Libdash文档

## XML解析

Node XML中的每个节点

|  |  |
| --- | --- |
|  |  |

DOM解析 利用xml解析，通过dashmanager构造mpd的dom树

|  |  |
| --- | --- |
|  |  |

## 5． Meida Presentation

MPD element

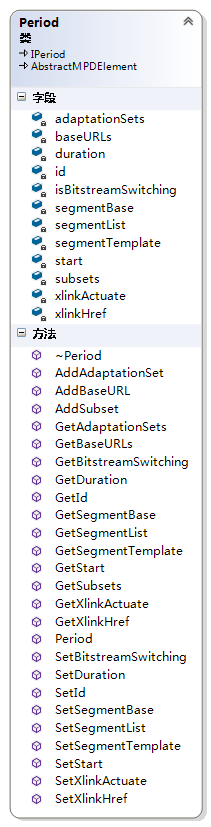
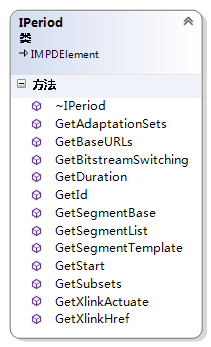
接口

GetAdditionalSubNodes用于获取标准中没有规定的XML节点信息

GetRawAttributes用于获取xml中的属性信息

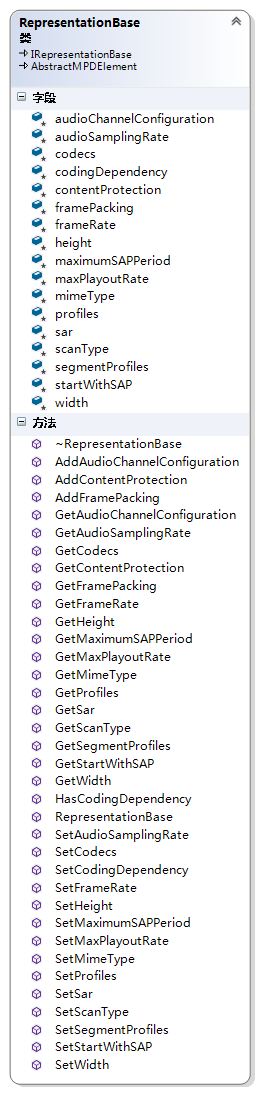
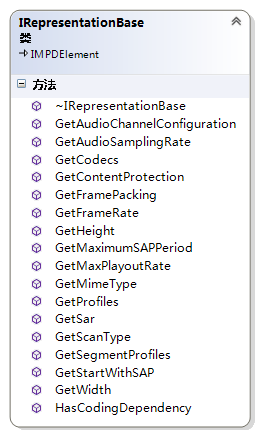
|  |  |
| --- | --- |
|  |  |

Period Element



注意：没有实现assetidentifier和Eventstream

Representation Base:代表了adaptation sets、representation以及sub-representation的共同属性，具体来说，在as、rep以及subrep中都是以属性的形式体现



注意：没有实现essentialproperty\supplementalproperty\inbandeventstream

Adaptation Sets 继承了Representation Base

|  |  |
| --- | --- |
|  |  |

Content component:一个adaptation set里面可以包含多个media content component，例如既有音频又有视频，它们的共同属性由content component或as定义，但不能重复定义。

|  |  |
| --- | --- |
|  |  |

Representation继承了Representation Base

|  |  |
| --- | --- |
|  |  |

Sub-representation: 继承了Representation Base

|  |  |
| --- | --- |
|  |  |

Subset

|  |  |
| --- | --- |
|  |  |

Segment

SegmentBase、SegmentList、SegmentTemplate不能同时出现在同一层，SegmentList和SegmentTemplate无论是否在同一层，都不能同时出现

SegmentInformation可以通过下面四种方法之一来表达，注意，必须是之一

1. 一个或多个SegmentList
2. 一个SegmentTemplate
3. 一个BaseUrl+一个SegmentBase
4. 一个BaseURL+SegmentList或SegmentTemplate

需要注意的是，对于SegmentList和SegmentTemplate来说，SegmentBase和MultiSegmentBase中的内容就是以普通的attribute形式体现的

