ICS 35.020
CCS L 70

DB23

黑 龙 江 省 地 方 标 准

DB23/T 3304—2022

大数据平台数据接入规范

2022-07-07 发布　　2022-08-06 实施

黑龙江省市场监督管理局　发布

目　　次

前　言

本文件依据GB/T 1.1-2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利，本文件的发布机构不承担识别专利的责任。

本文件由黑龙江省大数据产业协会提出。

本文件由黑龙江省工业和信息化厅归口。

本文件起草单位：黑龙江省大数据产业协会、哈尔滨财富通科技发展有限公司、黑龙江亿林网络股份有限公司、黑龙江省网络空间研究中心、黑龙江省标准化研究院、黑河市特种设备检验研究所、黑龙江大数据产业发展有限公司、黑龙江省信创科技有限公司、黑龙江交投信科科技有限责任公司、黑龙江交投千方科技有限公司、哈尔滨智路开发有限公司、黑龙江农投大数据公司、黑龙江省农投云产业有限公司。

本文件主要起草人：李璐昆、孙传友、杜飞、孙甲子、张驰、王阳、陈要武、杨大志、吕猛、王磊、唐丽、赵海洋、李冰冷、叶爽、王克云、李森、周全、何晨龙、叶爽、张新、关哲刚、杨旭、王晶。

大数据平台数据接入规范

1 范围

本文件规定了大数据平台数据接入规范的术语和定义，缩略语、总体框架和接入要求。

本文件适用于黑龙江省内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

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；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 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 29262、GB/T 35295和GB/T 35274界定的以及下列术语和定义适用于本文件。

3.1

数据接入

将数据传输进入数据平台所采用的形式。

3.2

数据采集

将数据收集进入数据平台所采用的形式。

3.3

全量采集模式

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。

3.4

增量采集模式

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。

3.5

轮询采集模式

轮询采集模式是增量采集模式的一种。

4 缩略语

下列缩略语适用于本文件。

DTS:数据传输服务(Data Transfer Service)。

ETL:将数据从来源端经过抽取、转换、加载至目的端的过程(Extract-Transform-Load)。

FTP：文件传输协议（File Transfer Protocol）。

HTTP:标准的超文件传输协议（Hyper Text Transfer Protocol ）。

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

NFS:网络文件系统（Network File System）。

KQS：消息集群数据接入（Kafka Queue Stream）。

SHA：安全哈希算法（Secure Hash Algorithm）。

5 总体框架

5.1 总体框架描述

大数据平台支持从关系型数据库、文件、数据流等来源获取数据，实现各类离线数据及实时数据的采集与接入，包括设备采集数据、企业管理业务数据、外部数据等。其中离线数据主要分为关系型数据库所存储的结构化数据及文件系统所存储的非结构化文件数据,实时数据主要是设备采集监控及业务系统产生的实时流数据。总体框架示意图见图1。

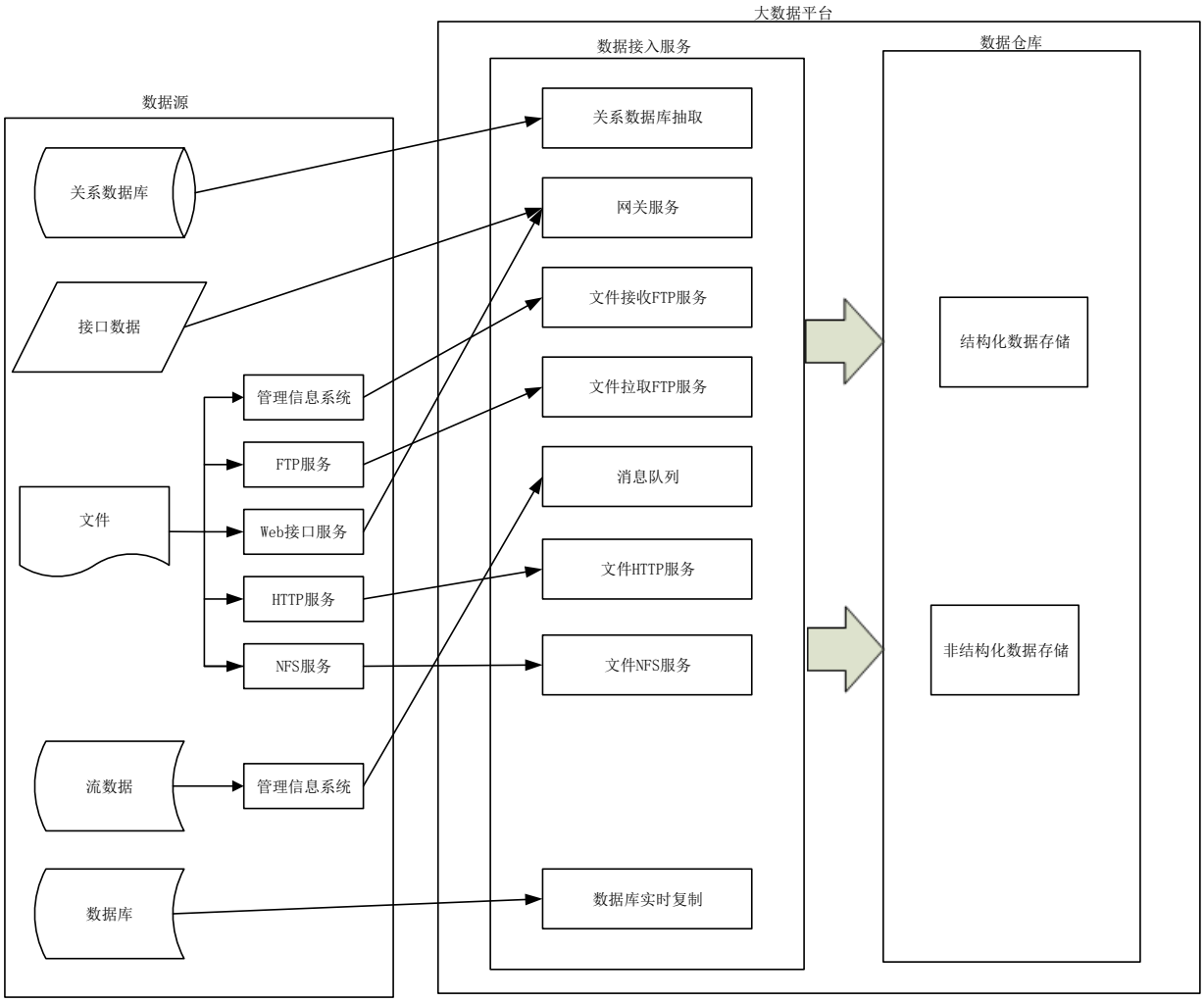


图1　总体框架示意图

5.2　数据源的接入和采集方式

大数据平台作为多维数据的处理平台，应支持各类数据源的接入和采集。常用数据源的接入和采集包含但不限于以下方式：

a)　关系数据库抽取；
b)　数据库实时复制；
c)　网关服务；
d)　消息队列服务；
e)　文件接收 FTP 服务；
f)　文件拉取 FTP 服务；
g)　文件 HTTP 服务；
h)　文件 NFS 服务。

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期批量抽取功能。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；
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、半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、定期调度运行。定期调度运行应提供多种调度策略，至少包括固定期间间隔运行、指定期间点运行、指定期间范围运行、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 应用场景描述

