

|  |
| --- |
| ALS7A信贷核算增强版1 |
| 程序规格说明书-批量 |

上海安硕信息技术股份有限公司授权:

2013年06月

上海安硕信息技术股份有限公司

THESE LISTED PEOPLE:

项目组或人员名单

使用本文件

ARE AUTHORIZED BY AMARSOFT INFORMATION & TECHNOLOGY CO., LTD. TO USE THIS DOCUMENT

上海安硕信息技术股份有限公司 保密声明

本文件及其附件中的所有的信息均应受到保密，受到法律的保护。本文件的信息仅限于指定使用者阅读。如果您并非指定的使用者，或者您只是本文件传递给指定使用者的转交人或机构，您不得使用、分发、复制本文件中的任何内容。如果本文件错误的转交给了您，请立即将其销毁并告知上海安硕信息技术股份有限公司。联系电话：86-21-65106600，联系人：翟先生。谢谢您的合作。

AMARSOFT INFORMATION & TECHNOLOGY CO., LTD. Proprietary

The information contained in this document and its attachments (if any) is confidential and subject to legal privilege. The information is intended only for use of the individuals(s) to whom it is addressed. If you are not an intended recipient, or the agent or employee responsible to deliver it to an intended recipient, you are hereby notified that any use, distribution or copying of this communication is strictly prohibited. If you have received this document in error, please destroy it and immediately notify Amarsoft Information & Information & Technology Co., Ltd. by calling 86-21-65106600, Mr. Zhai. Thank you.

上海安硕信息技术股份有限公司 特别申明

未经本公司书面同意，不得复制、翻译或摘录本文档。本公司明确依法保留有关版权的一切权利，保留更改权和解释权。

文档信息

|  |  |
| --- | --- |
| 文档名称： | ALS7A程序规格说明书-批量 |
| 初稿作者： | [xjzhao@amarsoft.com](mailto:xjzhao@amarsoft.com) |
| 初稿日期： | 2013/06/03 |
| 内容概述： | 核算批量新增内容 |

修订历史

| 版本 | 修订日期 | 修订人 | 复核日期 | 复核人 | 修改内容简述 |
| --- | --- | --- | --- | --- | --- |
| V1.0 | 2013/06/03 | 赵晓建 |  |  | 建立文档 |
| V1.1 | 2013/12/6 | 赵晓建 |  |  | 修改文档名称，调整目录 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

发布历史

| 发布日期 | 提交人 | 质检人 | 发布说明 |
| --- | --- | --- | --- |
| 2013/06/03 | 赵晓建 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

[1 概述 1](#_Toc375211237)

[1.1 编写目的 1](#_Toc375211238)

[1.2 背景 1](#_Toc375211239)

[1.3 定义 1](#_Toc375211240)

[1.4 阅读对象 2](#_Toc375211241)

[1.5 参考资料 2](#_Toc375211242)

[2 总体描述 3](#_Toc375211243)

[2.1 开发目标 3](#_Toc375211244)

[2.2 设计约束 3](#_Toc375211245)

[2.2.1 技术标准约束 3](#_Toc375211246)

[2.2.2 软件运行平台约束 3](#_Toc375211247)

[2.2.3 硬件运行平台约束 4](#_Toc375211248)

[2.2.4 通信运行平台约束 4](#_Toc375211249)

[2.3 设计方法 4](#_Toc375211250)

[3 基本应用框架 5](#_Toc375211251)

[3.1 批量程序框架 5](#_Toc375211252)

[3.2 任务概述 6](#_Toc375211253)

[3.3 任务装配 7](#_Toc375211254)

[3.4 执行单元 9](#_Toc375211255)

[3.5 配置路由 11](#_Toc375211256)

[3.6 ARE配置要点 12](#_Toc375211257)

[3.7 环境参数使用 13](#_Toc375211258)

[3.8 标准任务单元 14](#_Toc375211259)

[3.8.1 Table（SQL）->File 14](#_Toc375211260)

[3.8.2 File->Table 16](#_Toc375211261)

[4 应用总体设计 18](#_Toc375211262)

[4.1 划分原则 18](#_Toc375211263)

[4.2 Target划分 19](#_Toc375211264)

[4.3 Target说明 19](#_Toc375211265)

[5 组件详细说明 21](#_Toc375211266)

[5.1 组件基础说明 21](#_Toc375211267)

[5.1.1 组件基础抽象类 21](#_Toc375211268)

[5.1.2 执行SQL工具类 22](#_Toc375211269)

[5.2 开始核算批量 22](#_Toc375211270)

[5.2.1 基本说明 22](#_Toc375211271)

[5.2.2 程序逻辑 23](#_Toc375211272)

[5.3 表外业务垫款 23](#_Toc375211273)

[5.3.1 基本说明 23](#_Toc375211274)

[5.3.2 程序逻辑 23](#_Toc375211275)

[5.4 生成批量扣款数据 24](#_Toc375211276)

[5.4.1 基本说明 24](#_Toc375211277)

[5.4.2 程序逻辑 24](#_Toc375211278)

[5.4.2.1 数据转换 24](#_Toc375211279)

[5.5 生成批量扣款文件 25](#_Toc375211280)

[5.5.1 基本说明 25](#_Toc375211281)

[5.5.2 程序逻辑 26](#_Toc375211282)

[5.6 批量扣款结果文件处理 26](#_Toc375211283)

[5.6.1 基本说明 26](#_Toc375211284)

[5.6.2 程序逻辑 27](#_Toc375211285)

[5.7 批量贷款还款 27](#_Toc375211286)

[5.7.1 基本说明 27](#_Toc375211287)

[5.7.2 程序逻辑 27](#_Toc375211288)

[5.8 费用摊销、递延 28](#_Toc375211289)

[5.8.1 基本说明 28](#_Toc375211290)

[5.8.2 程序逻辑 28](#_Toc375211291)

[5.9 贷款日终交易 28](#_Toc375211292)

[5.9.1 基本说明 28](#_Toc375211293)

[5.9.2 程序逻辑 29](#_Toc375211294)

[5.10 年终结转交易 29](#_Toc375211295)

[5.10.1 基本说明 29](#_Toc375211296)

[5.10.2 程序逻辑 29](#_Toc375211297)

[5.11 汇总账务明细生成总账 29](#_Toc375211298)

[5.11.1 基本说明 29](#_Toc375211299)

[5.11.2 程序逻辑 30](#_Toc375211300)

[5.12 换日 30](#_Toc375211301)

