**Ftp-Parser-Framework总体说明文档**

**浪潮通信信息系统有限公司文档修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **更改人** | **描述（注明修改的条款或页）** |
| V0.5 | 2012/7/30 | 武玉刚 | 文档创建 |
| V0.6 | 2012/8/2 | 武玉刚 | 1. 更改了bin目录下的startup.sh脚本，修正了因全角冒号导致的classpath失效问题。 2. 修改了第三方ftp工具commons-net-2.0.jar，使其能够支持unix ftp中文环境的问题。 |
| V0.7 | 2012/8/13 | 武玉刚 | 1. 修改了bin目录下status.sh和shutdown.sh脚本，解决部分unix下无法使用的问题。 2. 完善了ftp文件下载机制，通过向ftp服务器上传文件的方式确定ftp服务器最新时间，下载20秒前3600秒内ftp服务器上生成的文件。 |
| V0.8 | 2012/9/5 | 武玉刚 | 1. 修改了第三方ftp工具commons-net-2.0.jar，解决了ftp服务器上文件名存在中文的问题（在内蒙古部署时遇到该问题）。 2. 完善了ftp文件下载机制，对每个下载线程进行超时监控，对于超过半小时（该值可调）仍然没有完成下载的线程进行主动重启，避免网络瞬断等原因导致的ftp下载线程僵死的问题（广西、山西、江苏反映过该问题）。 |
| V0.9 | 2012/9/6 | 武玉刚 | 1. 文档中增加了对于配置文件的说明。 2. 文档中增加了程序处理流程图，包括FTP采集流程和文件解析流程。 3. 修改了删除sqlldr入库过程中产生的过程文件的方法，只在每天的0、6、12、18点进行文件删除，避免每次都检索大量文件导致时间浪费，入库效率降低。(解决江苏反映的随着程序运行时间增长文件入库效率较低的问题) |
| V1.0 | 2012/10/23 | 武玉刚 | 1. 增加了ftp上传功能。 |
| V1.1 | 2012/11/20 | 武玉刚 | 1. 修正了部分程序中文件流未通过finally关闭的问题，避免程序异常时不释放文件流，导致文件打开过多出错（linux下的 too many open files 问题）。 2. 修正了中文环境下ftp下载时无法下载11、12月份文件的问题。 3. 优化了ftp线程超时监控机制，一旦有文件下载成功ftp线程的启动时间即更新为当前时间。 4. 优化了ftp下载机制，增加了文件大小为0的判断，当待下载的文件大小为0时不下载，当下载下来的文件大小为0时不记录到下载成功的列表中。 5. 增加了ftp手动补采功能。 6. 优化了文件解析判断机制，对于大小为0的文件不解析。 7. 补充了第8节Q&A部分。 |
| V1.2 | 2012/11/20 | 武玉刚 | 1. 完善了FTP下载时的日志信息，增加了源文件和目的文件的文件名、大小、时间等信息。 2. 完善了FTP补采功能，使之能够兼容老版本。 3. 规范了日志记录情况，使用源文件名标识处理过程中的日志信息。 4. 完善了已处理文件的记录信息，增加了记录时的时间。 |
| V1.3 | 2012/12/10 | 武玉刚 | 1. 完善了gunzip解压缩的部分，由于java自带的解压缩类存在bug导致部分gz文件解压不了。在通过java自带的类解压缩失败时调用系统的gunzip命令进行文件解压。 2. 在文件轮询处理时，每次开始之前增加了临时文件清理扩展点接口，清除解析过程中产生的临时文件，避免程序重启时一些过程文件被解析而引发错误。 3. 日志规范化：   (1)增加了对于sqlldr入库时成功条数和失败条数的解析，支持对sqlldr英文日志和中文的解析。  (2)日志规范化：按照今天制定的《FPF框架日志输出标准化设计.doc》对处理过程中的日志进行了规范化输出。  (3)日志规范化：进一步规范了FTP下载日志，增加了对FTP下载日志的解析功能。 |
| V2.0 | 2013/1/22 | 武玉刚 | 1. 对整体框架进行了重构，升级到2.0版本，在北京TD性能和TD参数解析中使用，运行稳定高效。 2. 更新了该文档中流程图及安装介质。 |
| V2.1 | 2013/2/18 | 武玉刚 | 1. 修正了删除备份的原始文件删不全的bug。 |
| V2.1 | 2013/2/25 | 武玉刚 | 1. 修改了commons-net-2.0.jar包，解决湖南综分下载文件是某个FTP服务器列出的文件时间月份和天之间没有空格的问题。 2. 修改了该文档的模板，修改了页眉部分，使用FPF图标作为页眉。 |
| V2.2 | 2013/3/8 | 武玉刚 | 1. 增加了通过RMI接口查看FPF状态的能力。 2. 增加了从配置文件中取ftp超时时间ftp.timeOut的能力，如果没有配置，则使用系统默认的时间，默认为300秒超时。 3. 将文档中的程序zip包更新为2013-3-8最新的程序。 4. 增加了FPF日志表建立的说明。 |