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。

6.1.3 应用场景图

关系数据库抽取应用场景见图2。

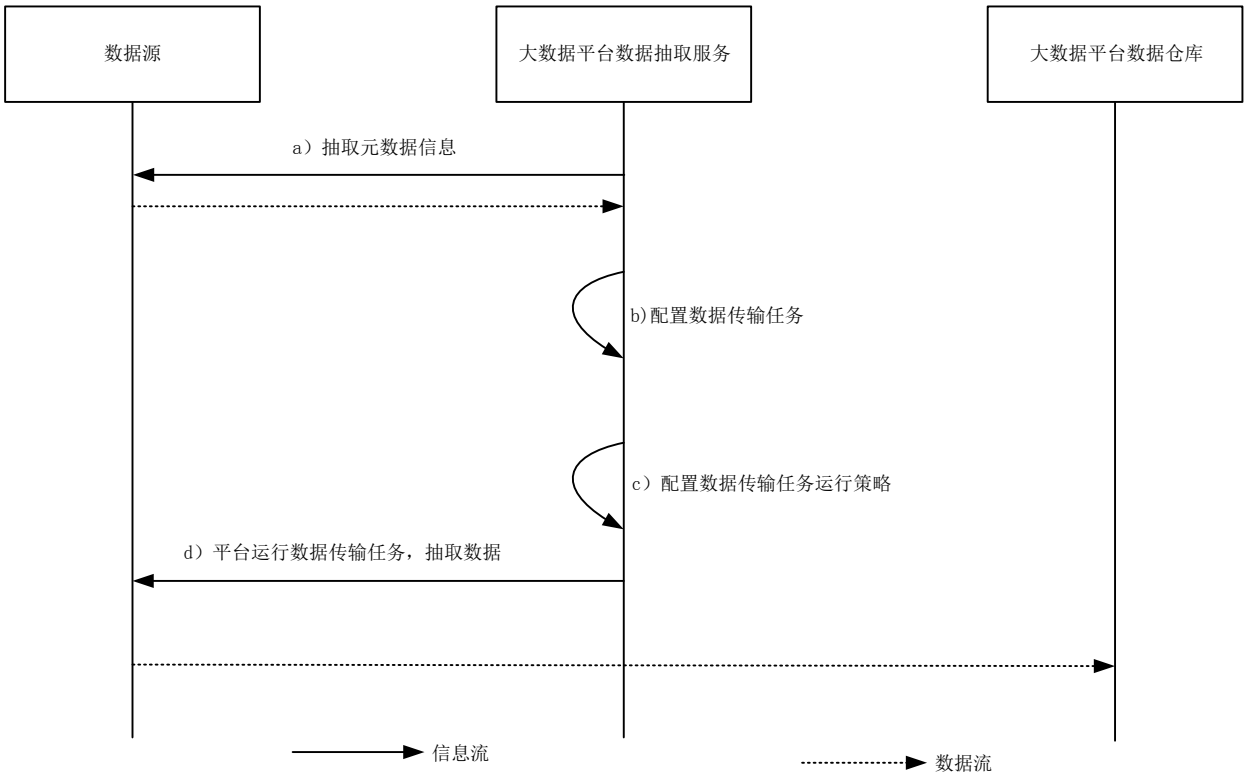


图2　关系数据库抽取应用场景

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期批量抽取场景，详细说明见附录 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　数据库实时复制

6.2.1　功能要求

数据实时复制应提供业务系统关系型数据库中的结构化数据到大数据平台数据存储的增量低时延复制功能，数据实时复制应具备以下主要功能：

a)　支持对主流的关系型数据库进行低时延增量复制，至少包括 Oracle、MySQL、PostgreSQL 等关系型数据库；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)　支持秒级时延的关系型数据库增量复制能力；同时支持在全量复制的基础上，无缝自动切换到增量复制模式；

c)　支持关系型数据库中结构化数据抽取到大数据平台关系型数据存储、非关系型数据存储、分布式文件存储、实时数据存储及消息队列；

d)　支持对关系型数据库数据的内容和类型转换操作，至少包括不同数据库差异转换、字段类型格式转换、时区转换等操作；

e)　支持对复制对象的过滤，至少包括模式过滤、表过滤、字段过滤、数据行过滤；

f) 支持数据复制操作的手动触发、定期调度及外部触发运行。定期调度运行应提供多种调度策略，至少包括固定期间间隔运行、指定期间点运行、指定期间范围运行、一次或指定次数运行等策略；外部触发支持标准 webservice 接口；
g) 应提供中心图形管理界面，应提供源端元数据查看、数据复制场景管理、数据表配置、表字段；
h) 选择配置、字段类型转换配置、触发机制配置、目标端输出配置、运行策略配置、运行监控等操作界面。

6.2.2 应用场景

应用过程如下：
a) 业务系统关系型数据库服务器上部署增量捕获程序；
b) 增量捕获程序捕获到增量数据通过 TCP 或消息队列发送到大数据平台增量接收服务；
c) 大数据平台解析增量数据，并将增量数据存储到大数据平台中；
d) 大数据平台分发增量数据到目标数据仓库中。

6.2.3 应用场景图

数据库实时复制应用场景见图3。

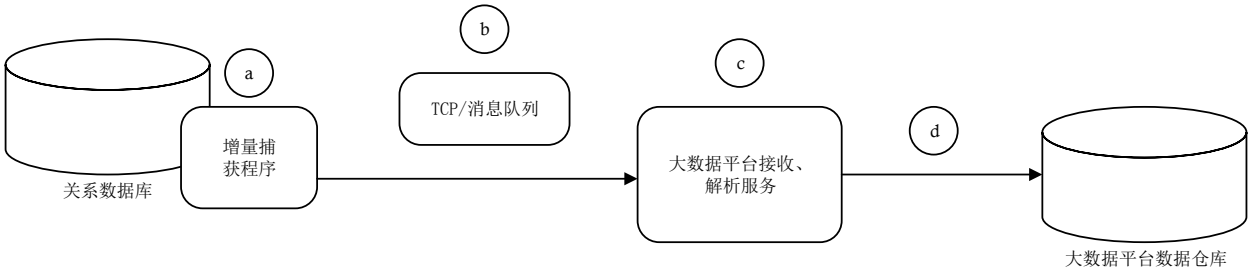


图3 数据库实时复制应用场景

6.2.4 应用要求

应用要求包括：
a) 源数据库应是关系型数据库；
b) 源数据库表应包含主键或唯一索引；
c) 源端数据库应开启附加日志和强制归档模式；
d) 源端数据库的在线日志文件、归档日志文件应存放在文件系统上。

6.3 网关服务

6.3.1 功能要求

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。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) 提供完善的日志和审计能力，应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） 具备熔断管理机制，接口访问异常情况下的处理策略，保证服务整体可用。

6.3.2 应用场景

网关服务应用场景描述如下:

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） 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） 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。

6.3.3 应用场景图

网关服务应用场景见图4。

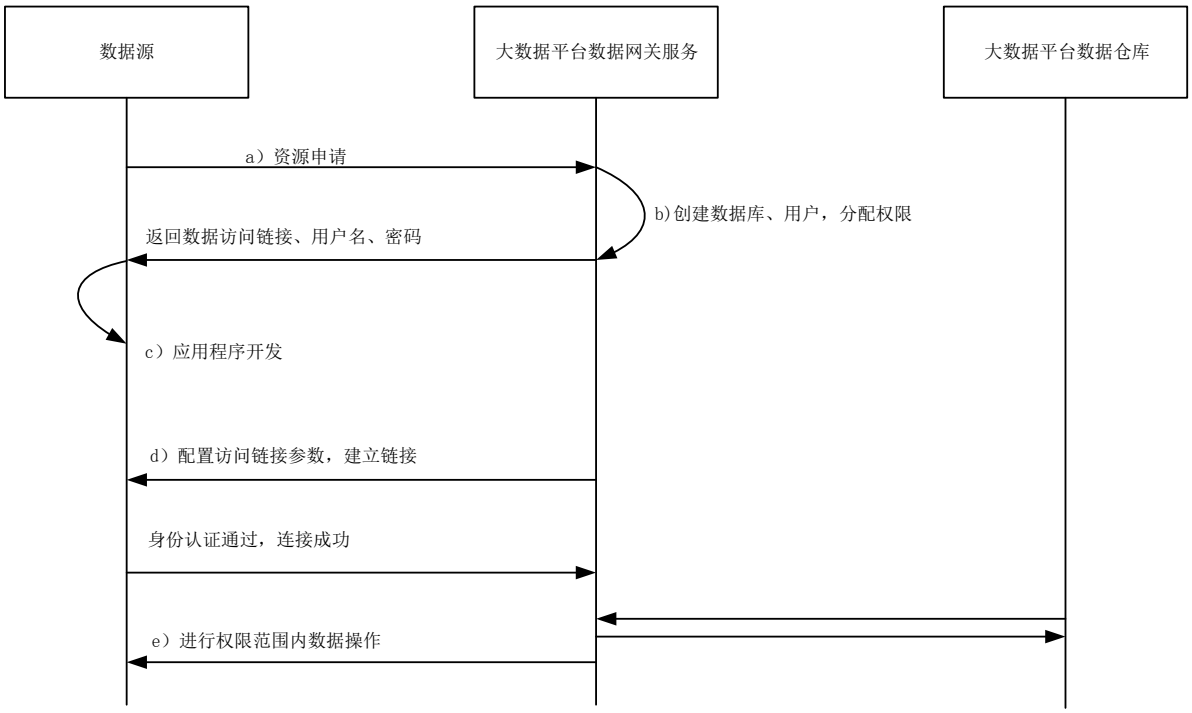


图4 网关服务应用场景

6.3.4 应用要求

网关服务应用要求包括：

a） 网关服务适用于提供接口类数据的数据源，详细说明见附录 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。

