# 一、Mpen码铺码场景需求汇总

## 1、书的纸张类型：

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **类型** | **细分说明** | **备注** |
| 1 | 书的每一页尺寸是统一大小的 | A、每页都是全铺码 | 现在的铺码系统已实现 |
| B、只进行局部铺码，一页中不铺码的部分空白，也存在有些页完全不铺码的情况 | 增加subpage管理 – subpage 的定义在哪里？ |
| C、某些页不使用mpen码，某些页中部分使用mpen码，其它部分使用其它码（如页码部分）。 | 现在的铺码系统已实现 |
| 2 | 书中存在某些页大小尺寸不一样的情况 | 如书中存在折叠页，展开后会比当前页大很多 | 现在每一页的大小是独立定义的。所以只要出版系统能够生成合适的PageInfo定义就可以了。 |
| 3 | 书中存在某些页尺寸不规则的情况 | 如该页为椭圆形的 | 建议：将这些当成标准页大小处理，如比标准页大则当成几页来处理；超出区域的码将来裁掉就是了。-- 一页就是一页，不要当成几页。以包含这个形状的最小长方形定义这一页就好了。 |

## 2、铺码类型说明：

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **铺码方法** | **铺码说明** | **备注** |
| 4 | 以某个版次为基准铺码 | a、码值不变，部分铺码参数改变（点大小，偏移等） |  |
| b、铺码参数不变，但手动指定一些页重新铺，这些页用新的码段。 |  |
| c、铺码参数不变，但需要随机抽取一定比例的页进行重铺，这些页用新的码段。 |  |
| 5 | 书中某些部分需要铺成单一码的情况 | 如贴纸页，一个贴纸铺一个码，一页中有很多个贴纸 | 建议：先标区域，然后再区域内铺单一码。铺设码系统需要提供一些模式设置以方便在一页中可以一次分成多个规则矩形区域。-- 老孔说还是沿用热区方式，不用单一码值，管理方便。我觉得单一码值或者热区都可以。---那就统一采用热区方式 |

# 服务端设计：

## 请求参数设计：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 类型 | 说明 | 可选值 | 备注 |
| type | Enum | 铺码类型 | ORDER（顺序铺码，默认值），SINGLE（单一铺码） | 用于解决5中单一铺码场景 |
| bookId | String | 铺码书籍id，非空 |  | 所有页共享参数 – matrixGap，margin，matrixSize，dotDistanceInPixels，dotSize，dotShift – 这些都应该放在 page 对象定义里面。然后这里用一个 defaultPage 参数 |
| defaultPageParam | PageParam | 默认全局铺码参数 |  |
| pageNum | int | 页数 |  |
| version | String | 版次，非空唯一 |  |
| page | Array[page] | 铺码页信息数组 |  | 用于解决2中页大小不一致场景 |
| baseVersion | String | 基础版次 |  | 用于解决4中以某一版次为基准铺码 |

PageParam对象：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| matrixGap | int | 两个点阵键的填充黑点个数 | 0（默认值），1 |  |
| margin | Array[int] | 页边界像素（left,top,right,bottom） | 0（默认值） |  |
| matrixSize | int | 点阵大小 | 5（默认值）,6,7,8 |  |
| dotDistanceInPixels | int | 黑点间距像素 | 16（默认值） |  |
| dotSize | int | 黑点大小 | 2,3（默认值） |  |
| dotShift | int | 数据黑点便宜像素 | 2（默认值）,3 |  |

page对象：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 类型 | 说明 | 可选值 | 备注 |
| num | int | 页码 |  |  |
| widthMm | float | 页宽（毫米） |  |  |
| heightMm | float | 页高（毫米） |  |  |
| subPage | Array[SubPage] | 铺码块 | 默认值为全铺 | 用于解决1中部分区域铺码场景 |
| pageParam | PageParam | 局部铺码参数 |  |  |

SubPage对象：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 类型 | 说明 | 可选值 | 备注 |
| top | int | 上边距 |  |  |
| left | int | 左边距 |  |  |
| bottom | int | 下边距 |  |  |
| right | int | 右边距 |  |  |
| num | int | 编号 |  |  |

注意：

A、4-a铺码场景时，只允许修改点大小（dotSize）和点偏移（dotShift），其他参数均不传，使用基准版次参数；

B、4-b和4-c场景部分页重铺时，page数组中只存放需要重铺的页数据，其余页的铺码结果直接复用基准版次中的数据；

C、1-b铺码场景时，各个铺码块不可以有重合部分；

D、block为空时，为全页铺码。

## 参数样例：

1. 场景1-a、2：

{

"type": "ORDER",

"bookId": "063546ad251d4e828e79e56ec50cdce9",

"matrixGap": 0,

"margin": [

0,

0,

0,

0

],

"matrixSize": 5,

"dotDistanceInPixels": 16,

"dotSize": 2,

"dotShift": 3,

"pageNum": 5,

"version": "test001",

"page": [

{

"num": 1,

"widthMM": 210,

"heightMm": 297

},

{

"num": 2,

"widthMM": 220,

"heightMm": 230

},

{

"num": 3,

"widthMM": 230,

"heightMm": 300

},

{

"num": 4,

"widthMM": 240,

"heightMm": 310

},

{

"num": 5,

"widthMM": 250,

"heightMm": 320

}

]