[5.12.1 基本说明 30](#_Toc375211302)

[5.12.2 程序逻辑 30](#_Toc375211303)

[5.13 贷款日初交易 31](#_Toc375211304)

[5.13.1 基本说明 31](#_Toc375211305)

[5.13.2 程序逻辑 31](#_Toc375211306)

[5.14 预约变更处理 31](#_Toc375211307)

[5.14.1 基本说明 31](#_Toc375211308)

[5.14.2 程序逻辑 31](#_Toc375211309)

[5.15 结束核算批量 32](#_Toc375211310)

[5.15.1 基本说明 32](#_Toc375211311)

[5.15.2 程序逻辑 32](#_Toc375211312)

# 概述

## 编写目的

本文档是Amarsoft信贷管理系统7.3A (ALS7)统一核算批量接口程序设计说明书，详细说明程序结构、数据流程、主要算法和约束条件，说明异常处理机制。本文档的目的是在需求文档的基础上、在项目组内部达成共识，指导程序开发人员书写稳定可靠，功能清晰的程序。

## 背景

ALS7.3A是Amarsoft信贷风险管理的最新成果，体现了国内最新的信贷风险管理水准。做为一个功能完整，结构复杂的大型金融管理产品，信贷风险管理系统需要与核心和其他系统交互数据，其中最重要的就是信贷系统联机核心进行批量扣款、总账上送等功能，这些处理统称为核算标准批量接口。

## 定义

为陈述方便，本文定义如下词语，词语定义在本文和本项目的后续文档中有效：

1. 信贷系统：Amarsoft银行信贷管理系统、风险管理系统、Amarsoft核算系统的通称；
2. 主机系统：别称会计系统、综合业务系统、核心系统、生产系统，指银行处理核心银行业务和记账的综合计算机系统，通常以字符终端的形式提供服务，是银行最主要的生产系统；
3. 总账系统。用于进行商业银行账务管理、生成财务报表等会计系统，一般的商业银行总账系统通常是主机系统的子功能。
4. 修改1，增加一个测试条款
5. 外围系统。即信贷系统以外的系统，上面提到的主机系统、总账系统就属于外围系统范畴。
6. 接口文件：从主机系统、信贷系统、国际业务系统等导出的，供其他系统接口导入的文件，以text的形式提供。

## 阅读对象

本文档的预期读者：

* 上海安硕信息技术股份有限公司信贷系统研发组
* $BANK相关部分
* 添加表格测试

|  |  |  |
| --- | --- | --- |
| 11 | 12 | 13 |
|  | 22 |  |
|  |  |  |

## 参考资料

* 《ALS7A功能规格说明书-贷款核算》
* 《ALS7A功能规格说明书-批量接口》
* AmarBank信贷管理系统相关资料

# 总体描述

## 开发目标

* 满足核算必须的批量处理功能，如批量扣款还款、贷款日终、贷款日初、日切等动作；
* 在设计上尽量考虑全面，尽量减少版本到客户方程序的改动，减少项目组代码开发的压力，多使用配置工具即可完成业务功能；

## 设计约束

### 技术标准约束

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **技术标准** | **版本** | **说明** |
| 1 | J2SE | 1.5.0 | 本系统开发使用Java 2 Standard Edition 平台 |
| 2 | ARE | 1.0 | 本系统开发基于安硕新运行环境 |
| 3 | JDBC | 2.0 Type 4 | 本产品中对数据库的连接使用JDBC2.0技术规范，并且推荐使用Type4的驱动。 |
| 4 | Language | GBK | 本程序的假定文件的标准语言使用GBK |

### 软件运行平台约束

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **软件平台** | **版本** | **说明** |
| 1 | JRE | 1.5.0+ | 运行环境需要JRE1.4.2以上版本 |
| 2 | Windows | 2000+ &SP4/2003 | 企业征信接口系统可以在Microsoft Windows的主流版本上运行，为稳定起见，推荐使用服务器版本 |
| 3 | zNix | Linux2.4+/Soloris8+/HP-UX10+/AIX5.1+ | 企业征信接口可以运行在Linux，Sun，HP和IBM的Unix系统上面 |
| 4 | Oracle | 10.2+ | 主流数据库版本 |
| 5 | Informix | 7.13+ |  |
| 6 | Sybase | 12.0+ |  |
| 7 | DB2 | 8.0+ |  |
| 8 | MS SQL Server | 2000+ |  |

### 硬件运行平台约束

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **硬件平台** | **版本** | **说明** |
| 1 | Intel | 32/64 |  |
| 2 | Sparc | 32/64 |  |
| 3 | Power PC | 32/64 |  |
| 4 | PA-RISK | 32/64 |  |

### 通信运行平台约束

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **通信平台** | **版本** | **说明** |
| 1 | 运行网络 | E100+ | 目标程序运行系统和数据库服务器之间的网络带宽应该达到独占1000M以上，推荐1000M带宽 |

## 设计方法

本系统是基于Java的系统，采用面向对象的设计方法，设计中注意使用多种模式设计的方法，减少系统的耦合性，提高系统的重用和灵活性。

ALS7在不同的项目实施会有很多变化和调整，核算标准批量程序设计为灵活可调整地架构。

# 基本应用框架

## 批量程序框架

本节说明应用程序总体框架，描述应用程序的外观视图和内在联系。批量程序是基于ARE-TASK任务调度器的应用架构，ARE任务调度器是纯Java的一个轻型应用程序运行容器，符合任务调度基本组件接口的程序模块可以放进任务调度器运行。

批量程序的主程序控制流程如下图所示。



批量程序最终发布后有下面的目录结构：

* APP\_HOME/：应用程序的起始根目录，一般放置程序的入口.sh和.bat文件
* APP\_HOME/etc：放置配置文件，通常包括了：are.xml, dbconfig.xml,metadata.xml, task.xml, jdklog.properties, log4jlog.properties,jbo相关xml文件
* APP\_HOME/src：源程序（可选）,放置项目本身的扩展程序。产品本身源代码不随着系统发布。
* APP\_HOME/classes：类目录，通常是放置项目扩展的类。产品本身都已经打包为jar
* APP\_HOME/lib：放置应用程序以及are、jdbc等所有的.jar文件
* APP\_HOME/log：放置日志文件
* APP\_HOME/docs：文档目录
* APP\_HOME/data：导入数据文件目录
* APP\_HOME/export：输出文件目录
* APP\_HOME/backup：备份数据的目录

