ICS 35.020
L 70

DB15

内 蒙 古 自 治 区 地 方 标 准

DB15/T 1872—2020

大数据平台 接入技术要求

Technical specification of big data platform for access

2020-04-03 发布　　2020-05-03 实施

内蒙古自治区市场监督管理局　　发布

目　　次

前　言

本标准按照GB/T　1.1—2009给出的规则起草。

本标准由内蒙古自治区大数据发展管理局提出并归口。

本标准起草单位：内蒙古自治区大数据发展管理局、新华三技术有限公司、中国电子技术标准化研究院、内蒙古大学、内蒙古电子信息职业技术学院、内蒙古自治区大数据与云计算标准化委员会、中国人民银行呼和浩特市中心支行、内蒙古自治区国土资源信息院、中信银行股份有限公司呼和浩特市分行、内蒙古自治区标准化院、浪潮软件集团有限公司、北京东方国信科技股份有限公司、中通服咨询设计研究院有限公司、天帆创新（北京）科技发展有限公司、同方知网（北京）技术有限公司、北京东方棱镜科技有限公司、内蒙古跃晨科技有限公司、内蒙古纵横云技术有限公司。

本标准主要起草人：周佳琪、崔连伟、张建军、崔娜、赵逢波、卫凤林、屈强、崔波、胡大伟、汪昆鹏、马逸群、王立权、王一丁、王海珠、徐小强、王楠、李敏、李建文、刘玉坤、胡南磊、付先路、王伟哲、冯国忠。

大数据平台 接入技术要求

1 范围

本标准规定了大数据平台与各数据提供单位管理支撑系统进行数据接入的技术要求及数据采集接口、方式。

本标准适用于内蒙古自治区大数据平台进行数据采集功能研发、数据采集工具选型及其数据接入场景提供规范要求。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 29262 信息技术 面向服务的体系结构（SOA） 术语

GB/T 35274 信息安全技术 大数据服务安全能力要求

GB/T 35295-2017 信息技术 大数据 术语

GB/T 37973 信息安全技术 大数据安全管理指南

3 术语和定义

GB/T 35295和GB/T 29262界定的术语和定义适用于本文件。为了便于使用，以下重复列出GB/T 35295中的一些术语和定义。

3.1

大数据 big data

具有体量巨大、来源多样、生成极快、且多变等特征并难以用传统数据体系结构有效处理的包含大量数据集的数据。

注：国际上，大数据的4个特征普遍不加修饰地直接用volume、variety、velocity和variablility予以表述，并分别赋予了它们在大数据语境下的定义：

a) 体量 volume：构成大数据的数据集的规模；

b) 多样性 variety：数据可能来自多个数据仓库、数据领域或多种数据类型；

c) 速度 velocity：单位时间的数据流量；

d) 多变性 variability：大数据其他特征，即体量、速度和多样性等特征都处于多变状态。

[GB/T 35295-2017,定义2.1.1]

3.2

非结构化数据 unstructured data

不具有预定义模型或未以预定义方式组织的数据。
[GB/T 35295-2017,定义2.1.25]

3.3

静态数据 data at rest

处于静止状态，有典型特征表现为大数据的体量和多样性特征的数据。
[GB/T 35295-2017,定义2.1.37]

3.4

关系数据库 relational database

数据按关系模型来组织的数据库。
[GB/T 35295-2017,定义2.2.5]

3.5

管理信息系统 management information system

是一个以人为主导，利用计算机硬件、软件、网络通信设备以及其他办公设备，进行信息的收集、传输、加工、储存、更新、拓展和维护的系统。

3.6

元数据 metadata

关于数据或数据元素的数据（可能包括其数据描述），以及关于数据拥有权、存取路径、访问权和数据易变性的数据。
[GB/T 35295-2017,定义2.2.7]

4 缩略语

下列缩略语适用于本文件。

FTP：标准的文件传输协议（File Transfer Protocol）

JDBC：java数据库连接（Java DataBase Connectivity）

HTTPS：超文本传输安全协议（Hyper Text Transfer Protocol over Secure Socket Layer 或 Hypertext Transfer Protocol Secure）