6.4 消息队列

6.4.1 功能要求

消息队列采集为管理信息系统提供消息推送和缓存功能。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） 应支持消息的持久化存储操作并且支持持久化周期设置；
d） 应提供消息分布式高可用的发送和消费接口，包括链接建立、消息发送、消息消费、链接关闭，支持消息分区和备份操作；
e） 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；
f） 具备完善的日志审计能力，应记录消息发送和消费时发生的各种事件。

6.4.2 应用场景

应用场景描述如下：
a） 管理信息系统应向大数据平台申请消息队列接入服务；
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，调用平台消息队列接口，发送数据或接收数据。

6.4.3 应用场景图

消息队列应用场景见图5。

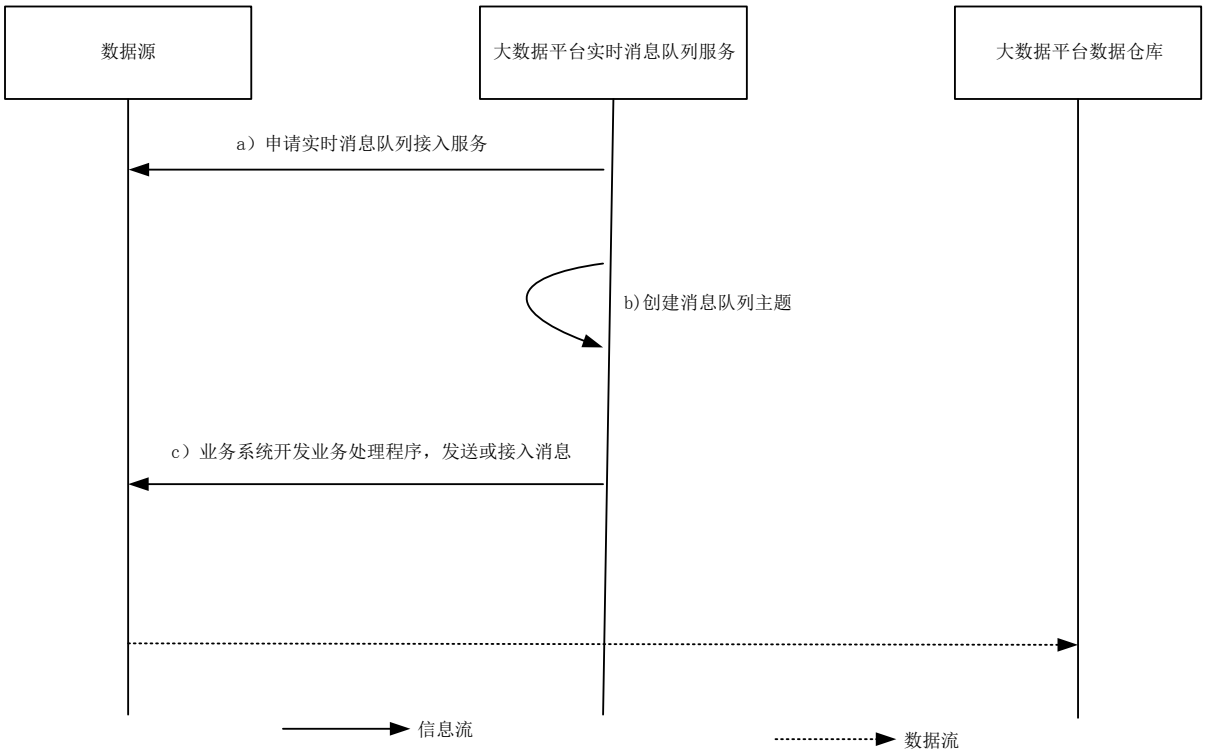


图5 消息队列应用场景

6.4.4 应用要求

应用要求如下：
a） 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消息队列中。基于消息队列的消息缓存进行数据分析，详细说明见附录 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。

6.5 文件接收 FTP 服务

6.5.1 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。

6.5.2 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。

6.5.3 应用场景图

文件FTP服务应用场景见图6。

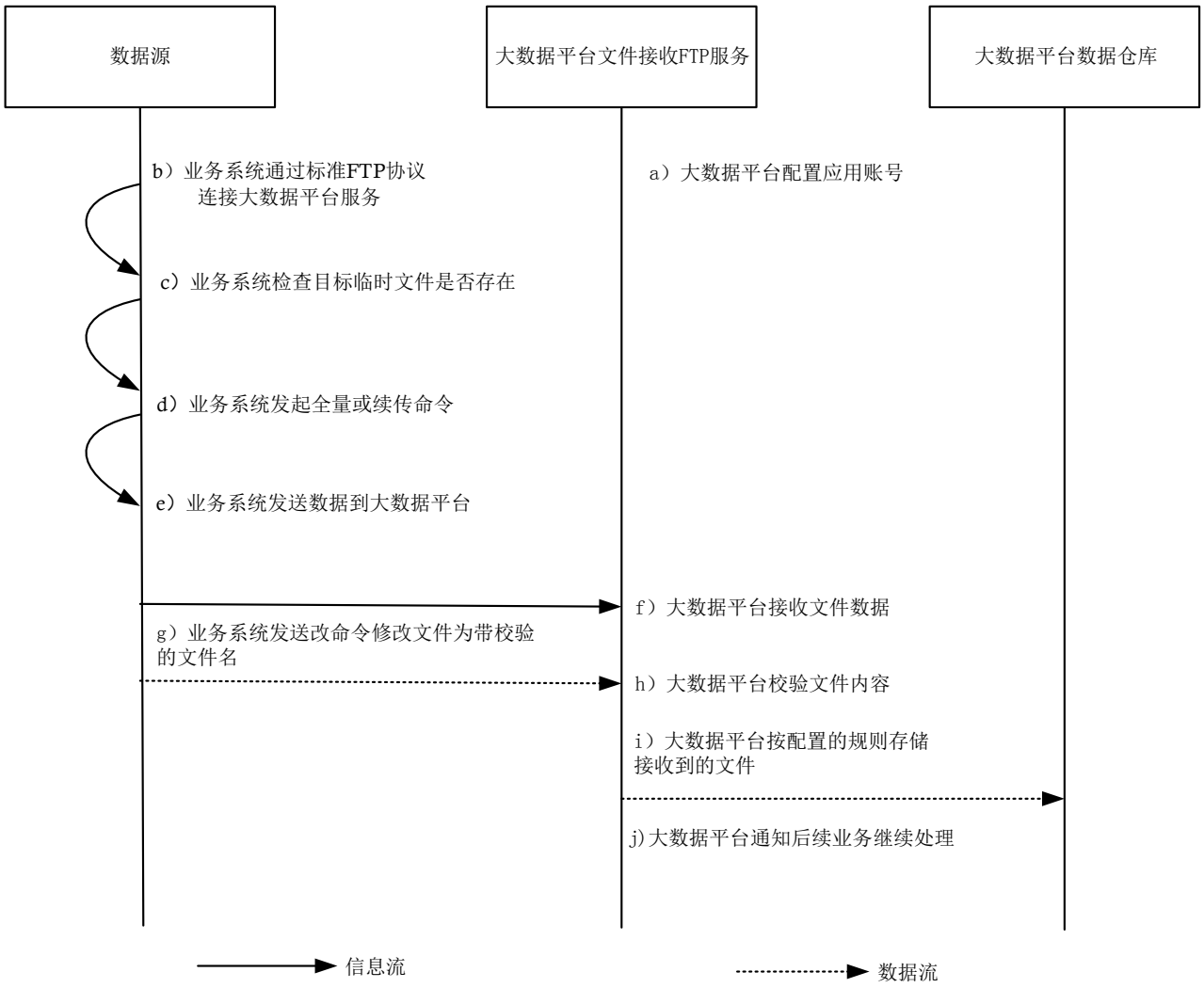


图6　文件 FTP 服务应用场景

6.5.4　应用要求

应用要求如下：

a)　管理信息系统应在大数据平台注册并申请账号；

b)　管理信息系统应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应同时生成对应的完整性校验码；

d)　具体文件接收 FTP 服务 API 接口详细说明见附录 D。

6.6　文件拉取 FTP 服务

6.6.1　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） 支持定期轮询文件采集，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、文件定期及实时策略配置、文件采集过滤配置。

