文件校验模板配置指导手册

|  |  |  |
| --- | --- | --- |
| 姓名 | 说明 | 日期 |
| 张海滨 | 文档创建v0.1 | 2017.01.04 |
| 张海滨 | 文档第一版完成v1.0 | 2017.01.06 |

目录

[文件校验模板配置指导手册 1](#_Toc471484933)

[1 文档说明 3](#_Toc471484934)

[2 模板示例 3](#_Toc471484935)

[3 配置方法 4](#_Toc471484936)

[3.1 各节点介绍 4](#_Toc471484937)

[3.1.1 <template/> 结点 4](#_Toc471484938)

[3.1.2 <filters/> 结点 5](#_Toc471484939)

[3.1.3 <filter/> 结点 5](#_Toc471484940)

[3.2 模板文件使用说明 18](#_Toc471484941)

[3.2.1 旧架构检验模板为配置文件 18](#_Toc471484942)

[3.2.2 产品化架构下存储在数据库 19](#_Toc471484943)

[4 总结 19](#_Toc471484944)

[5 附录 19](#_Toc471484945)

[5.1 高级功能校验模板示例——逻辑关联校验 19](#_Toc471484946)

[5.2 高级功能校验模板示例——子字段单独校验 19](#_Toc471484947)

[5.3 高级功能校验模板示例——逻辑运算校验 20](#_Toc471484948)

[5.4 文件返回码 20](#_Toc471484949)

# 文档说明

此文档旨在详细介绍当前清洁算系统的文件校验模块中，对账文件校验模板XML的配置方式，以便于开发、测试、运维和业务各方对校验规则的实现和配置有更深的了解。对于清洁算系统的开发人员来说，此文档属于必读且必须透彻理解，尤其是参与文件校验模块的开发人员。对于测试和运维人员，此文档也需要达到熟悉的程度，以便于在测试和运维工作中能理解各项校验规则的配置方式和实现方式。业务人员也需要了解校验模板的配置，便于日后开展验收工作，也有助于理解系统技术层的实现方式。

文档附录中有现在生产上正在使用的某些文件校验模板，这些模板中使用了某些比较复杂的逻辑校验配置功能，可以用来帮助理解。

# 模板示例

校验模板文件格式为xml格式，文件中结点和属性为自定义，主要包含template、filter、field等结点，这些结点中又各自包含不同的属性或者子节点，1.3中会详细介绍这些结点的实际意义和配置方法。

下面给出校验模板的整体结构图，便于了解校验模板xml格式下，各节点的位置和其属性的位置。

<template>

<filters>

<filter>

<type>fix|leftBlank</type>

<startWith>40</startWith>

<fixLength>581</fixLength>

<errorCode>F124</errorCode>

<operators>

<operator>

<operator>lessThan</operator>

<Result>RESERVED1</Result>

<isTime>yes</isTime>

<conditions>

<field>

<name>BUSS\_TYPE</name>

<fixValue>10</fixValue>

</field>

</conditions>

<factors>

<factor>

<factorName>SETTLE\_DATE</factorName>

<lessThanValue>30</lessThanValue>

</factor>

</factors>

</operator>

</operators>

<fields>

<field>

<name>FLAG</name>

<regx>^40$</regx>

<begin>0</begin>

<end>2</end>

</field>

<field>

…

</field>

</fields>

</filter>

</filters>

</template>

# 配置方法

如1.2节中示例模板所示，校验模板配置根据文件体中不同类型内容分为不同的校验器（filter），同时根据某些特殊功能拓展出一些单独校验模块。根据文件内容不同，校验器分为：文件头、尾校验器；文件体记录行校验器；文件签名行校验器。

## 各节点介绍

本节将详细介绍各主要结点及其属性的意义，及其配置方式。

### <template/> 结点

<template/>是自定义结点，也是校验模板文件中的最大根节点，代表“校验模板”，用于识别校验模板。在编写校验模板时，<template/>结点要配置在最外层，也是必须要配置的。

