江苏南大先腾信息产业有限公司  
数据交换平台产品白皮书

2014年6月

目 录

[1. 项目概述 3](#_Toc401753635)

[2. 总体架构 3](#_Toc401753636)

[2.1 平台架构 3](#_Toc401753637)

[2.2 设计原则 4](#_Toc401753638)

[2.3 关键技术 5](#_Toc401753639)

[2.4 业务模型 6](#_Toc401753640)

[3. 主要功能 8](#_Toc401753641)

[3.1 配置中心 8](#_Toc401753642)

[3.2 控制中心 11](#_Toc401753643)

[3.3 监控中心 12](#_Toc401753644)

[4. 应用场景 14](#_Toc401753645)

[4.1 基于数据库的交换 14](#_Toc401753646)

[4.2 基于WebService接口的交换 14](#_Toc401753647)

[4.3 组合交换 15](#_Toc401753648)

[5. 运行环境要求 16](#_Toc401753649)

[6. 服务与维护 16](#_Toc401753650)

[6.1 培训服务 16](#_Toc401753651)

[6.2 维护服务 16](#_Toc401753652)

[附件1：WebService接口配置规范 19](#_Toc401753653)

[1、用WebService提供数据 19](#_Toc401753654)

[2、用WebService接收数据 19](#_Toc401753655)

[3、WebService接口规范 19](#_Toc401753656)

[附件2：数据格式配置规范 24](#_Toc401753657)

[1、数据内容 24](#_Toc401753658)

[2、数据格式 25](#_Toc401753659)

1. 项目概述

为了有效整合分散异构的信息资源，消除“信息孤岛”现象，提高政府和企业的信息化水平。我公司采用SOA设计理念，自主研发了具有多种数据交换方式、灵活的业务规则定义、基于XML报文标准、全面的交换监控和统计、可视化的用户定制界面等特点的数据交换平台。

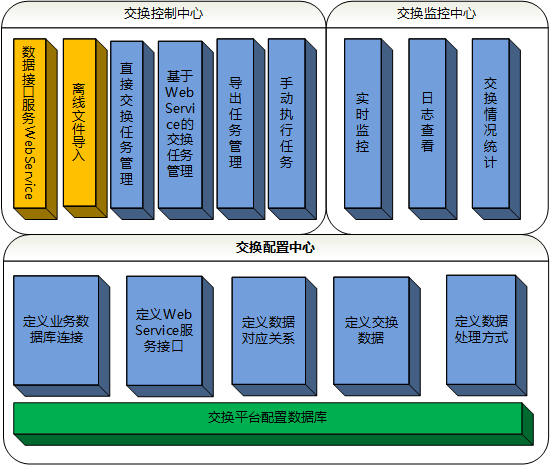
本平台把各种纷繁复杂的数据系统集成在一起完成特定业务，提供同构数据、异构数据之间的数据抽取、格式转换、内容过滤、内容转换、同异步传输、动态部署、可视化管理监控等方面功能，支持的数据包括各主流数据库、数据接口等各种格式。

数据交换平台主要作用是数据同步和数据集成。

* 数据同步：是指保持两个或者多个业务系统之间共享数据的一致性和完整性。
* 数据集成：是指将不同业务系统中的数据经过清洗、整理后汇总到一个数据中心，这样以便于数据共享、分析、挖掘。

1. 总体架构
   1. 平台架构

数据交换平台是基于J2EE开发的B/S结构的应用程序，其基本架构如下图所示：



* 1. 设计原则
     1. 稳定性原则

数据交换平台作为数据交换和集成的工具稳定时首要考虑的问题。系统有完整的异常处理方案，并采用以下设计准则：

* 多任务并行处理原则；系统中多个交换任务不能互相干扰，一个任务失败时不能影响其他任务顺利执行。
* 完善的异常处理逻辑，保证数据交换过程中的异常不会影响交换继续执行，整个交换作为一个事务的除外。
* 完善的日志记录；所有的操作都记录日志，对发生异常的操作日志同时记录交换的数据信息。
  + 1. 可扩展性原则

数据交换平台作为一个工具可扩展性使其适应能力的重要体现。交换工具在可扩展性方面采用一下设计原则：

* 交换的数据源、目标可以灵活配置。
* 交换的逻辑对应关系可以灵活配置。
* 交换前、交换后、出现异常时的处理方式可以灵活配置。
* 交换的途径是数据库直接访问、离线文件、还是WebService可以灵活配置。
* 交换的时机是定时任务、人工调度还是事件触发可以灵活配置。
  1. 关键技术
     1. JDBC技术

