
    //Liste de champs…
    Field "CORE_END_editKey" $CORE_END_editKey
    Block "start"
      Field "CORE_START_editKey" $CORE_START_editKey
      Field "CORE_START_hSoldePrec"
      Field "CORE_START_hsPrec"
      Field "CORE_START_hCongeAnPrc"
      Field "CORE_START_proghebdo" $CORE_START_proghebdo
    End
    Block "core"
      Field "CORE_date" $CORE_date
      //Liste de champs…
      Field "CORE_libelleEvene"
      Field "CORE_hCongeAnRep"
    End
    Block "Total"
      Field "CORE_END_hTotal" $CORE_END_hTotal
      Field "CORE_END_indPiqTotal" $CORE_END_indPiqTotal
      Field "CORE_END_indRepTotal" $CORE_END_indRepTotal
      Field "CORE_END_indDepTotal" $CORE_END_indDepTotal
    End
    Block "Heure_due"
      Field "CORE_END_hDues" $CORE_END_hDues
      Field "CORE_END_hNuitTotal" $CORE_END_hNuitTotal
    End
    Block "nouveau_solde"
      Field "CORE_END_hSolde" $CORE_END_hSolde Format "Nk=,d=."
      Field "CORE_END_hEcret" $CORE_END_hEcret
      //Liste de champs…
      Field "CORE_END_hCongeAnTotal" $CORE_END_hCongeAnTotal
    End
  End
```

I. Purging

It is possible to set up periodic purges, the syntax is as follows:
PURGE=<purge1>,<purge2>,etc..
PURGE_<purge1>_FOLDER=<folder to purge>.
PURGE_<purge1>_FILTERS=<filter on file names (MSDOS syntax)>
PURGE_<purge1>_DAYS=<Age in days of files: older files are deleted>
Alternative: PURGE_<purge1>_AGE=<number><unit of duration: day=d,hour=h,minute=mn>
PURGE_<purge1>_SCHEDULE=<expression CRON Quartz> or <number><unit of duration (same as above)>
CRON Quartz : http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/crontrigger.html
PURGE_<purge1>_RECURSIVE=true/false (search in sub folders)

Ex :

```
PURGE=bak,InputJDE
PURGE_bak_FOLDER=.\\bak
PURGE_bak_FILTERS=*
PURGE_bak_AGE=60d
PURGE_bak_RECURSIVE=true
PURGE_bak_SCHEDULE=1d
```

```
PURGE_InputJDE_FOLDER=.\\input\\jde
PURGE_InputJDE_FILTERS=*
PURGE_InputJDE_AGE=10d
```

PURGE_InputJDE_RECURSIVE=true
PURGE_bak_SCHEDULE=1d

J. Summary of parameters

Parameter	Description	Location

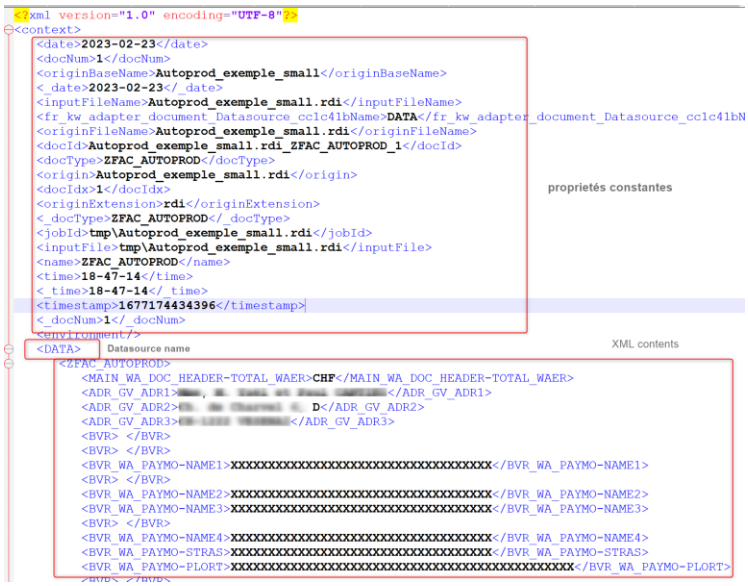
Parameter	Description	Location
KW_SERVER	name/ IP address for KW server	Base configuration file. Ex : main.properties
KW_SERVER_PORT	Listening port for KW	Base configuration file
SSL_ENABLED	true/false (protocole https ou http)	Base configuration file
LOG_LEVEL	Log level, java API syntax (https://docs.oracle.com/javase/7/docs/api/java/util/logging/Level.html) ALL, FINEST, FINER, FINE, CONFIG, INFO, WARNING, SEVER, OFF)	Base configuration file
PARENT	Location for a parent configuration file that will be used if a parameter is missing.	Any .properties file. Avoid « cycling » (inifinte loop)
KW_USER	KW user allowed to call web services.	Base configuration file
KW_PASSWORD_CRYPTED	Crypted password (KW encryption tool)	Base configuration file
KW_PASSWORD_PLAIN	Password (clear)	Base configuration file
KW_SYNCHRO_SETTINGS_PROJECT	Project in workbench that contains the Adapter configuration	Base configuration file
KW_SYNCHRO_SETTINGS_SCHEDULE	Update configuration schedule (CRON syntax from java Quartz or period + unit d, h, mn) http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/crontrigger.html	Base configuration file
KW_SYNCHRO_SETTINGS_FOLDER	Update configuration folder (in replacement of KW_SETTINGS_PROJECT)	Base configuration file
KEEP_PARSED	true/false Indicates if a backup must be done for the daa resulting of the processing (XML data, context, document information)	Base configuration file