6.6.2 应用场景

6.6.2.1 基于 FTP 协议的全量文件采集应用场景

应用场景描述如下：
a） 业务系统应提供初始文件存储位置；
b） 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；
c） 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仓库中。

6.6.2.2 应用场景图

基于FTP协议的全量文件采集应用场景见图7。

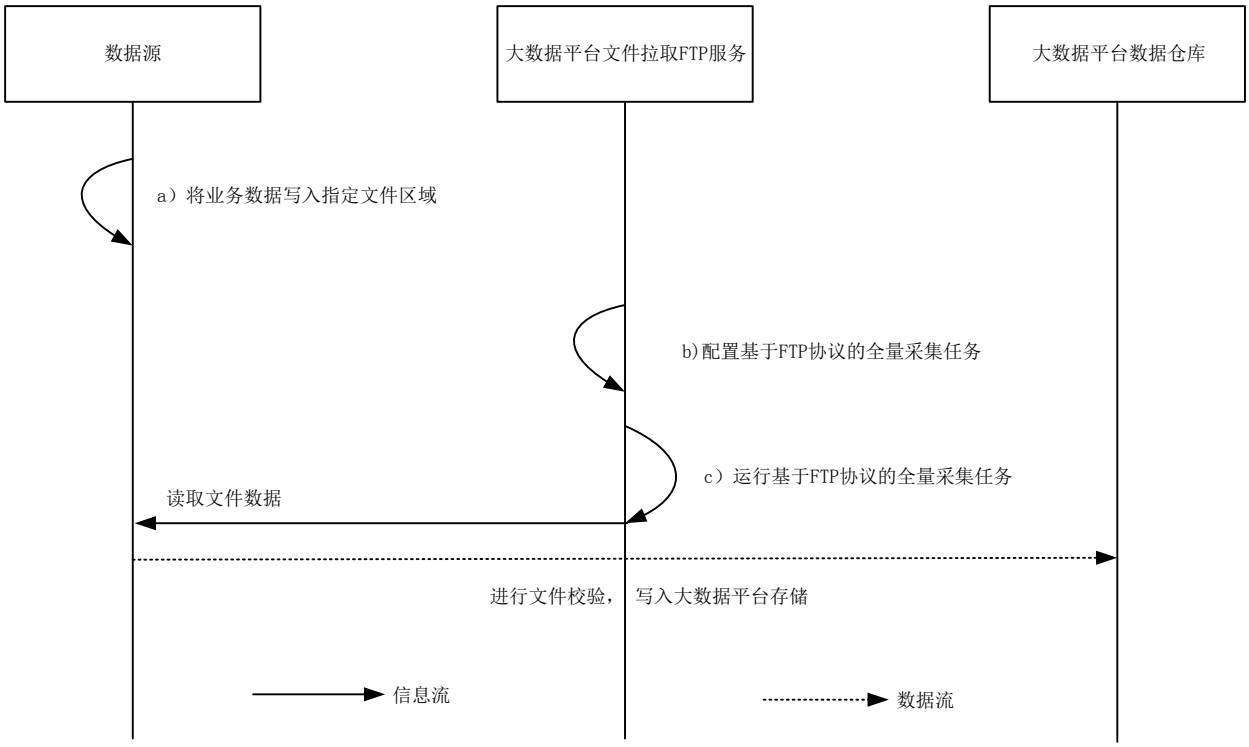


图7 基于 FTP 协议全量采集应用场景

6.6.2.3 基于 FTP 协议的定期轮询采集应用场景

应用场景描述如下：

a) 业务系统应按照业务需求生成数据文件，存储在指定文件区域；
b) 业务系统应根据校验算法 SHA1 在相同目录下生成校验文件，校验文件名称与数据文件相同；
c) 大数据平台轮询文件采集服务配置基于 FTP 协议的定期轮询采集任务，定期轮询采集文件；
d) 大数据平台定期读取解析索引及校验文件内容；如果业务系统数据文件出现新增时，大数据平台读取数据文件内容，并且进行文件校验，如果文件完整，则将数据文件写入大数据平台中。

6.6.2.4 应用场景图

基于FTP协议的定期轮询采集应用场景见图8。

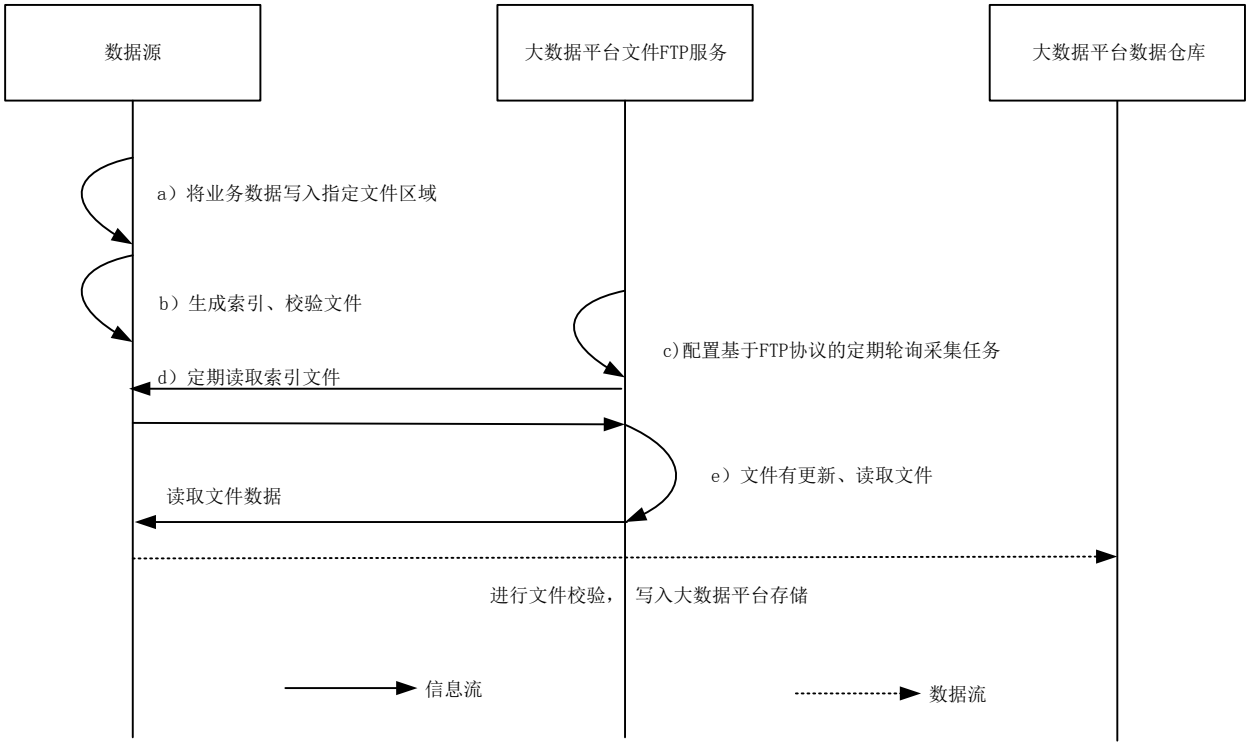


图8 基于 FTP 协议的定期轮询采集应用场景

6.6.3 应用要求

应用要求如下：
a) 业务系统应先将业务数据保存为文件，并设置访问权限；
b) 文件数据校验算法应支持标准的 SHA1 进行校验，结果转化为 16 进制 ASCII 字符表示；
c) 数据文件可通过 FTP 协议访问；
d) 业务系统生成文件数据时应同时生成对应的完整性校验码；
e) 具体文件拉取 FTP 服务 API 接口详细说明见附录 E。

6.7 文件 HTTP 服务

6.7.1 功能要求

文件HTTP服务，应提供通过访问HTTP协议实现将文件信息抽取到大数据平台存储的功能。文件HTTP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) 支持文件压缩传输，提供文件压缩传输规则设置；

c） 支持文件加密传输，提供文件加密传输规则设置；
d） 支持设置文件同步、异步拉取，支持设置拉取并行度；
e） 支持制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；
f） 支持全量文件采集，支持外部数据一次性初始化导入；
g） 支持定期轮询文件采集，采集新增的文件，支持文件列表规则过滤；
h） 支持图形管理功能，支持文件源配置、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、文件定期及实时策略配置、文件采集过滤配置。

