有没有时间介绍一下大概的业务流程

|  |  |  |  |
| --- | --- | --- | --- |
| 活动编号（ID）： | | | |
| 项目阶段 | ■计划阶段 | □需求阶段 | □概设阶段 |
| □详计阶段 | □编码阶段 | □单元测试阶段 |
| □集成测试阶段 | □系统测试阶段 | □关闭阶段 |
| 产品名称 | | 产品型号/版本 | 总页数 |
| 平台2.0 | | V2R1 | 共 ×× 页 |

CUI外设模块处理概要设计

（仅供内部使用）

|  |  |
| --- | --- |
| 文 件 编 号： | KIPD-MB-0001 |
| 版 本 号： | V2R2 |
| 实 施 日 期： | 2011-11-1 |
| B M 等 级： | ■M\*M □JI\*M □JUE\*M |
|  | |
| 编 制： |  |
| 审 核： |  |
| 会 签： |  |
|  |  |
|  |  |
| 批 准： |  |

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本号 | 描述 | 作者 |
| 2011-9-13 | 0.1 | 初稿完成 | 汪文轩 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1 前言 4](#_Toc326579245)

[2 范围 4](#_Toc326579246)

[2.1 项目名称 4](#_Toc326579247)

[2.2 项目目标 4](#_Toc326579248)

[2.3 项目时间 4](#_Toc326579249)

[3 定义 4](#_Toc326579250)

[4 运行环境 4](#_Toc326579251)

[5 主要功能 5](#_Toc326579252)

[5.1 业务层次图(外设) 5](#_Toc326579253)

[5.2 业务功能分析(外设) 5](#_Toc326579254)

[5.2.1 CU的会话维护(外设) 5](#_Toc326579255)

[5.2.2 CU的事务管理(外设) 5](#_Toc326579256)

[5.2.3 CU的断链处理机制(外设) 6](#_Toc326579257)

[6 架构设计（外设） 7](#_Toc326579258)

[6.1 CUI框架设计（外设） 7](#_Toc326579259)

[7 CUI流程描述 7](#_Toc326579260)

[7.1 CU操作外设流程 7](#_Toc326579261)

[7.2 CU下线断链处理(外设) 8](#_Toc326579262)

[7.3 CU录像下载操作(外设) 9](#_Toc326579263)

[7.4 CUI与模块断链(外设) 10](#_Toc326579264)

# 前言

平台2.0中，CU采用WebService接口与平台交互，外设模块都采用SIP协议相互通信。CUI从中起到两种协议相互转换的作用。同时由于外设模块对于CU来说是即插即用的，所以cui还起到了登陆断链等方面的问题(即：何时与外设模块建链，何时与外设模块断开)。因此CUI的设计模型必须满足这种应用场景。

# 范围

## 项目名称

科达视频监控平台ViewShot V2R2

## 项目目标

满足监控平台2.0的需要。

## 项目时间

随平台2.0一起发布。

# 定义

CUI：平台2.0中的负责接入CU的协议转换模块

CU：用户客户端。

# 运行环境

CUI：Linux

# 主要功能

## 业务层次图(外设)



图 5-1 CUI业务层次图

## 业务功能分析(外设)

### CU的会话维护(外设)

WebService是无连接协议，CU和CUI之间没有对应的OspNode，CUI也无法从请求中判断是来自哪个CU的，因此要有CU会话状态管理。

在CU进入服务态后，CUI收到CU操作外设请求，首先判断CUI是否向外设注册过，如果没有注册过则CUI先向外设进行注册(创建一个XXXAgent(外设task)维护CUI与当前外设的断链建链任务)，从外设获取一个SessionID与自身Task绑定，注册完毕后判断CU操作权限进行相应操作，允许操作则在XXXAgent内记录该CU的SessionID，同时记录该XXXAgent到Cui数据管理表中。

