SWS 自动脚本

SWS (STDF Workshop Script)脚本是 STDF Workshop 提供的简单脚本功能，可以用来编程并实现批量修改 STDF。在脚本编辑区修改脚本时，编辑器会自动设置不同的颜色显示。

紫色：内置命令

蓝色：内置对象/方法/属性

黑色：变量/常数等

SWS 使用流程如下：

- 编辑 SWS 脚本
- 编译 SWS 脚本并确认没有错误
- 打开文件(可以选择多个文件)执行编辑好的 SWS 脚本
- 或者打开文件夹执行编辑好的 SWS 脚本

注意：所有编辑好的 STDF 文件都会被另存到新的目录

可以将编辑好的 SWS 脚本另存为.sws 文件，然后用 exec_sws.exe 和 windows 计划任务后台定时执行 sws 脚本来实现自动 monitor 目录并把目录中的 STDF 自动修改并另存到指定的目录。(参看下一节内容)

SWS 语法介绍：

SWS 是一种简单脚本，通过提供的有限命令来实现自动打开文件，修改属性，执行方法和另存为的功能，同时提供了 FOR_EACH 循环和 IF 条件判断。SWS 的格式要求非常严格，所以需要用户严格按照要求编写脚本，建议每次都是在示例脚本的基础上修改而不是从完全从头编写。

重要：所有 SWS 命令都是大写字母，并且在命令、赋值符号、比较符号、数值之间必须要有空格。

FOR_EACH_FILE f

文件循环命令，循环每个 STDF 文件，f 代表 STDF 文件 (建议不要修改这个循环变量)，最后会以 **END_FOR_EACH** 结束。这个命令是 SWS 的最外层命令，整个 SWS 脚本必须以“**FOR_EACH_FILE f**”开始，并以“**END_FOR_EACH**”结束。在这个循环内部更新需要修改的记录的属性和执行相应的方法。

EXECUTE_METHOD

执行方法命令，在 SWS 中 STDF 文件“ f “ 提供可执行方法，用来做文件解析和文件另存为的操作等。每一种 Record 也有一个 r.Delete()方法可以执行。

f.Extract()　解析 STDF 文件，这个方法没有参数，这个方法一般是 FOR_EACH_FILE 循环里面的第一条需要执行的命令。

f.SplitFileName(“_”) 拆分文件名并把文件名拆分的内容保存在 f.FileNameFields 数组中，以便可以把文件名的特定字段的内容写入 STDF 的 MIR 或者其他的记录字段中。需要把文件名的分隔符通过参数传入方法。

注意：在调用 f.FileNameFields[3] 数组时，需要用户自己考虑 index 指向正确数组位置，确保 index 不会超出索引范围。

~~f.Repair() 方法可以用来修复不完整的 STDF, 然后执行 SaveAs()方法把修复好的 STDF 文件保存到新的目录中。~~

f.RepairAndSave(“C:\share”) 是用于修复 STDF 的新的方法，f.Repair()已经不可用了。新的方法直接修复并保存到目标目录，不需要再调用 f.SaveAs()方法。

f.UpdateSummaryRecords() 方法用来重新计算 HBR/SBR/PCR/TSR 的数量，并且更新到 STDF 中去，然后执行 SaveAs()方法把修复好的 STDF 文件保存到新的目录中。

f.SaveAs(“folder_name”) 把修改后的 STDF 另存到新的目录中(原来的 STDF 文件不会被修改), 需要把目录通过参数的形式传入。也接受没有引号的参数 f.SaveAs(folder_name)。确保在 FOR_EACH_FILE 循环内的最后一条命令是 f.SaveAs 命令，否则修改好的 STDF 不会被保存。

所有 RECORD 都支持 r.Delete()方法，用来删除当前记录。

调用外部接口获取 lot 信息并写入 STDF → f.GetInfoFromExternalInterface()

有时候需要根据 lot_id 或者其他信息从外部接口(例如：数据库，web server 等)查询并获取相关信息，比如产品名，handler id 等信息。再写入 STDF 文件。

这时候需要准备 External Interface DLL, 用户可以参看 STDF Workshop 安装目录下的“dev”目录的源代码，更新 SEIL.dll 从外部获取信息，然后返回 Dictionary<string, string>字典。再 sws 脚本中可以调用接口并传递参数来获取外部返回的字典，然后用这些信息更新 STDF.

在 sws 中调用编辑好的外部接口 SEIL.dll (位于 interface 目录下)，需要类似下面的代码，先传递参数(多个参数需要传递多次)，然后再调用 GetInfoFromExternalInterface()方法从外部获取信息字典，信息会保存在 f.ExternalInfoDict 中，再用字典里面的信息更新 STDF 相关记录的字段。由于返回信息是字典，所以在指定 Key 时请自行确认 key 是否存在，如果不存在的话，sws 执行会报错。

```
EXECUTE_METHOD f.AddExternalInterfacePara("EXLOT")       #Arg1 defined in external interface lib
EXECUTE_METHOD f.AddExternalInterfacePara("HOTCOLD") #Arg2 defined in external interface lib
EXECUTE_METHOD f.GetInfoFromExternalInterface()          #Get information from External interface
UPDATE_PROPERTY f.MIRs[0].LOT_ID = f.ExternalInfoDict["LOT"]
UPDATE_PROPERTY f.MIRs[0].PART_TYP = f.ExternalInfoDict["PRODUCT"]
```

UPDATE_PROPERTY

更新属性值，这个命令用来更新指定记录(Record)的属性值。STDF 解析之后在 STDF Workshop 中都是以记录数组的方式存储的，所以修改 STDF 其实就是修改指定记录的属性。