## 任务概述

任务调度器实行二级容器机制，一级容器是任务（Task），二级容器是是目标(Target)。一个任务由一到多个目标组成，每一个目标可以单独被调度，是任务调度的基本逻辑组件，目标间可以并行处理。目标内部是为了达成这个目标而放入的基本功能单元(Unit)。功能单元之间通过路由(Route)定义彼此之间的关系和执行顺序，功能单元和路由构成了目标的逻辑功能。

任务调度模型以ARE为基础，是ARE的一部分，运行环境中必然包含ARE基础运行环境。因此，任务组件可以透过任务调度器直接访问ARE的基础服务，等同与任务调度器提供了这些服务。这些服务包括：数据库连接、日志、元数据管理等多种和系统相关的资源，通过统一的AREService框架，应用组件透明的使用这些功能。

**J2SE&OpenLib JVM、JDOM、LOG4J….**

**ARE AREService(DBConnection、Log、MetaData。。。)**

**Task**

**Target**

**Unit**

**Unit**

**Unit**

**Unit**

**Unit**

**Target**

**Unit**

**Unit**

**Unit**

**Unit**

**Unit**

**Target**

**Unit**

**Unit**

**Unit**

**Unit**

**Unit**

**Target**

**Unit**

**Unit**

**Unit**

**Unit**

**Unit**

DataBase

Files

任务调度器模型是ARE基本应用平台，不需要开发和修改。应用程序开发的主要任务是根据业务需求开发各种基础功能，并把它们组装成Target和任务。路由是任务管理其的基础设施之一，用于组织Unit之间的逻辑，路由的表达形式可以用易于理解的描述形式在外部表达。

## 任务装配

任务管理器通过TaskBuillder构建任务，并启动运行。构建的过程称为装配，装配任务的原材料是路由和Unit，这些原材料通过任务描述文件的形式组织，任务描述文件是XML文件，包括了元素定义和逻辑定义两种信息。任务构建器通过解析XML文件装配真个任务，一旦任务装配完毕，就可以执行。

任务描述文件以固定的格式描述任务，一个任务描述文件的基本框架如下页所示。注意其中的元素的书写都是用的标准Java变量命名方法，即每个单词的首字母如果不是第一个大写，单词间不使用连接符号。

任务调度对象的基本元素定义有类似的头信息：name、describe、extendProperies，用于描述相应对象的基本特征。extendProperties下可以定义任意多的property，用于配置程序运行的参数。低层次的对象可以用到高层定义的property，因此可以把公用的部分在上层定义，在整个task都要用的在task定义，每个target公用的在target定义，每个unit要用的在executeUnit定义。

下面说明主要元素定义：

* task，任务是起点，可以定义的参数包括：parallelRun和traceOn，用于说明target之间是否并行运行，运行时是否输出跟踪信息。
* target，目标除了name、describe和extendProperties外，有executeUnits和routeTable构成，前者时执行单元的定义组，后者时执行单元之间的路由关系。
* executeUnit，执行单元时完成功能的最小单位，任务单元必须包含name、describe和executeClass，前两者用于在应用中表示和说明自己，同时name必须在一个target内唯一，因为路由是以unit的名字为标识的。executeClass指明了单元的执行类。可以在extendProperties中放置参数信息。
* routeTable有多条route组成，每条route有三个部分：源、状态和目标组成。源是出发的unit，状态是源unit执行的结果，目标unit是在此状态下要执行的下一个目标单元。所有的Route中必须有一条源和目标一致，状态是unexecuted，这条路由是target执行的起始点。

<?xml version="1.0" encoding="GB2312"?>

<task parallelRun="true" traceOn="true" >

<name>演示任务</name>

<describe>任务配置样板</describe>

<extendProperties>

<property name="property1-name" value="property1-name" />

</extendProperties>

<targets>

<target enabled="true">

<name>target1</name>

<describe>演示目标1</describe>

<extendProperties>

<property name="property1-name" value="property1-name" />

</extendProperties>

<executeUnits>

<executeUnit>

<name>unit0</name>

<describe>示例executeunit0</describe>

<executeClass>com.amarsoft.test.TestUnit</executeClass>

<extendProperties>

<property name="dataSource" value="sampledb" />

</extendProperties>

</executeUnit>

<executeUnit>

<name>unit1</name>

<describe>示例executeunit1</describe> <executeClass>com.amarsoft.test.TestUnit1</executeClass>

</executeUnit>

<executeUnit>

<name>unit2</name>

<describe>示例executeunit2</describe>

<executeClass>com.amarsoft.test.TestUnit2</executeClass>

</executeUnit>

</executeUnits>

<routeTable>

<route unit="unit0" executeStatus="unexecute" nextUnit="unit0"/>

<route unit="unit0" executeStatus="successful" extUnit="unit1"/>

<route unit="unit0" executeStatus="failed" nextUnit="unit2"/>

</routeTable>

</target>

</task>

## 执行单元

执行单元（ExecuteUnit）是任务调度的基本单位，每个单元独立的完成一项任务。所有的执行单元都有相同的接口，满足任务管理器的调度需要，类似于EJB组件有基本的管理接口。

执行单元在任务配置里面的外部展示如下：

<executeUnit>

<name>unit2</name>

<describe>示例executeunit2</describe>

<executeClass>com.amarsoft.test.TestUnit2</executeClass>

<extendProperties>

<property name="dataSource" value="sampledb" />

</extendProperties>

</executeUnit>

其中:

1）<name>指定这个单元的名称，在下面的路由表中要用到；

2）<describe>是一个简单的描述，说明单元

3）<executeClass>指出执行这个任务的具体的实现类

4）<extendProperties>包含多个配置参数的列表，每个配置参数是一个<property>项，包括name和value 两个属性。

执行单元的根类是com.amarsoft.are.task.ExecuteUnit。应用程序的执行单元组件必须是其子类。ExecuteUnit类是抽象类，定义了执行单元的基本信息和固定的一些方法。