Kafka：Kafka是一种高吞吐量的分布式发布订阅消息系统，它可以处理消费者在网站中的所有动作流数据。

5 概述

大数据平台作为多维数据的处理平台，支持各类数据源的采集与接入。以下是大数据平台六种常用数据源的接入方式：

a) 关系数据库抽取；
b) 服务网关服务；
c) 实时消息队列；
d) 文件接收 FTP 服务；
e) 文件拉取 FTP 服务；
f) 直报系统。

具体数据接入总体框架见图 1：

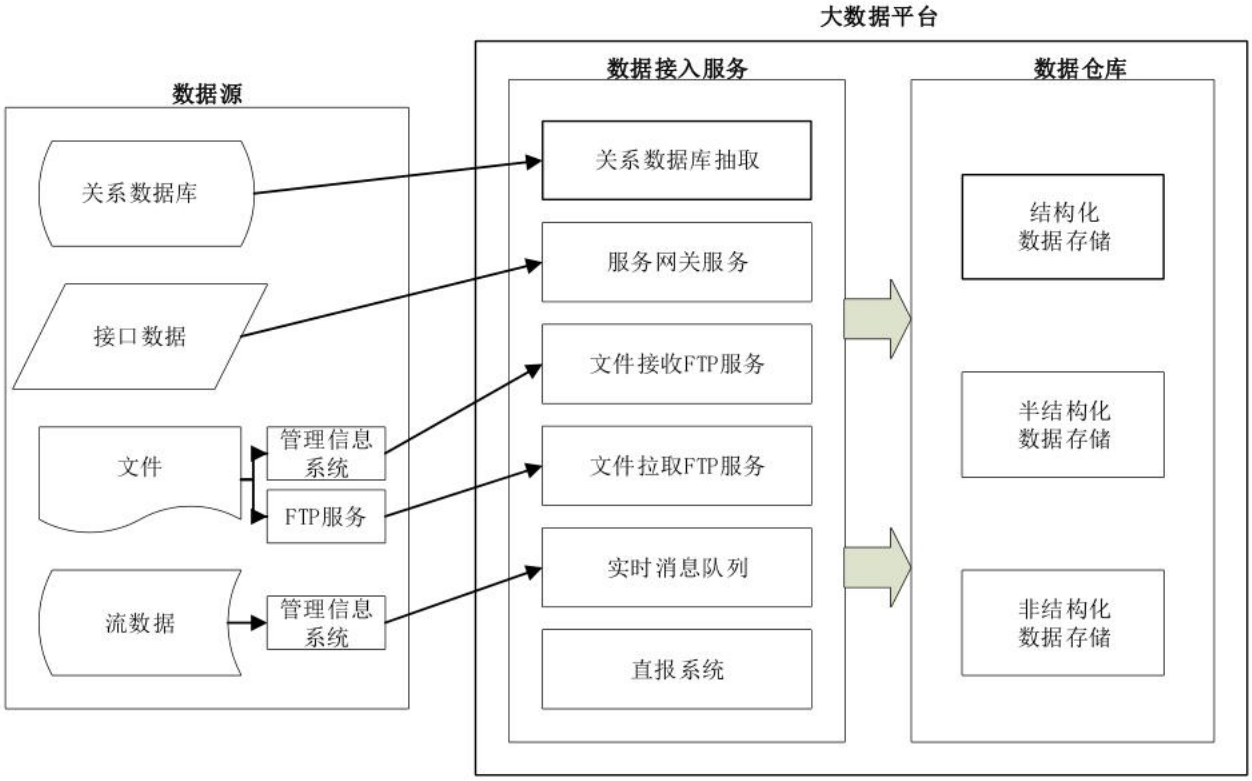


图 1 数据接入总体框架

6 接入要求

6.1 关系数据库抽取

6.1.1 功能要求

关系数据库抽取应提供管理信息系统关系数据库中的结构化数据到大数据平台数据存储的定时批量抽取功能。关系数据库数据抽取应具备以下主要功能：

a) 支持对主流的关系数据库进行数据抽取；支持对数据库中常用的数据类型进行数据抽取，至少包括数值型、字符型、日期/时间型等数据类型；
b) 支持“全量”和“ 增量”两种数据抽取模式；“全量”模式是指一次性将关系数据库中物理表的数据抽取到大数据平台。“增量”模式是指根据设置的抽取条件筛选符合条件的数据抽取到大数据平台；
c) 支持关系数据库中结构化数据抽取到大数据平台,包含结构化数据存储、半结构数据存储、非结构数据存储的数据仓库中；

d) 支持对关系数据库数据的采集内容和类型转换操作，至少包括选择具体的数据表、选择表中具体的字段、字段类型格式转换等操作；
e) 支持数据抽取操作的立即执行、定时调度运行。定时调度运行应提供多种调度策略，至少包括固定时间间隔运行、指定时间点运行、指定时间范围运行、一次或指定次数运行等策略；
f) 应提供图形化管理界面，应提供数据抽取模式设置、抽取源关系数据库配置、指定数据表配置、表字段选择配置、字段类型转换配置、大数据平台目标存储位置配置、运行策略配置等操作界面；
g) 应提供完善的日志和审计能力，可以记录数据抽取操作配置、运行时发生的各种事件；
h) 应提供完善的监控机制，运行过程中出现异常可快速的定位及解决。

6.1.2 非功能要求

关系数据库数据抽取服务应满足以下非功能性要求：
a） 数据抽取速度：不少于 1 万条/秒；
b） 数据抽取吞吐量：在千兆带宽的网络条件下， 数据抽取吞吐量不少于50 MB/秒。

6.1.3 应用场景

关系数据库抽取应用场景见图2：

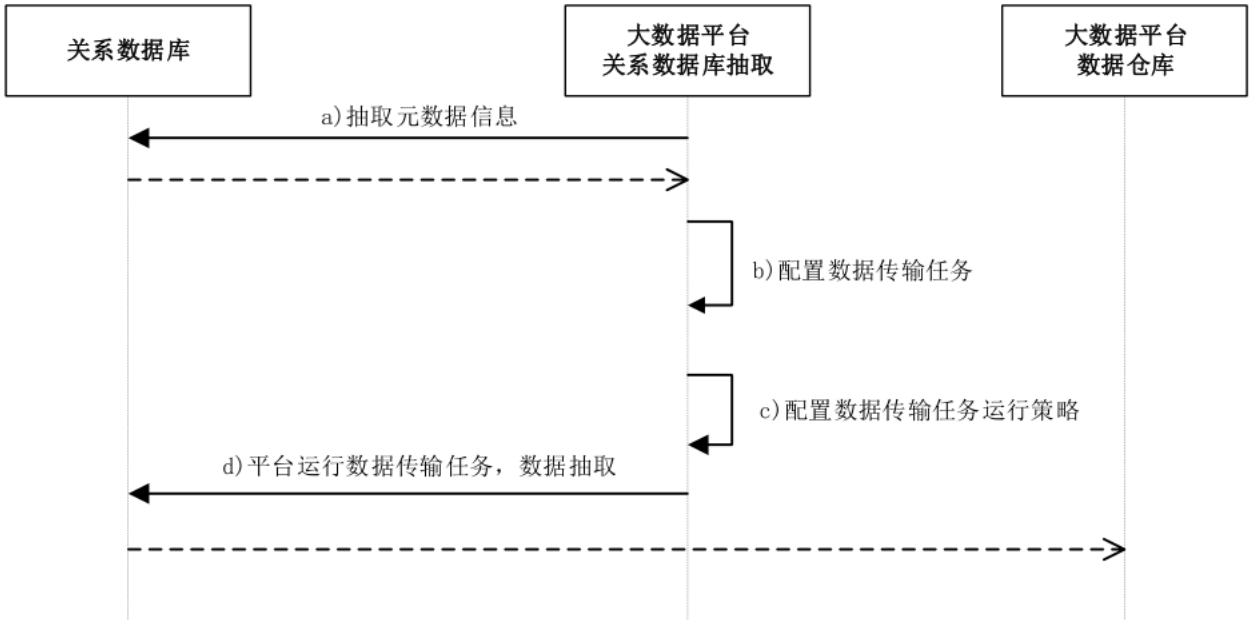


图 2 关系数据库抽取应用场景

应用场景描述如下:
a) 关系数据库抽取服务，抽取数据源中数据库的元数据信息，包括数据库用户所属的表、字段信息；
b) 关系数据库抽取服务配置数据传输任务，包括抽取数据库的源表和大数据平台对应的存储目标表；
c) 关系数据库抽取服务配置数据传输任务运行策略，包括运行的开始时间、结束时间、运行频度；
d) 关系数据库抽取服务运行数据传输任务，从数据源的数据库抽取数据到平台数据存储中。

6.1.4 应用要求

应用要求如下：

a) 关系数据库抽取服务适用于关系数据库定时批量抽取场景，详细说明参见附录A；
b) 数据源须提供关系数据库的访问链接，包括 IP、端口、数据库实例名、用户名、密码；
c) 数据源提供的数据库访问用户应具备数据库的元数据信息定义表的读取权限。