<template/>中只包含<filters/>子节点，且<template/>中只能配置一个<filters/>子节点。

### <filters/> 结点

<filters/>结点代表所有校验器集合，用于识别出真正参与校验处理的校验器（filter）。在编写校验模板是， <filters/>结点需要配置在<template/>内，也是必须要配置的。

<filters/>中只有<filter/>子节点属性，根据所要校验文件中不同记录体数量（比如文件包含文件头、文件尾、记录行和签名行这四种），可以并列配置多个<filter/>。

### <filter/> 结点

<filter/>结点代表校验器，是真正参与校验的部分，其中包含部分属性会用于全局的逻辑校验。<filter/>是比较重要的结点，包含多个属性配置项和子节点。<filter/>必须配置在<filters/>内，若要一个校验模板能实际进行校验操作，<filter/>结点是必须要配置的。

<filter/>中包含属性为：<type/>、<startWith/>、<fixRowNum/>、<seperator/>、<matcher/>、<errorCode/>、<fixLength/>；

<filter/>中包含子节点为：<operators/>。

这些属性和子节点都是文件实际校验时会用到的重要参数，下面将对这些参数和子节点做详细介绍。

#### <type/> 属性

<type/>代表所在filter需要处理的记录行类型，也是一个<filter/> 结点必须配置好的属性。

根据文件记录行字段间的分隔方式，我们将目前遇到的记录行类型可以归为两大类：记录行字段间用“|”或其他特定字符进行分隔的（如a|b|c|d），其type为“sep”，代表“分隔方式”，为英文单词“separate（分隔）”的缩写；记录行字段间没有特定字符进行分隔，而是通过字段定长来确定字段位置的（如字段A定长为4位，位置为0到4位的记录行： 000a000b000c000d，000a即为字段A），其type为“fix”，代表“定长方式”，为英文单词“fixLength（固定长度）”的缩写。

在大类型“sep”和“fix”的基础上，根据详细的字段分隔方式（目前只有fix类型下会有这种情况，且也只支持fix下如下配置），如：

文件记录为定长模式没有补空格，则为“fix”；

定长模式且有左补空格，则为“fix|leftBlank” ；

定长模式且有右补空格，则为“fix|rightBlank”。

如果确认的所需校验的文件记录行中是补空格方式的字段，则需要明确配置<type/>属性。

#### <seperator/> 属性

<seperator/>属性用于配置在type为“sep”时，所在filter要处理的记录行字段间的分隔符。

当filter的type为“sep”时，此处为必须配置属性；当type为“fix“时，此处不需要配置。如：

省BOSS对账文件数据记录行filter中需配置<seperator>|</seperator>，代表省BOSS对账文件数据记录行字段为“|”分隔。

#### <startWith/> 属性

<startWith/>属性代表所在filter需要处理的记录行的开头标识，用于匹配特定记录行，是<filter/> 结点中必须配置好的属性。

<startWith/>对应Java中String.startWith()方法，用于在校验文件时判断所处理的记录行是不是属于此filter处理。

此配置项需要配置对应记录行的开头特征字符串，如：

省BOSS对账文件的文件头记录行filter中，需配置<startWith>10</startWith>（文件头记录行固定以“10”开头），这样通过匹配开头为“10”的记录行就应为文件头记录行。

#### <errorCode/> 属性

<errorCode/>属性代表所在filter校验不通过时对应的错误返回码，是<filter/> 结点中必须配置好的属性。

根据所在filter需要处理的记录行种类不同，<errorCode/>中需要配置的错误返回码字符串也不同，如：

文件头校验器filter中，需配置为<errorCode>F105</errorCode>（F105代表头记录校验不通过）。

其他不同错误返回码信息请查询附录中5.1文件返回码章节

#### <fixRowNum/>属性

<fixRowNum/>属性代表固定行配置信息，用于判断匹配到的记录行行数是否满足此属性配置的行数，可根据情况选择是否配置。

<fixRowNum/>可用于校验特定行所在位置，如：

文件头filter中可配置<fixRowNum>1</fixRowNum>，用于校验匹配到的行是否为文件第一行；

文件签名行filter中可配置<fixRowNum>-1</fixRowNum>，用于判断匹配到的记录行石头为倒数第一行（现有业务的对账文件中，若存在签名行必定属于倒数第一行，此处为例子，需根据特定情况配置）。

