## 1、配置分析

网上下载下来的apache ftpserver，所有监听器ftplet都是空函数，在这种情况下，设计中转，让原始的ftpserver能支持更多的功能。首先需要一个enable字段决定是否开启中转配置，不开启的情况下，就是apache ftpserver原本的功能。然后尽量把所有的工作都交给接口来处理，某一些接口无法处理的，需要在ftplet中增加代码逻辑。

然后开始分析现有命令的处理

如删除文件，由dele命令自己处理，删除主目录下的文件，然后读取配置，是否开启中转配置，配置中是否有DELE这项，如果有，遍历它的值的每一项，当前ftpserver程序所处的监听器是否和配置中的where相同，相同就准备参数，请求对应url，最后根据配置的returnId来决定后续如何处理该命令。上述的某一步为否的都会结束当前的监听器，ftpserver进行下一步处理。

配置文件名为transfer.json，存放在res/conf下。配置格式为：

{

"enable": true,

"DELE": [{

"url": "<http://192.168.5.159/test/delete>",

"type": "POST",

"where": "afterCommand",

"sendLastResult": false

}]

........

}

说明：

"enable": true //开关中转配置

"DELE": [] //key为命令名，参考[附](#_附：)，全大写，value可以有多项的配置

"url": "请求地址" //url是其他处理程序的执行地址

"type": "POST" //请求url的参数传递类型，一般为POST，可能有些接口是GET的可以改为GET

每个url都会提交的参数有：

**username**：当前登录的用户名

**ip**：客户端的ip，当前使用哪个网络与ftp通讯就是该网络适配器的ipv4

**absolutePath**：当前目录绝对路径，如用户当前所在/a目录，该用户的主目录是C:\test，则absolutePath为C:\test\a

**commandLine**：完整命令，字符串格式：命令名 参数

"where": "监听器名" //每个命令都有对应的监听器，参考[附](#_附：)

"sendLastResult": Boolean //是否把上一次请求的返回值作为额外的参数传递到url中

接口的返回值要求格式统一：

{

"error": 0 //0代码成功，其他代表有错误，要查对应的错误消息

"returnId": 0 //0代表DEFAULT，1代表NO\_FTPLET，2代表SKIP，3代表DISCONNECT

"replyCode": 200 //ftp状态码，参考<https://blog.csdn.net/jinhao2003/article/details/2141378>

"replyMsg": "" //返回给客户端的提示语句

"filePath": "" //只在下载会使用，用于重定向下载的文件

"moduleId": 1 //模块id，一一对应，智能加解密模块为1

"params": "" //用于作为下一个请求（sendLastResult为true）的额外参数

}

注意：如果要阻止ftpserver的默认行为，returnId要为2，并且提供replyCode用来回复客户端已发送的命令请求。这里returnId如果设置不合适，可能会导致ftp出现问题，例如连接断开，程序报错等。

**DEFAULT**：这个返回值表示将调用下一个ftplet（监听事件）方法。如果没有其他可用的ftplet, FtpServer将处理请求。

**NO\_FTPLET**：此返回值指示不会调用其他ftplet方法，但FtpServer将继续处理此请求。

**SKIP**：它表示服务器将跳过所有内容。此请求将不作进一步处理。

**DISCONNECT**：它表示服务器将跳过并断开客户机。不会处理来自同一客户端的其他请求。

returnId为2的时候，不一定需要使用replyCode回复客户端；但需要使用replyCode回复客户端时，returnId一定要为2。这里就不增加字段来决定是否需要回复客户端，由程序来决定，如果请求是在beforeCommand或onXXXStart中，returnId为2就要回复客户端，如果请求是在afterCommand或onXXXEnd中，returnId为2就不回复客户端。

目前只考虑一个命令在一个监听器中只允许发送一个请求，也就是暂不考虑一个监听器处理多个事件，例如上传，可能想分为多个请求分别是拷贝文件、加解密、删除文件等，现在不考虑这种情况，在一个请求中完成所有的内容，后续再考虑细分的情况。

## 2、实现分析

### 2.1、通过上面的配置说明，我们增加一个处理，如修改密码：

"SETPASS": [{

"url": "xx",

"type": "POST",

"where": "beforeCommand",

"sendLastResult": false

}]