执行单元的方法包括：

|  |  |
| --- | --- |
| **方法摘要** | |
| void | [**addRoute**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\ExecuteUnit.html#addRoute%28com.amarsoft.are.task.Route%29)([Route](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\Route.html) route)            为本unit增加一个路由，重复的路由不会被加入 |
| abstract  int | [**execute**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\ExecuteUnit.html#execute%28%29)()            本方法是关键执行体，Target通过调用本方法达到执行应用的目标。 |
| [Route](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\Route.html)[] | [**getRouteTable**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\ExecuteUnit.html#getRouteTable%28%29)()            返回本uint的路由表，正常情况下，是从这个unit可以出发的多条路由信息 |
| [Target](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\Target.html) | [**getTarget**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\ExecuteUnit.html#getTarget%28%29)()            返回Unit所属的Targer |
| [ExecuteUnit](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\ExecuteUnit.html) | [**nextUnit**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\ExecuteUnit.html#nextUnit%28%29)()            返回下一个要执行的单元，target根据此方法调度路由的流转 |
| void | [**removeRoute**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\ExecuteUnit.html#removeRoute%28com.amarsoft.are.task.Route%29)([Route](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\Route.html) route)            删除一个路由，如果不存在，没有任何动作 |
| protected  void | [**setTarget**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\ExecuteUnit.html#setTarget%28com.amarsoft.are.task.Target%29)([Target](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\Target.html) target)            设置unit所在的target，注意此方法是为了本类包的内部调用。 |

 如上所述，ExecuteUnit的方法大部分都是为了任务调度设置的，子类和其他程序不应该改变这些方法，也不应该在程序外部调用。通常来说，子类需要做的唯一一件事情就是继承并实现抽象方法：execute，并且这个事情是必须的。

ExecuteUnit从父类继承的方法有：

|  |  |
| --- | --- |
| **方法摘要** | |
| java.lang.String | [**getDescribe**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#getDescribe%28%29)()            属性描述的get方法 |
| java.lang.String | [**getName**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#getName%28%29)()            属性名字的get方法 |
| java.lang.String | [**getProperty**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#getProperty%28java.lang.String%29)(java.lang.String property)            获取程序配置扩展属性 |
| boolean | [**getProperty**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#getProperty%28java.lang.String,%20boolean%29)(java.lang.String property, boolean defalutValue)            返回一个boolean类型的属性值，如果不存在，返回default值。 |
| java.util.Date | [**getProperty**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#getProperty%28java.lang.String,%20java.util.Date%29)(java.lang.String property, java.util.Date defalutValue)            返回一个日期类型的属性值，如果不存在或者转换出错，返回default值。 |
| java.util.Date | [**getProperty**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#getProperty%28java.lang.String,%20java.util.Date,%20java.lang.String%29)(java.lang.String property, java.util.Date defalutValue, java.lang.String format)            返回一个日期类型的属性值，如果不存在或者转换出错，返回default值。 |
| double | [**getProperty**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#getProperty%28java.lang.String,%20double%29)(java.lang.String property, double defalutValue)            返回一个数字类型的属性值，如果不存在或者转换出错，返回default值。 |
| int | [**getProperty**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#getProperty%28java.lang.String,%20int%29)(java.lang.String property, int defalutValue)            返回一个整数类型的属性值，如果不存在或者转换出错，返回default值。 |
| java.lang.String | [**getProperty**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#getProperty%28java.lang.String,%20java.lang.String%29)(java.lang.String property, java.lang.String defalutValue)            返回一个字符串类型的属性值，如果不存在，返回default值。 |
| boolean | [**isTraceOn**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#isTraceOn%28%29)()            判断跟踪状态是否置位true，如果跟踪状态为true，会自动的在日志中记录跟踪信息 |
| void | [**setDescribe**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#setDescribe%28java.lang.String%29)(java.lang.String describe)            属性描述的set方法 |
| void | [**setName**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#setName%28java.lang.String%29)(java.lang.String name)            属性名字的set方法 |
| void | [**setProperty**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#setProperty%28java.lang.String,%20java.lang.String%29)(java.lang.String name, java.lang.String value)            设置一个应用程序扩展属性 |
| void | [**setTraceOn**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#setTraceOn%28boolean%29)(boolean traceOn)            设置跟踪状态，如果跟踪状态为true，会自动的在日志中记录跟踪信息 |
| protected  void | [**trace**](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html#trace%28com.amarsoft.are.task.TaskObject%29)([TaskObject](file:///E:\studio\release\are\docs\api\com\amarsoft\are\task\TaskObject.html) obj)            跟踪信息记录 |

 这些方法通常也不需要改变和调用，子类在实现execute方发的时候，经常使用的是getProperty方法获取参数。

根据上述描述，在task程序体系里面，实现一个任务单元的步骤是：

* 1. 配置任务单元，设置需要传递的参数。可以认为这些参数就像是命令行的参数一样，在程序中可以调用；
  2. 继承com.amarsoft.are.task.ExecuteUnit，实现execute方法，完成程序逻辑。Execute方法相当于一般程序的main方法，使单元的入口，当然程序逻辑可以有其他方法和类实现，但入口仅此一个。实现方法过程中可以通过getProperty获取外部命名参数。

## 配置路由

配置路由，体现了批量程序运行的逻辑关系，从宏观上规定了批量作业的程序结构。路由表有多个路由组成，每个路由规定了一个执行单元在特定执行结果下要执行的另一个单元。

一般形式如下：

<routeTable>

<route unit="unit0" executeStatus="unexecute" nextUnit="unit0"/>

<route unit="unit0" executeStatus="successful" extUnit="unit1"/>

<route unit="unit0" executeStatus="failed" nextUnit="unit2"/>

</routeTable>

每条路由有三个属性：

* unit-出发的单元
* executeStatus-单元执行的结果
* nextUnit-在这个结果下面应该执行的单元

单元必须是前面的单元定义中出现的单元，路由的几个隐性规则需要在配制的时候注意：

* 1. 有且必须有一个unit=nextUnit，executeStatus=”unexecuted”的路由表示，整个目标执行的起始点；
  2. 如果存在unit x，存在到x的路由，从x出发不存在任何一个终结点不是他自己的的路由，则x是终结单元，执行到x后路由表自然完毕，一个路由表中允许多个终结单元，表示在不同状态下的终结；
  3. 执行状态分为三种预定的状态：unexecute、successful、failed，除此之外必须是正整数。unit要负责返回各种执行状态供路由表调度；
  4. 配置时必须自己注意避免循环路由。

## ARE配置要点

一个典型的ARE批量程序配置包括：

* 运行环境配置（are.xml）
* 元数据配置（metadata.xml）
* 数据库连接配置（dbconfig.xml）
* 日志配置（simplelog.properties/audit.properties）