JDBC（Java Data Base Connectivity, java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC为工具/数据库开发人员提供了一个标准的API，据此可以构建更高级的工具和接口，使数据库开发人员能够用纯 Java API 编写数据库应用程序。Java数据库连接体系结构是用于Java应用程序连接数据库的标准方法。JDBC对Java程序员而言是API，对实现与数据库连接的服务提供商而言是接口模型。作为API，JDBC为程序开发提供标准的接口，并为数据库厂商及第三方中间件厂商实现与数据库的连接提供了标准方法。JDBC使用已有的SQL标准并支持与其它数据库连接标准，如ODBC之间的桥接。JDBC实现了所有这些面向标准的目标并且具有简单、严格类型定义且高性能实现的接口。

* + 1. WebService技术

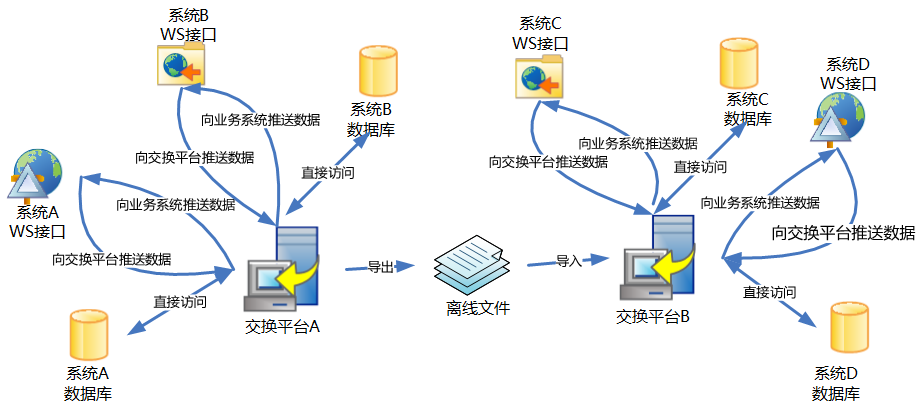
Web Service技术， 能使得运行在不同机器上的不同应用无须借助附加的、专门的第三方软件或硬件， 就可相互交换数据或集成。依据Web Service规范实施的应用之间， 无论它们所使用的语言、 平台或内部协议是什么， 都可以相互交换数据。Web Service是自描述、 自包含的可用网络模块， 可以执行具体的业务功能。Web Service也很容易部署， 因为它们基于一些常规的产业标准以及已有的一些技术，诸如[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)下的子集[XML](http://baike.baidu.com/view/63.htm)、HTTP。Web Service减少了应用接口的花费。Web Service为整个企业甚至多个组织之间的业务流程的集成提供了一个通用机制。

* + 1. XML技术

可扩展的标记语言XML是WebService平台中表示数据的基本格式。除了易于建立和易于分析外，XML主要的优点在于它既与平台无关，又与厂商无关。XML可以对文档和数据进行结构化处理，从而能够在部门、客户和供应商之间进行交换，实现动态内容生成，企业集成和应用开发。

* 1. 业务模型

数据交换平台实现的交换业务如下图所示：



* + 1. 数据交换的业务对象

在数据交换平台的业务中涉及到以下几类对象：

* 源数据：数据交换过程中提供数据的一方，一般是一个业务系统或者业务系统数据库。
* 目标数据：数据交换过程中接收数据的一方，本平台中为数据库、指定格式文件或WebService接口。
* 配置数据：用来定义平台上执行的数据交换任务。包括源数据、目标数据、数据交换规则、交换任务等数据。
* 中心数据库：在数据集成中的概念，在数据交换中的角色等同于一般的应用数据库，只是没有具体的业务与之对应。这个数据库可以作为企业的数据中心供统计分析适用；也可以作为数据交换过程中的临时数据库或者中转数据库使用。
  + 1. 数据交换的业务分类

数据交换按照交换的时间点可以分为定时交换和实时交换，按照源数据和目标数据是否可以直接通讯分为在线交换和离线交换。

|  |  |  |
| --- | --- | --- |
| **交换类别** | **在线交换** | **离线交换** |
| **定时** | 通过定时任务交换数据 | 需要人为干预 |
| **实时** | 通过接口调用实时交换 | —— |