说明：

(1)url和type没有什么问题，根据远程接口进行配置。

(2)where参考[附](#_附)，得到自定义的不存在的命令要处理的地方可以是beforeCommand或afterCommand，但因为发送的是命令不是原生的命令，ftpserver会当成是无效命令，即ftpserver执行beforeCommand --> NotSupportedCommand --> afterCommand，在中间NotSupportedCommand会返回502 Command XXX not implemented.给客户端，所以我们需要更改默认逻辑，不让ftpserver执行NotSupportedCommand，即我们要在NotSupportedCommand之前的beforeCommand中进行处理并响应客户端并返回SKIP，所以where填写beforeCommand。

(3)sendLastResult大多时候都是false，除了某些特殊的接口，执行完一个url还要把结果作为参数传到另一个url继续处理，而考虑到修改密码接口，只需要原来的用户名和命令的参数作为新密码就行，不需要其他额外的参数，所以填写false。

伪代码：

**public** FtpletResult beforeCommand(FtpSession session, FtpRequest request){

*String command = request.getCommand().toUpperCase();*

if (command 在 配置中) {

value = 获取它的value数组

for (遍历value) {

if (value[i].where == beforeCommand) {

请求value[i].url，post/get 固定的参数

if (返回值error == 0){

if(returnId == 2){

session.write(**new** DefaultFtpReply(replyCode, **replyMessage**));

}

}

根据returnId值执行如 **return** FtpletResult.***SKIP***;

}

}

} else{

**return** FtpletResult.***DEFAULT***;

}

}

当接口处理成功返回要求：

{

"error": 0,

"returnId": 2,

"replyCode": 200,

"replyMsg": "new password set successfully",

"filePath": "",

"moduleId": 1,

"params": ""

}

接口处理失败返回示例，可以有多种：

{

"error": 1,

"returnId": 0, //ftp执行默认处理NotSupportedCommand，告诉客户端502

"replyCode": 0,

"replyMsg": "",

"filePath": "",

"moduleId": 1,

"params": {}

}

或者

{

"error": 1,

"returnId": 2, //结束ftplet

"replyCode": 502,

"replyMsg": "fail", //回复客户端502 fail

"filePath": "",

"moduleId": 1,

"params": {}

}

关键是失败后是否希望ftpserver处理原本的默认处理，如果不是就returnId为2并提供replyCode

### 2.2、上传例子

先说明一下现在的上传功能，ftpserver先执行beforeCommand，onUploadStart，都是空函数，然后执行STOR，把文件上传到了主目录，然后执行afterCommand空函数，执行onUploadEnd，把源文件拷贝到明密文目录并对应地加解密，没有可以提取出来供其他程序来处理的代码，除非远程接口为本地的程序，那么可以把拷贝文件、加解密、删除文件这些内容提取出来，然后调用接口，由其他程序处理，暂时先考虑一个接口完成所有事情，后续再考虑把事情细分为多个接口，多个接口的时候可能就会需要sendLastResult为true了。

"STOR": [{

"url": "xx",

"type": "POST",

"where": "onUploadEnd",

"sendLastResult": false

}]

说明：

(1)url和type没有什么问题，根据远程接口进行配置，

(2)where参考[附](#_附：)，得到STOR命令可能会触发的监听器有：beforeCommand、onUploadStart、afterCommand、onUploadEnd，开始分析，上传时，我们是希望文件上传到主目录，即默认的命令逻辑，根据默认的逻辑看它是成功还是失败再作后续的处理（加解密），所以where可以选择afterCommand或onUploadEnd，因为接口可能需要判断文件是否上传成功，所以这里where填写afterCommand，额外提交reply参数给接口用于判断文件是否上传到主目录成功了。

(3)sendLastResult仍然是false，只要知道absolutePath和commandLine就能知道源文件的路径

伪代码：

这里我们假设远程接口是本地程序，能进行文件操作

**public** FtpletResult afterCommand(FtpSession session, FtpRequest request, FtpReply reply){

*String command = request.getCommand().toUpperCase();*

if (command 在 配置中) {

value = 获取它的value数组

for (遍历value) {

if (value[i].where == onUploadEnd) {

请求value[i].url，post/get 固定的参数，并增加reply参数

//这里远程接口会根据absolutePath和commandLine得到源文件路径，根据root得到明密文目录地址，然后就拷贝文件并对应的加解密

if (返回值error == 0){

//不需要回复客户端，在STOR就已经回复了

//if(根据返回值的result.reply是否为true){

// session.write(**new** DefaultFtpReply(replyCode, **replyMessage**));

