****

**版本号 1.00**

**共 册/共 页**

**杭州登虹科技有限公司**

**转码基础库**

**StreamConvert概要设计说明书**

**编 写 人：Closeli**

**编写时间：2018.06.04**

**部 门 名：ATR**

**审 核 人：**

**审核时间：**

修订记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **编号** | **变更时间** | **版本** | **变更人** | **审批人** | **变更说明** |
| 1 | 2018/6/5 | V0.1 | 田野 |  | 新建，StreamConvert库概要设计文档 |
| 2 | 2018/6/7 | V0.2 | 田野 |  | 根据意见，进行格式修改 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[**1 引言** 4](#_Toc516146517)

[**1.1编写目的** 4](#_Toc516146518)

[**1.2背景** 4](#_Toc516146519)

[**1.3定义** 4](#_Toc516146520)

[**1.4参考资料** 5](#_Toc516146521)

[**2 总体设计** 5](#_Toc516146522)

[**2.1目标** 5](#_Toc516146523)

[**2.2设计约束** 5](#_Toc516146524)

[**2.3系统结构** 5](#_Toc516146525)

[**2.4详细结构** 6](#_Toc516146526)

[**3 模块设计** 7](#_Toc516146527)

[**3.1 线程池模块** 7](#_Toc516146528)

[**3.2 任务队列模块** 8](#_Toc516146529)

[**3.3 任务模块** 9](#_Toc516146530)

[**3.4 转封装模块** 10](#_Toc516146531)

[**4 运行设计** 10](#_Toc516146532)

[**4.1外部接口调用流程** 10](#_Toc516146533)

[**4.2内部任务创建流程** 11](#_Toc516146534)

[**4.3任务调度流程** 12](#_Toc516146535)

[**5 特定需求** 13](#_Toc516146536)

[**5.1安全** 13](#_Toc516146537)

[**5.2 性能** 13](#_Toc516146538)

[**5.3 可用性** 13](#_Toc516146539)

[**5.4易用性** 13](#_Toc516146540)

[**5.5可测试性** 13](#_Toc516146541)

[**5.6可移植性** 13](#_Toc516146542)

[**5.7 可维护性** 13](#_Toc516146543)

[**5.8 兼容性** 13](#_Toc516146544)

[**5.9 系统** 13](#_Toc516146545)

[**6 外部依赖** 14](#_Toc516146546)

[**7 改动/影响** 14](#_Toc516146547)

[**8 测试建议** 14](#_Toc516146548)

[**9 当前问题** 14](#_Toc516146549)

[**10潜在问题** 14](#_Toc516146550)

**1 引言**

**1.1编写目的**

本文主要描述了转封装基础库里面的基本结构和模块之间的交互流程，定义了模块的接口和数据成员，旨在帮助读者快速的了解转封装库，同时为代码开发开发人员和测试人员提供一定的指导。

本文档预期读者包括：产品经理，项目经理，软件开发工程师，测试工程师等项目相关人员

**1.2背景**

随着互联网、安防、物联网等行业的快速爆发，摄像头，手机等设备在实际中产生了海量音视频数据。这些数据在不同的平台上处理时，会存在一定的兼容性问题。公司将提供转码服务用于实时流和云存储等数据的格式转换，其中转码基础库(StreamConvert)作为转码服务的核心模块，将进一步提高代码的复用性。同时可用于公司其它项目中快速集成，这样将大大降低开发和维护成本。

转码基础库StreamConvert作为媒体业务的基础库，主要用于视频数据的封装格式转换。该库对外提供相应接口，可以用于相关媒体服务和公司重大项目的快速开发。目前基础库用在转码服务StreamTranscode和OneNet项目中。