#### <fixLength/> 属性

<fixLength/>属性代表所在filter要处理的记录行若是固定长度的，则其匹配到的记录行应满足此属性配置的长度。<fixLength/>应根据需求选择是否配置，此属性不为必须配置属性。

<fixLength/>用于校验匹配到的记录行应该满足的长度要求，如：

省BOSS对账文件的文件头校验filter中，需配置<fixLength>33</fixLength>，因为现有业务的省BOSS对账文件的文件头固定为33位。

#### <matcher/> 属性

<matcher/>属性代表所在filter做字段详细校验时，可以使用的matcher接口的实体类对象。<matcher/>是模板配置校验规则校验的替代校验方式，不建议使用。

<matcher/>属性可以配置校验器接口matcher的实体类对象，需配置对象的java类路径，这样就可以在类对象中用代码的方式实现一些校验操作或者复杂逻辑校验操作。若想通过这种方式进行某些校验工作，则需要理解校验部分的代码，然后手动实现matcher接口。如老的咪咕清洁算的支付宝文件体校验filter中配置：

<matcher>com.cmsz.cmup.clearing.validator.impl.fieldMatcher.MGAlipayFieldMatcher</matcher>

新架构下所有复杂校验，如多字段逻辑校验，均可通过模板配置方式实现自动校验，具体配置方法见下文。

#### <operators/> 子节点

<operators/>子节点代表逻辑运算表达式校验配置结点，用于配置需要多字段参与的文件全局逻辑预算表达式信息。<operators/>子节点需要根据需求来配置，并且配置较复杂。

<operators/>子节点包含属性：<opName/>、<operator/>、<Result/>、<opDesc/>、<isTime/>；

<operators/>子节点包含子节点：<factors/>、<conditions/>。

通过配置<operators/>，可以实现多字段间的四则运算，以及两个字段间的大小比较。下面将详细介绍这些属性和子节点，并对各种运算的配置实现进行样例进行说明。

##### <opName/> 属性

<opName/>表示<operators/>结点的名字，即当前逻辑运算表达式校验器的别名。<opName/>可根据需要配置。

##### <operator/> 属性

<operator/>属性代表运算符配置，即此逻辑运算校验器所进行的运算种类，为必须配置属性。

<operator/>中可以配置3种：

加法运算配置为<operator>add</operator>；

乘法运算配置为<operator>multi</operator>；

小于比较运算配置为<operator>lessThan</operator>

由于减法是加法的逆运算，除法时乘法的逆运算，大于与小于相逆，所以减法、除法和大于的逻辑运算校验可以通过配置加法、乘法和小于的运算实现，比如Y=X-Z即X=Y+Z。各种运算详细配置方法下边会阐述。

##### <Result/> 属性

<Result/>属性表示运算结果字段或阈值名，是必须要配置的字段，需要注意Result中第一个字母R大写，因为此配置项容易与因子配置项弄混所以比较特殊。

<Result/>用以配置所在逻辑运算表达式的等号左边，即要与运算结果去比对的字段名（阈值名）。比如现有A、B、C三个字段需要参与逻辑运算，且要满足A=B+C的加法运算，则此时需要配置为<Result>A</Result>。

在加减乘除的运算中，<Result/>属性配置的是结果字段名；而在大小比较的运算校验中，<Result/>需要配置的也是小于号左边，比如有D、E两个字段，要满足D<E，则此时需要配置为<Result>D</Result>。