//}

}

根据return值执行如 **return** FtpletResult.***SKIP***;

}

}

}else{

**return** FtpletResult.***DEFAULT***;

}

}

### 2.3、下载例子

"RETR": [{

"url": "xx",

"type": "POST",

"where": "onDownloadStart",

"sendLastResult": false

}]

说明：

(1)url和type没有什么问题，根据远程接口进行配置，

(2)where参考[附](#_附：)，得到RETR命令可能会触发的监听器有：beforeCommand、onDownloadStart、afterCommand、onDownloadEnd。开始分析，下载时，我们希望ftpserver能清楚下载明文还是密文，即不能执行RETR类的方法，所以where能选的只有beforeCommand和onDownloadStart，优先onDownloadStart，所以where的值填onDownloadStart。

(3)sendLastResult仍然是false

伪代码：

**public** FtpletResult onDownloadStart(FtpSession session, FtpRequest request)  
 **throws** FtpException, IOException {  
 *String command = request.getCommand().toUpperCase();*

if (command 在 配置中) {  
 value = 获取它的value数组

for (遍历value) {

if (value[i].where == onUploadEnd) {

请求value[i].url，post/get 固定的参数

//这里远程接口分析该用户应该下载明文还是密文，然后返回文件路径filePath

if (返回值error == 0){

**long** skipLen = session.getFileOffset();

返回的filePath，来构造文件对象

然后一系列的判断文件对象的正确性

打开数据通道传输文件

//这里就不判断reply了，函数中固定，在什么情况下回复什么状态码

//if(根据返回值的result.returnId是否为2){

// session.write(**new** DefaultFtpReply(replyCode, **replyMessage**));

//}

}

根据return值执行如 **return** FtpletResult.***SKIP***;

}

}

}

**else** {  
 **return super**.onDownloadStart(session, request);  
 }  
  
}

## 附

onConnect：客户端连接通知方法。这是FtpServer将调用的第一个方法。如果返回跳过、IP限制检查、连接限制检查将不执行，服务器将不发送欢迎消息(220)。

onDisconnect：客户端断开通知方法。这是FtpServer将调用的最后一个方法。无论返回什么，客户机连接都将关闭。

onConnect和onDisconnect是没有对应的命令的，不能获取当前命令，即transfer.json配置中的key那部分需要制定

如果想在onConnect中执行请求，key就为"CONNECT"，where为"onConnect"

如果想在onDisconnect中执行请求，key就为"DISCONNECT"，where为"onDisconnect"

beforeCommand：在服务器调用命令之前调用。

afterCommand：在服务器调用命令后调用。如果where填写afterCommand，那么会额外提交reply参数，该参数是字符串，格式为“200 ok”（状态码 消息）。供接口通过状态码判断命令函数的执行结果。

onLogin: 客户端登录通知方法。这将在用户身份验证之后调用。在本例中，FtpServer已经发送了login OK(230)响应。这是在FTP PASS请求期间调用的。当返回值为FtpletResult.DISCONNECT时FTP会话将断开连接。

onDeleteStart：在文件删除之前，将调用此方法。在此之前FtpServer不会检查任何类似文件存在或权限。这在DELE FTP命令中调用。如果返回SKIP，该方法应该发送一些响应(比如250、450、550)。在这种情况下，服务器将跳过命令处理，ftplet必须发送适当的响应值。

onDeleteEnd：此方法将在文件删除后调用，无论是否成功。在本例中，FtpServer已经发送了回复消息。这在DELE FTP命令中调用。

onUploadStart：此方法将在文件上载之前调用。我们可以从请求中获得数据输入流。这将在权限检查之前调用。这在STOR命令期间调用。如果方法返回SKIP，它必须在处理前后发送响应。例如，在打开数据输入流之前，该方法必须使用响应代码150通知客户机。类似地，在数据传输之后，方法必须使用响应代码226通知客户机。如果出现错误，该方法应该发送不同的响应码如450，425，426，551。