该库目前由ATR团队负责开发和维护，相关的开发包和成果物可以联系ATR团队提供。

**1.3定义**

本文档中会出现一些缩略语和技术用语，为了简化描述和避免歧义，下面表格将给出详细解释说明。后续在本文档遇到相关词汇，不特殊说明情况下，一切以表格给出的解释为准。

**表 1‑1：技术用于/缩略语**

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **缩写** | **原词** | **详细说明** |
| 1 | RAW |  | 公司内部定义的私有码流封装格式 |
| 2 | TS | TransportStream | 是一种视频封装格式MPEG2-TS，特点是可以再视频流的任一一段都可以解码播放 |
| 3 | FLV | Flash Video | FLV流媒体格式是随着Flash MX的推出发展而来的视频格式. |
| 4 | MP4 | MPEG-4 | MP4是一套用于音频、视频信息的压缩编码标准 |
| 5 | FFmpeg |  | FFmpeg是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序. |

**1.4参考资料**

**参考文档:**

1.《转码服务StreamTranscode产品用户使用说明书》；

2.《转码基础库StreamConvert需求说明书》；

**2 总体设计**

**2.1目标**

转封装基础库(StreamConvert)，输入源码流，指定配置参数，经过转封装基础库(StreamConvert)内部数据转换处理，回调输出目标码流。

目标码流

(flv ts raw mp4)

待转码流

(flv ts raw mp4)

转封装基础库(StreamConvert)

**图<1>:StreamConvert处理流程示意图**

**2.2设计约束**

**2.2.1 运行环境**

网络： 标准配置 万兆网卡 最低配置千兆网卡

硬件： Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz 4核 最低2核

内存： 标准8GB 最低配置2GB

**2.2.2 软件环境**

内核版本： Linux version 3.10.0-693.el7.x86\_64

发行版本： CentOS Linux release 7.4.1708 (Core)

**2.2.3 编译环境**

编译器： gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-16)

**2.3系统结构**

转封装StreamConvert基础库作为转码服务的基础组件，在业务中的位置如下图所示：



**图<2>:StreamConvert基础库业务位置**

**2.4详细结构**

下图提供转码基础库的核心功能模块结构图，该库目前分为三层。从上面的接口层，中间的业务层，和最下面的转封装模块层构成。各个模块的详细结构如下：



**图<3>:StreamConvert基础库详细框架图**

**3 模块设计**

**3.1 线程池模块**

**功能描述：**

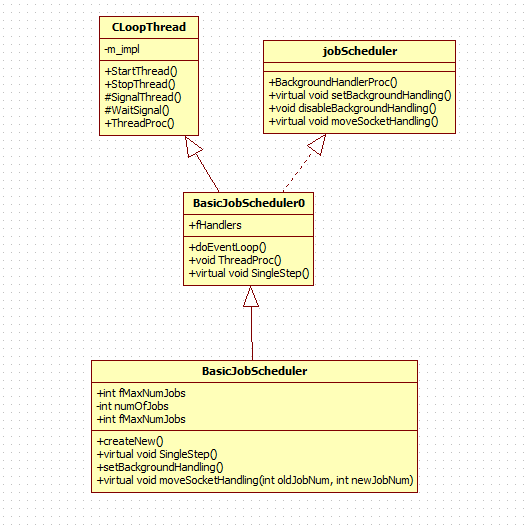
目前服务器处理器CPU一般都是多核的，为了进一步发挥处理器的并发性能和提升转封装的并发路数，该基础库内部实现了一个线程池。

1、线程池由固定数目的线程组成，用户可以在调用接口时进行设置。目前给的参考初始线程数目为：cpu核心数\*8。

2、每个线程都持有一个属于自己的任务队列，当程序启动后。则每个线程通过调度算法来轮询自己的任务队列中的任务，拿到任务后，则读取相应缓存中数据，进而在转封装模块中进行处理。

**依赖的模块：**任务队列模块

**类图设计：**



**图<4>:线程池和调度器类图**

**模块关键实现：**

BasicJobSchedule继承了线程基类CLoopThread和jobSchedule。其中CLoopThread提供了线程创建，销毁等操作；

jobSchedule则提供了向任务队列fHandles里面插入任务的操作。这样BasicJobSchedule则可以用自己创建的线程，来通过线程函数ThreadProc执行调度算法来遍历任务队列里面的任务，从而完成转封装的操作。