}

1. 场景1-b：

{

"type": "ORDER",

"bookId": "063546ad251d4e828e79e56ec50cdce9",

"matrixGap": 0,

"margin": [

0,

0,

0,

0

],

"matrixSize": 5,

"dotDistanceInPixels": 16,

"dotSize": 2,

"dotShift": 3,

"pageNum": 5,

"version": "test001",

"page": [

{

"num": 1,

"widthMM": 210,

"heightMm": 297,

"block": [

{

"num": 1,

"top": 10,

"left": 10,

"bottom": 100,

"right": 100

}

]

},

{

"num": 2,

"widthMM": 210,

"heightMm": 297,

"block": [

{

"num": 1,

"top": 10,

"left": 10,

"bottom": 100,

"right": 100

}

]

},

{

"num": 3,

"widthMM": 210,

"heightMm": 297,

"block": [

{

"num": 1,

"top": 10,

"left": 10,

"bottom": 100,

"right": 100

}

]

},

{

"num": 4,

"widthMM": 210,

"heightMm": 297,

"block": [

{

"num": 1,

"top": 10,

"left": 10,

"bottom": 100,

"right": 100

}

]

},

{

"num": 5,

"widthMM": 210,

"heightMm": 297,

"block": [

{

"num": 1,

"top": 10,

"left": 10,

"bottom": 100,

"right": 100

}

]

}

]

}

1. 场景4：

{

"bookId": "063546ad251d4e828e79e56ec50cdce9",

"dotSize": 2,

"dotShift": 3,

"version": "test002",

"baseVersion": "test001",

"page": [

{

"num": 2,

"widthMM": 220,

"heightMm": 230

}

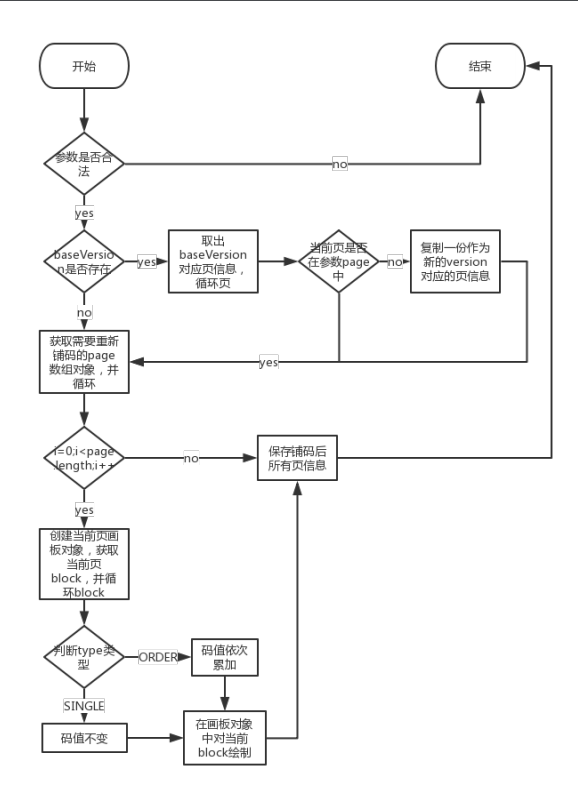
]

}

## 处理逻辑：

1. 服务端接收请求参数，判断参数合法性；
2. 判断baseVersion（基准版次）是否存在，如果存在则为场景4；
3. 取出baseVersion对应的所有铺码页信息，循环页；
4. 如果当前页为参数page中存在的页，则重新铺码，否则复制出一份作为新的version对应的铺码页信息；
5. 如果baseVersion不存在，则为场景1、2、5，循环page数组；
6. 判断block是否为空，如果为空，全页铺码；
7. 如果block不为空，循环block数组，对当前页不同block分别铺码；
8. 判断铺码类型type，ORDER时码值一次累加，SINGLE时根据block依次累加，同一个block内码值不变；
9. 铺码信息保存。

## 流程图：



## PageInfo

public class PageInfo {

public String id; // special UUID string, one without -

public String bookId; // special UUID string, one without -

public int pageNum;

public int xCodeNum; // x 轴点阵计数

public int yCodeNum; // y 轴点阵计数

public byte[] margin;

public MpCode startCode;

public byte matrixGap; // 两个点阵键的填充黑点个数，一般0，或者1

public byte matrixSize; // 5, 6, 7, 8

public byte dotDistanceInPixels = 16; // 黑点间距：缺省 16 像素点

public byte dotSize; // 黑点大小，一般是 2:2X2 或者 3:3X3

public byte dotShift; // 数据黑点偏移的pixel数量，一般是2

public SubPageInfo[] subPageInfo;

public ResourceVersion version;

}

Public class SubPageInfo{

public int xCodeNum; // x 轴点阵计数

public int yCodeNum; // y 轴点阵计数

public MpCode startCode;

public int[] margin;

public int num;

}