onUploadEnd：将调用此通知方法，以指示文件传输成功，服务器已发送了响应。如果出现任何错误，将不调用此方法。这在STOR命令中调用。

onDownloadStart：这是在RETR命令期间调用的文件下载请求通知方法。这将在下载文件之前调用。类似地，可以从请求中获得数据输出流。这将在文件存在检查和权限检查之前调用。如果方法返回SKIP，它必须在处理前后发送响应。例如，在打开数据输出流之前，该方法必须使用响应代码150通知客户机。类似地，在数据传输之后，方法必须使用响应代码226通知客户机。如果出现错误，该方法应该发送不同的响应代码，比如450，425，426,551。

onDownloadEnd：此通知方法将被调用，以指示文件传输成功，并且服务器已在RETR命令中发送了响应。

onRmdirStart：在目录删除之前，此方法将在RMD命令期间调用。在此之前FtpServer不会检查任何类似目录存在或权限。如果方法返回SKIP，它必须向客户端发送适当的响应代码，比如250、450和550。

onRmdirEnd：此方法将在调用RMD命令之后调用。在这情况下，FtpServer已经发送了回复消息。

onMkdirStart：在创建目录之前，将在MKD命令期间调用此方法。在此之前FtpServer不会检查任何类似目录存在或权限。如果它返回SKIP，它必须向客户端发送适当的响应代码，比如250,550。

onMkdirEnd：如果MKD命令中目录创建成功，将调用此方法。在这情况下，FtpServer已经发送了回复消息。

onAppendStart：这是在APPE命令中调用的文件追加请求通知方法。我们可以从请求中获得数据输入流。这将在权限检查之前调用。如果方法返回SKIP，它必须在处理前后发送响应。例如，在打开数据输入流之前，该方法必须使用响应代码150通知客户机。类似地，在数据传输之后，方法必须使用响应代码226通知客户机。如果出现错误，该方法应该发送不同的响应代码，如450、425、426、551。

onAppendEnd：这是在APPE命令中调用的文件追加成功通知方法。

onUploadUniqueStart：这是在STOU命令中调用的惟一文件创建请求通知方法。我们可以从请求中获得数据输入流。这将在权限检查之前调用。如果方法返回SKIP，它必须在处理前后发送响应。例如，在打开数据输入流之前，该方法必须使用响应代码150通知客户机。类似地，在数据传输之后，方法必须使用响应代码226通知客户机。如果出现错误，该方法应该发送不同的响应代码，比如450、425、426、551。

onUploadUniqueEnd：这是在STOU命令中调用的惟一文件创建成功通知方法。将调用此通知方法来指示服务器已发送了响应。

onRenameStart：这是在RNTO命令中调用的文件重命名启动通知方法。这将在文件存在或权限检查之前调用。如果返回SKIP，则必须发送适当的响应代码，比如503,55,250。

onRenameEnd：这是在RNFR命令中调用的文件重命名成功通知方法。这将在文件存在或权限检查之前调用。此通知方法将在调用RNTO命令后调用，并且服务器已发送了应答消息。

onsite：这是SITE命令启动通知方法。它提供了一个实现自定义SITE命令的机会。如果此方法返回跳过或断开连接，则不会执行现有SITE命令。

当同时具有afterCommand和onXXXEnd，where配置优先选择onXXXEnd。同时具有beforeCommand和onXXXStart，where优先选择onXXXStart。

注意：如果一个中转配置处理完后要告诉客户端结果的，需要回复客户端的，例如下载，那么该请求要有reply的返回值及replyCode和replyMessage，return为2，并且where只能选择beforeCommand和onXXXStart，否则ftpserver就会默认执行并返回了消息给客户端，当你在后续的其它监听器处理后就无法再通知客户端并会发生异常。