6.2 服务网关服务

6.2.1 功能要求

服务网关服务为数据源提供大数据平台中结构化数据或非结构化数据的接口数据接入。服务网关服务应具备以下主要功能：

a) 支持接入Webservice、RESTful方式的接口；
b) 支持包括结构化数据、非结构化数据的接口；
c) 支持接口编排，轻松实现多个接口的功能集成；
d) 提供图形化管理界面，用于接口数据存储位置、操作用户、目标存储位置的配置；
e) 应提供完善的日志和审计能力，可以记录接口数据配置及数据抽取操作配置、运行时发生的各种事件；
f) 应具备熔断管理机制，保证服务整体可用，是接口访问异常情况下的处理策略。

6.2.2 非功能要求

服务网关服务应满足以下非功能要求：

a) 数据写入速率：在千兆带宽的网络条件下,数据抽取吞吐量不少于30 MB/秒；
b) 操作并发数：并发数大于200 个/秒。

6.2.3 应用场景

服务网关服务应用场景见图3：

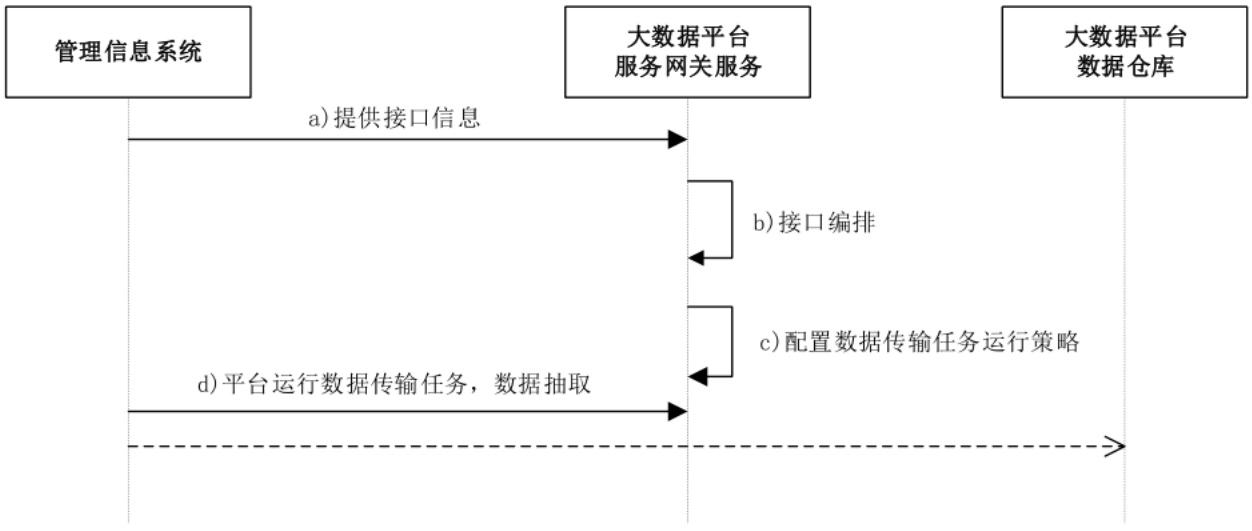


图 3 服务网关服务应用场景

应用场景描述如下:

a) 数据源向大数据平台提供接口信息，包括：接口访问地址、输入参数、输出参数、验证方式等接口信息；
b) 大数据平台根据数据源提供的数据接口进行定义及编排；
c) 服务网关服务配置数据传输任务运行策略，包括运行的开始时间、结束时间、运行频度；
d) 服务网关服务运行数据传输任务，从数据源的数据接口中抽取数据到大数据平台数据仓库中。

6.2.4 应用要求

a) 服务网关服务适用于提供接口类数据的数据源，详细说明参见附录 B；
b) 提供数据接口的数据源需做好自身数据操作接口程序的开发。

6.3 实时消息队列

6.3.1 功能要求

实时消息队列采集为管理信息系统提供实时消息推入和缓存功能。实时消息队列应具备以下主要功能：

a） 应提供分布式消息队列的管理功能，支持消息主题的创建、删除、修改；
b） 应提供支持“点对点”和“发布-订阅”两个消息模式；
c） 支持消息的持久化存储操作并且支持持久化周期设置；
d） 应提供消息的发送和消费接口，包括链接建立、消息发送、消息消费、链接关闭；
e） 应提供分布式高可用的消息队列操作接口,支持消息的发送和消费;支持消息分区和备份操作；
f） 具有风格统一的图形化管理界面，支持消息队列主题的创建、删除、测试、授权访问的操作；
g） 具备完善的日志审计能力，可以记录消息发送和消费时发生的各种事件。