本基础库线程池的大小，则取决于基础库加载后，实例化BasicJobScheduler的个数，这个可以通过提供给用户的接口来决定线程池的大小。

**提供的接口：**

**1、**

|  |  |
| --- | --- |
| **接口：** | **void doEventLoop(char volatile\* watchVariable)** |
| **接口描述：** | 线程处理函数，是线程池初始化后，该线程的执行函数 |

**2、**

|  |  |
| --- | --- |
| **接口：** | **virtual void SingleStep() = 0** |
| **接口描述：** | 调度函数，线程调用该函数，调度任务队列里面的任务；关于该调度算法会在后面进一步阐述； |

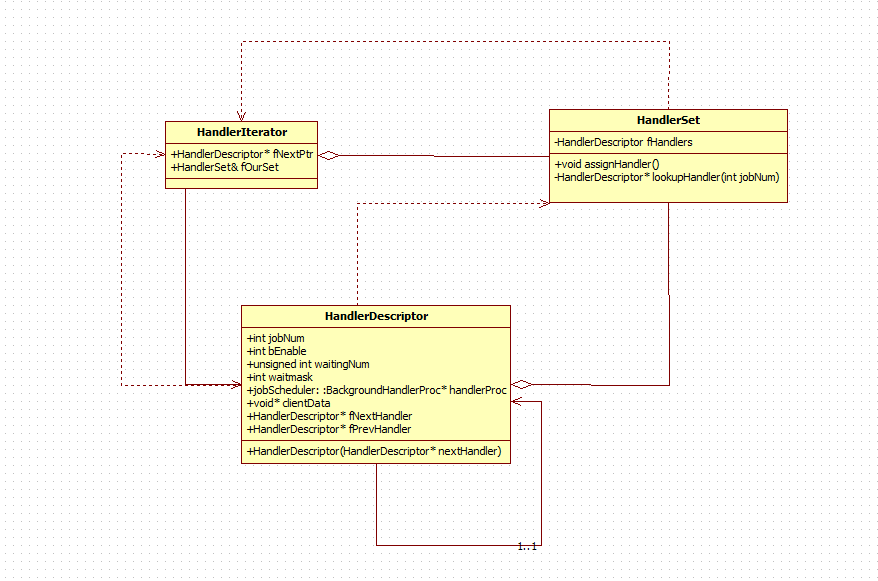
**3.2 任务队列模块**

**功能描述：**

1. 目前转封装(StreamConvert)基础库提供了两种任务队列，包括后台任务和延时任务。任务队列中保存着用户创建的任务;
2. 一种任务时延时任务，该任务需要定时器触发，只有达到一定的超时时间才能被线程池中的线程调度;
3. 还有一种是后台任务，这种任务，线程池的线程会一直通过调度算法执行该队列的任务，目前基础库主要使用了该任务模型;

**依赖模块：**线程池模块 任务模块

**类图设计：**



**图<5>:任务队列类图**

**模块关键实现：**

1. HandlerDescriptor类将具体的任务类型和执行任务时应该执行函数封装成了结点，同时这样的结构是一个双向链表的形式，每一个任务都会链在上一个任务结点后面，其中前向指针指向了前面的任务节点，后向指针指向了后面的任务节点；
2. HandlerSet类则是任务集合，其中通过迭代器指针执行了所有任务的起始节点，从而利于迭代器则可以操作所有的任务。体重主要提供了两个方法assignHandler和lookupHandle，一个是新来任务的插入和通过任务编号对相应任务节点的查询。