* + 1. 数据交换的实现过程

数据交换过程就是从源数据中读取数据按照交换平台配置的对应关系写入到目标数据中，在交换的过程的各个时间点可以通过事件的形式触发交换平台定义的操作。

源数据的形式可以是：对一个业务数据库的查询操作、一个离线文件、接口中的数据内容参数。目标数据只能是一个数据库。系统提供的不同的交换类别的交换过程参见下表。

|  |  |  |
| --- | --- | --- |
| **交换过程** | | **过程描述** |
| **直接连接交换** | | 定时在线交换，系统定义一个查询语句作为数据源，根据映射关系保存到目标数据库中。 |
| **离线 交换** | **数据导出** | 定时或者人工触发，以查询语句作为数据源，根据规定的格式写成离线文件。 |
| **数据导入** | 人工触发或者定时监控目录，以离线文件作为数据源，根据规定的映射关系写入目标数据库中。 |
| **实时交换** | | 通过接口调用实时交换，通过调用交换平台中的WebService接口提供数据源，并根据规定的映射关系写入目标数据库中。 |

1. 主要功能

数据交换平台是一个独立运行的B/S结构应用系统，其主要功能由配置中心、控制中心和监控中心三部分构成。

* 1. 配置中心

配置中心提供一系列与数据交换业务相关的配置功能，可以对数据连接、任务、数据、关系等进行定义。

* + 1. 数据库连接配置

在交换平台中定义业务数据库，供数据直接交换的源数据或者目标数据使用。定义数据库连接需要配置以下内容：

* 数据库名称，全局唯一。
* Jdbc连接url。
* 数据库用户名。
* 数据库用户密码，用加密的方式存储。

数据交换平台支持的数据库包括：Oracle、SqlServer、DB2这三种。

* + 1. WebService服务接口配置

数据交换平台中的WebService接口规范参见附件1。平台中的通过接口数据交换都是采用推送方式来实现的，都是数据提供方发起请求的。平台提供的WebService服务供业务系统调用向平台提供交换数据，同样业务系统需要接受数据就要实现WebService供平台调用。

WebService接口定义需要配置以下内容：

* 服务名称，全局唯一。
* WebService服务的url。
* 用于验证的用户名。
* 对应验证用户的密码，用加密的方式存储。
  + 1. 数据对应关系配置

定义源数据和目标数据之间数据对应关系，需要定义的内容：

* 源数据库中的一条查询语句。
* 目标数据库中的一个表。
* 源数据查询语句的字段和目标数据库中的表字段对应关系。
* 交换触发器，数据交换过程中需要的处理。
  + 1. 数据导出格式配置

数据格式可以用于离线文件导出，也可以作为调用业务系统WebService接口的数据。数据格式需要定义的内容包括：

* 源数据库中的一条查询语句。
* 查询语句中各个字段的保存方式和格式。
* 数据导出操作触发器配置。
* 接收方处理标识，这个标识供接收方识别数据内容以判断如何处理。

具体配置规范参见附件2。

* + 1. 数据处理方式配置

平台通过接收方处理标识来识别处理方式。平台提供的处理方式有两种，写入数据库和调用业务系统WebService接口发送数据。

写入数据定义内容包括：

* 业务数据库中对应的表。
* 字段对应关系。
* 写入相关触发器。

调用WebService接口：

* 服务器名，通过名称与3.1.2 WebService服务接口配置中定义的接口对应。
  + 1. 触发器配置

在数据对应关系配置、数据导出配置、数据处理方式中数据导入操作配置中都涉及到对应的触发器配置。这里说的触发器是借用了数据库中相对应的名词；**他的主要属性有**：

* 执行数据库对象；是在源数据库还是在目标数据库中执行。针对导出操作只有源数据库，对导入操作只有目标数据库。
* 执行时机；是在整个同步操作之前或者之后，还是在每条记录的同步前或者之后，还是发生导出错误之后。
* 执行顺序；同一个时机点可能有多个触发器，他们的执行顺序。
* 执行内容；可以是一条sql语句，也可以是一个存储过程调用。需要引用同步记录中的变量是请用命名参数。

**触发器主要作用一般有：**

* 设置同步是否成功标记、同步时间等等。
* 记录同步日志，包括错误日志。
* 对同步数据进行合法性检测，通过抛出异常来阻止不合法的数据交换。
* 对同步后的数据进行相关业务处理，根据具体的项目的需求而定。
  1. 控制中心

