[Impeach 1](#_Toc499018809)

[Create 1](#_Toc499018810)

[~~Update~~ 2](#_Toc499018811)

[Search 3](#_Toc499018812)

[Delete 3](#_Toc499018813)

[Impeach\_action 3](#_Toc499018814)

[Create 4](#_Toc499018815)

[ImpeachComments 4](#_Toc499018816)

[Create 4](#_Toc499018817)

# Impeach

## Create

**每次状态变更，都会创建，以便记录。**

**普通用户和管理员采用不同的流程。**

**在用户点击（文档或者评论）举报后，在后台自动插入一条内容为空（其他字段根据实际情况获取）的新记录，status为”新建”。然后页面跳转至对应的页面，执行update操作。**

**url:** <http://domain/impeach/> POST

**输入参数：**

Values:{

RecordInfo:{

~~Title:’ 新举报’~~

Type: int, article|comment

impeachedArticleId| impeachedCommentId 被举报对象的id

impeachedUserId:

~~creatorId~~

}

method:0

}

**备注：**

1. 普通用户

**步骤：**

1. 检查用户类型是否正确

**2.** 检查是否有session（用户是否登录）

3. 用户是否有权限创建举报

4. 根据articleId或者commentId，获得其authodId，加入到记录

5. 检查是否有同类别的处罚，且没有过期。如果有，且没有过期，报错通知管理员（用户已经有此处罚存在）

5. 创建新的记录

## ~~Update~~

~~用户新建举报后对举报进行完善，只能对某些字段进行修改（~~**~~以下字段都是可选~~**~~）~~

**~~url:~~** [~~http://domain/jb/~~](http://domain/jb/) ~~POST~~

**~~输入参数：~~**

~~Values:{~~

~~RecordInfo:{~~

~~Title~~

~~Content~~

~~impeachImages~~

~~impeachAttachments~~

~~impeachComments~~

~~status 如果是创建者，只能~~**~~是~~**~~”新建”或者”提交”~~

~~}~~

~~method:1~~

~~}~~

~~Admin只能修改状态（现阶段，不考虑存储处理状态）~~

~~Values:{~~

~~RecordInfo:{~~

~~impeachComments~~

~~status 如果是管理员，只能是~~**~~非~~**~~”新建”|”提交”~~

~~}~~

~~method:1~~

~~}~~

**~~步骤：~~**

~~1. 检查参数~~

~~2. 从session获取当前用的user\_id~~

~~3. 从adminUser中，根据step2中的id检索对应的记录~~

~~4. 判断当前用户是否有权处理（1. Type＝0（admin）。 2. Priority中包含impeach）~~

~~5. 更新~~

~~5.1 如果是普通用户更新~~

~~5.1.1 如果加入、删除图片或者附件，首先在impeachImage/impeachAttachment表中插入记录或者更改dDate字段。~~

~~5.2.2 更新impeachImage/impeachAttachment字段~~

~~5.2 如果是admin更新（更改）~~

~~5.3 如果管理员或者用户，增加了comment，获得对应的id，直接更改记录（而不是通过API）~~

~~2. 对字段content，使用DOMPurify进行XSS检测~~

~~3. 根据sessionId，读取当前用户id，和impeachId指定的记录中的authorId比较~~

~~4. 如果status存在，根据sessionId读取用户Id，判断status是否为新建/提交~~

## Search

读取所有举报（只包括title、content、status等，不包括之后的处理（comment））

**url:** <http://domain/jb/> POST

**输入参数：**

Values:{

searchParams:{

creatorId:[{value:’adf’}]

dealer:[{value:’adsf’}]

impeachedArticleId| impeachedCommentId

impeachedUserId: 某人有多少文档评论被举报

status 处于某种状态的举报

},

method:2

}

## Delete

举报如果没有admin借如果，可以被物理删除（impeach和action一起，因为无需care未被处理的impeach）。否则无法删除

# Impeach\_action

只有create操作，每次发生改变，都要记录在案

Admin和普通用户使用同样逻辑，主代码在express中完成，copy到express\_admin中

## Create

**url:** [http://domain/impeach\_state/](http://domain/jbfp/) POST 举报分配

**输入参数：**

Values:{

RecordInfo:{

ImpeachId: objectId,

State:

owenrId:

ownerColl:

dealerId:

dealerColl:

}

method:0

}

**备注：**

1. dealerId从session中获得

2. 每次state变化，插入新记录

**步骤：**

1. 检查输入参数是否正确（FK是否存在，enum的值是否重复）
2. 对应的impeach是否已经删除，如果已经删除，则无法create任何state
3. 如果变更state的是普通用户，判断impeach是否为用户所创
4. impeach的初始state必须为NEW
5. 如果最后一个state为DONE或者reject，则无法继续输入state
6. 输入的state不能为NEW（NEW是创建impeach时自动创建的）
7. 如果是admin，根据其权限，判断是否允许输入当前的state
8. 根据前一个state，判断当前state是否valid（例如，submit之后只能为accept而不能为DONE）
9. 如果是REVOKE（user+submit的条件下），state自动变成NEW；如果是DONE或者REJECT，owner变成当前操作用户
10. 创建新的记录

# ImpeachComments

## Create

后台默认创建一个新的

**1. 初始化文档举报(**server自动生成数据。 如果要插入图片/附件，是通过专门的rest API完成**)**

**url:** http://domain/impeach\_comment POST

Values:{

“recordInfo”:{

impeachId:{value:”asdf” },

}

Method:UPDATE

}

**步骤：**

**普通用户**

1. 用户类型检测（普通用户）
2. FkValue（impeachId）是否存在
3. 权限检查（impeachId是否为当前用户）
4. Impeach是否已经submit过且被admin处理过
5. 是否有当前用户创建的未完成的impeach comment，有则直接返回此comment id
6. Content进行XSS检测
7. 添加内部值，检测
8. Db 操作