各个配置文件详细情况，参阅对应文件。下面是配置的时候得要点：

* are.xml要点：
  1. are.xml是所有are应用程序配置的总入口，可以按图索骥找到其他配置信息
  2. 应用属性自动传输到系统属性
  3. 服务根据需要配置，简单服务属性自包含，复杂服务应用其他配置文件
* simplelog.properties要点：
  1. 原则：简单的提示信息输出到屏幕，详细调试信息输出到文件
  2. 文件名字通配符含义：%g=唯一区分文件的序号，d%=创建文件的datetime，D%=创建文件的日期
  3. com.amarsoft.are.log.SimpleLog.level = ALL，特别定义ARE程序的日志输出
* meatadata.xml要点
  1. column的index必须从1开始连续
  2. 日期型的format必须是java的日期的format,特别注意M=月份，m=分钟
  3. 数值型的precision和scale，通常都是0，
  4. displaySize只有在使用定长文件的时候才起作用

## 环境参数使用

为保证基于ARE的程序就有良好的可配置行和适应性，ARE支持环境参数的使用。环境参数为程序提供了一定的变量代换机制，为程序提供更多的灵活性。

ARE参数是运行时刻ARE和环境属性的对外表现形式，在运行时刻，ARE保存一些运行属性，这些属性是全局可见的，通过ARE.getProperty()方法获取。利用这个特征，在应用程序中可以简单的一致的机制访问全局属性，可以做为系统模块间传递简单数据的一种有效方法。ARE参数的表现形式有两种：{ARE.xxxx}，或者{SYSTEM.xxxx}，表示是ARE或者系统的属性xxxx。

环境参数可以用于主要的配置文件中的属性值里面：

* are.xml
* task.xml
* jdklog.properties

下面are.xml是一个简单例子：

<SystemProperties>

**<Property name="GCI\_HOME" value="d:\\amargci " />**

<Property name="gciTaskFile" value=**"{ARE.GCI\_HOME}\\gci\_task.xml**" />

</SystemProperties>

…

<Service id="LOG" initOnStart="true">

<Provider>Amarsoft</Provider>

<Version>1.0</Version>

<Describe>日志服务</Describe>

<ServiceClass>com.amarsoft.are.log.impl.LogFactoryImpl</ServiceClass>

<Properties>

<Property name="com.amarsoft.are.log.logClass" value="com.amarsoft.are.log.impl.JDKLog" />

<Property name="com.amarsoft.are.log.JDKLog.properties" value="**{GCI\_HOME}/etc/jdklog.properties**" />

</Properties>

</Service>

在上面的例子中，首先在ARE的首部定义了一个环境变量GCI\_HOME，然后在第二个属性gciTaskFile中使用了这个变量，这样系统在实际运行的时候，就变成了“d:\\amargci\\etc\\gci\_task.xml”。方便的解决了绝对路径和相对路径的问题。同样的用法在下面的Service中再次出现。总结起来：**凡是在are.xml中定义的属性值中都可以出现ARE变量。**

同样的使用方法可以在task.xml中使用，task定义中的task、target和unit的扩展属性（<extendProperties>）中的属性值都可以包含ARE变量。

关于环境变量，和上面的论述对比，还有两个可以利用的特点：

* 凡是出现{ARE.xxx}的地方同样可以用{SYSTEM.xxx}来使用jvm的系统变量；
* 凡是可以出现ARE变量的地方都可以使用ARE注释，形式如{#我是注释，不会被转载到程序中}

例如：<Property name=”test” value=”{ARE.test}{#这个是ARE变量，比如－测试}，{SYSTEM.java.home}{#这个下取的系统变量}”> 的value解析实际等价于：

“测试，c:\java1.4.2”

关于ARE变量的使用，需要掌握几个原则：

* 系统变量（{$SYSTEM.xxx}）先于are.xml加载，因此总是可用的；
* ARE变量全部定义在are.xml的<SystemProperties>节点下；
* are.xml是先于其他所有配置文件加载的，因此在其他的配置文件中总是可以使用{$ARE.xxx}
* are.xml的属性是顺序加载的，因此，在<SystemProperties>只能后面的属性引用前面已经定义的变量
* are.xml中的<SystemProperties>是先于其他服务加载的，因此可以在服务的属性中引用are变量

## 标准任务单元

ARE提供了一些标准的的任务管理单元，用于最常用的任务处理。和批量接口程序关系密切的有两个：

* Table（SQL）->File（com.amarsoft.cabp.units.ExportFile）
* File->Table（com.amarsoft.cabp.units.ImportFile）

这两个单元分别完成数据传输中的两个重要任务。第一个单元根据元数据定义将SQL导出成文本文件；

第二个单元完成将文本文件按照元数据定义导入到数据库表中。

### Table（SQL）->File

**单元：**数据表或SQL导出成TXT文件

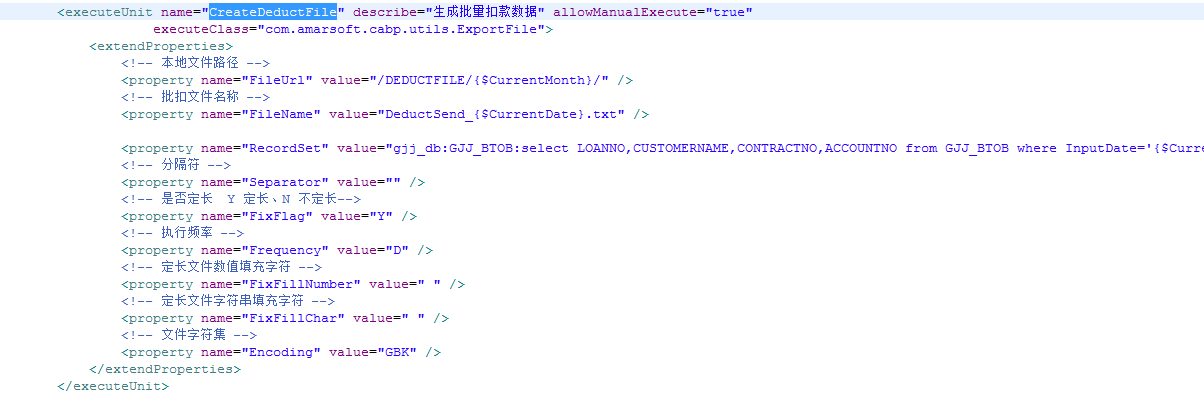
**类名：**com.amarsoft.cabp.units.ExportFile