# 框架简介

fpt-parser-framework是一个轻量级多线程调度框架，用于实现数据采集中常见的ftp文件采集及解析、入库的整体调度和控制。其主要功能包括：

1. **FTP轮询采集：**

轻量级多线程文件下载框架，该框架可为每一个FTP服务器即时开启一个采集线程，每个线程负责扫描其负责的FTP服务器上是否有新文件生成，如果有新文件生成且尚未进行下载的则该线程自动进行下载处理，每次扫描后该线程sleep 60秒（可配置修改）再进行下一轮扫描，以此来保证一旦服务器上有新文件生成就能很快的下载下来，减少了文件漏采的几率。

该框架会开启一个FTP采集超时监控器，对于由于网络瞬断、FTP服务器问题等导致的FTP采集线程僵死的情况（单次采集历时超过30分钟），则主动停掉原来的线程并重新启动一个线程，以此来保证文件的正常及时下载。

该框架能够提供基于《FPF框架日志输出标准化设计.doc》要求的规范化下载日志输出。

1. **文件解析入库：**

轻量级多线程文件处理框架，该框架能够维持一个线程数可配置的线程池，线程池中的每个线程用于处理一个文件，而每个线程可以提供包括：临时文件清理、判断文件是否需处理、调用扩展点配置的解析入库程序对文件进行处理、备份原始数据文件、记录已处理的文件列表、删除过期数据文件、删除过期的中间文件等文件处理过程的基本动作。在此框架的基础上，针对具体某个厂家的某类文件进行处理时，可以将焦点集中在通过扩展点配置其具体的解析程序上面，而不需要关注其他辅助的处理环节。

该框架能够提供基于《FPF框架日志输出标准化设计.doc》要求的规范化文件处理日志输出。

该框架对于使用者提供三个扩展点以便适配不同的文件处理场景，包括：

* 1. 临时文件清理扩展点接口(com.inspur.ftpparserframework.extend.ItempFileCleaner)：

用于使用者根据自身情况删除文件处理过程中产生的临时文件，该扩展点在每次轮询扫描待处理文件时执行，用于清理临时文件，以便在程序意外停止或手动重启时保持一个干净的运行环境，保证程序的正常运行。框架提供的默认临时文件清理器（com.inspur.ftpparserframework.extend.DefaultTempFileCleaner）可以用于递归的删除待处理文件夹中以.tmp或以.temp结尾的文件。

* 1. 文件过滤扩展点接口(com.inspur.ftpparserframework.extend.IFileFilter)：

用于使用者根据自身情况对待处理的文件进行过滤。框架提供了基础的文件过滤功能，主要实现对文件的基础过滤处理，包括大小为0的文件不处理、已处理的文件不重复处理、近10秒内生成的文件不处理（避免文件在生成的过程中被扫描处理）。使用者实现的过滤器是在基础过滤的基础上做的叠加。