6.3.2 非功能要求

实时消息队列应满足以下非功能性要求：

a） 高可行性：支持消息主题的分区和备份；
b） 负载均衡：支持消息发送和消费时的负载均衡操作；
c） 消息发送速度：可接受的数据量大于5 万条/秒。

6.3.3 应用场景

实时消息队列应用场景见图4：

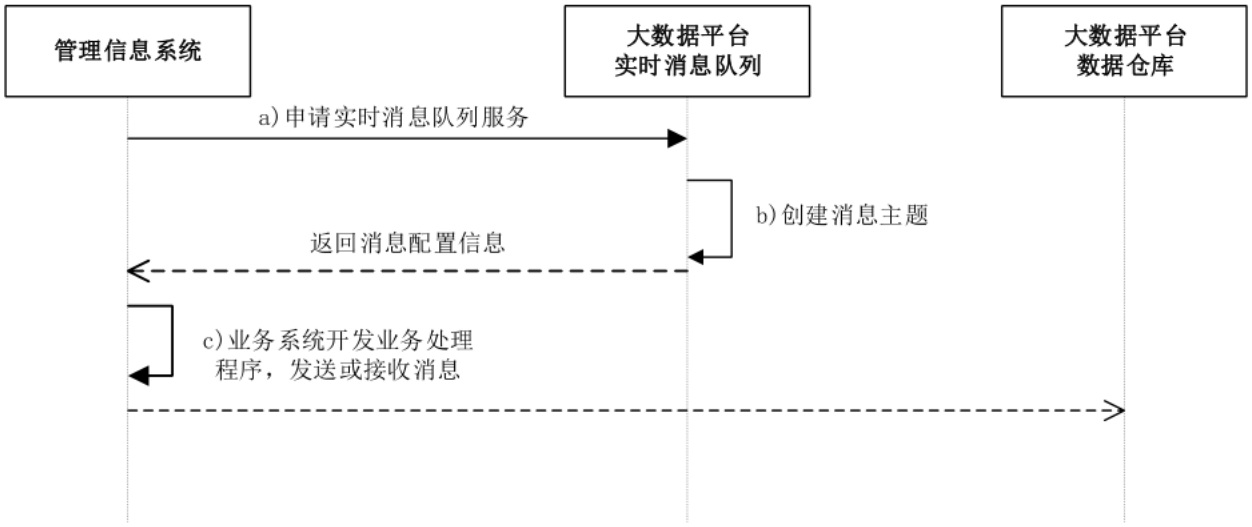


图 4 实时消息队列应用场景

应用场景描述如下：

a） 管理信息系统须向大数据平台申请实时消息队列接入服务；

b） 大数据平台根据申请创建消息队列主题， 返回消息队列名称；

c） 管理信息系统开发业务处理程序， 调用平台实时消息队列接口，发送数据或接收数据。

6.3.4 应用要求

应用要求如下：

a） 实时消息队列采集适用于管理信息系统主动将数据封装为消息，发送到大数据平台的实时消息队列中。基于实时消息队列的消息缓存进行数据分析，如流计算实时处理等，详细说明参见附录C；

b） 发送的消息内容格式支持字符串，发送的数据对象可以通过对象序列化机制转换为字符串格式的消息内容；

c） 管理信息系统应依照大数据平台提供的实时消息队列采集接口完成自身数据发送或接收接口的开发。

6.4 文件接收 FTP 服务

6.4.1 功能要求

文件接收FTP服务应提供外部系统文件数据的接收并存入到大数据平台数据仓库的功能。文件采集应具备以下主要功能：

a） 支持标准 FTP 协议接收数据；

b） 支持顺序型断点续传功能；

c） 支持接收的文件的重命名及指定存储目录；

d） 应支持对接收文件的完整性校验；

e） 应支持对客户端进行认证；

f） 支持图形管理功能，支持认证配置、文件目标位置配置、校验处理配置。

6.4.2 应用场景

文件接收FTP服务应用场景见图5：

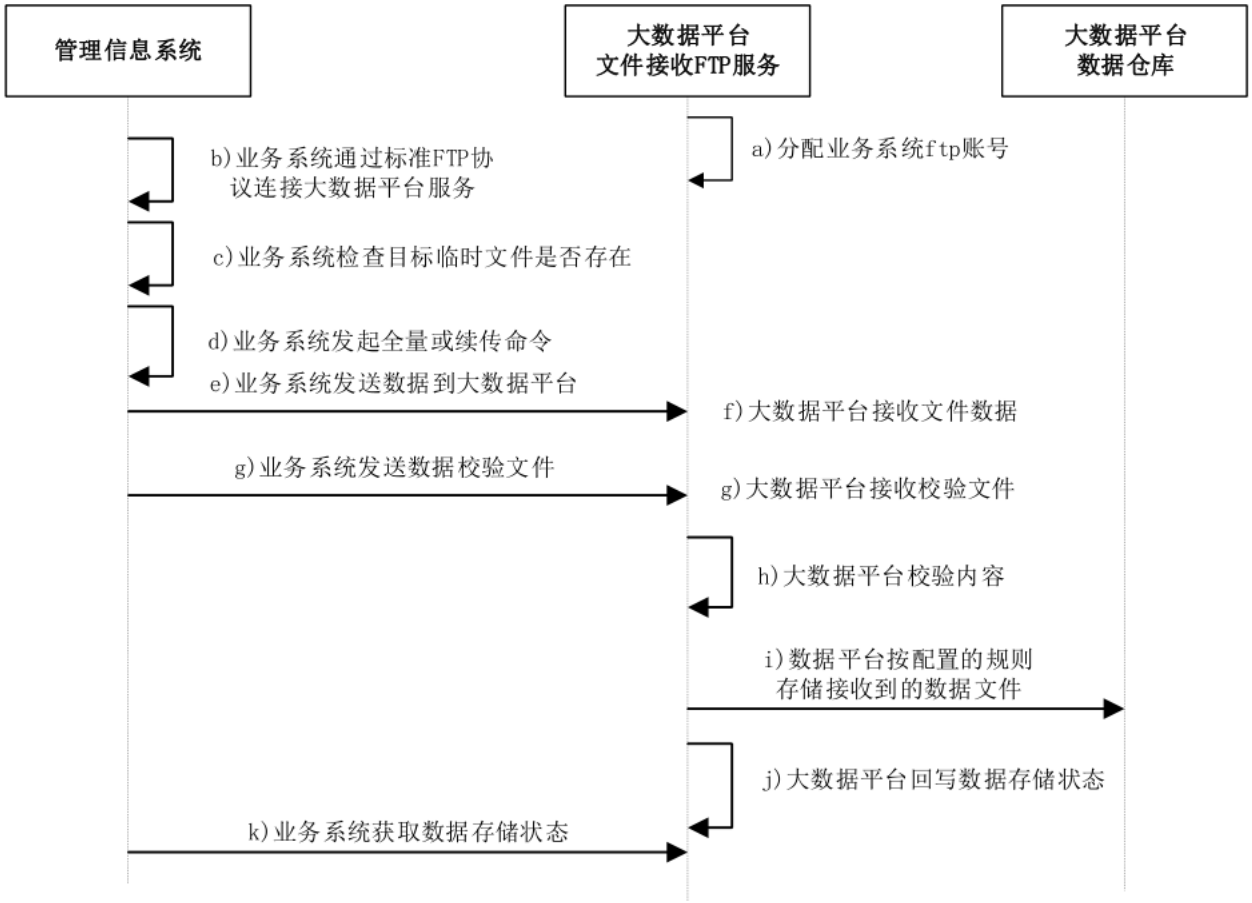


图 5 文件接收 FTP 服务应用场景

应用场景描述如下：

a) 大数据平台配置应用账号、接收参数及存储位置；
b) 管理信息系统通过标准 FTP 协议连接大数据平台服务；
c) 管理信息系统检查目标临时文件是否存在；
d) 管理信息系统发起全量或续传指令；
e) 管理信息系统发送数据到大数据平台；
f) 大数据平台接收文件数据；
g) 管理信息系统发送数据校验文件；
h) 大数据平台根据校验文件校验数据文件内容；
i) 大数据平台按配置的规则存储接收到的数据文件；
j) 大数据平台回写数据存储状态；
k) 管理信息系统获取数据存储状态。

6.4.3 应用要求

应用要求如下：

a) 管理信息系统须在大数据平台注册并申请账号；
b) 管理信息系统须按平台协议规范开发上传功能；
c) 管理信息系统生成文件数据时须同时生成对应的完整性校验码；
d) 具体文件接收 FTP 服务 API 接口参见附录 D.1，具体实现逻辑参见附录 D.2。