3. HandlerIterator类则是迭代器类，主要是对双向链表的实际操作，其中主要通过fNextPtr执行双向链表的起始位置，通过指针的遍历实现对双向链表的操作。

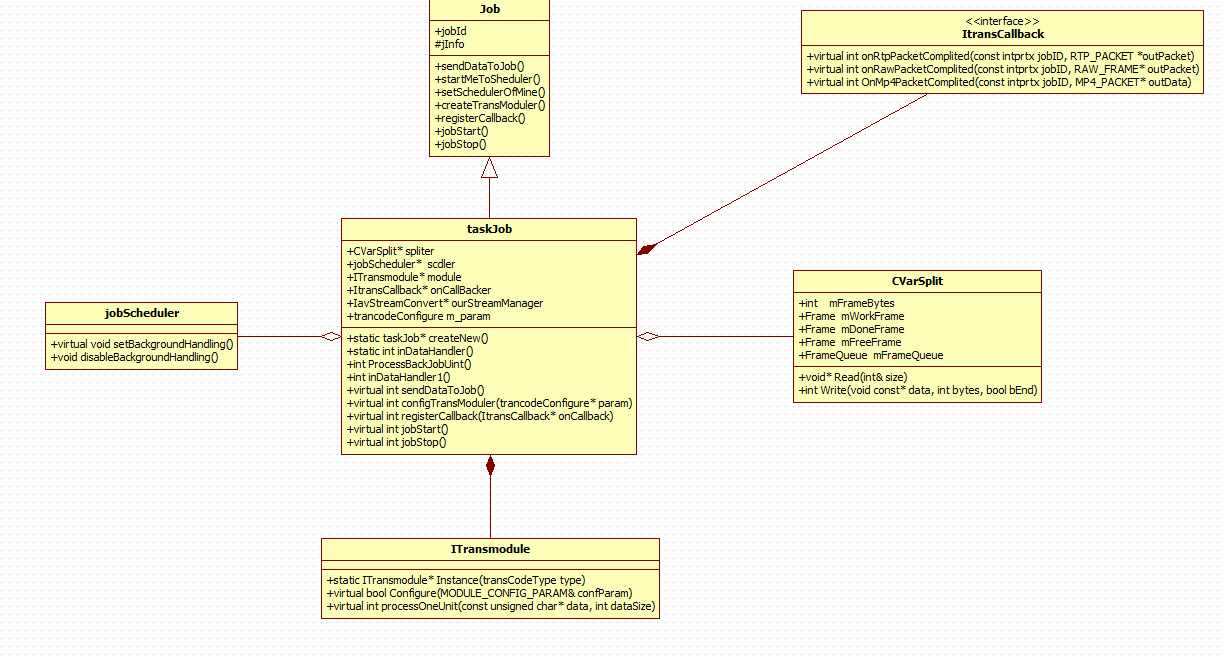
**3.3 任务模块**

**功能描述：**

1. 任务模块taskJob是对用户接口的具体实现；
2. 也是对数据缓存队列模块HandlerSet、调度模块jobScheduler以及转封装模块Itransmode的粘合;
3. 当用户创建任务后，任务实例化转封装模块Itransmode对象，将自己这个任务添加到调度模块中负载最轻的任务队列HandlerSet里;
4. 然后等待线程池的线程来调度自己。当任务被调度时，将读取数据缓存CVarSplit里面的源目标格式数据，然后在转封装模块ITransmode中进行转封装，转完成后，转封装模块将目标格式的数据通过ItransCallback里面注册的回调函数回调给使用者;

**依赖模块：**线程池模块 任务队列模块 转封装模块

**类图设计：**



**图<6>:任务类图**

**提供的接口：**

taskJob类是具体接口的实现类，其中主要起桥接作用。

**1、**

|  |  |
| --- | --- |
| **接口：** | **int taskJob::createTransModuler(trancodeConfigure\* param)** |
| **接口描述：** | 接口主要用来实例化转封装模块对象，根据传进来的参数决定用哪个模块进行完成转封装。 |

**2、**

|  |  |
| --- | --- |
| **接口：** | **int taskJob::registerCallback(ItransCallback\* onCallback)** |
| **接口描述：** | 接口主要用来注册回调函数，用于转封装模块能够将数据回调给调用者； |

**3、**

|  |  |
| --- | --- |
| **接口：** | **int taskJob::sendDataToJob(const unsigned char\* exterData, int exterSize, int offSet,const unsigned char\* insideData, int insideSize, CONVERT\_BUFFER\_INFO\* bInfo)** |
| **接口描述：** | 接口主要用来用户层向该任务的数据缓存里发送数据；。 |