控制中心提供相应的功能用于控制数据交换业务的运行，包括定时调度控制、人工控制和WebService服务控制。

* + 1. 定时调度控制
       1. 定时数据库直接交换调度

一个直接交换任务包括一个或者多个数据对应关系，并指定一个执行时间计划。

* + - 1. WebService接口交换调度

一个基于WebService接口交换任务包括一个或多个接收方处理标识为调用WebService接口的交换数据，并指定一个执行时间计划。

* + - 1. 离线数据导出调度

一个数据导出任务包括一个或多个交换数据；忽略接收方处理标识的类别，这个接收方处理标识是给数据导入时失败的。并指定一个执行时间计划。

无论有多少个交换数据，数据导出的都是一个zip文件。

* + 1. 人工调度控制
       1. 人工执行调度

在交换平台中可以手动执行定时调度中定义的三类任务。

* + - 1. 数据导入

数据导入的文件为数据交换平台导出的文件，或者符合相关格式的文件。数据导入主要用于网络物理隔离的场景中。

上传离线文件到数据交换平台服务器上，交换平台解压并读取文件中的数据，根据离线文件中的接收方处理标识将文件中的数据写入到目标数据库或者调用业务系统的WebService接口将数据提交给业务系统。

这里有一个关键点接收方处理标识，在数据交换平台中必须有对应的数据处理方式。

* + 1. WebService接口控制

业务系统可以调用交换平台的WebService接口服务，向交换平台提供数据，交换平台根据数据中的接收方处理标识将文件中的数据写入到目标数据库或者调用业务系统的WebService接口将数据提交给业务系统。和数据导入一样数据交换平台中必须有对应的数据处理方式。

* 1. 监控中心

系统中所有的交换操作都会记录日志，日志内容包括：

* 执行的开始时间、结束时间。
* 执行方式：自动、手动、操作人员（自动为系统）。
* 执行结果：成功多少条、失败多少条。
* 失败数据记录，如果失败了会记录失败的数据，供跟踪和补录。

监控中心基于交换日志提供实时监控、日志查看、情况统计等功能。

* + 1. 实时监控

实时查看交互平台中的交换任务和交互情况，包括以下内容：

* 查询系统中所有有效的交换任务。
* 查看系统中正在执行中的交换任务。
* 查看当天交换日志，包括各个任务交换数据总量和失败的条数。
  + 1. 日志查看

平台对所有的交换任务包括定时调度、人工调度和WebService触发的都进行详细的操作日志记录。并且平台提供多种方式查看日志：

* 按照任务查看；查看某个任务历史交换情况。
* 按照时间查看；查看某个时间段，比如一天内的所有交换情况。
  + 1. 交换情况统计

对交换结果进行统计分析，比如：

* 按时间段统计数据交换总数、成功数、失败数。
* 按照交换类别统计分析。
* 按照数据源或目标进行统计分析。

1. 应用场景

数据交换平台支持以下应用场景下的数据交换业务。

* 1. 基于数据库的交换

基于数据库的交换是在应用系统提供数据的读写权限的基础上执行的。

* + 1. 直接相连的定时交换

在数据交换平台中定义源数据和目标数据之间的对用关系，并创建定时任务。数据交换平台定时的执行数据同步。

一个交换任务可以包括多个数据对应关系，这些数据对应关系可以有不同的数据源和目标，在交换任务中可以知道数据对应关系的执行顺序。

* + 1. 物理隔离的离线交换

离线交换需要分两步，数据导出和数据导入。在物理隔离的系统中，需要在隔离的网络中分别安装数据交换平台。在源数据所在的交换平台上定义数据导出对应关系，在目标数据所在的交换平台上定义数据导入对应关系。这两个对应关系是通过XML格式文件和接收方处理标识来耦合的。离线数据文件需要人工的移转。

* 1. 基于WebService接口的交换

在应用系统不允许直接访问数据库的情况下可以通过实现WebService接口的方式来实现数据交换。

* + 1. 应用作为数据提供方实时交换

应用程序实现WebService接口规范的客户端，通过调用交换平台的WebService接口向交换平台提供数据，数据交换平台根据dataoptid来查找对应的映射关系，并将数据保存到对应的系统或者数据库中。

* + 1. 应用作为数据接收方实时交换

应用程序实现WebService接口规范的服务端，供交换平台调用。应用程序将交换平台提供的数据写入到应用系统数据库中。

* 1. 组合交换
     1. 基于接口的实时分发数据

提供数据的应用程序通过调用平台的WebService接口向平台提供数据，数据交换平台根据对应的映射关系可以同时调用多个目标应用程序的WebService服务向多个目标应用程序实时分发数据。