SegmentBase：只适用于每个rep只有一个seg的情况

|  |  |
| --- | --- |
|  |  |

MultipleSegmentBase

|  |  |
| --- | --- |
|  |  |

URLType

|  |  |
| --- | --- |
|  |  |

SegmentList：包含一组SegmentURL，SegmentURL指向可以被下载的Segment

所以先来看一下和网络有关的内容

|  |  |  |
| --- | --- | --- |
| Chunk接口    DownloadableChunk接口    AbstractChunk | Connection接口 | 字节块    字节流    同步的字节流（支持多线程） |
| Downloadobserver接口      下载状态管理（支持多线程） |
| Segment接口 | BaseURL接口  BaseURL |  |
| SegmentURL接口    SegmentURL |
| SegmentList | 继承了MultipleSegmentBase | |

SegmentTemplate 继承了MultipleSegmentBase

先来看一下跟他紧密相关的Timeline

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | |  | |

Program Information

|  |  |
| --- | --- |
|  |  |

Descriptor 每种具体的descriptor都有专门的Uri，例如content protection、Role等等

|  |  |
| --- | --- |
|  |  |

Event 略，libdash中也没有实现

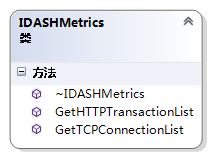
## Annex D DASH Metrics

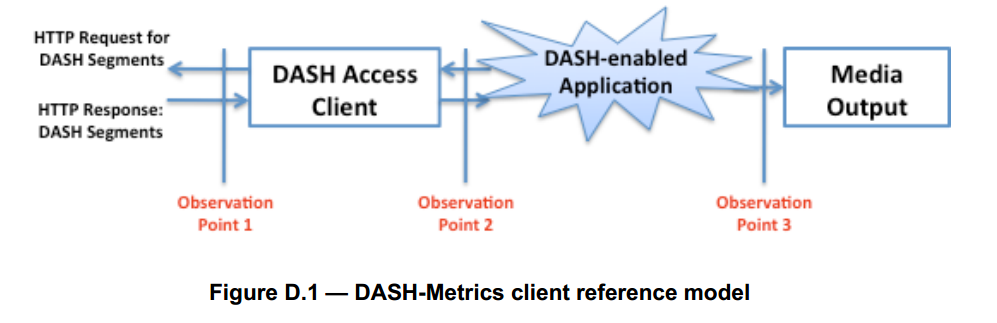
Dash QoE metric，跟所有的标准一样，只是给出了指标的定义，至于具体的指标报告方法则没有定义

首先定义了一个汇报时间range，然后才是metrics

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

但还有专门的DASH Metrics，对应Metrics中的metrics成员





观测点1：dash access client与网络之间的输入输出接口，这里输出的是HTTP请求，请求mpd、segmeny或partial segments。

在这个接口上观测到的内容有：

一组TCP连接，每一个连接由其目标IP，初始化、连接、关闭时间来标识；

一系列的HTTP请求，每一个请求由其传输时间、内容、所依赖的TCP连接来标识；

一系列的HTTP响应，由接收时间、response header的内容、以及response body中每个字节的接收时间决定（需要注意的是，response body的内容完全由request和response的header内容决定）

|  |  |
| --- | --- |
| TCPConnection： | HTTP Transcation： |
|  |  |
| HTTPTranscation中的throughput | IHTTPTranscation中的type |

观测点2：dash access client与dash-enabled application之间的接口，这里前者将编码的媒体数据传给后者，供下一步处理（解复用和解码），同时后者也可能向前者传递命令。