**4、**

|  |  |
| --- | --- |
| **接口：** | **int taskJob::inDataHandler(void\* instance, int& mask)** |
| **接口描述：** | 是线程池轮询的任务处理函数; |

**5、**

|  |  |
| --- | --- |
| **接口：** | **int taskJob::ProcessBackJobUint(unsigned char\* data, int dataSize)** |
| **接口描述：** | 是读取数据函数，是任务被执行时，从源缓存读取数据的函数； |

**3.4 转封装模块**

**功能描述：**

该模块是真正完成转封装动作的模块，里面包含了ts2raw、flv2raw、raw2raw等模块。

**依赖模块：**任务模块

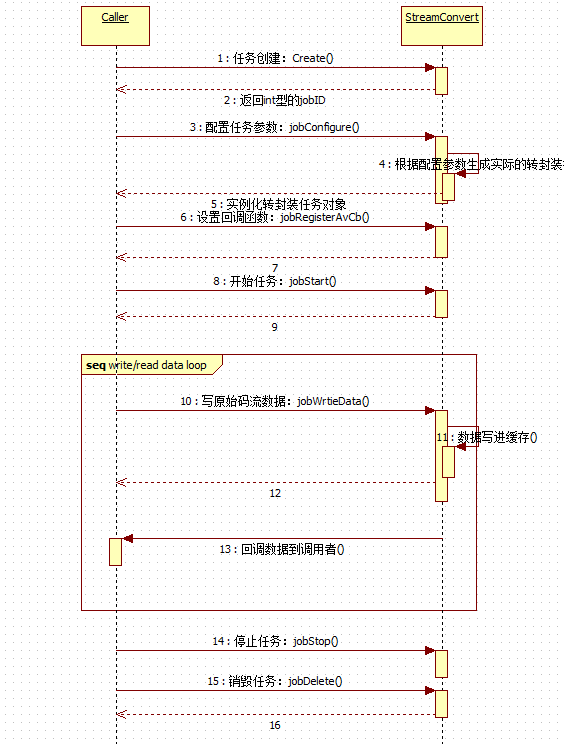
**4 运行设计**

**4.1外部接口调用流程**

转封装StreamConvert库，提供了一组用户接口。用户可以根据这些接口，创建不同的转封装任务，完成从源格式码流向目标格式码流的转换动作。下面将用时序图示例使用者调用接口的调用流程；