##### <opDesc/> 属性

<opDesc/>为所在<operators/>运算校验器的文字描述，可以根据需求配置，建议用简单描述配置一下，可以体现在日志当中。

##### <isTime/> 属性

<isTime/>表示是否时间逻辑运算校验标识配置，若此逻辑运算是对时间字段进行运算，则必须配置<isTime/>属性。

<isTime/>属性用以分辨出参与逻辑运算的字段是否是时间格式，因为时间的运算进制不与数字运算相同，比如月份是在日期满30、31、28或者27才会进一。

##### <factors/> 子节点

<factors/>子节点表示参与此逻辑运算的因子集合，若要进行正常的逻辑运算此项是必须要正确配置的。

前边介绍的<Result/>属性用来配置结果字段名（大小比较中是较小字段名），那么就需要配置因子字段名，<factors/>子节点就是用来配置参与运算各因子信息的。<factors/>子节点中又可以配置多个<factor/>子节点，用以单独配置参与逻辑运算校验的各因子信息。其结构为：

<factors>

<factor>

…

</factor>

<factor>

…

</factor>

</factors>

###### <factor/> 子节点

</factor>子节点是用来配置参与逻辑运算校验中的各因子信息的，是必须配置好的。

</factor>子节点中包含配置项：<factorName/>、<lessThanValue/>。

<factorName/> 属性

<factorName/>属性用于配置此运算因子子节点所对应字段的字段名，为必须配置属性。

若文件记录字段A、B、C需要满足A=B+C，则B、C均为参与因子，此时必须有个<factor/> 子节点的<factorName/>配置为<factorName>B</factorName>，C同理。

<lessThanValue/> 属性

<lessThanValue/>属性表示大小比较时差值范围，需要根据实际情况进行配置。

</factor> 配置实例

若文件记录中D、E两个字段的大小需要满足D<E，且D、E的差距要在100以内，则此时必须将<factor/> 子节点作如下配置：

<operator>

<operator>lessThan</operator>

<Result>D</Result>

<isTime></isTime>

<factors>

<factor>

<factorName> E </factorName>

<lessThanValue>100</lessThanValue>

</factor>

</factors>

</operator>

此配置就表示D<E，且D比E小的差值在100以内（包括100）。

##### <conditions/> 子节点

<conditions/>子节点用于配置其所在的逻辑运算校验器的触发条件，需要按照需求进行配置。

<conditions/>子节点又包含<field/>子节点，<field>子节点也是下文会说到的字段校验器配置子节点，是很重要的一个节点，可以事项很多功能，详情见下文。

<conditions/>子节点中可以配置多个<field/>子节点，代表一个逻辑运算校验可有多个触发条件，只有所有触发条件同时满足是才会进行逻辑运算校验。其大体结构为：

<conditions>

<field>

…

</field>

<field>

…

</field>

…

</conditions>

###### <field/> 子节点

<field/>子节点本为字段校验器配置子节点，功能很多，在<conditions/>中配置主要用到其规则校验部分，用来当作触发条件校验器。<field/>子节点需要根据需求进行正确配置。

<field/>子节点中的属性和子节点详解见“3.1.3.9.1<field/> 子节点”章节。

###### <conditions/> 配置实例

若所需校验的文件记录行字段中有A、B、C、N四个字段，且需要满足A=B+C，但是前提条件是N=10，则此时<conditions/>中需要配置一个触发条件校验器<field/>，其配置如下：

<conditions>

<field>

<name>N</name>

<fixValue>10</fixValue>

</field>

</conditions>

其中<field/>的属性有<name/>和<fixValue/>，分别用于配置触发条件字段名及触发条件固定值大小。此处配置表示名为“N”的字段在其值大小为“10”时满足触发条件。

#### <fields/> 子节点

<fields/>子节点为各字段校验器的集合，为必须配置的子节点。

<fields/>子节点中有子节点<field/>，<field/>表示字段校验器。<fields/>中可以配置多个<field/>子节点，且配置的<field/>数量应该与当前filter需要处理的文件记录行中字段数量一致，即一个字段对应配置一个<field/>字段校验器子节点，前后顺序也要和字段在数据记录行中的数序一致，而且必须这样配置。