* 1. 文件解析入库扩展点接口(com.inspur.ftpparserframework.transformator. ITransformator)：

用于使用者根据自身情况实现对文件主体的解析和入库，其中入库时推荐使用框架中提供的Sqlldr入库工具com.inspur.ftpparserframework.util.OracleUtil来实现。

注意：使用者在基于该框架进行扩展开发时，必须按照《FPF框架日志输出标准化设计.doc》的要求进行规范化的日志输出。

1. **实用工具：**
   1. **默认的临时文件清理器：**能够递归的删除待处理文件夹下以.tmp或以.temp结尾的文件(需同时满足文件生成10秒以上，避免正在ftp下载的文件被删除)，清理掉各文件解析器生成的中间文件。
   2. **默认的文件过滤器：**过滤掉大小为0的文件、已处理过的文件、近5秒内生成的文件、以.tmp或以.temp结尾的文件。
   3. **文件解压工具**：能够对gzip压缩格式的文件进行解压缩处理。
   4. **ORACLE SQLLDR数据入库工具**：能够基于外部传递的数据库表、列和csv格式的数据文件通过Sqlldr进行数据加载，同时能够解析其log日志，提取数据加载成功和失败的条数，并基于《FPF框架日志输出标准化设计.doc》进行标准日志输出。
   5. **规范化日志解析工具**：能够对程序中基于《FPF框架日志输出标准化设计.doc》要求输出的文件解析入库日志进行解析，解析结果可以输出为csv文件，包括：
      1. 文件总体解析日志，命名如parse\_yyyyMMddHHmmss.csv，日常主要看这个解析结果，可以以透视图的方式分析各网元对应的日志解析情况。
      2. 文件中间转换日志，命名如middle\_yyyyMMddHHmmss.csv，体现的是文件在从原始文件到最终入库的过程中进行的多次格式转换情况。
      3. sqlldr执行及分析日志，命名如sqlldr\_yyyyMMddHHmmss.csv，体现的是数据通过sqlldr加载到Oracle数据库的成功和失败情况。
   6. **规范化日志入库工具**：能够对程序中基于《FPF框架日志输出标准化设计.doc》要求输出的日志自动入库，以便于后续分析。

# 安装介质





# 环境要求

1. java环境为jdk 6及以上版本

# 目录结构及说明

1. **bin：** 用于存放该框架运行所需的shell脚本，该目录下的shell脚本是使用该框架的其他程序所必须的脚本，不可改动。包括：
   1. startup.sh：启动进程。
   2. status.sh：已启动进程状态查看。
   3. statusrmi.sh：通过RMI接口查看在指定端口启动的FPF的内部状态。
   4. shutdown.sh：停止已启动的进程。
2. **conf\_template：**用于存放该框架所需的配置文件的模板，该目录下的文件在程序运行时不会起作用，仅是作为配置模板供参考使用。包括：
   1. SystemConfig.xml：程序的系统配置文件样例。
   2. FtpServerConfig.xml ：FTP连接信息配置文件样例。
   3. log4j.xml：日志记录配置文件样例。
3. **lib：**用于存放该框架所需的所有jar包。

# 部署方法

1. 建立目录：登陆数据采集服务器，在相应目录下建立etl目录（如果已经建立则不需要重复建立），用于存放数据采集工具及各厂商采集程序。
2. 安装介质上传及解压：将ftp-parser-framework-yyyyMMdd.zip上传到数据采集服务器上etl目录下，执行命令unzip ftp-parser-framework-yyyyMMdd.zip即可解压缩。
3. 添加shell脚本的执行权限：进入ftp-parser-framework/bin 目录，执行 chmod +x \*.sh
4. 检测部署情况：进入ftp-parser-framework/bin 目录，执行./startup.sh，看到如下提示信息则表明部署成功。

命令说明: ./startup.sh [-ftp y|n] [-parse y|n] [-rmiport portNo][-input inputDir] [-server svr1,svr2,svr3] [-debug y|n]

该框架是用于实现ftp文件下载及解析入库的调度框架。

命令选项解释:

-h 帮助

-ftp 可选项。是否启动FTP下载 (y 是|n 否，默认为n否)。

-parse 可选项。是否启动文件解析(y 是|n 否，默认为n否)。

-rmiport 可选项。启动RMI服务所需的端口号，如果不指定则不启动RMI服务。

-server 可选项。本次需要采集和解析的服务器信息,多个服务器用半角逗号隔开，不设置该参数则默认处理所有服务器的数据。

-input 可选项。待解析文件路径(默认为程序根目录下的../data/input目录)。

-debug 可选项。是否调试状态(y 是|n 否，默认为n否),启动调试状态后解析过程中产生的中间文件将不会删除。

1. 建立FPF标准日志表：连接到需要建表的Oracle数据库，执行《FPF日志分析建表语句-V2.0.sql》中的语句即可。

# 其他

1. 关于第三方FTP工具，该框架使用的FTP是基于Apache的commons-net-2.0.jar进行封装和开发的，但是由于commons-net-2.0原始的版本不支持中文FTP环境，因此武玉刚对该jar包的源码进行了一定的修改，主要是将其默认的码流编码格式由UTF-8改为了GBK，使之能够支持中文FTP环境。

# 附录

1. 配置文件说明
   1. SystemConfig.xml：程序的系统配置文件样例。

<?xml version="1.0" encoding="gb2312"?>

<!--说明：项目现场需要修改dbuserid参数，其他标记为“高级”的参数一般不需要修改。-->

<Config>

**<!-- FTP文件下载相关参数 -->**

<ftp>

<!-- 高级：ftp采集轮询间隔时间，影响文件采集及时性；单位秒，默认60秒；建议配置为文件生成周期的1/4。-->

<scanInterval>60</scanInterval>

<!-- 高级：ftp采集最近多长时间内生成的文件，单位秒，默认1小时。-->

<fileLastModifyTime>3600</fileLastModifyTime>

<!-- 高级：ftp采集线程超时时间,单位秒 ，默认5分钟-->

<timeOut>300</timeOut>

</ftp>

**<!-- sqlldr入库时所需的信息 -->**

<sqlldr>

<!-- 项目现场需修改：数据入库时SQLLDR所需的dbuserid，使用解析服务器ORACLE客户端tnsnames.ora中的配置。 -->

<dbuserid>pm/pm@nwom2</dbuserid>

<!-- 高级：入库过程中产生的中间文件保存天数，受限于存储的容量，默认1天。-->

<fileSavedDays>1</fileSavedDays>

</sqlldr>

**<!-- 文件解析相关参数 -->**

<parser>

<!-- 高级： 解析线程池最大线程数，影响文件处理及时性，受限于服务器CPU总核数和JVM最大可用内存，建议不超过CPU核数的80%，默认15个。-->

<threadpool>15</threadpool>

<!-- 高级：文件解析轮询扫描间隔时间，单位秒，默认30秒。-->

<scanInterval>30</scanInterval>

<!-- 高级：原始文件保存天数，受限于存储的容量，默认3天。-->

<fileSavedDays>3</fileSavedDays>

</parser>

**<!-- 项目现场需修改：过程日志入库参数设置，用于实现基于log4j的标准日志入库 -->**

<log>

<dbDriverClassName>oracle.jdbc.driver.OracleDriver</dbDriverClassName>

<dbUrl>jdbc:oracle:thin:@10.224.141.182:8521:NMSW1</dbUrl>

<dbUser>nwom</dbUser>

<dbPassword>wxwy-NWOM.1</dbPassword>

</log>

**<!-- 扩展点 -->**

<extend>

<!-- 临时文件清理扩展点，由开发者修改，部署时不需修改-->

<tempFileCleaner>com.inspur.ftpparserframework.extend.DefaultTempFileCleaner</tempFileCleaner>

<!-- 文件过滤扩展点，由开发者修改，部署时不需修改-->

<fileFilter>com.inspur.ftpparserframework.extend.DefaultFileFilter</fileFilter>

<!-- 基本信息解析扩展点，由开发者修改，部署时不需修改-->

<basicInfoParser>com.inspur.ftpparserframework.extend.DefaultBasicInfoParser</basicInfoParser>