6.5 文件拉取 FTP 服务

6.5.1 功能要求

文件拉取FTP服务，应提供通过访问FTP协议实现将文件数据抽取到大数据平台数据仓库的功能。文件拉取FTP服务应具备以下主要功能：

a） 支持顺序型断点续传功能，支持外部文件存储断点续传能力的自动识别及模式匹配；
b） 支持 FTP 服务登录用户名和密码设置；
c） 支持文件压缩传输，提供文件压缩规则设置；
d） 支持文件加密传输，提供文件加密传输规则设置；
e） 支持设置文件同步、异步拉取，支持设置拉取并行度；
f） 支持指定目标文件存储位置、文件名，提供文件类型转换规则，支持常见类型转换；
g） 支持全量文件采集，支持外部数据一次性初始化导入；
h） 支持定时轮询文件采集， 采集新增的文件， 支持文件列表规则过滤；
i） 支持图形管理功能，支持 FTP 连接配置、文件源配置、文件目标存储配置、文件压缩和加密传输规则配置、文件同步/异步传输规则配置、文件传输并行度配置、文件定时及实时策略配置、文件采集过滤配置。

6.5.2 应用场景

文件拉取FTP应用场景见图6：

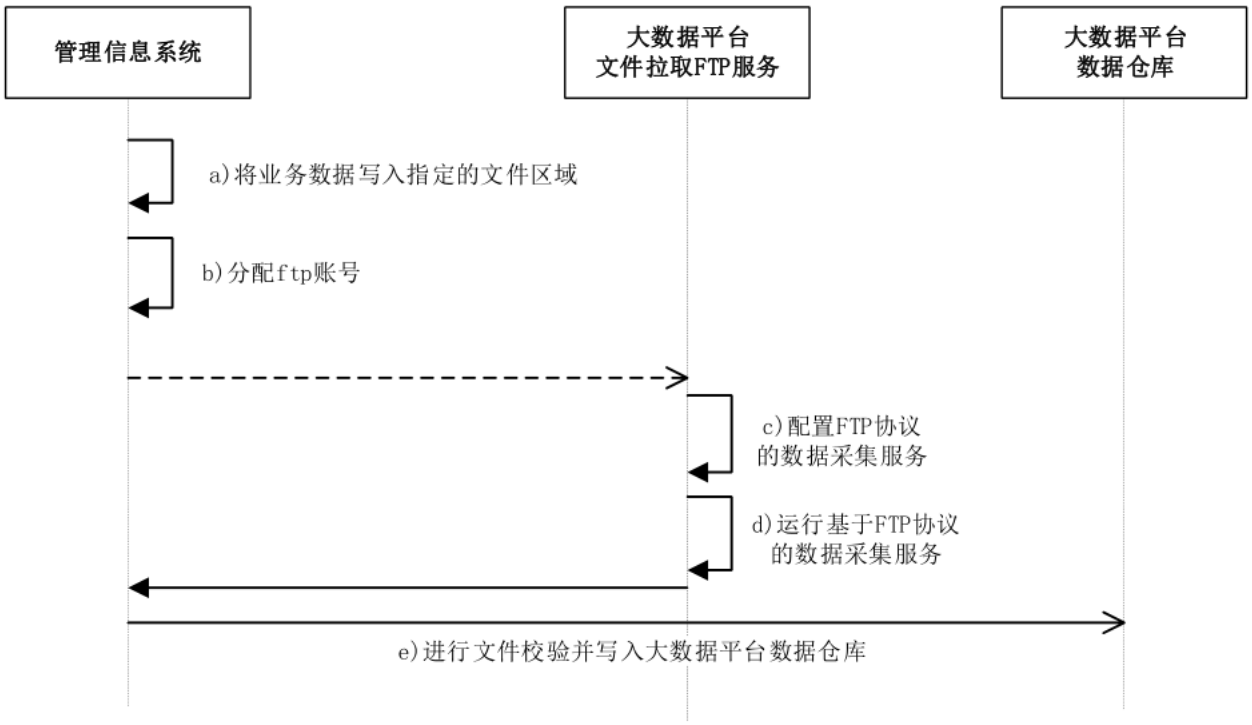


图6 文件拉取FTP服务应用场景

应用场景描述如下：

a) 管理信息系统应提供初始文件存储位置并将业务数据写入到文件中；
b) 管理信息系统搭建FTP服务并为大数据平台分配FTP账号信息；
c) 大数据平台全量文件采集提供基于 FTP 协议的采集任务配置，包括：存储文件位置、采集文件、列表方式、是否文件校验、存储目标位置及文件存储命名规则；
d) 运行基于FTP协议的数据采集任务，包括：包括运行的开始时间、结束时间、运行频度；
e) 大数据平台运行文件采集任务， 读取文件存入大数据平台数据存储中。

6.5.3 应用要求

应用要求如下：

a) 管理信息系统须先将业务数据保存为文件，并设置访问权限；
b) 数据文件可通过 FTP 协议访问；
c) 管理信息系统生成文件数据时须同时生成对应的完整性校验码；
d) 具体文件拉取 FTP 服务 API 接口参见附录 E.1，具体实现逻辑参见附录 E.2。

6.6 直报系统

6.6.1 功能要求

直报系统是大数据平台为各数据接入单位提供的在线填写、上传。直报系统应具备以下主要功能：

a) 支持数据接入单位注册功能，支持管理单位用户审批功能，支持管理单位为用户分配权限功能；
b) 支持用户仅能同时登录一次功能；
c) 支持模板管理功能，应具备模板的新增、删除、修改、搜索操作功能；
d) 支持模板下载、数据上传、数据提交操作功能；