BACKUP	true/false Indicates if the input files must be saved.	Base configuration file
PURGE	List of purging definitions (sperated by comma ",")	Base configuration file
PURGE_<purge name>_FOLDER	Folder to purger	Base configuration file
PURGE_<purge name>_FILTERS	File name filter. Several filters can be set, separator « , ».	Base configuration file
PURGE_<purge name>_DAYS	Maximum age for files (they will be deleted if older)	Base configuration file
PURGE_<purge name>_AGE	Maximum age for files, with unit (d, h, mn). In replacement of PURGE_<purge name>_DAYS.	Base configuration file
PURGE_<purge name>_RECURSIVE	true/false Search in sub folders or not.	Base configuration file
PURGE_<purge name>_SCHEDULE	Purging schedule Number+unit (d,h,mn), or CRON expression from java Quartz. http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/crontrigger.html	Base configuration file
PROCESSING_FOLDER	Working folder for the adapter. (Usually tmp\processing)	Base configuration file
<document>_TEMPLATE	KW template for the document identifiedt. Ex : Invoice_TEMPLATE=//Invoicing/templates/Invoice.template	Base configuration file
<document>_DATASOURCE	Datasource for the template. Ex : Invoice_DATASOURCE=DATA	Base configuration file
<document>_METADATA	Metadata list to send when submitting the dcument. Syntax : <name> :<value>;<name>:<value> ;etc... We can use Freemarker syntax. Ex : Invoice_METADATA=JobId :${jobId !} ;InputFile:${originBaseName!}	Base configuration file
<document>_OMS_CONTROL_JOB_TEMPLATE	M/OMS Control Job Freemarker template (for preformatted documents such as PDF or OTF). If not set, the default value will be <document>.ftl. These templates are mandatorily put in the folder « xf ».	Base configuration file
<document>_ADDITIONAL_FILES	Additional files to parse when processing the document. These additional files should produce new documents and therefore new datasources (for the initial document <document>_ADDITIONAL_DATASOURCES).	Base configuration file
<document>_ADDITIONAL_DATASOURCES	Additional datasources to send for this document. The can come from preceding documents in the flow (like a technical header document having its own parser) or from additional data files indicated by <document>_ADDITIONAL_FILES.	Base configuration file
<document>_PRINTER	Printer that should be used by M/Text (usually « OMS »)	Base configuration file
<document>_UPDATE_OMS_PARAMETERS	List of OMS paramters that will be updated if ALL the documents in the flow are processed without error. Usually used to set a status that allows the stack to process the whole document set at one time. Ex : Invoice_UPDATE_OMS_PARAMETERS=JOB_ID:${jobId!};JOB_SUBMIT_STATUS:SUCCESS	Base configuration file
<document>_TONIC_PATH	Tonic path if the document must be sent as a Tonic interactive document. Freemarker syntax possible.	Base configuration file
<document>_RESPONSE_PATH	Path for a PDF as response. Only individual PDFs can be created, the documents of an input stream cannot be grouped into a global single PDF with this method.	Base configuration file
<document>_FILTER	Filter to apply on the main datasource of the document. Could be an external command (Stdin/stdout) or an XSLT stylesheet.	Base configuration file
<document>_XML_SPLITTER	Indicate the XML Splitter to use in KW workspace (file managed in the Workbench and deployed to the KW server).	Base configuration file

Parameter	Description	Location
PARSERS	List of parsers to use for a given scanner. Separate parsers with « ; ». Ex : PARSERS=dsc/Invoice.dsc;dsc/rdi;dsc/routing.dsc	File .properties in « scanners » folder