在这个接口上观测到的内容是编码的媒体samples，由媒体类型、解码时间、显示时间、representation id、delivery时间标识

观测点3：dash enabled application将解码后的数据传递给media output来向用户做最终显示，这其中的接口即为观测点3。

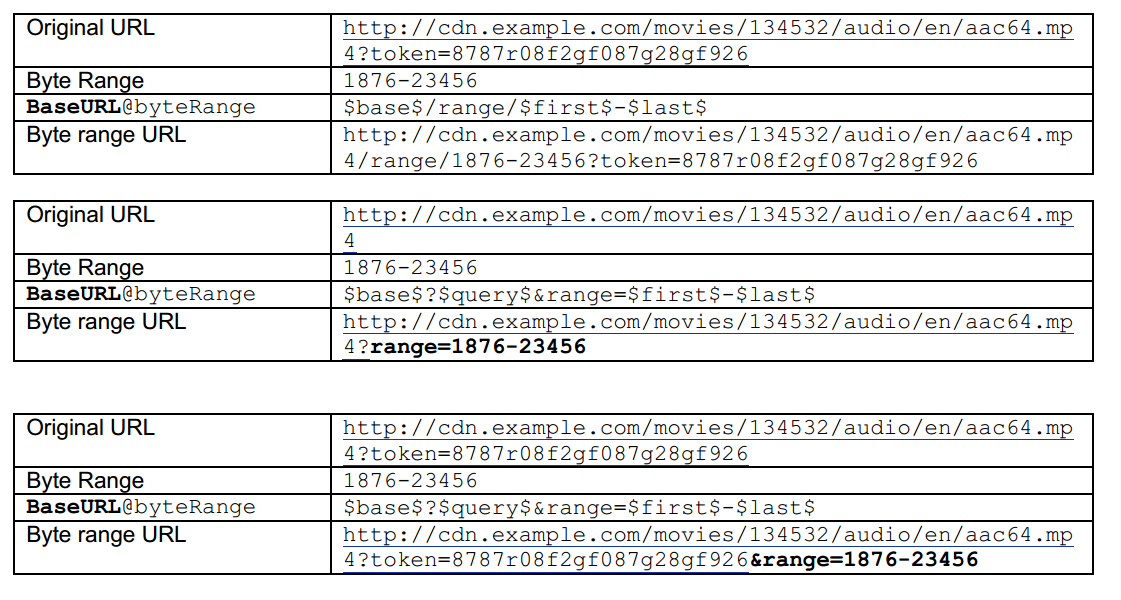
在这个接口上观测到的内容是解码的媒体samples，由媒体类型、预定显示时间戳（media time）、真实显示时间戳（real time）、representation id

需要注意的是：标准中的rep switch event\buffer level\play list没有在libdash中实现

## Annex E Byte Range Request

如果有不支持HTTP partial GET的环境，那么就要用**BaseURL**@byteRange来实现相同的功能，在IBaseUrl和BaseUrl类中有体现。

具体来说，如下所示



## Annex F 扩展DASH所支持的封装格式

只要这种封装格式具有如下特性：只需要很小的一部分数据，就能完成任何媒体片段的解码或播放。

其实大部分的流媒体封装格式应该都是可以的（有待考证）。

此外，如果这种封装格式能有一个index，标识了媒体中各个byte range的位置，从而可以不用下载整个文件，而只下载某一个位置，这样就更好了。

## Annex G MPD示例及用法

略

## Dash播放器

MediaObject及其buffer

|  |  |
| --- | --- |
|  |  |

渲染显示

|  |  |
| --- | --- |
|  |  |

解码 不支持音频，同步方法待商榷

|  |  |
| --- | --- |
|  |  |

播放流程

初始化SDLRender

用mpd地址初始化DASHReceiver{

创建一个DASHmanager，打开mpd地址

DOMparse根据mpd地址建立DOM模型

由DOM模型的根节点构建整个MPD以及所有相关的类，过程中给各个变量赋值

根据MPD和相关类初始化切换logic类

创建dobuffering线程

初始化libavdecoder，绑定SDLrender，设置帧率，初始化

主线程开始等待mediaobjectbuffer不为空的时刻，通过ffmpeg的读取回调函数来实现

while(this->mediaobjects.size() == 0 && !this->eos)

SleepConditionVariableCS(&this->full, &this->monitorMutex, INFINITE);

随后进行解码，当got——picture时，调用SDL进行显示

}

Dobuffering线程中{

根据切换logic得到目标mediaobject对象（有Segment和Representation构造）

开始下载，创建DownloadInternalConnection线程

while(this->state != COMPLETED && this->state != ABORTED)

SleepConditionVariableCS(&this->stateChanged, &this->stateLock, INFINITE);一直等到状态改变

}

DownloadInternalConnection线程中{

}