* + 1. 多应用系统集中离线交换

数据导出时根据数据对应关系进行导出的，一次导出可以执行多个对应关系，这些对应关系可以来自不同的应用系统。同样数据导入可以将数据导入到多个目标应用系统中。这样在隔离的两个网段中的多个应用系统数据可以通过一次导出和一次导入集中交换。

* + 1. 通过中转数据库集中离线交换

在应用系统数据库不允许或者不能直接访问时，如果需要进行离线交换可以通过中转数据库对数据进行中转。源应用系统通过调用交换平台的WebService接口向平台提供数据，平台将数据写入到中转数据库中。然后定时/手动进行数据导出和数据导入，实现离线交换。

离线交换都是定时/手动的无法做到实时交换。

1. 运行环境要求

系统运行环境为最基本要求，网站后台管理界面在1024×768分辨率下为最佳效果。请参阅以下配置要求：

|  |  |
| --- | --- |
| 项目 | 详细 |
| 操作系统 | Windows 2003/2008 Server 或者 linux |
| Web服务器 | Tomcat6.0 以上 + JDK 1.6 以上 |
| 平台配置数据库 | Oracle Database 10g/11g |
| 业务系统数据库 | 支持oracle 10g以上版本、SqlServer2005以上版本，DB2 8.2以上版本 |
| 硬件要求 | Intel至强 CPU, 8GM内存或更高，硬盘空间视用户需要存储的数据而定 |
| 带宽要求 | 10M共享或更高 |

1. 服务与维护
   1. 培训服务

确保用户能够用好系统。培训重点是系统的操作使用。培训具体内容如下：

* 操作系统和应用软件介绍。
* 系统总体介绍，包括：系统特点、界面介绍、功能介绍。
* 系统系统的使用与操作：各个模块的具体使用方法。

培训材料和培训方式包括：

|  |  |  |
| --- | --- | --- |
|  | 培训课程 | 培训方式 |
| 1 | 《系统整体演示》 | 授课 |
| 2 | 《系统设计介绍》 | 授课 |
| 3 | 《系统安装、配置、维护、管理》 | 授课、实际操作 |
| 4 | 《系统使用操作》 | 授课、实际操作 |

* 1. 维护服务
     1. 服务形式

结合投标方售后服务规程以及本项目的相关要求，本项目提供如下服务形式：

* 电话热线服务：通过公司设立的（固定电话+移动电话）的服务热线，提供7\*24小时热线响应服务，由一名工程师负责组织和协调用户的系统维护工作，这名工程师将配备手机，保证客户在7×24小时内都能得到服务响应，对于用户方软件系统发生的一切问题，这名工程师都将负责安排帮助解决，
* 传真技术支持：在工作时间，提供5×8小时传真技术支持服务。
* 互联网技术支持：通过公司互联网网站的售后服务支持功能，用户工作人员可以远程登录，了解相关系统的使用方法和常见问题，或者通过电子邮件通知我们遇到的问题，我们的技术支持人员每隔1小时会浏览一次电子邮箱，并在收到邮件后一个工作日内予以答复。
  + 1. 服务内容

1. 缺陷管理：针对本次招标的系统中存在的bug、缺陷，不论在保期内、外，投标人均应持续提供修正与消缺服务。
2. 系统升级：提供应用平台的软件补丁版本的升级服务。
3. 需求变更：对于招标方自身业务规则的变化导致的非模块级功能需求变更、性能要求提升导致的部署结构变化，可经双方协商提供限定次数的服务支持。
4. 运行支持：对系统运行过程中用户及业务部门的问题提供解答和问题解决跟踪，对于关键业务点的上线推广与运行提供现场保障。服务期期间，提供7×24（每周7天，每天24小时，）
5. 服务期内免费为用户提供产品的软件维护、版本升级、补丁程序及技术支持等服务，提供客户化软件升级服务。
6. 在最终用户使用主流品牌的硬件、软件产品出现兼容性问题时，积极配合，与有关硬件、软件厂商和最终用户接洽，及时定位问题原因、寻求解决方案。
7. 文档服务：整个服务过程均需有完善的文档记录，便于跟踪、分析问题；对各项服务提供详细的书面报告，包括故障处理报告、维护总结报告、服务年度报告等。每次现场服务后均须向甲方提交《维护技术服务工作报告》，对未完全解决的问题，服务工程师须与甲方工程师联系，跟踪问题，分析协商处理方案，直至问题解决。
   * 1. 免费服务期