ENCODING	Encoding to use to read the input data streames for (streamin, fieldin, rdi). (https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html)	File .properties in « scanners » folder
FILTER_TYPE	Filter type to apply on received input streams, before the parsing stage. COMMAND, JDE, XSLT	File .properties in « scanners » folder
FILTER_ARGUMENTS	type JDE : JDE filter arguments (Ex : -arg dsc/jde.txt -field) type XSLT : stylesheet location XSLT type COMMAND : command to call	File .properties in « scanners » folder
FILTER_BACKUP_FOLDER	Backup file to save the original input data streams before the filter process.	File .properties in « scanners » folder
FILTER_RESULT_EXTENSION	Extension to describe the kind of data produced by the filter. This can be used to find the right parser to be used. (For instance avoid to open an XML as a PDF)	File .properties in « scanners » folder
PARSE_ONLY	true/false If « true », the documents are not sent to KW, th process will stop after the parsing stage (the parsing result will be saved if FILTER_BACKUP_FOLDER is set)	File .properties in « scanners » folder
DELETE_FILES	true/false If « false », the input files found in the scanned folder won't be deleted (and will be reprocessed at the next scan loop)	File .properties in « scanners » folder
FOLDER	Folder to scan.	File .properties in « scanners » folder
FILTER	Filter on file names (MSDOS syntax)	File .properties in « scanners » folder
SCHEDULE	Scan period. CRON Quartz syntax (http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/crontrigger.html) or a number and a time unit (d,h,mn,s,ms)	File .properties in « scanners » folder

K. Freemarker context

Parameters supporting Freemarker expressions or MOMS control files have the document context available to insert dynamic values.
See the Freemarker documentation (https://freemarker.apache.org/docs/index.html) for more details on the possibilities offered (loops, calculations, functions, tests, etc.).

Variable	Comment
${docNum} ${_docNum} ${docIdx}	Document position in the global flow.
${originBaseName}	Input file name, without path nor extension.
${originFileName} ${inputFileName} ${origin}	Input file name without path, with extension.
${originExtension}	Input file name extension
${_date} ${date}	Current date (yyyy-mm-dd)
${docId}	Id created by the adapter based on input file name, document name, document position.
${docType} ${_docType} ${name}	Document type
${jobId}	Job id managed by adapter, based on input file name
${time} ${_time}	Current hour (hh-mm-ss)
${timestamp}	Timestamp in ms
${<datasource>.<root>.<field>}	Access the document data. Ex : ${DATA. ZFAC_AUTOPROD. HEAD_P_ADRESSE01}



IV. Startup

A. Scan directory

java -cp KWAdapter.jar;KWAdapter_lib* -Dkwsoft.env.globalcontext=ini\client.ini -Djava.io.tmpdir=tmp -Djavax.net.ssl.trustStore=ini\cacerts **fr.kw.adapter.engine.scanner.StartScanners** -config **main.properties**

- - The client.ini file is available in the KW installation, it is used by some KW libraries used in the Adapter
- - The option -Djavax.net.ssl.trustStore=ini\cacerts.fr can be useful in an SSL context to manage a list of trusted certificates

B. Windows service

It is possible to install the Adapter as a Windows service with "Apache Commons Daemon" https://commons.apache.org/proper/commons-daemon/procrun.html.
It is possible to rename the Apache executable to "KWAdapter-service,exe" as this utility uses its name to create the service.