如果远程接口并不改变原来命令的逻辑，即不改变与客户端的通讯，作为一些额外处理，那么比较好的处理地方是afterCommand和onXXXEnd。

另外需要注意：

当在afterCommand和onXXXEnd中进行选择时，如果不需要判断命名函数执行是否成功，where可以选择两个之一，如果是要分析命令是否成功，那么where只能选择afterCommand，因为afterCommand中的reply参数可以知道命令函数的处理结果。在afterCommand中会额外发送reply参数给接口，接口可以判断命令执行是成功还是失败。

|  |  |  |  |
| --- | --- | --- | --- |
| 命令名 | 解析 | 可能触发的监听器（顺序执行） | 可以额外提交的参数 |
| ABOR | 中止前面的FTP服务命令和任何相关的数据传输。 | beforeCommand  afterCommand | 暂无 |
| ACCT <account> | 系统特权账号 | beforeCommand  afterCommand | 暂无 |
| APPE <filename> | 将数据附加到远程主机上文件的末尾。如果文件不存在，就创建它。此命令之前必须有PORT或PASV命令。 | beforeCommand  onAppendStart  afterCommand  onAppendEnd | 暂无 |
| CDUP <dir path> | 切换到父目录。 | beforeCommand  afterCommand | 暂无 |
| CWD <dir path> | 更改工作目录。如果没有指定目录名，则假定根目录(/)。 | beforeCommand  afterCommand | 暂无 |
| DELE <filename> | 删除由提供的路径指定的文件。 | beforeCommand  onDeleteStart  afterCommand  onDeleteEnd | 暂无 |
| EPRT | EPRT命令允许指定数据连接的扩展地址。扩展地址必须由网络协议以及网络和传输地址组成。EPRT的格式是EPRT |||| | beforeCommand  afterCommand | 暂无 |
| FEAT | 显示特性列表。 | beforeCommand  afterCommand | 暂无 |
| HELP <command> | 显示帮助信息。 | beforeCommand  afterCommand | 暂无 |
| LANG | 一个新的命令“LANG”被添加到FTP命令集中，以允许server-FTP进程确定用哪种语言表示服务器问候和命令响应的文本部分。 | beforeCommand  afterCommand | 暂无 |
| LIST <name> | 此命令将一个列表从服务器发送到被动DTP。如果路径名指定了一个目录或其他文件组，服务器应该传输指定目录中的文件列表。如果路径名指定了一个文件，那么服务器应该发送该文件的当前信息。null参数表示用户当前工作目录或默认目录。数据传输通过数据连接进行。此命令之前必须有PORT或PASV命令。 | beforeCommand  afterCommand | 暂无 |
| MDTM | 返回文件被修改的日期和时间。 | beforeCommand  afterCommand | 暂无 |
| MKD <directory> | 此命令将路径名中指定的目录创建为目录(如果路径名是绝对的)或当前工作目录的子目录(如果路径名是相对的)。 | beforeCommand  onMkdirStart  afterCommand  onMkdirEnd | 暂无 |
| MLSD | 使列表从服务器发送到被动DTP | beforeCommand  afterCommand | 暂无 |
| MODE <mode> | 设置数据传输模式。有效值是S和Z。 | beforeCommand  afterCommand | 暂无 |
| NLST <directory> | 此命令将目录清单从服务器发送到用户站点。路径名应指定目录或其他特定于系统的文件组描述符;null参数表示当前目录。服务器将返回一个文件名称流，没有其他信息。 | beforeCommand  afterCommand | 暂无 |
| NOOP | 没有操作。 | beforeCommand  afterCommand | 暂无 |
| OPTS | 使服务器为指定的命令使用可选特性。 | beforeCommand  afterCommand | 暂无 |
| PASS <password> | 参数字段是指定用户密码的Telnet字符串。此命令必须紧跟在USER命令之前。 | beforeCommand  afterCommand  onLogin | 暂无 |
| PASV | 此命令请求server-dtp侦听数据端口(这不是它的默认数据端口)，并等待连接，而不是在收到传输命令时启动连接。此命令的响应包括此服务器正在监听的主机和端口地址。 | beforeCommand  afterCommand | 暂无 |
| PBSZ | 保护缓冲区大小。 | beforeCommand  afterCommand | 暂无 |