e) 应提供完善的日志和审计能力，可以记录各数据接入单位在数据配置及直报运行时发生的各种事件；

f) 应具备熔断管理机制，保证服务整体可用，是直报系统访问异常情况下的处理策略。

6.6.2 非功能要求

直报系统应满足以下非功能性要求：

a) 直报系统需支持主流的浏览器版本；

b) 直报系统网络需支持互联网、政务网；

c) 数据上传速度不少于5 M/秒；

d) 系统响应时间在200并发下不低于3 秒。

6.6.3 应用场景

直报系统应用场景见图7：

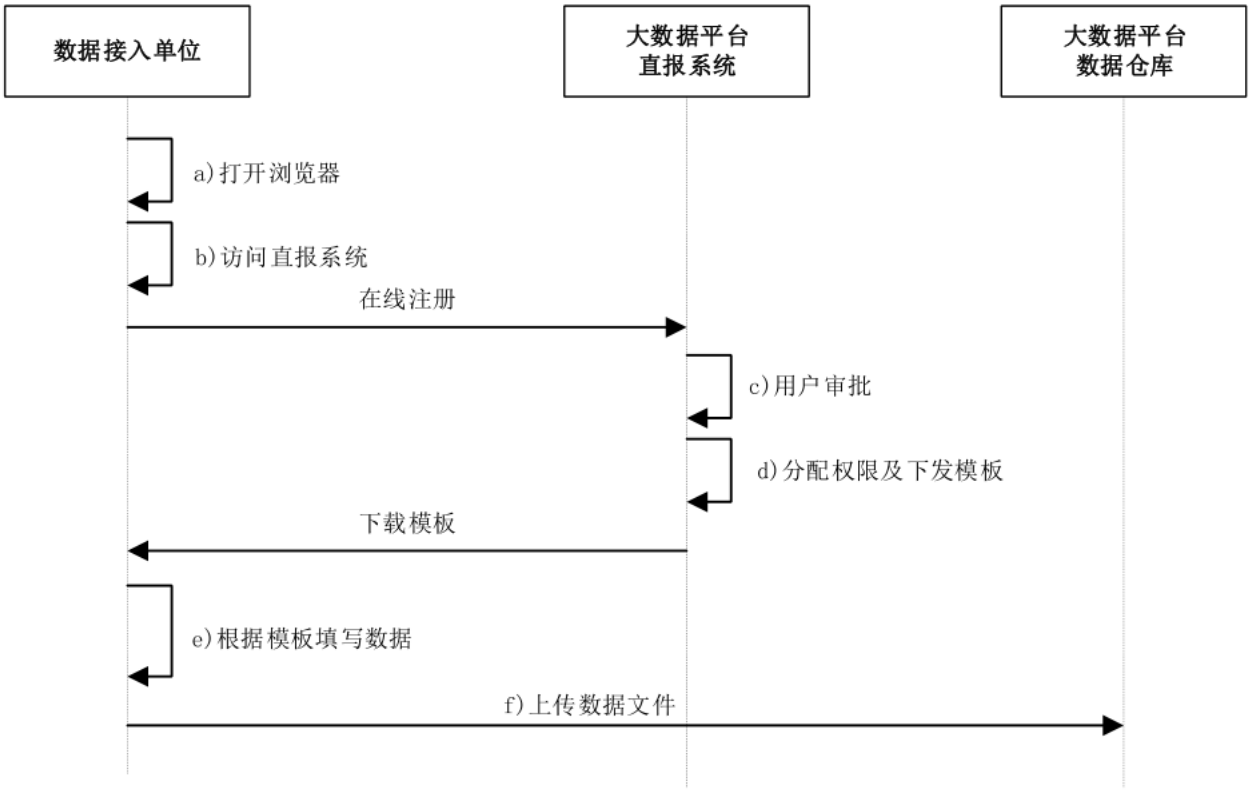


图 7 直报系统应用场景

应用场景说明如下：

a) 数据接入单位用户打开浏览器；

b) 在地址栏输入大数据平台提供的直报系统访问地址，首次使用按照指引进行用户在线注册（非首次使用直接到e）；

c) 大数据平台管理员进行用户合法性审批；

d) 审批通过后为注册用户分配权限并根据业务需求制定数据模板；

e) 数据接入单位用户登录直报系统后下载模板文件，并根据模板填写需上报的具体数据；
f) 数据接入单位用户根据模板填写完成后在直报系统中直接上传数据文件并提交，直报系统收到数据接入单位用户上传请求后进行数据格式校验，校验通过后抽取数据到平台数据仓库中。

6.6.4 应用要求

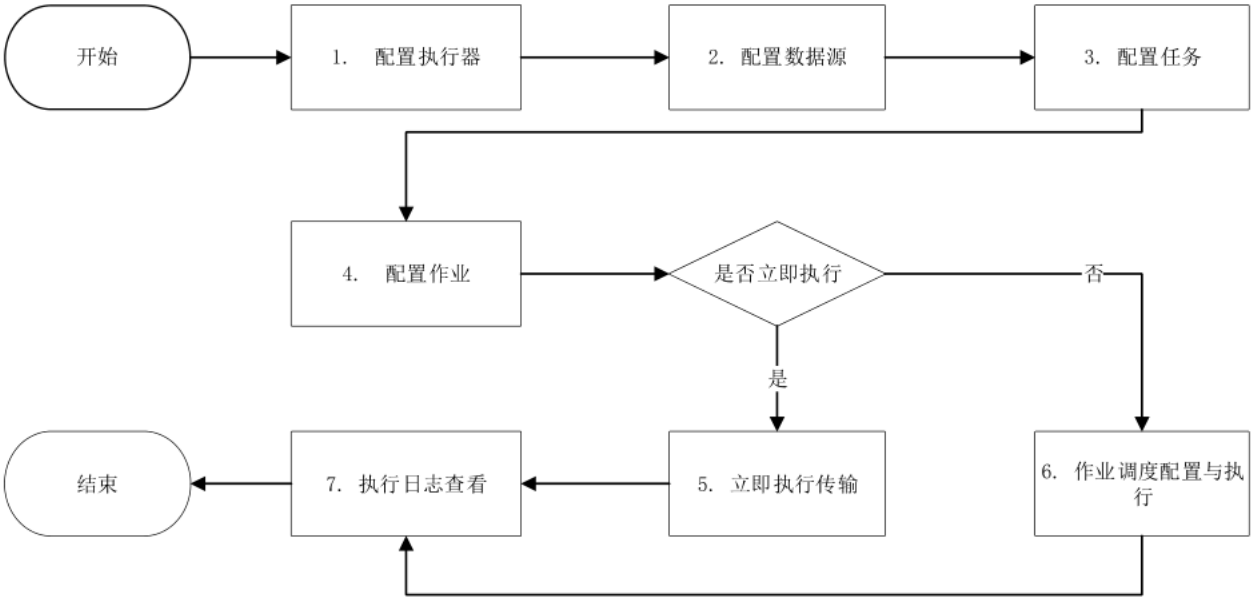
应用要求如下：
a) 对信息化建设较弱且有计算机基本操作知识的数据接入单位用户提供；
b) 大数据平台应提供直报系统的访问地址、操作手册，酌情组织、安排定期培训以普及直报系统的使用流程。