其结构大致为：

<fields>

<field>

…

</field>

<field>

…

</field>

…

</fields>

##### <field/> 子节点

<field/>子节点是用来配置文件记录行中各个具体字段校验配置信息的，数据记录行中每个字段都单独对应一个自己的字段校验器field（或域校验器），是必须要配置的子节点。

<field/>子节点又包含多个属性和子节点：

<field/>子节点包含属性为<name />、<regx/>、<unique/>、<begin/>、<end/>、<fixValue/>、<related/>、<passParam/>；

<field/>子节点包含子节点为<component/>、<cases/>。

###### <name /> 属性

<name />属性用于配置字段名，是必须配置的属性

###### <regx/> 属性

<regx/>属性用于配置正则表达式，是此field校验器要校验的字段需要满足的正则表达式规则，是必须要配置的属性。

<regx/>属性中需要配置包含开始和结束标识符的正则表达式，如：

<regx> ^\d{0,3}$</regx>

其中“^”为开始标识符，“$”为结束标识符。

###### <unique/> 属性

<unique/>属性是用于配置所校验字段是否需要在所在文件记录中唯一的标识，需要根据需求来配置。

若对应字段为唯一，<unique/>属性需要配置“yes”，否则不需要配置。

比如文件记录中有不大于3位的数字字段A需要满足每条记录的A字段内容在所在文件中都是唯一的，那么需要如下配置：

<field>

<name>A</name>

<regx>^\d{0,3}$</regx>

<unique>yes</unique>

</field>

###### <begin/> 属性

<begin/>属性是在文件记录行字段是“fix”（定长）模式下，所在field需要校验字段的在记录行中的起始位置，用于从记录行中截取出对应字段的值。<begin/>属性在定长模式下是必须要配置的属性。

###### <end/> 属性

<end/>属性同<begin/>属性，是在文件记录行字段是“fix”（定长）模式下，所在field需要校验字段的在记录行中的结束位置，用于从记录行中截取出对应字段的值。<end/>属性在定长模式下是必须要配置的属性。

比如需要校验的字段A是fix模式的文件记录行中第0至第3位（注意起始位置从0开始算，结束位置不算入字段长度内），那么需要如下配置：

<field>

<name>A</name>

<regx>^\d{0,3}$</regx>

<begin>0</ begin >

<end>3</end>

</field>

若此时正在校验的记录行内容是“123456”，则对应A字段内容会是“123”

###### <fixValue/> 属性

<fixValue/>属性如之前3.1.3.8.7.1小节的简单介绍一样，<fixValue/>用于配置field校验器需要校验的字段需要满足的固定值条件。<fixValue/>属性可以根据需求配置。

实际上通过<regx/>属性的正则表达式也能实现字段的固定值匹配校验，<fixValue/>属性只是提供了另一种方式，而且只做固定值匹配。<fixValue/>主要用在条件匹配。

###### <related/> 属性

<related/>是用来配置是否参与多字段逻辑校验的标识，是搭配<field/>中的子节点<cases/>来使用的。<related/>属性需要按照需求来配置，若对应字段参与了多字段需要参与某逻辑校验case，则对应field需要如下配置：

<related>yes</related>

逻辑校验器<cases/>的具体配置方法以及与<related>属性的搭配使用方式会在下面的讲解<cases/>小节中详细说明。

###### <passParam/> 属性

<passParam/>属性是用来配置需要校验的字段是否需要和流程传入参数做匹配校验，需要根据需求进行设置。

目前的业务流程下，流程的传入参数主要有“settleDate（帐期日）”、“busiLine（业务线）”、“province（省份编码）”以及在流程文件XML中自定义配置的参数。若需要配置<passParam/>属性，则需要配置“yes”字符串。比如“fix”类型的文件数据记录行中，第0至第4位之间是业务线字段，需要和传入的业务线参数进行匹配校验，则需要如下配置：