**功能描述：**本单元是通过执行SQL获取的结果集与元数据格式定义进行匹配，以元数据定义的字段格式化要求生成文本文件。

**使用说明：**本单元包含以下属性：

|  |  |  |  |
| --- | --- | --- | --- |
| **属性定义** | **属性含义** | **必须性** | **缺省值** |
| FILE\_HOME | 用于作为输出文件的保存路径根目录，该属性隶属于ARE作为公共变量使用，和APP\_HOME变量类似。 | 必须 |  |
| FileUrl | 基于FILE\_HOME项下的文件保存路径，FILE\_HOME+FileUrl构成完整的文件存放路径。 | 必须 |  |
| FileName | 文件名 | 必须 |  |
| recordSet | 数据抽取方式，其结构如下：  DataSource:TableSource:Sql  DataSource:ARE Metadata中定义的元数据源。  TableSource:ARE MetaData中定义的元数据表 | 必须 |  |
| FixFlag | 定长标示 Y 定长文本、N 非定长文本 | 必须 | N |
| Frequency | 执行频率 D 每天、M 每月末、Y 每年末 | 必须 | D |
| FixFillNumber | 定长文件格式时，对数值未达到长度的左边填充字符 | 必须 | 一个空格 |
| FixFillChar | 定长文件格式时，对字符串未达到长度的右边填充字符 | 必须 | 一个空格 |
| Encoding | 输出文件采用字符集 | 必须 | GBK |
| Separator | 分隔符，非定长文件格式时使用 | 必须 | 空串 |
| FirstSeparator | 每行首字符是否采用分隔符开始，Y 是 N 否 | 必须 | Y |

**单元范例：**



### File->Table

**单元：**将TXT文件导入到数据表中

**类名：**com.amarsoft.cabp.units.ImportFile

**功能描述：**本单元是通过Txt文件与元数据格式定义进行匹配，将txt文件数据按照元数据配置更新到数据表中。

**使用说明：**本单元包含以下属性：

|  |  |  |  |
| --- | --- | --- | --- |
| **属性定义** | **属性含义** | **必须性** | **缺省值** |
| FILE\_HOME | 用于作为输出文件的保存路径根目录，该属性隶属于ARE作为公共变量使用，和APP\_HOME变量类似。 | 必须 |  |
| FileUrl | 基于FILE\_HOME项下的文件保存路径，FILE\_HOME+FileUrl构成完整的文件存放路径。 | 必须 |  |
| FileName | 文件名 | 必须 |  |
| CommitNum | 导入文件数据，数据库提交笔数上线 | 必须 | 1000 |
| SqlType | SQL类型 Insert/Update | 必须 | Insert |
| UpdateKeyColumns | 更新where条件，SqlType=Update时该值必须 | 有条件必须 |  |
| UpdateField | 更新字段，以逗号分隔，SqlType=Update时该值必须 | 有条件必须 |  |
| DeleteClause | 删除条件，SqlType=Insert时该值必须 | 有条件必须 |  |
| Separator | 分隔符，非定长文件格式时使用 | 必须 | 空串 |
| TableName | 数据库更新表名 | 必须 |  |
| SourceName | Metadata 所定义的数据源 | 必须 |  |
| SourceTable | Metadata 所定义的数据源表名 | 必须 |  |

**单元范例：**



# 应用总体设计

本节说明应用的总体设计逻辑，应用是基于第3节的基本应用框架的，因此本节讲述如何把系统分解target和unit，如何组织他们之间的关系的。

信贷系统一般数据流转：



核算批量接口处理数据的位置在图中表示，程序是围绕这个逻辑开展。

## 划分原则

根据上述核算批量接口逻辑，在设计整个程序时候按照下面的原则进行：

* 每个逻辑单元尽量功能单一，通过组合来完成任务
* 有逻辑关系的单元放在一个目标之中
* 每个数据倒入单元可以自动适应初始化和正式倒入
* 程序有一定的数据自我修复能力，避免错误扩大
* 程序应该尽量就有类似的逻辑，降低维护的成本
* 逻辑单元尽量利用已有的稳定功能模块
* 逻辑单元需要保证事务一致，需支持多次执行数据不能有影响

## Target划分

批量接口程序的功能清楚，任务之间有一定的逻辑关系，系统分为两个Target。一个Target处理核算批量扣款业务功能，另一个Target处理核算系统内账务处理和计算功能。

## Target说明

根据核算需求，核算批量接口流程图如下：



# 组件详细说明

本节详细说明每个单元的数据逻辑，每个单元按照基本说明和程序逻辑来说明。基本说明用表格的形式，说明所用的类，配置参数的含义，是必须等。算法说明阐述这些类的具体计算逻辑。

## 组件基础说明

### 组件基础抽象类

com.amarsoft.cabp.AbstractCreditBatch类是所有其他类继承的公共抽象类。该类继承于ExecuteUnit类。该类完成的工作有：

* 获取数据库连接；
* 初始化日志设置；
* 从SYSTEM\_SETUP取到系统营业日期，然后初始化一些变量，这些变量分别表示当前日期、下一日、上一日、当月、上月、下月、当年、下年、上年、年末日等等；
* 向数据表BATCH\_TASKSTATUS中记录批量处理路由单元，通过表中Status字段进行判断对应单元是否已经成功执行，如果已执行则不再执行直接跳过，否则就正常执行该单元。
* 通过SysConfig.properties加载核算需要的资源，如组件信息、利率信息、机构信息、还款方式、还款周期、业务品种、科目信息、交易信息等等；
* 实现ExecuteUnit的execute方法，该程序中执行抽象方法process，并判断其执行结果得出整个单元执行的状态，并将该状态写入数据库表BATCH\_TASKSTATUS.Status中，如果执行成功同时记录BATCH\_TASKSTATUS.EndTime字段。

此外，该类还提供一些公共方法，包括：

* 从task.xml中批量读取SQL语句自动执行提交；
* 替换task.xml配置文件中的变量内容，如将${deductDate}替换为当前营业时间。

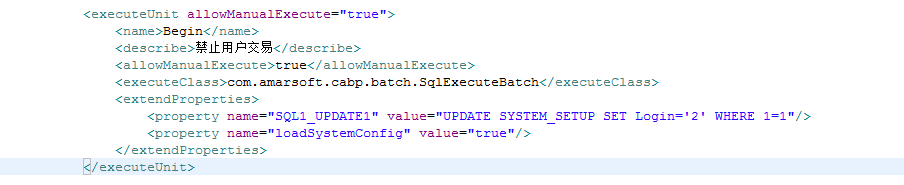
### 执行SQL工具类

类com.amarsoft.cabp.batch.SqlExecuteBatch是用于从task.xml配置文件中自动读取SQL语句然后自动执行提交的类。该类只能执行单一的SQL语句，不能处理复杂的业务逻辑。