6.7.2 应用场景

6.7.2.1 基于 HTTP 协议的全量采集应用场景

应用场景描述如下：
a） 业务系统应提供初始文件存储位置；
b） 大数据平台全量文件采集提供基于 HTTP 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、传输是否压缩、加密、同步/异步、并行度、存储目标位置及文件存储类型转换和命名规则；
c） 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仓库中。

6.7.2.2 应用场景图

基于HTTP协议的全量采集应用场景见图9。

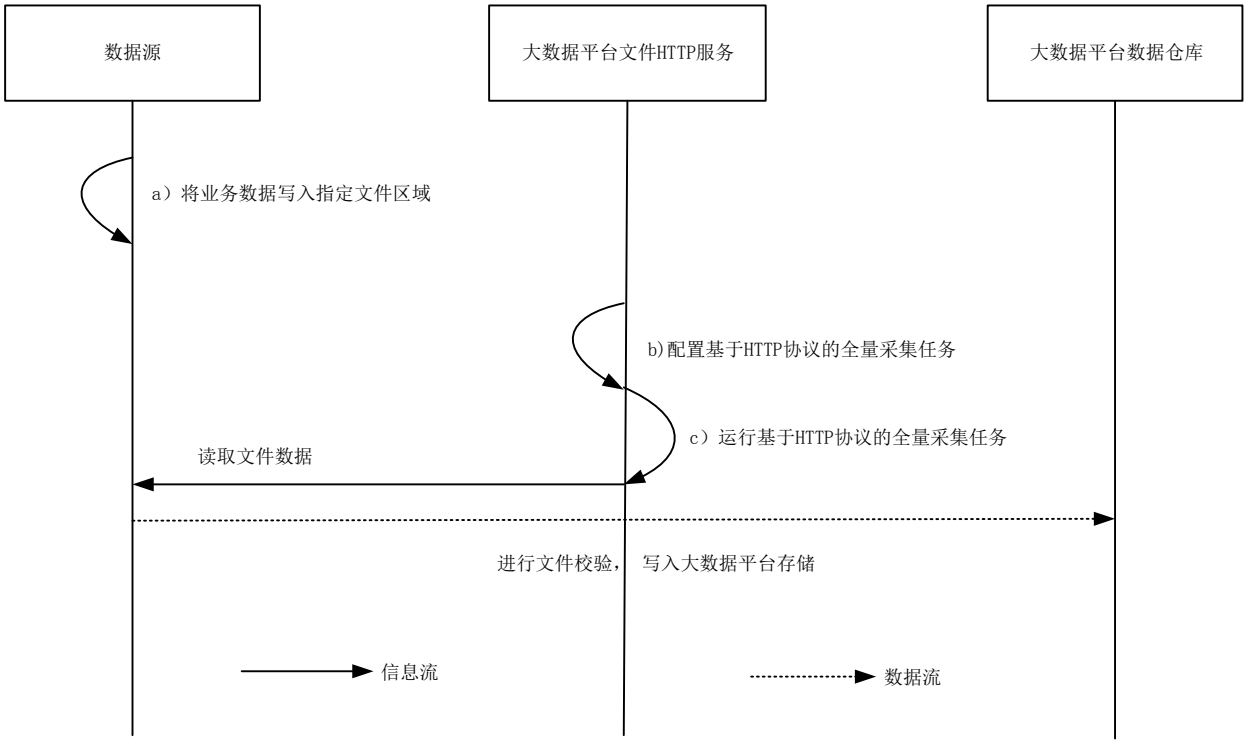


图9 基于 HTTP 协议的全量采集应用场景

6.7.2.3 基于 HTTP 协议的定期轮询采集应用场景

应用场景描述如下：

a) 业务系统应按照业务需求生成数据文件，存储在指定文件区域；
b) 业务系统应根据校验算法 SHA1 在相同目录下生成校验文件，校验文件名称与数据文件相同；
c) 大数据平台轮询文件采集服务配置基于 HTTP 协议的定期轮询采集任务，定期轮询采集文件；
d) 大数据平台定期读取解析索引及校验文件内容；如果业务系统数据文件出现新增时，大数据平台读取数据文件内容，并且进行文件校验，如果文件完整，则将数据文件写入大数据平台中。

6.7.2.4 应用场景图

基于HTTP协议的定期轮询采集应用场景见图10。

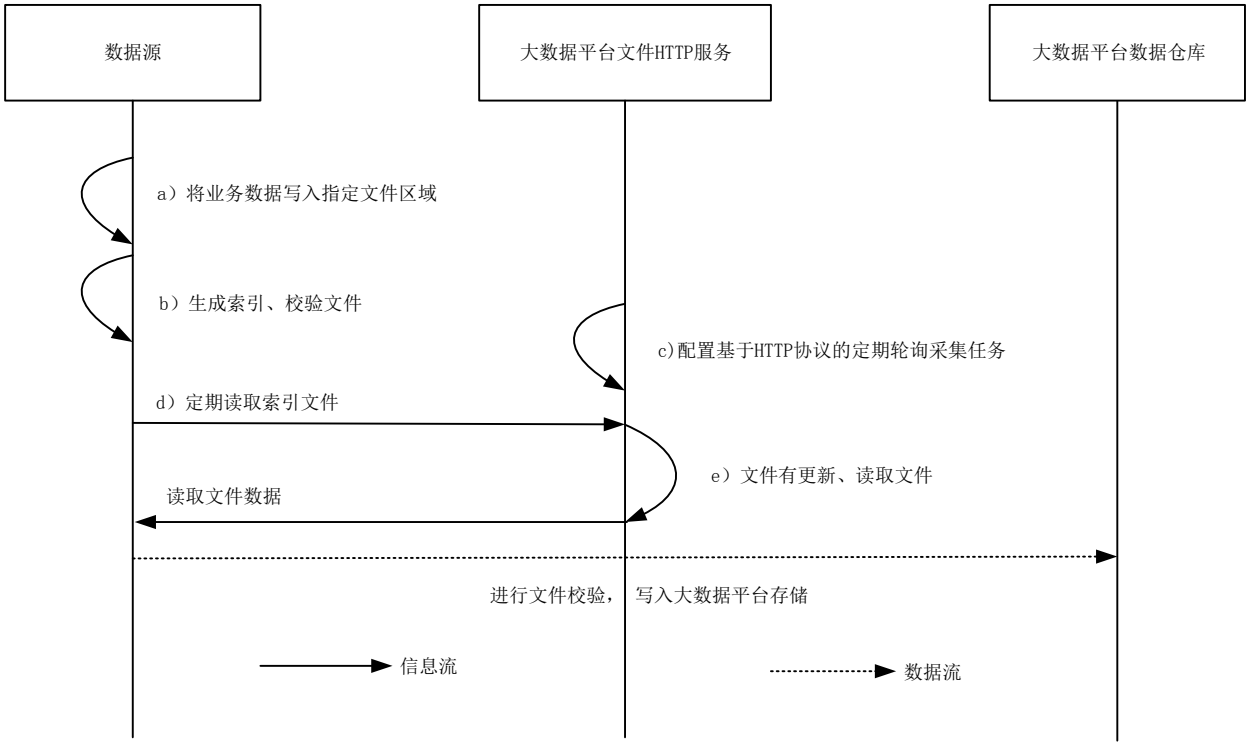


图10 基于 HTTP 协议的定期轮询采集应用场景

6.7.3 应用要求

应用要求如下：

a) 业务系统须先将业务数据保存为文件；
b) 文件数据校验算法须支持标准的 SHA1 进行校验，结果转化为 16 进制 ASCII 字符表示；
c) 数据文件可通过 HTTP 协议访问；
d) 业务系统生成文件数据时应同时生成对应的完整性校验码；
e) 具体文件 HTTP 服务 API 接口详细说明见附录 F。

6.8 文件 NFS 服务

6.8.1 功能要求

文件NFS服务，应提供通过访问NFS文件系统，实现将文件信息抽取到大数据平台数据存储的功能。文件NFS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；

c) 支持全量文件采集，支持外部数据一次性初始化导入；
d) 支持定期轮询文件采集， 采集新增的文件， 支持文件列表规则过滤；
e) 支持图形管理功能，支持文件源配置、文件目标存储配置、文件压缩和加密传输规则配置、文件定期及实时策略配置、文件采集过滤配置。

6.8.2 应用场景

6.8.2.1 NFS 全量文件采集应用场景

应用场景描述如下：
a) 业务系统应提供初始文件存储位置；
b) 大数据平台全量文件采集提供采集 NFS 文件任务配置，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；
c) 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仓库中。