<field>

<name>busiLine</name>

<begin>0</ begin >

<end>4</end>

<passParam>yes</passParam>

</field>

注意，配置<passParam/>属性时，field的<name/>属性中一定要配置传入参数的名称，而不能自动以名称，不然无法从传入参数map中找到对应名称的key，也无法取得value，匹配校验就无法进行。

###### <component/> 子节点

<component/>子节点用于配置需要校验的字段中包含的子字段校验信息，需要按照需求进行配置。

<component/>子节点中又可以配置<compFields/>子节点。

<compFields/> 子节点

<compFields/>子节点用于保存需要校验的字段中包含的子字段校验器集合，需要根据需求配置。

子字段校验器也是复用<field/>子节点来配置校验信息。<compFields/>子节点中可以配置多个<field/>子节点，因为一个字段中可能有多个子字段。

<compFields/>中配置<field/>的结构如下：

<component>

<compFields>

<field>

<name>settleDate</name>

<begin>0</begin>

<end>8</end>

<passParam>yes</passParam>

</field>

<field>

……

<field>

…….

</compFields>

</component>

<field/> 子节点

<field/>子节点在<compFields/>中配置属于配置子字段校验器，需要根据需求配置。

<compFields/>中配置的<field/>子节点又可以配置多种属性和子节点，且与field校验器的配置方式一样，这样就形成了类似递归的处理逻辑。<field/>子节点的具体配置方式也就是3.1.3.9.1小节，即本章节所描述的配置方式。

<component/> 配置实例

比如，现在需要校验字段A，是32位纯数字，其中前3位是省份编码，需要与流程参数做匹配校验，第3至11位的ODER\_NO字段是纯数字，且是唯一性校验字段。那么可以如下配置：

<field>

<name>A</name>

<regx/>^\d{32}<regx/>

<component>

<compFields>

<field>

<name>province</name>

<begin>0</begin>

<end>3</end>

<passParam>yes</passParam>

</field>

<field>

<name>ODER\_NO</name>

<begin>3</begin>

<end>11</end>

<unique>yes</ unique >

</field>

</compFields>

</component>

</field>

更多配置方式可以根据需求配置。

###### <cases/> 子节点

<cases/>子节点用于保存需要校验的字段会参与的不同字段间逻辑关联校验器的集合。<cases/>子节点也是需要根据需求来配置。

<cases/>子节点又包含多个<case/>子节点，每个<case/>子节点标识对应字段参与的一种逻辑关联校验，其配置结构如下例子：

<cases>

<case>

<caseName>USER\_ID\_TYPE\_case1</caseName>

<regx>^01$</regx>

<relaField>

<name>USER\_ID\_NO</name>

<regx>^1\d{10}$</regx>

</relaField>

</case>

<case>

······

<case>

······

</cases>

<case/> 子节点

<case/>子节点用于配置需要校验的字段会参与的不同逻辑关联校验场景，如上文3.1.3.9.1.7小节所述，需要搭配<related/>一同配置。<case/>子节点的需要根据需求配置，其配置较复杂。

<case/>子节点包含属性<caseName/>、<regx/>、<fixValue/>、<massage/>、<isEqual/>；

<case/>子节点包含子节点<relaField/>。

<caseName/> 属性

<caseName/>属性用于保存其所在的逻辑关联校验器的别名或代号，是case中必须要配置的，建议配置一个有意义且易于理解的别名。

<regx/> 属性

<case/>子节点中的<regx/>属性与<field/>子节点中的<regx/>属性类似，都是用来存储正则表达式的，也是必须要配置的。

<case/>子节点中的<regx/>属性的意义是在当前的逻辑关联校验场景下，所校验的字段需要满足的规则，并将这些规则转换成正则表达式的形式进行匹配校验，可以理解为进行当前逻辑关联校验case的前提条件。若校验字段满足<case/>子节点<regx/>属性中的正则表达式，则说明所校验字段已满足条件，需要进行当前<case/>的逻辑关联校验；若不匹配正则表达式，则不进行执行的<case/>逻辑关联校验。