在task.xml文件中配置的SQL语句需要遵循如下规则：

* 需自动执行的SQL必须以大写的SQL开头，然后紧跟SQL组顺序。
* SQL头与组顺序之后，必须是英文半角下划线。之后是SQL类型与组内顺序，SQL类型只能是SELECT/DELETE/INSERT/UPDATE中的一种区分大小写，组内顺序是指以相同的SQL+组顺序开头的SQL语句，顺序索引号。如第三组的第八条为查询语句，则命名为SQL3\_SELECT8。
* 如果INSERT语句需要使用之前的查询结果集，则INSERT需与SELECT语句在同一组，且需标明使用的RS在组内的顺序。方法为在最后添加下划线，然后是RS，最后是组内顺序，如SQL3\_INSERT10\_RS8,表明该条语句是第三组的第10条，类型为新增语句，新增使用的结果集为第三组的第八条语句查询出来的。

配置举例：



## 开始核算批量

### 基本说明

**名称：**StartCABP

**Java类：**com.amarsoft.cabp.batch.SqlExecuteBatch

**功能：**更新系统表SYSTEM\_SETUP的Login字段为2，标志批量已经启动

**属性：**可参见前面对该SQL执行工具类的介绍

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | SQL1\_UPDATE1 | 更新SQL脚本 | UPDATE SYSTEM\_SETUP SET Login='2' WHERE 1=1 |
| 2 | loadSystemConfig | 是否加载系统定义的参数信息，参见SysConfig. properties | true，批量启动的第一个单元和日期切换单元才设置该值为true，其他单元都为false不加载缓存 |

**必要性：**必须，本单元时整个核算批量的起点。

### 程序逻辑

参见[执行SQL工具类](#_执行SQL工具类)。

## 表外业务垫款

### 基本说明

**名称：**

**Java类：**com.amarsoft.are.task.units.FileDataSourceCheckUnit

**功能：**数据文件加查单元，用于对数据文件进行基本检查，以便确定所对应的文件的字段个数等没有和metadata要求不一致。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | resultFile | 检查结果输出文件 | 合法的文件名 |
| 2 | dataSource\_x | 待检查的数据源，所有已dataSource\_开头的属性被认为是一个数据源 | dataSource\_x中的x是一个自然的序号如\_0、-1等，用于唯一的区分这些属性。 |

**必要性：**可选，本单元是为了保证后面的数据源定义的原文件符合要求，这在程序试运行阶段比较有必要，如果可以确定所给的文件就是符合要求的，则可以去掉此单元提高到性能。

### 程序逻辑

本程序按照dataSource\_x定义的每一个数据源，检查期合法性，这个数据源是统一数据源定位中的文件类型，如：datasource:ndb:datax:select \* from SubjectHistory:data/SubjectHistory.dat:\|。

程序检查逻辑是：

1. 检查定义的文件是否存在；
2. 检查文件中的每行数据的格式和原数据定义中的样式是否一致

如果检查逻辑没有通过，除非所有的dataSource检查都通过，返回成功的状态，否则返回失败。

## 生成批量扣款数据

### 基本说明

**名称：**GenerateDeductData

**Java类：**com.amarsoft.cabp.batch.GenerateDeductData

**功能：**根据贷款还款计划、费用还款计划、贴息计划生成当日应该还款的贷款还款记录。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | SQL1\_DELETE1 | Delete批扣数据表BATCH\_LAS\_CORE当日扣款数据 | DELETE FROM BATCH\_LAS\_CORE WHERE INPUTDATE='${deductDate}' and BatchNo = '01' |
| 2 | BatchNo | 扣款数据批次，其值需和属性1中SQL的where条件保持相同 | 01，如果每日有多次扣款则配置多个task在这之上累加 |

**必要性：**必选

### 程序逻辑

该程序逻辑主要是将还款计划表（ACCT\_PAYMENT\_SCHEDULE）、费用计划表（ACCT\_FEE\_SCHEDULE）当日以及以前应还款还未还的记录按照核算主题（贷款、费用）生成对应的批量扣款数据到（BATCH\_LAS\_CORE）。

#### 数据转换

数据倒入过程中进行了下面主机-信贷的转换：

* 机构代码
* 币种

## 生成批量扣款文件

### 基本说明

**名称：**CreateDeductFile

**Java类：**com.amarsoft.cabp.units.ExportFile

**功能：**根据抽数SQL和元数据定义生成对应的。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | FileUrl | 基于FILE\_HOME项下的文件保存路径，FILE\_HOME+FileUrl构成完整的文件存放路径。 | /DEDUCTFILE/{$CurrentMonth}/ |
| 2 | FileName | 文件名 | DeductSend\_{$CurrentDate}.txt |
| 3 | recordSet | 数据抽取方式，其结构如下：  DataSource:TableSource:Sql  DataSource:ARE Metadata中定义的元数据源。  TableSource:ARE MetaData中定义的元数据表 | db:BATCH\_LAS\_CORE:select \* from BATCH\_LAS\_CORE where InputDate='{$CurrentDate}' and BatchNo = '01' |
| 4 | FixFlag | 定长标示 Y 定长文本、N 非定长文本 | Y |
| 5 | Frequency | 执行频率 D 每天、M 每月末、Y 每年末 | D |
| 6 | FixFillNumber | 定长文件格式时，对数值未达到长度的左边填充字符 | 空格 |
| 7 | FixFillChar | 定长文件格式时，对字符串未达到长度的右边填充字符 | 空格 |
| 8 | Encoding | 输出文件采用字符集 | GBK |
| 9 | Separator | 分隔符，非定长文件格式时使用 | 空串 |
| 10 | FirstSeparator | 每行首字符是否采用分隔符开始，Y 是 N 否 | N |