### CU的事务管理(外设)

当收到CU的操作外设请求之后，XXXAgent就会创建相应的操作事务，并由该事务处理该请求，XXXAgent记录该事务信息。这些事务直接由XXXAgent管理，外设模块回复消息时，则直接通过TaskID找到新创建事务，交由新创建的事务处理。

在CUI中，所有的资源都是由事务Task掌控的，只要删除XXXAgent并连带删除其任务集合中的所有Task事务，CU的所有分配XXX外设相应资源都会回收。

CUI收到XXXAgent的主动通知(只存在订阅和查询的主动通知)的消息时，首先通过找到订阅的子Task处理该消息，子Task找到相应的CuAgent填到对应的消息队列中。由于WebService只有请求应答模式，没有逆向请求应答以及主动通知模式。所以这些主动通知消息被CuAgent缓存，等待CU轮询时再回复。

CUI与服务端之间的会话协议需要满足事务三元组<SessionID，TaskID，SeqNum>，在SIP会话中SeqNum和TransID是等价的。

### CU的断链处理机制(外设)

WebService是无连接的协议，而业务上CUI必须知道CU是否在线，以此决定是否继续保留为该CU分配的资源。所以CUI业务层上必须有对CU的保活机制。

当CUI与XXXAgent断链时，在子Task内如果给CU相应的错误码回复，删除所有的子Task，最后删除该XXXAgent。

当某CU下线或者断链时，遍历XXXAgent中的子Task，删除相关的Task。

注意：CU对于XXXAgent来说是不存在的，或者说对于XXXAgent来说，只有CUI。CUI作为SIP的UA注册到XXXAgent上，XXXAgent只对CUI设置心跳。

# 架构设计（外设）

## CUI框架设计（外设）



图 6-1 CUI框架设计

# CUI流程描述

下面我们描述几个常用的消息流程。

## CU操作外设流程

CU的操作分为以下几个步骤：1，CUI注册XXX外设；2，创建相应的task处理；3，获取平台外设数据。



图 7-1 CU操作外设时序图

如图7-1所示，CU的操作顺序如下：

1. CU向CUI发起获取外设数据信息请求。
2. CUI向XXXAgent发起注册请求。
3. XXXAgent发起注册响应给CUI。
4. 注册成功后，CU以CUI身份，由CUI向XXXAgent发送获取外设数据请求。
5. XXXAgent向CUI发送获取外设数据响应。
6. CUI将消息转发给CU。

## CU下线断链处理(外设)

外设模块无需感知CU用户的存在，只感知提供服务的事务状态以及CUI的状态。当CUI发现CU用户下线或者意外断链时，中断所有与之相关的外设事务，但是订阅事务(录像、下载)则发送BYE请求给外设模块，通知外设模块结束相应的业务并删除该事务。当CUI下线或者断链，则外设模块中断并删除所有相关事务。

## CU录像下载操作(外设)



图 7-3 CU视频浏览时序图

这一流程主要描述CUI处理一般操作的内部流程，步骤如下：

1. CU请求录像下载。
2. Cui\_Server收到请求收，首先查找Cu对应的CuAgent。将请求交由CuAgent处理。
3. CuAgent创建一个录像下载事务，并向外设模块发送录像下载请求。此消息也是异步的，含带浏览事务号。
4. 外设模块返回允许录像下载响应给Cui\_Server。Cui\_Server依据事务号找到对应的视频浏览事务。并将该应答消息丢该事务处理。
5. 录像下载事务判断本次事务生命周期是否终结。如果是最后一条期望应答消息到来，回复CU操作结果。如果应答消息超时，则直接写入失败结果。
6. CU收到允许录像下载的结果后，通知CU上层准备下载录像，然后向WebService接入线程发送请求，通知目的准备好下载。
7. 剩余流程同上。

## CUI与模块断链(外设)

CUI与某外设模块断链，中断与该外设模块的所有业务，并删除相应的事务以及事务管理表中的数据。当CU再次发起请求时则进行注册，如果注册不上则返回相应的错误码。