<massage/> 属性

<massage/>属性用来配置逻辑关联校验器的简单说明，可以根据需要进行配置。

<isEqual/> 属性

<isEqual/>属性是用来配置当前字段是否与其做关联校验的字段进行值相等匹配校验的标识位，需要根据需要进行配置。

<isEqual/>的使用场景为：参与逻辑关联校验的另一字段要与当前字段内容值相等，需要在当前的case中配置<isEqual/>属性为“yes”。

<isEqual/>属性需要与下面的<relaField/>子节点搭配使用才能完成字段等值逻辑校验，集体配置方法下边会给出。

<relaField/> 子节点

<relaField/>子节点是配置当前逻辑关联校验case中，参与的另一方字段的信息。<relaField/>子节点是必须要配置的。

在上一小节“3.1.3.9.1.10.1.4小节<isEqual/> 属性”中说道，<isEqual/>属性需要搭配<relaField/>子节点使用，比如：

需要校验的文件记录行（类型为“sep”）中，有字段A、B，其中字段A为不大于32位纯数字，且当A满足值为“123”时，B字段的内容要与A字段的内容相等。

以上就是一个比较基本的字段间等值逻辑关联校验的场景，具体配置如下：

<field>

<name>settleDate</name>

<regx>^\d{0,32}$</regx>

<related>yes</related>

<cases>

<case>

<caseName>A\_case1</caseName>

< regx >^123$</ regx >

< isEqual>yes</ isEqual>

<relaField>

<name>B</name>

</relaField>

</case>

</cases>

</field>

<relaField/>子节点又包含多个属性和子节点，具体为：

<relaField/>包含属性<name/>、<regx/>、<unique/>、<fixValue/>、<passParam/>；

<relaField/>包含子节点<component />、<cases />。

<relaField/>实际上和子节点<field/>类似，其包含的属性都能在<field/>子节点中找到，而且以上这些属性的代表意义与配置实用方式与<field/>子节点同名属性是一样的，可以查看<field/>子节点中对应的同名属性或子节点的介绍，这里就不在赘述。

<cases/> 配置实例

需要校验的文件记录行（类型为“sep”）中，有字段A、B、settleDate，其中字段A为不大于32位纯数字，且当A满足值为“123”时，B字段的内容要与A字段的内容相等；当A满足值为“456”时，settleDate字段需要与流程传入参数“settleDate”相等。

以上就是一个比较复杂的字段间等值逻辑关联校验的场景，具体配置如下：

<field>

<name>settleDate</name>

<regx>^\d{0,32}$</regx>

<related>yes</related>

<cases>

<case>

<caseName>A\_case1</caseName>

< regx >^123$</ regx >

< isEqual>yes</ isEqual>

<relaField>

<name>B</name>

</relaField>

</case>

<case>

<caseName>A\_case2</caseName>

< regx >^456$</ regx >

<relaField>

<name>settleDate</name>

<passParam>yes</passParam>

</relaField>

</case>

</cases>

</field>

## 模板文件使用说明

校验模板XML文件有两种放置方式，在旧架构中是以配置文件的形式放置和使用的，而在新机构（即产品化架构）中是保存在文件处理模块的信息配置表中。

### 旧架构检验模板为配置文件

旧架构的校验模板和入库模板都是以配置文件的方式存储和使用的，比如在能力开放平台业务中，支付宝文件的校验模板存放路径就为：/cmosp-clearing-osp/src/test/resources/templates/validationTpl/CmospAlipayRegxFile.xml。

而在代码中具体调用模板时，需要根据其包相对路径进行调用，比如能力开放平台业务中支付宝文件校验模板的调用路径为：templates/validationTpl/CmospAlipayRegxFile.xml。