**通信主体：**外部调用者 转封装库

**时序图：**



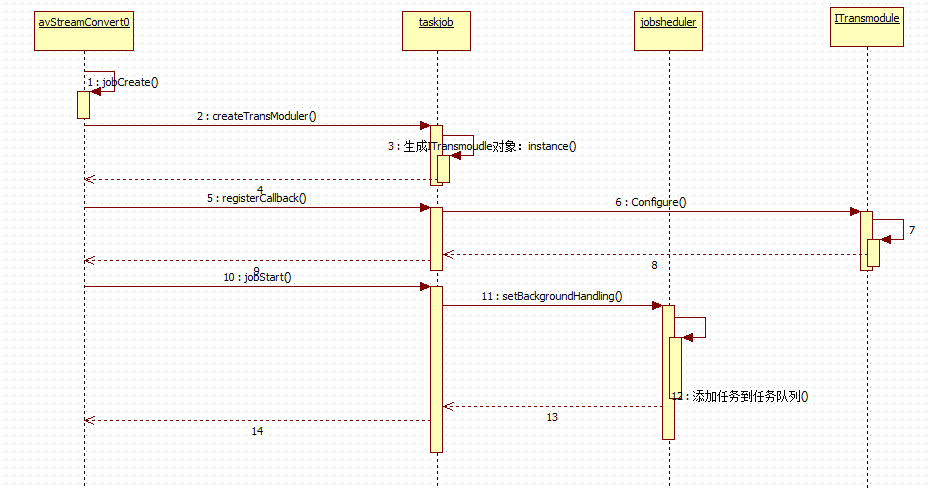
**图<6>:外部调用接口序列图**

**4.2内部任务创建流程**

外部在创建任务后，需要将该任务添加到任务队列里面，下面序列图展示这个过程。

**通信主体：**调用者 任务模块 任务队列

**时序图：**



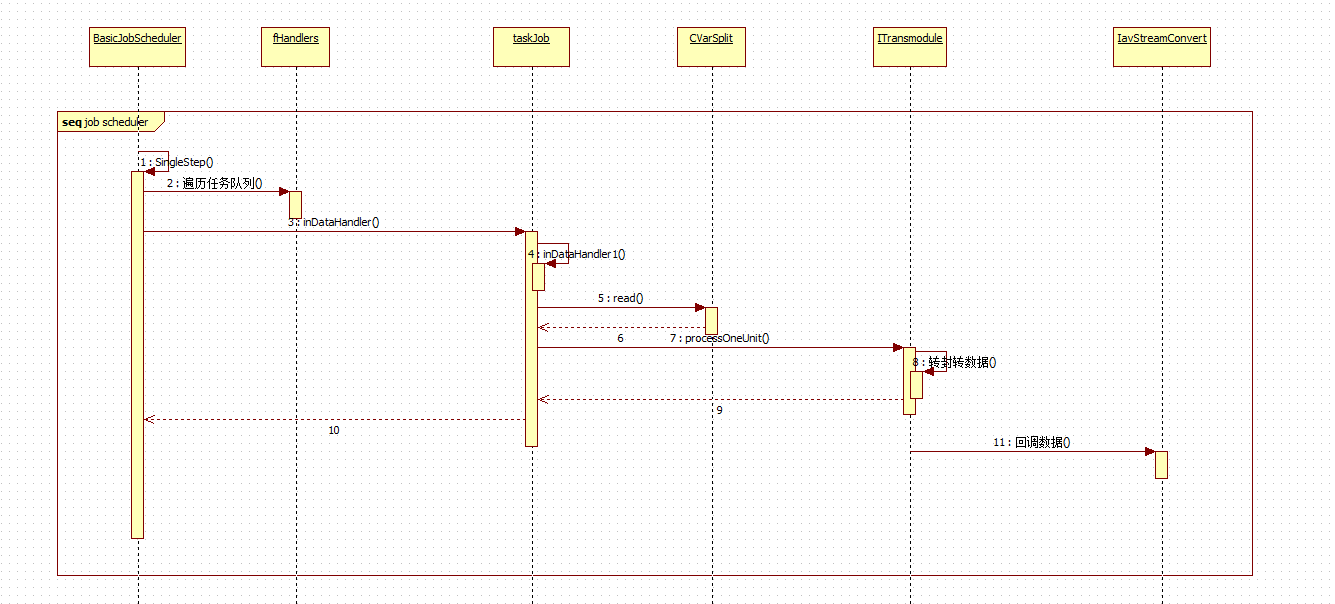
**图<7>:任务创建时序图**

**4.3任务调度流程**

当一系列任务不断进行创建添加到任务队列中，线程池中的线程就会通过调度算法来执行这些线程，进而执行每个任务。

**通信主体：**调度器 任务队列 任务 数据缓存 转封装模块 调用者

**时序图：**



**图<8>:调度器处理任务时序图**

**5 特定需求**

**5.1安全**

**5.2 性能**

**5.3 可用性**

**5.4易用性**

**5.5可测试性**

**5.6可移植性**

**5.7 可维护性**

**5.8 兼容性**

**5.9 系统**

**5.9.1 Linux**

目前支持linux64位版本

**5.9.2 Windows**

N/A

**6 外部依赖**

**7 改动/影响**

**8 测试建议**

**9 当前问题**

**10潜在问题**

**10.1**目前转封装库还不支持日志回调;

**10.2** 功能上支持的格式组合还不是很多，后续会逐渐丰富该库的基本转封装能力；

**10.3** 全链路没有唯一ID，导致定位问题困难；