6.8.2.2 应用场景图

NFS全量文件采集应用场景见图11。

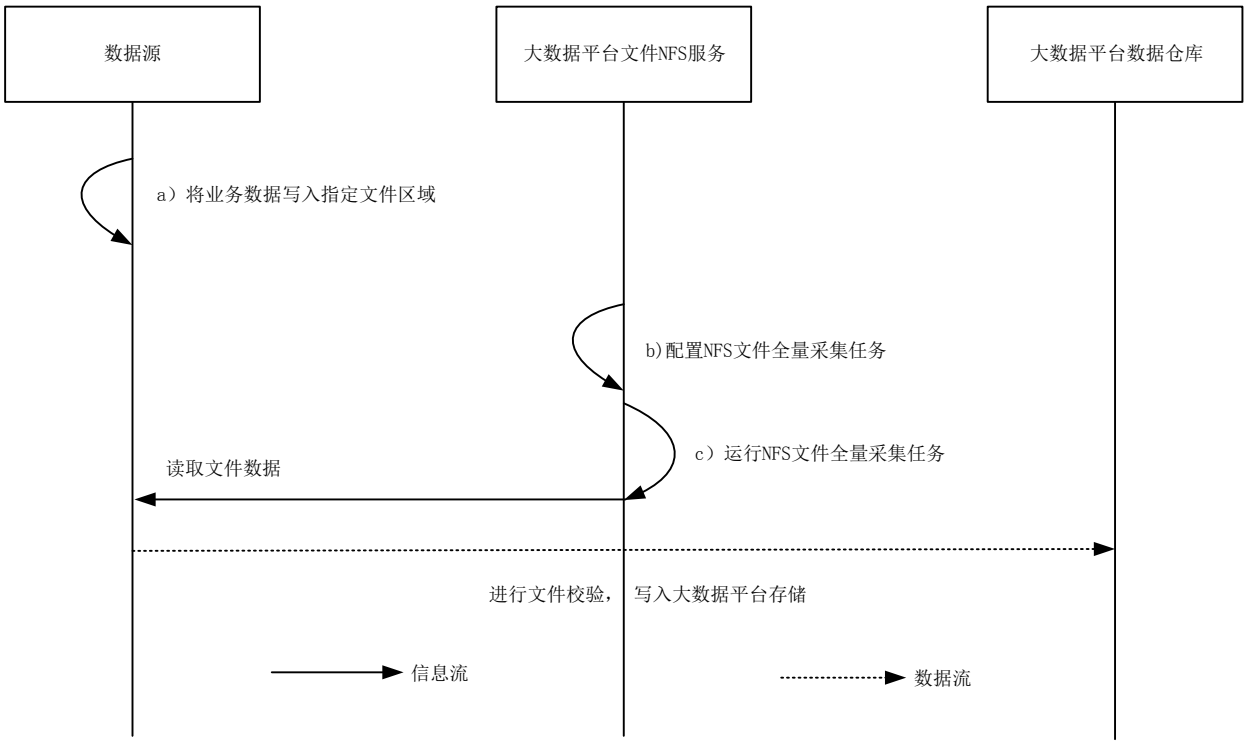


图11 NFS 全量文件采集应用场景

6.8.2.3 NFS 文件定期轮询采集应用场景

应用场景描述如下：
a) 业务系统应按照业务需求生成数据文件，存储在指定文件区域；
b) 业务系统应根据校验算法 SHA1 在相同目录下生成校验文件，校验文件名称与数据文件相同，文件扩展名为.SHA1；
c) 大数据平台轮询文件采集服务配置 NFS 文件定期轮询采集任务，定期轮询采集文件；

d) 大数据平台定期读取解析索引及校验文件内容；如果业务系统数据文件出现新增时，大数据平台读取数据文件内容，并且进行文件校验，如果文件完整，则将数据文件写入大数据平台中。

6.8.2.4 应用场景图

NFS文件定期轮询采集应用场景见图12。

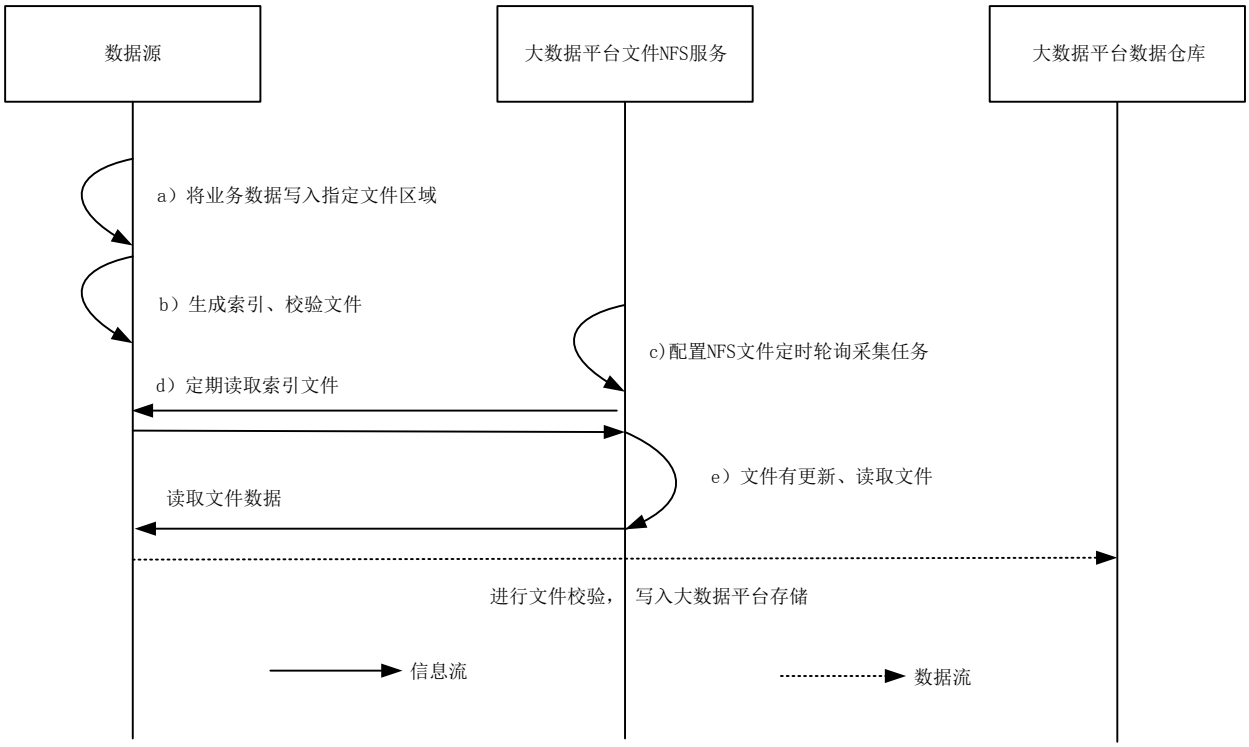


图12 NFS 文件定期轮询采集应用场景

6.8.3 应用要求

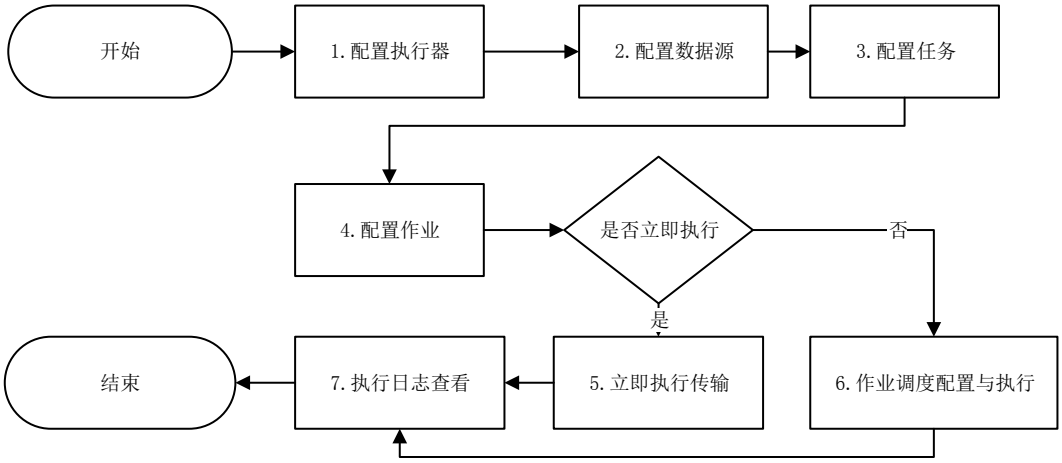
应用要求如下：

a) 业务系统应先将业务数据保存为文件，并设置访问权限；
b) 文件数据校验算法应支持标准的 SHA1 进行校验，结果转化为 16 进制 ASCII 字符表示；
c) 数据文件可通过 NFS 访问；
d) 业务系统生成文件数据时应同时生成对应的完整性校验码；
e) 具体文件 NFS 服务 API 接口详细说明见附录 G。