### 产品化架构下存储在数据库

产品化架构下，校验模板和入库模板都已经存入了数据库的配置表中。校验模版存储位置具体在FILE\_SERVICE\_INFO\_PRO中的REGX\_TPL字段中，REGX\_TPL字段以CLOB格式存储模板数据。

# 总结

现阶段校验模板的配置，主要还是两个方面：对校验逻辑的抽象和对模板支持的理解。当需要校验某个文件时，首先要收集文件中精确到每个字段的校验规则，整理出一份校验规则文档，用最简单的方式将规则表达出来。做完这一步其实就已经完成了校验逻辑的抽象，下一步只不过是将这些表达转化成正则表达式或者各种模板支持的逻辑校验器的配置。而在校验逻辑的转化过程中，需要对校验模板的各种功能有清晰的理解，理解越透彻，通过模板配置实现的功能越强大，比如可以熟练配置文件全局的多字段之间的四则运算逻辑校验。

无论是现阶段的产品化架构，还是未来的微服务化平台化架构，清洁算系统的文件校验模块对于文件字段校验规则的配置都是一个重点。用这种可配置的、模块化的、功能强大的校验模板形式去存储和执行文件校验，是一种比较敏捷的实现方式。当然，现如今的配置方式还是太复杂，而且未来随着新技术的不断接入和新架构不断衍生，这种方式会被更简易的，功能也更强的实现方式替换掉。

# 附录

## 高级功能校验模板示例——逻辑关联校验

互联网返利业务的互联网公司对账文件的校验中，USER\_ID\_TYPE字段就通过配置与USER\_ID\_NO字段进行了逻辑关联校验，具体校验规则可见互联网公司对账文件的校验规则文档。下边给出相应的校验模板附件。



## 高级功能校验模板示例——子字段单独校验

在能力开放平台的浙商对账文件校验中，通过配置子字段校验器实现了对TRADE\_START\_TIME字段的前八位做出帐期日匹配校验，如下校验模板中所示：



## 高级功能校验模板示例——逻辑运算校验

在天猫商城业务对天猫侧对账文件的校验中，通过配置实现了当BUSS\_TYPE字段值为“10”时，日期格式下RESERVED1字段比SETTLE\_DATE字段要小，且差值在30以内的逻辑大小运算校验，如下校验模板所示：



## 文件返回码

|  |  |  |
| --- | --- | --- |
| **返回码** | **定义** | **错误级别** |
| **F000** | 对平记录 | **成功返回码** |
| F102 | 文件格式错误 | 文件级返回码 |
| F104 | 文件名校验不通过 |
| F105 | 头记录校验不通过 |
| F201 | 对账文件在规定时间未到（返回空的错误文件） |  |
| F202 | 对账文件为空（大小为0） |  |
| F203 | 文件头记录总数与文件体不一致 | 文件级返回码 |
| F106 | 用户身份不存在 | 记录级返回码 |
| F107 | 银行账户信息不存在 |
| F108 | 用户身份异常 |
| F109 | 银行账户状态异常 |
| F110 | 不存在该用户的签约关系 |
| F111 | 银行账号余额不足 |
| F112 | 签约协议号不存在 |
| F113 | 统一支付系统有BOSS无 |
| F114 | BOSS有统一支付系统无 |
| F115 | 记录不一致 |
| F116 | 银行已扣款省BOSS未销账 |
| F117 | 银行未扣款省BOSS已销账 |
| F118 | 批扣成功 |
| F119 | 超过累计缴费门限 |
| F120 | 该签约信息已经存在 |
| F121 | 该用户已经签约其它账户 |
| F122 | 该笔交易不存在 |
| F123 | 批扣失败 |
| F124 | 记录级格式错误 |
| F125 | 该笔交易记录已退款 |  |
| F126 | 对账文件存在重复记录 |  |
| F127 | 对账文件存在空行 |  |
| F128 | 对账文件字段缺失 | 记录级返回码 |