7 安全要求

安全要求须满足 GB/T 35274 规范中“数据服务安全要求”。

附录 A
（资料性附录）
关系数据库抽取接入说明

关系数据库抽取通过大数据平台提供的Web界面进行操作， 完成数据接入操作。关系数据库应用流程见图A.1：



图A.1 关系数据库应用流程

应用流程描述如下：

a) 大数据平台配置执行器，用于作业调度执行的服务；

g) 配置数据源，分别配置源数据库及目标数据库。根据业务源端的数据库的 IP、 端口、 实例名、用户名、密码，进行数据源链接配置，抽取数据库用户所属的表、字段信息，目标端数据库配置， 配置抽取目标数据库信息配置，包括目标数据库的 IP、端口、用户名、密码、数据库名；

h) 配置任务，配置源端与目标端字段对应关系、是否抽取、格式化公式等，一个抽取可配置多个任务；

i) 配置作业，根据抽取任务执行的先后顺序配置成作业；

j) 立即执行传输，手动开启作业输立即执行；

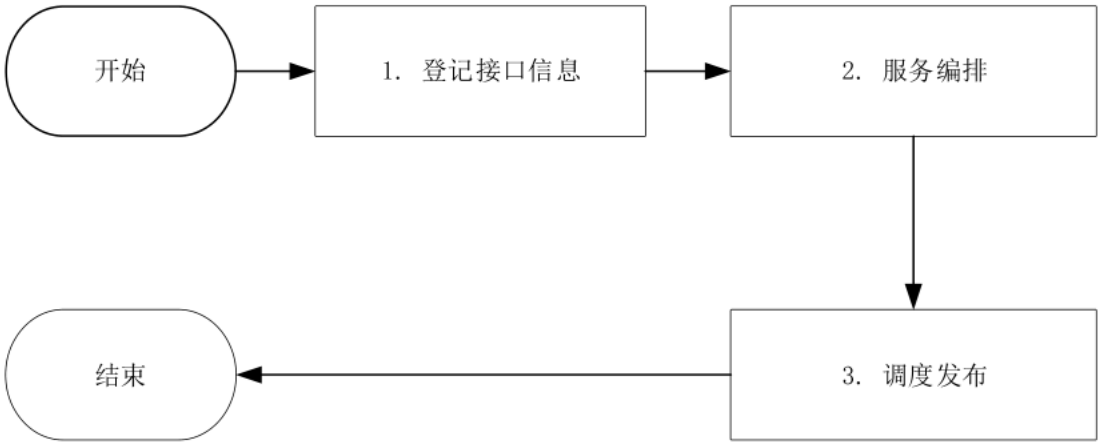
k) 作业调度配置与执行，将数据库抽取配置为作业，支持配置抽取作业的执行策略，包括作业开始时间、结束时间、运行频率等；

l) 执行日志查看，提供作业执行结果信息查看，包括作业开始执行时间、结束时间、运行时长、作业状态、运行结果、日志详情等。

附录 B
（资料性附录）
服务网关服务接入说明

B.1 服务网关服务应用流程

通过大数据平台的服务网关服务提供的Web界面进行操作。首先服务网关服务录入接口信息，然后通过服务编排进行接口的编排，生成新的数据接口进行调度发布。 服务网关服务应用流程见图B.1：



图B.1 服务网关服务应用流程

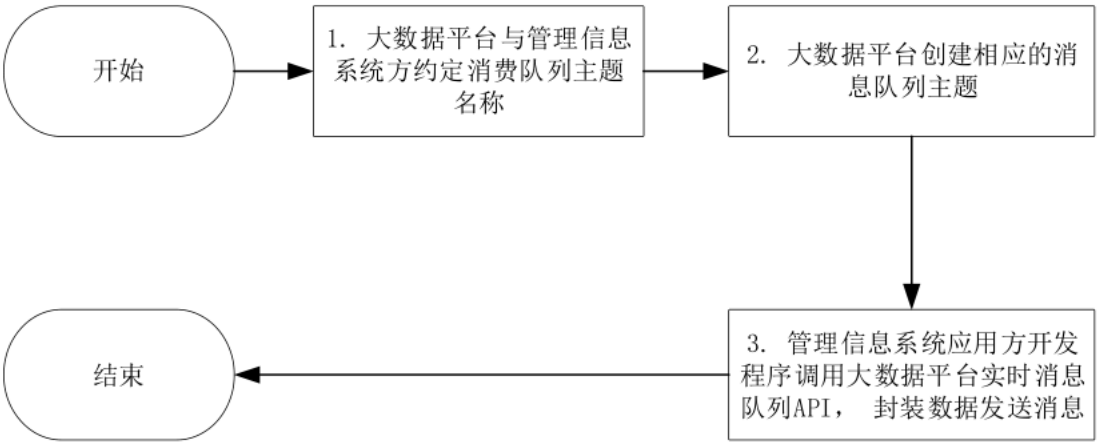
应用流程描述如下：

a) 根据数据源提供的数据接口地址、请求方式、请求头配置、传输协议、请求参数信息在服务网关进行数据接口的登记、熔断保护配置及测试验证；
b) 针对登记的接口信息进行服务编排，定义接口的输入参数、输出参数、请求头等信息，并生成新的接口地址；
c) 服务编排后的数据接口配置调度策略，包括调度时间、调度周期及频次等信息，然后进行发布。

附录 C
（资料性附录）
实时消息队列接入说明

C.1 实时消息队列应用流程

管理信息系统通过Java代码开发方式调用大数据平台提供的实时消息队列API接口，往消息队列主题中发送数据。实时消息队列应用流程见图C.1：



图C.1 实时消息队列应用流程

说明：

1. 大数据平台与管理信息系统应用方约定消息队列主题名称；
2. 大数据平台在大数据集群中创建相应的消息队列主题；
3. 管理信息系统开发程序调用大数据平台实时消息队列 API，封装数据往消息队列主题中发送消息。

附录 D
（资料性附录）
文件接收 FTP 服务接入说明

D.1 文件接收 FTP 服务接口

文件接收FTP服务接口见表D.1：

表D.1 文件接收FTP服务接口

序号	接口方法	接口说明
1	public void initFtpClient(String hostname, Integer port,String username,String password)	初始化FTP服务器，参数说明如下： hostname：FTP服务器地址； prot：FTP服务器端口 username：FTP登录账号； password：FTP登录密码。
2	public boolean uploadFile(String pathname, String fileName, InputStream inputStream)	上传文件。 参数说明如下： pathname：ftp服务保存地址 fileName：上传到ftp的文件名 inputStream：输入文件流
3	public boolean CreateDirecroty(String remote)	创建文件目录。参数说明如下： remote：文件目录