自项目验收完成之日起**壹年**免费的系统软件维保服务，包括软件版本免费升级服务。

1. WebService接口配置规范

WebService服务提供了实时交换的方案。业务可以通过调用交换平台的WebService接口将数据传递给交换平台。接口中的数据是以XML格式上传的，格式和数据导出中的表数据格式一致。同样根据dataoptid来找对应的映射关系，保存到对应的表中。WebService作为数据实时交换的技术方案，它也是双向的，应用程序可以通过WebService向交换工具提供数据，同时也可以提供相同的WebService来接收数据。

* 1. 用WebService提供数据

应用程序实现WebService接口规范的客户端，通过调用交换平台的WebService接口向交换平台提供数据，数据交换平台根据dataoptid来查找对应的映射关系，并将数据保存到对应的系统或者数据库中。

* 1. 用WebService接收数据

应用程序实现WebService接口规范的服务端，供交换平台调用。应用程序将交换平台提供的数据写入到应用系统数据库中。

* 1. WebService接口规范

WebService对应的WSDL文件为：

 。

其接口的java语言描述参见下面的代码。系统中提供了两种参数形式的接口，另外为了操作方便还提供了单独更改大字段内容的接口。

public interface UploadData {

/\*\*

\* 上传数据接口，写入数据库之前需要进行用户身份验证

\* @param userName 用户名

\* @param userPin 密码，加密形式

\* @param tableDate 数据格式参见离线文件数据格式，为了控制大小可以将大字段单独上传，也可以一起上传

\* <!ELEMENT table (row\*)>

<!ATTLIST table name CDATA #REQUIRED>

<!ATTLIST table dataoptid CDATA #REQUIRED>

<!-- 表可以是一个单独文件存储，也可以在这个文件中 -->

<!ATTLIST table sourceosid CDATA #IMPLIED>

<!ATTLIST table sourcedatabase CDATA #IMPLIED>

<!ATTLIST table exportdesc CDATA #IMPLIED>

<!ELEMENT row (item+)>

<!ELEMENT item (#PCDATA)>

<!ATTLIST item name CDATA #REQUIRED>

<!ATTLIST item type (text|date|datetime|number|blob|clob) "text">

<!-- type 为date是 这个format为日期的格式，默认为"YYYY-DD-MM"； type为blob时这个格式为 plain 或则base64，clob默认为 plain， blob默认为base64 -->

<!ATTLIST item format CDATA #IMPLIED>

\* @return

\*/

@WebMethod

String uploadTableAsXml(

@WebParam(name="userName") String userName,

@WebParam(name="userPin") String userPin,

@WebParam(name="xmlData") String tableData);

/\*\*

\* 上传数据接口，写入数据库之前需要进行用户身份验证

\* @param userName 用户名

\* @param userPin 密码，加密形式

\* @param tableDate 数据格式参见离线文件数据格式，为了控制大小可以将大字段单独上传，也可以一起上传

\* @return

\*/

@WebMethod

String uploadTableAsDataHandler(

@WebParam(name="userName") String userName,

@WebParam(name="userPin") String userPin,

@WebParam(name="xmlData") DataHandler tableData);

/\*\*

\* 单独上传大字段接口

\* @param userName 用户名

\* @param userPin 密码，加密形式

\* @param mapinfoid 处理标记ID

\* @param tableName 表名

\* @param columnName lob字段名

\* @param keyDesc 主键描述可以是复合主键，形式为 key=value and key2=value2 ...

\* @param isBase64 是否为 base64编码，一般Clob不需要，Blob需要编码

\* @param lobData 大字段字符串

\* @return

\*/

@WebMethod

String uploadLobAsString(

@WebParam(name="userName") String userName,

@WebParam(name="userPin") String userPin,

@WebParam(name="database") String database,

@WebParam(name="tableName") String tableName,

@WebParam(name="columnName") String columnName,

@WebParam(name="keyDesc") String keyDesc,

@WebParam(name="isBase64") boolean isBase64,

@WebParam(name="lobData") String lobData);

/\*\*

\* 单独上传大字段接口

\* @param userName 用户名

\* @param userPin 密码，加密形式

\* @param mapinfoid 处理标记ID

\* @param tableName 表名

\* @param columnName lob字段名

\* @param keyDesc 主键描述可以是复合主键，形式为 key=value and key2=value2 ...