附 录 A
（资料性）
关系数据库抽取接入说明

A.1 关系数据库应用流程图

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图

A.2 应用流程

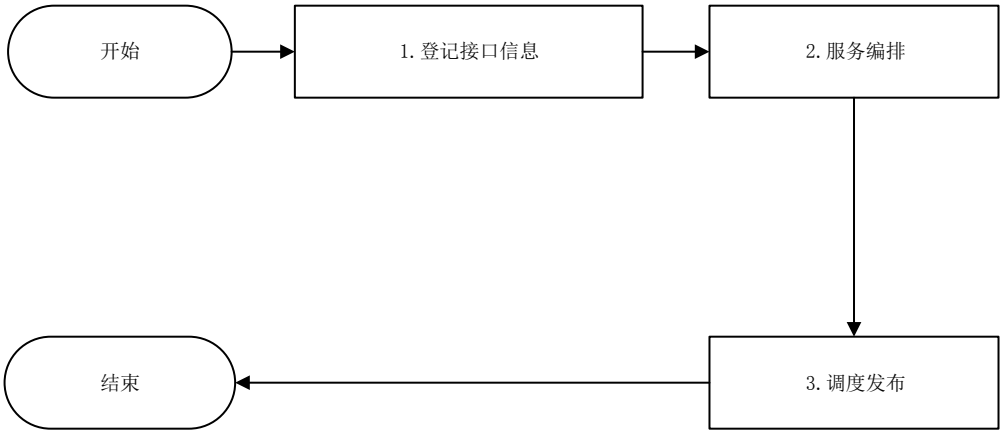
应用流程描述如下：

a) 大数据平台配置执行器，用于作业调度执行的服务；
b) 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；
c) 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；
d) 配置作业，根据抽取任务执行的先后顺序配置成作业；
e) 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；
f) 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）
网关服务接入说明

B.1 网关服务应用流程图

通过大数据平台的网关服务提供的Web界面进行操作。首先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。 网关服务应用流程见图B.1：



图B.1 网关服务应用流程图

B.2 应用流程

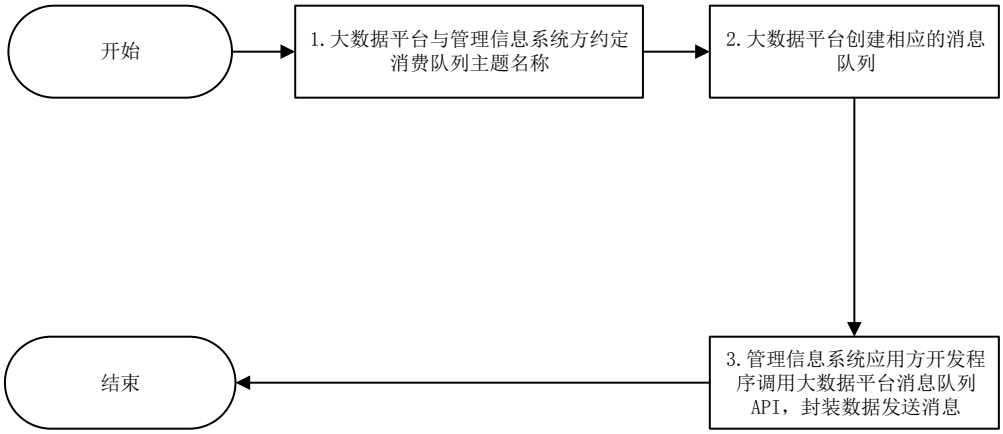
应用流程描述如下：

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）
消息队列接入说明

C.1 消息队列接入说明

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



图C.1 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消息队列 API，封装数据往消息队列主题中发送消息。

C.2 消息队列API

消息队列API接口清单见表C.1。

表C.1 消息队列 API 接口清单

序号	接口方法	接口说明
1	Producer(String username, String password , List<String>)	构建消息队列主题发送实例。参数说明如下： username:用户名 password:密码 topics：消息队列主题名称
2	send(String topic, String message)	往主题发送单条消息。参数说明如下： topic：主题名称 message：数据内容
3	send(String topic, String messages)	往主题发送多条消息。参数说明如下： topic：主题名称 messages：多条数据内容
4	send(String topic, String message , Boolean isSensitive)	往主题发送单条消息,进行加密后发送。参数说明如下： topic：主题名称 message：数据内容 isSensitive：是否敏感信息
5	send(String topic, String messages , Boolean isSensitive)	往主题发送单条消息,进行加密后发送。参数说明如下： topic：主题名称 messages：多条数据内容 isSensitive：是否敏感信息
6	Producer.close()	关闭消息队列主题发送实例的连接
7	Consumer(String username , String password , List<String> topics)	构建消息队列主题接收实例。参数说明如下： username:用户名 password:密码 topics：消息队列主题名称
8	fetch(String topic,EventHander handler)	监听消息主题的消息。参数说明如下： topic：主题名称 handler:接收到数据后的处理类
9	Consumer.close()	关闭消息队列主题接收实例的连接

附 录 D
（资料性）
文件接收 FTP 服务接入说明

D.1 文件接收FTP服务API接口

文件接收FTP服务API接口见表D.1。

表D.1 文件接收 FTP 服务 API 接口

序号	接口方法	接口说明
1	public void initFtpClient (String hostname, Integer 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

业务系统在生成好文件后，通过调用大数据平台FTP文件接收服务。调用服务示例如下:
示例：

```
    public void initFtpClient (String hostname, Integer port, String username, String password) {
    ftpClient = new FTPClient0();
    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 filelame, InputStrean inputStrean) {
try {
    Systen. 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, inputStrean);
    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 != inputStrean) {
    try {
    inputStrean close ( ) ;
    } catch (IOException e) {
```

```
e. printStackTrace () ;
}
}
}
return true;
};
public boolean CreateDirecroty(String remote)  throws IOException {
boolean success = true;
String directory = renote + "/";
//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. startsith("/")) {
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"), "is0-8859-1);
path = path + "/" + subDirectory;
if (lexistFile(path)) {
if (makeDirectory (subDirectory)) {
changeorkingDirectory (subDirectory);
} else {
Systen. 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）
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接口清单见表E.1。

表E.1 文件拉取 FTP 服务 API 接口清单

序号	接口方法	接口说明
1	downloadViaFTP (String ip, int port, String usermame,String password,String remotePath, String fileName, String localPath)	FTP文件下载。参数说明如下: ip: FTP服务地址 port: FTP 服务端口 username:FTP服务用户名 password: FTP 服务密码 remotePath:远程文件路径 fileName:文件名 localPath:本地临时中转目录
2	getFileSize(String fileName)	获取文件大小。参数说明如下: fileName:文件名

E.2 文件拉取FTP服务示例

业务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。调用服务示例如下：
示例：

```
private static FTPClient fpClient = new FTPClient();
private static String encoding = System.getProperty(" file.encoding");

public static boolean downFileViaFtp(String ip, int port, String username,
        String password, String remotePath, String fileName,
        String localPath) {
    boolean result = false;
    try {
        int reply;
        ftpClient.setC ontrolEncoding(encoding);

        ftpClient.connect(ip, port);
        ftpClient,login(username, password);
        ftpClient.setFileType(FTPClient.BINARY_ FILE_TYPE);
        reply = fpClient.getReplyCode();
        If (!FTPReply.isPositiveCompletion(reply)) {
            fipClient.disconnect();
            System.err. printn("FTP server refused conection.");
            return result;
        }
        ftpClient.changeWorkingDirectory(new String(remotePath.getBytes(encoding).
        "iso-8859-1"));
        FTPFile[] fs = ftpCient.listFiles();
        for (FTPile ff:fs) {
            if (ff.getName0.equals(fileName)) {
                File localFile = new File(localPath + "/" + ff.getName());
                OutputStream is = new FileOututstream(localFile);
                ftpClient.retrieveFile(ff.getName(), is);
                is.close();
            }
        }
        ftpClientlogout();
        result = true;
  } catch (IOException e) {
      e.printStackTrace(;
  } finally {
      if (ftpClient.isConnected()) {
         Try{
              ftpClient.disconnect();
           }catch (IOException ioe){
                        if (ftpClientisConnected0) {
                            try {
```