D.2 文件接收 FTP 服务示例

管理信息系统在生成好文件后，编写代码实现上传文件到大数据平台的FTP服务中，调用服务示例（ Java） 见表D.2：

表D.2 文件接收FTP服务示例

```
public void initFtpClient(String hostname, Integer port,String username,String password) {
ftpClient = new FTPClient();
ftpClient.setControlEncoding("utf-8");
try {
ftpClient.connect(hostname, port);
ftpClient.login(username, password);
int replyCode = ftpClient.getReplyCode();
if (!FTPReply.isPositiveCompletion(replyCode)) {
  System.out.println("connect failed...ftp服务器:" + this.hostname + ":" + this.port);
} catch (MalformedURLException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
```

```
    }
    }
public boolean uploadFile(String pathname, String fileName, InputStream inputStream) {
try {
    System.out.println("开始上传文件");
    initFtpClient();
    ftpClient.setFileType(FTP.BINARY_FILE_TYPE);
    CreateDirecroty(pathname);
    ftpClient.makeDirectory(pathname);
    ftpClient.changeWorkingDirectory(pathname);
    ftpClient.storeFile(fileName, inputStream);
     inputStream.close();
    ftpClient.logout();
    System.out.println("上传文件成功");
    } catch (Exception e) {
    System.out.println("上传文件失败");
    e.printStackTrace();
    } finally {
    if (ftpClient.isConnected()) {
    try {
    ftpClient.disconnect();
    } catch (IOException e) {
    e.printStackTrace();

    }
    }
    if (null != inputStream) {
    try {
    inputStream.close();
    } catch (IOException e) {
    e.printStackTrace();
    }
    }
    }
    return true;
    };
    public boolean CreateDirecroty(String remote) throws IOException {
    boolean success = true;
    String directory = remote + "/";
    // 如果远程目录不存在，则递归创建远程服务器目录
    if (!directory.equalsIgnoreCase("/") && !changeWorkingDirectory(new String(directory))) {
    int start = 0;
```

```
int end = 0;
if (directory.startsWith("/")) {
start = 1;
} else {
start = 0;
}
end = directory.indexOf("/", start);
String path = "";
String paths = "";
while (true) {
String subDirectory = new String(remote.substring(start, end).getBytes("GBK"), "iso-8859-1");
path = path + "/" + subDirectory;
if (!existFile(path)) {
if (makeDirectory(subDirectory)) {
changeWorkingDirectory(subDirectory);
} else {
System.out.println("创建目录[" + subDirectory + "]失败");
changeWorkingDirectory(subDirectory);
}
} else {
changeWorkingDirectory(subDirectory);
}
paths = paths + "/" + subDirectory;
start = end + 1;
end = directory.indexOf("/", start);
// 检查所有目录是否创建完毕
if (end <= start) {
break;
}
}
}
return success;
}
```

附录 E
（资料性附录）
文件拉取 FTP 服务接入说明

E.1 文件拉取 FTP 服务 API 接口

文件拉取FTP服务API接口见表E.1：

表 E.1 文件拉取 FTP 服务 API 接口

序号	接口方法	接口说明
1	public void initFtpClient(String hostname, Integer port,String username,String password)	初始化FTP服务器，参数说明如下： hostname：FTP服务器地址； prot：FTP服务器端口 username：FTP登录账号； password：FTP登录密码。
2	public boolean downloadFile(String pathname, String filename, String localpath)	FTP 文件下载。参数说明如下： pathname：FTP服务器文件目录 filename：文件名称 localpath：下载后的文件路径
3	public boolean CreateDirecroty(String remote)	创建文件目录。参数说明如下： remote：文件目录

E.2 文件拉取 FTP 服务示例

管理信息系统在生成好文件后，通过调用大数据平台FTP文件拉取服务，调用服务示例（Java） 见表E.2：

表E.2　文件拉取FTP服务示例

```
    public void initFtpClient(String hostname, Integer port,String username,String password) {
    ftpClient = new FTPClient();
    ftpClient.setControlEncoding("utf-8");
    try {
    ftpClient.connect(hostname, port);
    ftpClient.login(username, password);
    int replyCode = ftpClient.getReplyCode();
    if (!FTPReply.isPositiveCompletion(replyCode)) {
      System.out.println("connect failed...ftp服务器:" + this.hostname + ":" + this.port);
    } catch (MalformedURLException e) {
    e.printStackTrace();
    } catch (IOException e) {
    e.printStackTrace();
    }
    }

public boolean downloadFile(String pathname, String filename, String localpath) {
    boolean flag = false;
    OutputStream os = null;
    try {
       initFtpClient();
       boolean changeFlag = ftpClient.changeWorkingDirectory(pathname);
       System.err.println("changeFlag==" + changeFlag);
       ftpClient.enterLocalPassiveMode();
       ftpClient.setRemoteVerificationEnabled(false);
       String[] a = ftpClient.listNames();
       System.err.println(a[0]);
       FTPFile[] ftpFiles = ftpClient.listFiles();
       for (FTPFile file : ftpFiles) {
          if (filename.equalsIgnoreCase(file.getName())) {
             File localFile = new File(localpath + "/" + file.getName());
             os = new FileOutputStream(localFile);
             ftpClient.retrieveFile(file.getName(), os);
             os.close();
          }
       }
      ftpClient.logout();
      flag = true;
       System.out.println("下载文件成功");
    } catch (Exception e) {
    System.out.println("下载文件失败");
```

```
e.printStackTrace();
} finally {
if (ftpClient.isConnected()) {
try {
ftpClient.disconnect();
} catch (IOException e) {
e.printStackTrace();
}
}
if (null != os) {
try {
os.close();
} catch (IOException e) {
e.printStackTrace();
}
}
}
return flag;
}

public boolean CreateDirecroty(String remote) throws IOException {
boolean success = true;
String directory = remote + "/";
// 如果远程目录不存在，则递归创建远程服务器目录
if (!directory.equalsIgnoreCase("/") && !changeWorkingDirectory(new String(directory))) {
int start = 0;
int end = 0;
if (directory.startsWith("/")) {
start = 1;
} else {
start = 0;
}
end = directory.indexOf("/", start);
String path = "";
String paths = "";
while (true) {
String subDirectory = new String(remote.substring(start, end).getBytes("GBK"), "iso-8859-1");
path = path + "/" + subDirectory;
if (!existFile(path)) {
if (makeDirectory(subDirectory)) {
changeWorkingDirectory(subDirectory);
} else {
```

```
System.out.println("创建目录[" + subDirectory + "]失败");
changeWorkingDirectory(subDirectory);
}
} else {
changeWorkingDirectory(subDirectory);
}
paths = paths + "/" + subDirectory;
start = end + 1;
end = directory.indexOf("/", start);
// 检查所有目录是否创建完毕
if (end <= start) {
break;
}
}
}
return success;
}
```