| PORT <address> | 参数是数据连接中使用的数据端口的主机端口规范。用户和服务器数据端口都有默认值，在正常情况下不需要此命令及其响应。如果使用此命令，则参数是32位internet主机地址和16位TCP端口地址的连接。该地址信息被分成8位字段，每个字段的值都以十进制数字(以字符串表示)传输。字段之间用逗号分隔。一个端口命令将是:  port h1, h2, h3, h4, p1, p2 | beforeCommand  afterCommand | 暂无 |
| PROT | 数据通道保护级别。支持的级别值是C、P。 | beforeCommand  afterCommand | 暂无 |
| PWD | 此命令将在应答中返回当前工作目录的名称。 | beforeCommand  afterCommand | 暂无 |
| QUIT | 关闭连接。 | beforeCommand  afterCommand | 暂无 |
| REIN | 刷新用户，而不影响正在进行的传输 | beforeCommand  afterCommand | 暂无 |
| REST <offset> | 参数字段表示要重新启动文件传输的服务器标记。此命令不会导致文件传输，而是跳过文件到指定的数据检查点。此命令之后应立即执行相应的FTP服务命令，该命令将导致文件传输恢复。 | beforeCommand  afterCommand | 暂无 |
| RETR <filename> | 此命令导致server-dtp将路径名中指定的文件副本传输到数据连接另一端的server-或user-dtp。服务器站点上文件的状态和内容不受影响。 | beforeCommand  onDownloadStart  afterCommand  onDownloadEnd | 暂无 |
| RMD <directory> | 此命令将路径名中指定的目录作为目录(如果路径名是绝对的)或当前工作目录的子目录(如果路径名是相对的)删除。 | beforeCommand  onRmdirStart  afterCommand  onRmdirEnd | 暂无 |
| RNFR <old path> | 此命令指定要重命名的文件的旧路径名。这个命令必须立即后跟一个指定新文件路径名的RNTO命令。 | beforeCommand  afterCommand | 暂无 |
| RNTO <new path> | 此命令指定前一个RNFR命令中指定的文件的新路径名。这两个命令一起导致文件重命名。 | beforeCommand  onRenameStart  afterCommand  onRenameEnd | 不额外提交RNFR的参数，如果想知道old path，请增加RNFR配置 |
| SITE <params> | 处理Apache FTP服务器特定的定制命令。请参阅SITE命令页。 | beforeCommand  onSite | 暂无 |
| SIZE | 返回文件大小(以字节为单位)。 | beforeCommand  afterCommand | 暂无 |
| STAT <directory> | 此命令应以应答的形式通过控制连接发送状态响应。 | beforeCommand  afterCommand | 暂无 |
| STOR <filename> | 此命令使server-dtp接受通过数据连接传输的数据，并将数据作为文件存储在服务器站点上。如果路径名中指定的文件存在于服务器站点上，则其内容应由正在传输的数据替换。如果路径名中指定的文件不存在，则在服务器站点上创建一个新文件。 | beforeCommand  onUploadStart  afterCommand  onUploadEnd | 暂无 |
| STOU <filename> | 这个命令的行为类似于STOR，只是生成的文件将在当前目录中以该目录惟一的名称创建。250传输开始响应必须包含生成的名称。 | beforeCommand  onUploadUniqueStart  afterCommand  onUploadUniqueEnd | 暂无 |
| STRU <type> | 参数是指定文件结构的单个Telnet字符代码。允许的参数是F。 | beforeCommand  afterCommand | 暂无 |
| SYST | 此命令用于查明服务器上操作系统的类型。 | beforeCommand  afterCommand | 暂无 |
| TYPE <data type> | 参数指定表示类型。允许的类型是A和I。 | beforeCommand  afterCommand | 暂无 |
| USER <username> | 参数字段是一个标识用户的Telnet字符串。用户标识是服务器访问其文件系统所需的标识。这个命令通常是用户在建立控制连接后发送的第一个命令。 | beforeCommand  afterCommand  注意：USER在命令作为第一个命令，重写了该命令来实现一些必要的内容，所以不要配置为跳过USER命令，即不要在where中填beforeCommand并同时返回SKIP | 暂无 |
| 其他自定义的不存在的命令 |  | beforeCommand  afterCommand  尽量在beforeCommand中进行处理，然后返回SKIP，如果不返回SKIP，就会执行NotSupportedCommand并返回502 Command XXX not implemented.然后执行afterCommand | 暂无 |