一般情况下每个 STDF 只有一个 MIR 和一个 MRR，所有修改 MIR 记录的属性时，可以直接指定 index 为 0 (也就是数组中的第一个记录)，比如：f.MIRs[0].LOT_ID=”new lot id”。

STDF 中有很多记录的数量非常庞大，所以在修改属性的时候需要用到 FOR_EACH_REC 循环和 IF 条件判断来实现。

设置记录属性的值，可以直接在 SWS 中使用字符串常数或者也可以指定其他属性值(目前只支持使用文件名拆分数组 f.FileNameFields[3])

Note: 关于 STDF 有哪些记录，每种记录有哪些属性，需要用户对 STDF 的结构有一些了解，可以同时参照此帮助文件的第一章 STDF 结构和第四章 详解记录修改。

在 STDF Workshop 中同类型的记录会被保存在同样的记录数组中，记录数组的名称都是记录类型后面加一个 ”s”, 例如：MIRs, PIRs, PTRs, HBRs 等。

FOR_EACH_REC

记录循环命令，这个命令用来循环记录数组中的每一条记录，在这个循环中可以修改每条记录的属性，或者借助 IF 条件判断来修改符合条件的记录的属性。建议用 r 来作为循环变量代表每个记录。这个循环命令同样以 END_FOR_EACH 命令结束.

记录循环嵌套，FOR_EACH_REC 可以嵌套以实现复杂的功能，每个 FOR_EACH_REC 的循环变量必须不一样，否则编译会出错。

示例代码（scripts\tsr_255_updates.sws）：用来把 TSR 记录中 SITE_NUM!=255 的记录的 EXEC_CNT 和 FAIL_CNT 复制到同样 TEST_NUM 的 SITE_NUM==255 的记录中去，并且把 SITE_NUM==255 的记录的 TEST_NAM 复制到 SITE_NUM!=255 的对应记录中(注意循环变量 r，t)。

```
FOR_EACH_REC r IN TSRs
        IF r.SITE_NUM == "255"
            FOR_EACH_REC t IN TSRs
                IF t.TEST_NUM == r.TEST_NUM && t.SITE_NUM != "255"
                    UPDATE_PROPERTY t.TEST_NAM = r.TEST_NAM
                    UPDATE_PROPERTY r.EXEC_CNT = t.EXEC_CNT
                    UPDATE_PROPERTY r.FAIL_CNT = t.FAIL_CNT
                    UPDATE_PROPERTY r.HEAD_NUM = "255"
                END_IF
            END_FOR_EACH
        END_IF
END_FOR_EACH
```

IF 条件判断命令，这个命令用来判断条件，一般是用来检查记录的属性是否符合特定条件，来筛选记录，以便修改符合条件的记录的属性。请务必在条件判断符号左右都留空格，支持的条件判断符号如下：== , != , > , >= , < , <=。 **IF** 命令最终以 **END_IF** 命令结束。

```
IF  f.MIRs[0].RTST_COD != "R"
```

多条件 IF 语句，IF 语句支持多个条件可以用 AND(&&)或者 OR(||)连接，其中&&的优先级高于||的优先级，所以在多条件语句中&&会优先计算，最后在计算||来完成所有的条件判断。

优先级示例：A && B || C && D　*（实际代码）*

→　*(A && B) || (C && D)*　*（执行顺序，实际代码中不支持括号）*

实际代码演示（*scripts/default.sws*）：

```
IF f.MIRs[0].MODE_COD == "D" && f.MIRs[0].PROT_COD == "E"
    UPDATE_PROPERTY f.MIRs[0].RTST_COD = "R"
END_IF
```

说明：SWS 的编译功能并不能检查出所有的语法和逻辑错误，需要用户在编写 SWS 脚本的时候特别留心。如果有任何问题可以联系 support@nornion.com. 在 STDF Workshop 的安装目录下的 scripts 文件夹中可以找到我们提供的 sws 常用实例脚本。

利用计划任务自动执行 SWS 脚本

既然 STDF Workshop 可以执行 SWS 脚本自动修改 STDF，是否可以设置 Windows 计划任务自动执行 SWS 脚本呢？答案是肯定的，只需要利用我们提供的 exec_sws.exe (SWS 脚本执行器)就可以实现这个功能。

exec_sws.exe 是一个命令行工具，调用它的时候需要指定一些参数，可以使用 windows 计划任务执行这个命令配上相应的参数即可。也可以把调用的命令保存在 BAT 文件中，然后用计划任务执行这个 BAT 文件 (可以参看 STDF Workshop 安装目录下的 exec_sws.bat)

exec_sws.exe -s .\scripts\example.sws **-d** c:\temp\input **-a** c:\temp\archive **-e** c:\temp\error **-g** c:\temp\log

参数介绍：

-s 用来指定 sws 脚本文件，路径中如果有空格请使用 “ ”

-d 用来指定 STDF 输入目录，脚本执行器会从这个目录寻找文件并为每个文件执行 sws 脚本

-a 用来指定 STDF 文件 archive 目录，原来的 STDF 文件会被备份到此目录

-e 用来指定错误目录，STDF 执行 sws 时如果遇到错误，STDF 会被移动到这个目录

-g 用来指定 log 的目录，这个参数可选，如果指定，sws 执行器会把详细 log 写入这个目录

注意：

STDF 输出目录(修改后的保存目录)是在 sws 脚本中定义的 *f.SaveAs(“c:\temp\output”)*。在调用 exec_sws.exe 的时候参数指定的顺序可以随意。

在创建计划任务的时候，请务必设置其“**起始目录**”为 STDF Workshop 的安装目录，exec_sws.bat 也需要放在这个目录。当然 sws 脚本可以放在任何地方，只要用 ”-s” 指定完整路径即可。