**必要性：**必选

### 程序逻辑

参见标准任务单元

## 批量扣款结果文件处理

### 基本说明

**名称：**ImportDeductFile

**Java类：**com.amarsoft.cabp.units.ImportFile

**功能：**从文件中将数据解析导入到表BATCH\_CORE\_LAS。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | FileUrl | 基于FILE\_HOME项下的文件保存路径，FILE\_HOME+FileUrl构成完整的文件存放路径。 | /BATCHFILE/{$CurrentMonth}/ |
| 2 | FileName | 文件名 | DeductRecieve\_{$CurrentDate}.txt |
| 3 | CommitNum | 导入文件数据，数据库提交笔数上线 | 1000 |
| 4 | SqlType | SQL类型 Insert/Update | insert |
| 5 | UpdateKeyColumns | 更新where条件，SqlType=Update时该值必须 | 空 |
| 6 | UpdateField | 更新字段，以逗号分隔，SqlType=Update时该值必须 | 空 |
| 7 | DeleteClause | 删除条件，SqlType=Insert时该值必须 | INPUTDATE = '{$CurrentDate}' and BatchNo='01' |
| 8 | Separator | 分隔符，非定长文件格式时使用 | 空 |
| 9 | TableName | 数据库更新表名 | BATCH\_CORE\_LAS |
| 10 | SourceName | Metadata 所定义的数据源 | db |
| 11 | SourceTable | Metadata 所定义的数据源表名 | BATCH\_CORE\_LAS |

**必要性：**必选

### 程序逻辑

参见标准任务单元

## 批量贷款还款

### 基本说明

**名称：**PaymentBatch

**Java类：**com.amarsoft.cabp.batch.RepaymentBatch

**功能：**该单元根据BATCH\_CORE\_LAS记录中实际还款金额针对每笔贷款或费用进行还款操作，更新贷款还款计划表、费用还款计划表、贷款账务、费用账务。

**属性：**无

**必要性：**必选。

### 程序逻辑

该单元根据BATCH\_CORE\_LAS记录中实际还款金额针对每笔贷款或费用进行还款操作，更新贷款还款计划表、费用还款计划表、贷款账务、费用账务。

## 费用摊销、递延

### 基本说明

**名称：**BatchAmortizeFee

**Java类：**com.amarsoft.cabp.batch.BatchAmortizeFee

**功能：**计算需摊销、递延的费用信息进行入账处理。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | transCode | 费用摊销和递延调用的交易 | 3551 |

**必要性：**必选

### 程序逻辑

计算需摊销、递延的费用信息，将计算的数值进行入账处理。

## 贷款日终交易

### 基本说明

**名称：**LoanEOD

**Java类：**com.amarsoft.cabp.batch.DailyLoanBatch\_EOD

**功能：**贷款日终交易，完成贷款形态的转换，贷款罚息、复利的计算等。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | TransCode | 贷款日终调用的交易编号 | 9091 |

**必要性：**必选

### 程序逻辑

该程序抽取未结清贷款且贷款交易日期不大于当前日期的贷款信息，对这些贷款信息做贷款日终账务交易，出现错误回滚数据库，正常执行则提交数据库。

## 年终结转交易

### 基本说明

**名称：**AccountingEOY

**Java类：**com.amarsoft.cabp.batch. AccountingEOY

**功能：**年终结转交易，完成年终结算将损益类科目结转到结算科目中。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | FromSubjectNo | 待结转的科目，可以存在多个，多个以逗号分隔 |  |
| 2 | ToSubjectNo | 年结科目，单一值 |  |

**必要性：**可选，根据各家行对年结的要求。

### 程序逻辑

系统判断每年12月31日执行本单元，首先备份ACCT\_GENERAL\_LEDGER当日数据到ACCT\_GENERAL\_LEDGER\_HIS表中，然后在处理根据输入的FromSubjectNo和ToSubjectNo做系统的内部转账动作，将总账（ACCT\_GENERAL\_LEDGER）中FromSubjectNo科目余额转入到ToSubjectNo科目中，将FromSubjectNo科目余额清零。

## 汇总账务明细生成总账

### 基本说明

**名称：**InitGeneralLedger

**Java类：**com.amarsoft.cabp.batch.InitGeneralLedger

**功能：**根据明细账汇总生成总账，总分账平衡。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | checkSubjectNo | 总分平衡检查科目 |  |

**必要性：**必选

### 程序逻辑

根据交易明细表（ACCT\_SUBLEDGER\_DETAIL）当日发生额和总账表（ACCT\_GENERAL\_LEDGER）当日余额生成今日的总账余额。

## 换日

### 基本说明

**名称：**ToNextDate

**Java类：**com.amarsoft.cabp.batch.SqlExecuteBatch

**功能：**更新系统交易日期为下一日。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | SQL1\_UPDATE1 | 更新SQL脚本 | update SYSTEM\_SETUP set businessDate='${nextDate}' WHERE 1=1 |
| 2 | loadSystemConfig | 是否加载系统定义的参数信息，参见SysConfig. properties | true，批量启动的第一个单元和日期切换单元才设置该值为true，其他单元都为false不加载缓存 |

**必要性：**必选

### 程序逻辑

程序按照配置执行SQL更新系统交易日期到下一日。

## 贷款日初交易

### 基本说明

**名称：**LoanBOD

**Java类：**com.amarsoft.cabp.batch.DailyLoanBatch\_BOD

**功能：**贷款日初交易，完成贷款结息，基准利率调整等。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | TransCode | 贷款日初调用的交易编号 | 9092 |

**必要性：**必选

### 程序逻辑

该程序抽取未结清贷款且贷款交易日期不大于当前日期的贷款信息，对这些贷款信息做贷款日初账务交易，出现错误回滚数据库，正常执行则提交数据库。

## 预约变更处理

### 基本说明

**名称：**RunTransactions

**Java类：**com.amarsoft.cabp.batch.RunTransactions

**功能：**预约变更处理完成客户提前申请好的当日生效的变更信息。

**属性：无**

**必要性：**必选

### 程序逻辑

预约变更处理完成客户提前申请好的当日生效的变更信息，调用变更对应的交易进行数据处理。

## 结束核算批量

### 基本说明

**名称：**EndCABP

**Java类：**com.amarsoft.cabp.batch.SqlExecuteBatch

**功能：**更新系统表SYSTEM\_SETUP的Login字段为1、nextBatchExecuteDate为交易日当日、BatchDealFlag为2，标志批量已经结束

**属性：**可参见前面对该SQL执行工具类的介绍

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **属性名称** | **含义** | **取值说明** |
| 1 | SQL1\_UPDATE1 | 更新SQL脚本 | update SYSTEM\_SETUP set NextBatchExecutedate='${deductDate}',Login='1',BatchDealFlag='2' WHERE 1 = 1 |

**必要性：**必须，本单元时整个核算批量的终点。

### 程序逻辑

参见[执行SQL工具类](#_执行SQL工具类)。