\* @param lobData 大字段数据

\* @return

\*/

@WebMethod

String uploadLobAsDataHander(

@WebParam(name="userName") String userName,

@WebParam(name="userPin") String userPin,

@WebParam(name="database") String database,

@WebParam(name="tableName") String tableName,

@WebParam(name="columnName") String columnName,

@WebParam(name="keyDesc") String keyDesc,

@WebParam(name="lobData") DataHandler lobData);

}

import javax.activation.DataHandler;

import javax.jws.WebMethod;

import javax.jws.WebParam;

import javax.jws.WebService;

@WebService(targetNamespace = "http://dde.centit.com/ws/")

public interface UploadData {

/\*\*

\* 上传数据接口，写入数据库之前需要进行用户身份验证

\* @param userName 用户名

\* @param userPin 密码，加密形式

\* @param database 目标数据库或者业务

\* @param tableDate 数据格式参见离线文件数据格式，为了控制大小可以将大字段单独上传，也可以一起上传

\* @return

\*/

@WebMethod

String uploadTableAsXml(

@WebParam(name="userName") String userName,

@WebParam(name="userPin") String userPin,

@WebParam(name="database") String database,

@WebParam(name="xmlData") String tableDate);

/\*\*

\* 上传数据接口，写入数据库之前需要进行用户身份验证

\* @param userName 用户名

\* @param userPin 密码，加密形式

\* @param database 目标数据库或者业务

\* @param tableDate 数据格式参见离线文件数据格式，为了控制大小可以将大字段单独上传，也可以一起上传

\* @return

\*/

@WebMethod

String uploadTableAsDataHandler(

@WebParam(name="userName") String userName,

@WebParam(name="userPin") String userPin,

@WebParam(name="database") String database,

@WebParam(name="xmlData") DataHandler tableDate);

/\*\*

\* 单独上次大字段接口

\* @param userName 用户名

\* @param userPin 密码，加密形式

\* @param database 目标数据库或者业务

\* @param tableName 表名

\* @param columnName lob字段名

\* @param keyDesc 主键描述可以是复合主键，形式为 key=value and key2=value2 ...

\* @param isBase64 是否为 base64编码，一般Clob不需要，Blob需要编码

\* @param lobData 大字段字符串

\* @return

\*/

@WebMethod

String uploadLobAsString(

@WebParam(name="userName") String userName,

@WebParam(name="userPin") String userPin,

@WebParam(name="database") String database,

@WebParam(name="tableName") String tableName,

@WebParam(name="columnName") String columnName,

@WebParam(name="keyDesc") String keyDesc,

@WebParam(name="isBase64") boolean isBase64,

@WebParam(name="lobData") String lobData);

/\*\*

\* 单独上次大字段接口

\* @param userName 用户名

\* @param userPin 密码，加密形式

\* @param database 目标数据库或者业务

\* @param tableName 表名

\* @param columnName lob字段名

\* @param keyDesc 主键描述可以是复合主键，形式为 key=value and key2=value2 ...

\* @param lobData 大字段数据

\* @return

\*/

@WebMethod

String uploadLobAsDataHander(

@WebParam(name="userName") String userName,

@WebParam(name="userPin") String userPin,

@WebParam(name="database") String database,

@WebParam(name="tableName") String tableName,

@WebParam(name="columnName") String columnName,

@WebParam(name="keyDesc") String keyDesc,

@WebParam(name="lobData") DataHandler lobData);

}

1. 数据格式配置规范
   1. 数据内容
      1. 元数据配置

交换元数据配置的内容包括：

* 事务属性，整体作为一个事务，还是每一条数据作为一个事务。默认每一条数据作为一个事务。
* 从属表属性：字表单独一个事务还是和主表中的记录一起作为一个事务。默认从属于主表作为一个事务。（这句话不太懂）
* 主键冲突是覆盖（更新）还是报错。默认更新。
* 执行数据录入前所做的准备工作，一组数据库更新语句、或者建表操作。这部分可以没有。
* 回执数据内容。这部分默认记录成功条数，失败条数和失败数据内容。

注：这部分内容全部有默认值，也就是说这部分内容可以缺省。

* + 1. 数据内容

数据内容格式配置的内容包括：

* 表；内容包括：表名、表的元数据（字段名、字段类型、lob字段值存储方式）、和其他表的关系。
* 表中的记录；每一条记录中每字段一个元素，名称对应字段名内容对应字段值。
  + 1. LOB字段的处理