<!-- 文件转换器，由开发者修改，部署时不需修改-->

<transformators>com.inspur.ftpparserframework.transformator.DefaultTransformator</transformators>

</extend>

</Config>

* 1. FtpServerConfig.xml ：FTP连接信息配置文件样例。

<?xml version="1.0" encoding="gb2312"?>

<Config version="1.0">

<ftpServer name="JN\_HUAWEI\_OMC">

<ip>10.18.1.26</ip>

<port>21</port>

<user>nwom</user>

<password>nwom123</password>

<dir>/home/nwom/user/wuyg</dir>

<!--文件名举例:TD-SCDMA\_MRO\_HUAWEI\_OMC\_1921\_20120116084400.bmr.gz-->

<filter>\*MRS\*.gz</filter>

<!—本地存储路径-->

<storeDir></storeDir>

</ftpServer>

</Config>

* 1. log4j.xml：日志记录配置文件样例。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration debug="false" xmlns:log4j="http://jakarta.apache.org/log4j/">

<!-- 标准输出即控制台输出 -->

<appender name="stdoutAppender" class="org.apache.log4j.ConsoleAppender">

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="[%d{yyyy-MM-dd HH:mm:ss}][%l][%p]-%m%n"/>

</layout>

</appender>

<!-- 记录流水日志，每天产生一个文件 -->

<appender name="fileAppender" class="org.apache.log4j.DailyRollingFileAppender">

<param name="File" value="../logs/ftpparser.log"/>

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="[%d{yyyy-MM-dd HH:mm:ss}][%l][%p]-%m%n"/>

</layout>

</appender>

<!-- 数据库日志 -->

<appender name="dbAppender" class="com.inspur.ftpparserframework.log.log4j.DbAppender">

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="[%d{yyyy-MM-dd HH:mm:ss}][%l][%p]-%m%n"/>

</layout>

</appender>

<!-- 设置输出的日志级别，调试时可调整为DEBUG，能够输出更详细的信息-->

<root>

<level value="INFO"/>

<appender-ref ref="fileAppender"/>

<appender-ref ref="dbAppender"/>

</root>

</log4j:configuration>

1. 程序处理流程
   1. FTP采集流程



* 1. 文件解析流程



# Q&A

***Q：***

文件如何补采集：因各种原因导致FTP上的文件采集缺失了，问题解决后如何补充采集？

***A：***

第一步：使用实际程序bin目录下的shutdown.sh脚本停掉在运行的程序，使用ftpshutdown.sh脚本停掉采集程序。（切记，需先停掉程序）。

第二步：使用实际程序bin目录下的ftpstartup.sh -redownload启动补采。

第三步：补采结束后，使用实际程序bin目录下的ftpshutdown.sh停止补采程序。

第四步：启动正常的采集解析程序。

***Q：***

文件如何补解析：因各种原因导致部分文件解析失败了，问题解决后如何重新解析？

***A：***

第一步：使用实际程序bin目录下的shutdown.sh脚本停掉在运行的程序。（切记，需先停掉程序）。

第二步：将需要重新解析的文件从bak目录中恢复出来。

第三步：重新启动程序，正常运行即可。

***Q：***

文件解析时有时报内存溢出错误，该如何处理？

即程序有时僵死了，log中无错误，但nouhup.out中报“java.lang.OutOfMemoryError: GC overhead limit exceeded?”错误。

***A：***

此问题是由于java最大内存的限制导致同时解析多个文件时内存溢出，处理该问题的办法有两个：

第一个办法：修改ftp-parser-framework/bin/startup.sh脚本中的jvm最大内存-Xmx参数，现在默认是2048M，可以根据现场服务器情况适当调大。

第二个办法：修改改程序conf/SystemConfig.xml中的“解析线程池最大线程数”，即threadpool参数，默认配置是15，可以适当降低该参数。