Exemple of setings :
KWAdapter-service.exe //IS//KWAdapter-service
--Install=**D:\KWSOFT\adapter**\WindowsServiceTool\KWAdapter-service.exe
--Description="KWAdapter"
--Jvm="C:\Program Files\Eclipse Adoptium\jdk-11.0.14.101-hotspot\bin\server\jvm.dll"
--Classpath=D:\KWSOFT\adapter\KWAdapter.jar;D:\KWSOFT\adapter\KWAdapter_lib*
--StartMode=jvm --StartClass=fr.kw.adapter.engine.scanner.StartScanners
--StartMethod=main
--StartParams=start
++StartParams=-config=D:\KWSOFT\adapter\main.properties
--StopMode=jvm --StopClass=fr.kw.adapter.engine.scanner.StartScanners
--StopMethod=stopService
--StopParams=stop --LogPath=**D:\KWSOFT\adapter\log**
--StdOutput=auto --StdError=auto
--StartPath=D:\KWSOFT\adapter
--JvmOptions=-Dkwsoft.env.globalcontext=**D:\KWSOFT\adapter\ini\client.ini**
++JvmOptions=-Djava.io.tmpdir=**D:\KWSOFT\adapter\tmp**
++JvmOptions=-Djavax.net.ssl.trustStore=**D:\KWSOFT\adapter\ini\cacerts**

Once the arguments are correct, run KWAdapterService.bat from an MSDos window running in administrator mode. The service is created under the name "**KWAdapter-service**".
If after testing, the service does not work properly, it can be removed with the command : « **`sc delete KWAdapter-service`** »

C. Unitary run

The adapter can be launched by command line to process a single input file.
Ex :
kw.adapter.engine.ProcessFile
java -cp KWAdapter.jar;KWAdapter_lib* -Dkwsoft.env.globalcontext=ini\client.ini -Djava.io.tmpdir=tmp -Djavax.net.ssl.trustStore=ini\cacerts **fr.kw.adapter.engine.ProcessFile** -config **scanners\<mon scanner>.properties** -baseConfig **main.properties** -in **<file to process>** [-noRefresh] [-noPurge] [-refreshOnly]

Note that the configuration usually starts from a scanner configuration as it will contain the information about the parsers to be used.
The -baseConfig option is not mandatory (but recommended) if the parser configuration references main.properties via the "PARENT" property.
-noRefresh: the Adapter is asked not to refresh its settings when connecting to the KW server.
-noPurge: the Adapter is asked not to worry about purge settings.

-refreshOnly: The Adapter just updates its settings (by connecting to the KW0 server.

D. Sending a PDF to M/OMS

The Adapter can be used to send PDFs to M/OMS with a control file. This functionality also offers the possibility to resize the pdf.
Example of use (Gerflor context):

- the received JDE PDF is saved by the adapter and converted to XML by the JDE filter.
- The XML generates a "dummy" document which is sent to M/OMS in a "PRE-PRINT" stack
- The PRE-PRINT stack launches an external command that will use the possibilities of the adapter to send the original PDF to M/OMS (for processing by the classic mail, archiving, printing, etc. stacks). In passing, the pdf can be resized via a parameter (cf. Common\scripts\OMS\getPPDFScale.js)

Ex :

```
java -cp KWAdapter.jar;KWAdapter_lib\* -Dkwsoft.env.globalcontext=ini\client.ini -Djava.io.tmpdir=tmp -
Djavax.net.ssl.trustStore=ini\cacerts fr.kw.adapter.engine.ProcessFile -config main.properties -noPurge -noRefresh -
pdfDirect -pdfScale <redimensionnement> -input <fichier PDF> -pdfControlJob <fichier de contrôle M/OMS>
```

-pdfscale : scale for resizing
Ex :
1.0= no change
0.5 = reduce of 50%
-input : the PDF file to preocess
-pdfControlJob : The control job file for M/OMS
-config <main.properties> : The main configuration file (with KW server connexion settings)
-noPurge : Do not run purging tasks (if set in main.properties)
-noRefresh : Do not refresh configuration (if set in main.properties)

The italic arguments dépend on context.
-Djava.io.tmpdir=tmp : folder for java
-Djava.nst.ssl.trustStore=ini\cacerts : use the ini\cacerts file for the list of trusted digital certificates. This is used when the KW server is using HTTPS, in which case the server certificate must be one of the trusted certificates in order for the client program (in this case Java and the Adapter) to accept to connect.