数据交换平台对lob的处理有两种方式：

* CDATA区段；clob指端直接放入其中，blob字段通过base64编码后放入cdata区段中。
* 文件形式；每个lob字段一个文件，在记录中记录文件名，一般这个文件名可以和记录的主键相同。使用这种方式必需把xml文件和这些lob文件压缩都一个zip文件中。
  1. 数据格式
     1. 文件格式

离线数据文件是一个XML文件，可以每个表一个文件也可以多个表一个文件。Lob字段可以单独存放，一个字段一个文件，也可以经过base64编码后和其他字段一起放在项目了文件中。所以一次导出可能有多个离线文件，为了传输方便提高效率对离线文件进行打包。压缩包中有一个主文件一般为 导出任务名.xml，如果表的数据单独存放还包括多个表数据文件，以表名.xml命名；压缩文件中可以包括lob字段的文件，这些lob文件可以有自己的子目录，一般子目录的层次为表名\字段名，当然用户也可以定义任意的层次结构，和xml文件中字段值的描述一致皆可。

压缩算法采用java.util.zip中的通用zip算法。

* + 1. 离线文件XML格式(DTD)

<!DOCTYPE exchange [  
 <!ELEMENT exchange (metadata?,data)>  
 <!ATTLIST exchange id CDATA #REQUIRED>  
   
 <!ELEMENT metadata (pretreatment?,post\_processing?,receipt?)>  
 <!ELEMENT pretreatment (sql\*,call\*)>  
 <!ELEMENT post\_processing (sql\*,call\*)>  
 <!ELEMENT receipt ANY>  
   
 <!ELEMENT data (table+)>  
 <!ELEMENT table (row\*)>  
 <!ATTLIST table name CDATA #REQUIRED>  
 <!-- dataoptid 为交换平台中定义的数据写入对应关系，这个很重要 ,没有这个属性系统将无法导入 -->  
 <!ATTLIST table dataoptid CDATA #REQUIRED>  
 <!--表可以是一个单独文件存储，也可以在这个文件中 -->  
 <!ATTLIST table store (infile|embedded ) "embedded">  
 <!ATTLIST table sql CDATA #IMPLIED>  
 <!ATTLIST table transaction (each\_row|whole) "each\_row">  
 <!ATTLIST table conflict (replace|abandon) "replace">  
   
 <!ELEMENT row (item+,sub\_table\*)>  
 <!ELEMENT item (#PCDATA)>  
 <!ATTLIST item name CDATA #REQUIRED>  
 <!ATTLIST item type (text|date|timestamp|number|blob|clob) "text">  
 <!-- 如果type为blob、clob这个字段才有效 -->  
 <!ATTLIST item store (infile|embedded ) "embedded">  
 <!-- type 为date是 这个format为日期的格式，默认为"YYYY-DD-MM"； type为blob时这个格式为 plain 或则base64，clob默认为 plain， blob默认为base64 -->  
 <!ATTLIST item format CDATA>   
]>

* + 1. 表数据XML格式(DTD)

表数据XML格式同时也是WebService接口中数据的XML格式。

<!DOCTYPE table[  
 <!ELEMENT table (row\*)>  
 <!ATTLIST table name CDATA #REQUIRED>  
 **<!-- dataoptid 为交换平台中定义的数据写入对应关系，这个很重要 ,没有这个属性系统将无法导入 -->**  
 <!ATTLIST table dataoptid CDATA #REQUIRED>  
 <!--表可以是一个单独文件存储，也可以在这个文件中 -->  
 <!ATTLIST table store (infile|embedded ) "embedded">  
 <!ATTLIST table sql CDATA #IMPLIED>  
 <!ATTLIST table transaction (each\_row|whole) "each\_row">  
 <!ATTLIST table conflict (replace|abandon) "replace">  
   
 <!ELEMENT row (item+,sub\_table\*)>  
 <!ELEMENT item (#PCDATA)>  
 <!ATTLIST item name CDATA #REQUIRED>  
 <!ATTLIST item type (text|date|timestamp|number|blob|clob) "text">  
 <!-- 如果type为blob、clob这个字段才有效 -->  
 <!ATTLIST item store (infile|embedded ) "embedded">  
 <!-- type 为date是 这个format为日期的格式，默认为"YYYY-DD-MM"； type为blob时这个格式为 plain 或则base64，clob默认为 plain， blob默认为base64 -->  
 <!ATTLIST item format CDATA>  
]>