```
                                ftpClient.disconnect(;
                            } catch (IOException ioe) {
                            }
                        }
                    }
                    Return result;
                }
public static void main(String[] args) {
      boolean flag = downFileViaFtp("127.0.0.1", 21, "user",
                "password", "/", "测试txt", "home/ftpfile/");
      。。。。
      IHdisService hdfsService = (IHdfsService) UtilFactory.getHadoopUtil
      (UtilType.HDFSService, "hdfs");
       ......
      hdfsService.upload(hdfspath, localpath, overwrite);
          }
```

附 录 F
（资料性）
文件 HTTP 服务接入说明

F.1 文件HTTP服务API接口

文件HTTP服务API接口清单见表F.1。

表F.1 文件 HTTP 服务 API 接口清单

序号	接口方法	接口说明
1	uploadFile(String actionUrl, String[] uploadFilePaths)	上传文件。参数说明如下： actionUrl：请求地址 inStream：文件路径列表
2	getFileSize(String fileName)	获取文件大小。参数说明如下： fileName：文件名

F.2 文件HTTP服务示例

业务系统在生成好文件后，通过调用大数据平台HTTP文件服务。调用服务示例如下:
示例：

```
public class HttpTransFile{
    @Suppess Warnings("finally")
    public static String uploadFile(String actionUrl, String[] uploadFilePaths) {
        String end = "\r\n";
        String twoHyphens="- -";
        String boundary="******";
        DataOutputStream ds = null;
        InputStream inputStream = null;
        InputStreamReader inputStreamReader = null;
        BufferedReader reader = null;
        StringBuffer resultBuffer = new StringBuffer();
        String tempLine = null;
        try {
            URL url = new URL(actionUrl);
            URLConnection urlConnection = url.openConnection();
            HttpURLConection httpURLConnection = (HttpURLConnection) urlConnection;
            try {
                URL url = new URL(actionUrl);
                URLConnection urlConnection = url.openConnection();
                HttpURLConection httpURLConnection = (HttpURLConnection) urlConnection;
                httpURLConnection.setDolnput(true);
                httpURLConnection.setDoOutput(true);
                httpURLConection.setUseCaches(false);
                httpURLConection.setRequestMethod("POST");
                httpURLConnection.setRequestProperty("Connection", "Keep-Alive");
                httpURLConnection.setRequestProperty("Charset", "UTF-8");
            httpURLConnection.setRequestProperty"Content-Type","multipart/form-data;boundary="          +
boundary);
                ds = new DataOutputStream(httpURLConnection.getOutputStream());
                for (int i= 0; i < uploadFilePaths.length; i++){
                    String uploadFile = uploadFilePaths[i];
                    String filename = uploaFile.substring(uplodFile.lastTndexOF("//")+ 1);
                    ds.writeBytes(twoHyphens + boundary + end);
                    ds.writeBytes("Content-Disposition: form-data; " + "name =\"file" + i +"\",filename =\""
+ filename+ "\"" + end);
                    ds.writeBytes(end);
                    FileInputStream fStream = new FileInputStream(uploadFile);
                    int bufferSize = 1024;
                    byte[] buffer = new byte[bufferSize];
                    int length=-1;
                    while ((length = fStream.read(buffer))!=-1){
```

```
                    ds.writer(buffer,0,length)
}
                    ds.writeBytes(end);
                    /* close streams */
                    fStream.close();
                }
                ds.writeBytes(twoHyphens+boundary+twoHyphens+end);
                /* close streams */
                ds.flush();
                if(httpURLConnection.getResponseCode()>=300){
                    throw new Exception("HTTP Request is not success , Response code is "+
httpURLConnection.getResponseCode());
                }
                if(httpURLConnection.getResponseCode()==HttpURLConnection.HTTP_OK){
                    inputStream=HttpURLConnection.getInputStream();
                    inputStreamReader=new InputStreamReader(inputStream);
                    reader=new BufferedReader(inputStreamReader);
                    tempLine=null;
                    resultBuffer-new StringBuffer();
                    while((tempLine=reader.readLine())!=null){
                       resultBuffer.append(tempLine);
                       resultBuffer.append("\n");
                        }
                }
            }catch(Exception e){
                e.prinstStackTrace();
            }finaly{
                if(ds!=null){
                    try{
                        ds.close();
                    }catch(IOException e) {
                        e.printStackTrace();
                    }
                }
                if(reader!=null){
                    try{
                        reader.close();
                    }catch(IOException e) {
                        e.printStackTrace();
                    }
                }
```

```
  if(inputStreamReader!=null){
                    try{
                        inputStreamReader.close();
                        }catch(IOException e) {
                        e.printStackTrace();
                    }
                }
                if(inputStream!=null){
                    try{
                        inputStream.close();
                    }catch(IOException e) {
                        e.printStackTrace();
                    }
                }
                return resuItBuffer.toString();
            }
        }
    }
    public static void main(String[]args){
                String str = uploadFile("地址链接".new String[]{"需上传文件所在地址 1","需上传文件所在地
址 2"});
        }
    }
```

附 录 G
（资料性）
文件 NFS 服务接入说明

G.1 文件NFS服务API接口

文件NFS服务API接口清单见表G.1。

表G.1 文件 NFS 服务 API 接口清单

序号	接口方法	接口说明
1	downloadViaNFS (final String ip,final String user,final String password,final String dir)	NFS文件下载。参数说明如下: ip:NFS服务器地址 user:NFS服务用户名 password: NFS 服务密码 dir:文件所在路径
2	getFileSize(String fileName)	获取文件大小。参数说明如下: fileName:文件名

G.2 文件NFS服务示例

业务系统在生成好文件后，通过调用大数据平台NFS文件拉取服务。调用服务示例如下：
示例：

```
public void downloadViaNFS(final String ip,final String user,final String password,final String dir){
    logger.debug("NFS download begin!");
    try {
        String url = "nfs://"+ip+"/"+dir;
        XFile xf = new XFile(url);
        if (xf.exists()){
            logger.debug("URL is OK!");
        }else{
            logger.debug("URL is bad!");
            return;
        }
        XFileExtensionAccessor nfsx = (XFilcExtensionAccessor)xf.getExtensionAccessor();
        if(!nfsx.loginPCNFSD(ip. user, passwo rd){
            logger.debug("login failed"!);
            return;
        }
        private static FTPClient fpClient = new FTPClient();
        private static String encoding = System.getProperty(" file.encoding");
public static boolean downFileViaFtp(String ip, int port, String username,String password, String remotePath,
String fileName,String localPath) {
            boolean result = false;
            try {
                int reply;
                ftpClient.setC ontrolEncoding(encoding);
                ftpClient.connect(ip, port);
                ftpClient,login(username, password);
                ftpClient.setFileType(FTPClient.BINARY_ FILE_TYPE);
                reply = fpClient.getReplyCode();
                If (!FTPReply.isPositiveCompletion(reply)) {
                    fipClient.disconnect();
                    System.err. println("FTP server refused conection.");
                }
                String[]fileList = xf.list();
                XFile temp = null;
                long startTime = System.currentTimeMillis();
                int filesz= 0;
                for(String file:fileList){
                temp = new XFile(url+ "/"+file);
                XFileInputStream in = new XFileInputStream(temp) ;
                XFileoutputStream out = new XFileoutputStream(tempDir +File separator+file);
```

```
        int c;
                    byte[] buf = new byte[8196];
                     while ((c = in.read(buf)>0) {
                        filesz += c;
                        out.write(buf, 0, c);
                        if (temp.can Write()){
                            temp.delete();
                            logger.debug(file + "is deleted!");
                        }else{
                            logger.debug(file + " can not be delted!");
                        }
                    }
                    long endTime = System.currentTimeMillis();
                    long timeDiff = endTime - starTime;
                    int rate = (int) ((filesz /1000) / (timeDiff / 1000.0));
                    logger.debug(filesz + " bytes copied @ " + rate + "Kb/sec");
                }catch (lOException e) {
logger.debug(e);
                }


            }


        }
    }
}
public static void main(String[] args) {
    downloadViaNFS(ip, username, password,dir );
    ...
    IHdfsService      hdfsService      =      (IHdfsService)
    UtilFactory.getHadoopUtil(UtilType.HDFSService, "hdfs");
    ...
    hdfsService.upload(hdfspath, localpath, overwrite);
}
```