**2. 用户完成文档举报**（对于用户，是新建操作，添加内容/图片/附件，更改状态等）

**url:** <http://domain/impeach_comment> POST

**传入参数：**(类似创建CR，先创建记录)

Values:{

“recInfo”:{

Content:{value:”asdf” },

Type:{value:enum.commit},

}

“recId”: impeach\_objectId

method:UPDATE

}

**步骤：**

1. 检查数据结构和值

2. 对字段content，使用DOMPurify进行XSS检测

3. 根据sessionId，读取当前用户id，和impeachId指定的记录中的authorId比较

4. 如果status存在，根据sessionId读取用户Id，判断status是否为新建/提交

**3. 客服更新举报状态：**（对于举报，客服只能修改dealer和status字段）

**url:** <http://127.0.0.1/jb/update> POST

**传入参数：**

Values:{

“recInfo”:{

dealer:{value:objectId },

status:{value:enum.xxxx},

}

~~“curColl”:”impeach”,~~

“recId”: impeach\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 如果dealer存在

2.1 dealer是否可用（可能使用了另外一个coll，例如admin而不是普通的user）

2.2 dealer是否有权更新此文档（是否有权处理此举报）

3. status必须存在，判断是否为有效的操作（接受/分配/处理中/处理完）

**4. 用户举报插入图片**（impeachImages）

**url:** <http://domian/jb/upload> Image POST

**传入参数：**

Values:{

~~“curColl”:”impeach”,~~

“recId”: impeach\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 根据sessionId获得用户id，从impeach根据recId获得记录，判断用户id是否等于记录的authorId（判断用户是否有权对当前举报插入图片）

3. 判断上传文化是否符合（名称/类型/大小，其他限制）

4. 获得上传文件的信息，server选取合适的path，生成{name:”aaaa”, hashName:”bbbb”, pathId:”asdf”, size:1234, authorId:”adsfas”}，写入coll: impeachImages。

**5. 用户举报插入附件**（impeachAttachments）

**url:** <http://domian/jb/upload> Attachment POST

**传入参数：**

Values:{

~~“curColl”:”impeach”,~~

“recId”: impeach\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 根据sessionId获得用户id，从impeach根据recId获得记录，判断用户id是否等于记录的authorId（判断用户是否有权对当前举报插入附件）

3. 判断上传文化是否符合（名称/类型/大小，其他限制）

4. 获得上传文件的信息，server选取合适的path，生成{name:”aaaa”, hashName:”bbbb”, pathId:”asdf”, size:1234, authorId:”adsfas”}，写入coll: impeachAttachments。

**6. 用户（客服） 初始化举报对话（**impeachComments，**）**

**url:** <http://domian>(127.0.0.1)/jbdh /create POST

**传入参数：**

Values:{

“recInfo”:{

impeachId:{value:objectId },

}

~~“curColl”:”impeachComments”,~~

}

**步骤：**

1. 判断传入数据格式和值

2. 根据sessionId读取用户id，然后根据impeachId读取对应记录的authorId（dealerId），判断此举报是否由当前用户创建（用户只能对自己创建的举报进行处理）

3. 将空内容和step 2中读取的用户id填入记录，并存入db。 { content:””, authorId: userId}

**7. 用户（客服）完成举报对话内容（**impeachComments**）**

**url:** <http://domian>(127.0.0.1)/jbdh /update POST

**传入参数：**

Values:{

“recInfo”:{

content:{value:objectId },

}

~~“curColl”:”impeachComments”,~~

“recId”: impeachComments\_objectId,

}

**步骤：**

1. 判断传入数据格式和值

2. 根据sessionId读取用户id，然后根据impeachCommentsId读取impeachComments中对应记录的authorId，判断此举报是否由当前用户（客服）创建（用户（客服）只能对自己创建的举报进行处理）

3. 对上传的content字段，用DOMPurify进行XSS检测

4. 上传的内容更新到coll中

**8. 用户（客服）为impeachComment添加图片：（**impeachImages**）**

**url:** <http://domain/jbdhtp/uploadImage> POST

**传入参数：**

Values:{

~~“curColl”:”impeachImages”,~~

“recId”: impeachComments\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 根据sessionId获得用户id，从**impeachComments**根据recId获得记录，判断用户id是否等于记录的authorId（dealerId）（判断用户（客服）是否有权对当前**举报对话**插入附件）

3. 判断上传文化是否符合（名称/类型/大小，其他限制）

4. 获得上传文件的信息，server选取合适的path，生成{name:”aaaa”, hashName:”bbbb”, pathId:”asdf”, size:1234, authorId:”adsfas”}，写入coll: impeachImages。

**9. 用户（客服）为impeachComment添加附件：（**impeachAttachments**）**

**url:** http://domain/jbdhtp/uploadIAttachments POST

**传入参数：**

Values:{

~~“curColl”:”impeachImages”,~~

“recId”: impeachComments\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 根据sessionId获得用户id，从**impeachComments**根据recId获得记录，判断用户id是否等于记录的authorId(dealerId)（判断用户是否有权对当前**举报对话**插入附件）

3. 判断上传文化是否符合（名称/类型/大小，其他限制）

4. 获得上传文件的信息，server选取合适的path，生成{name:”aaaa”, hashName:”bbbb”, pathId:”asdf”, size:1234, authorId:”adsfas”}，写入coll: impeachAttachments。