Amendment history 修改历史记录

版本号	修改说明	修改批准人	修改人	日期	签收人
101	创建文档		沈胜文	2008/4/30	

Table of Content 目录

1.Introduction 简介

1.1. Objective 编写目的

本文档是根据法电项目 MuFTAD 项目要求，按照《doc00_SoftProtocol_0.1.0_RC1.doc》协议，在《MuFTAD 需求说明书》的基础上进行详细设计和开发的说明文档；

本文档尽可能详细地说明法电定义的各操作码实现的每一个细节，但是在由于其他原因导致设计不合理或是考虑不充分的地方，需要开发人员在实施中予以完善，但是所作修改需要讨论决定，并且修改本文档；

最终设计以代码为准；

1.2. Background 背景

见《MuFTAD 需求说明书》中说明

1.3. Terms & Abbreviation 术语 & 缩写解释

见《MuFTAD 需求说明书》中说明

1.4. Reference Material 参考资料

详细参考本《MuFTAD 需求说明书》及所需要模块的详细设计文档

2. Rules 规则

2.1. Name Rules 命名规则

本软件是针对法电需要所设计的一款专有的应用软件，并不可使用于其他项目，该程序中，所有的函数均以 MuFTAD_开头，因此，在整个程序中，所有的函数名形如：MuFTAD_XXX()；

2.2. Illuminate 说明

针对每个程序，都必须注明其开发目的，开发者，开发时间等等。以下字段必须被包含于程序的开头部分。

```
/*****************************************************************
*
*                    ====Microunit Techonogy Co,.LTD.====
* File Name:
*
*         XMLParse.c
*
* Description:
*
*         This file get the file name, which store the XML Contents.
*        The Functions Open the file, parse it, then return the Information we need.
*
* Revision History:
*
*         10-3-2008 ver1.0
*
*Author:
*
*         ssw    (fzqing@gmail.com)
*
*                    ***PROTECTED BY COPYRIGHT***
*****************************************************************/
```

2.3. Note Rules 注释规则

程序中的各个函数均需要明确注释其功能，并能简要描述其实现，及注意点。特别应该注意的是：在描述时，应该详细包括对锁，输入和输出进行详细说明；

可参考格式；

```
/*************************************************************
*Description:
*         This Function is Parse the XML, return the Informations to caller use the
*         Value pointer;
*Input:
*         filename: the file name , which stored the XML contents
*Output:
*         Pointer: which is a pointer, point to the buffer stored the Information
*LOCK:
```

```
*           NONE
*Modify:
*           ssw (fzqing@gmail.com    10-3-2008)
**************************************************************/
```

2.4. Libaries 库

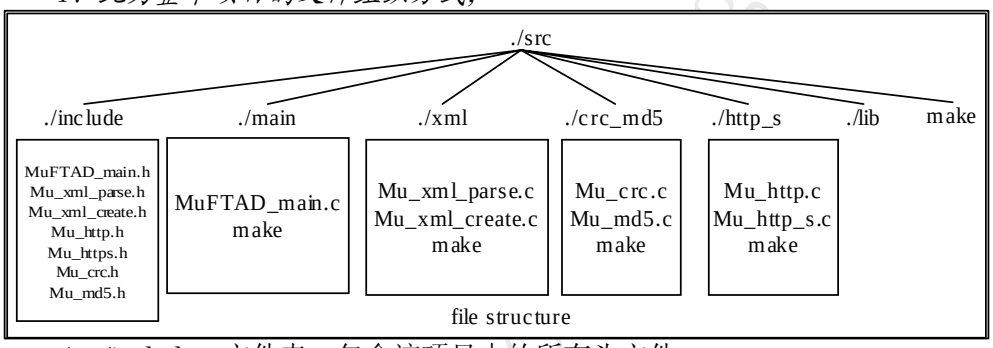
待开发的软件所使用的模块均由微元提供，各模块均以库的形式对本软件提供支持；

各模块的详细说明参考相应模块的详细设计文档；

2.5. File Structure 文件组织

[注意]:

1、此为整个项目的文件组织方式;



./src/include：文件夹，包含该项目中的所有头文件；

./src/main：文件夹，包含所有按法电《do3c00_SoftProtocol_0.1.0_RC1》流程所开发的程序；

./src/xml：文件夹，包含项目中所需要的 xml 处理库函数源代码；

./src/crc_md5：文件夹，包含项目中所需要的校验函数源代码，包括 CRC 和 MD5校验代码；

./src/http_s：文件夹，包含项目中所需要的与服务器交互的方式，包括 HTTP(s) GET、POST 方式；

./src/lib：文件夹，用于存储编译所生成的 xml、http 和 https、crc/md5库。软件编译连接时使用该文件夹下的库；

./src/make：文件，总的编译入口；

[注意]:

1、各对应文件夹下的源文件按需要添加，但是所作修改必须对 makefile 文件作相应的修改，以正确编译;

3. Structure Of Routines 程序结构

3.1. Overview 总述

按照《MuFTAD 需求说明书》中所描述的，程序必须具备如下功能：

初始化

在初始化阶段，可分为二个子阶段：变量初始化和设备初始化；

变量初始化阶段，应该初始化本程序及各模块应用时所需要的内存空间和变量名等作用；

设备初始化阶段，要求使 Device 更新至最新固件，解析设备和服务器信息；

预登录

访问服务器（GET），获得登录方式；

登录

从服务器获得临时的帐号信息；

若有最新的固件，则下载至设备内；

状态查询

获得订阅节目信息；

将节目信息按2XX 和4xx 分别存储于相应的结构体中；

DIR 下载

下载4xx 类型的节目；

FDDF 表查询

取得各节目的 FDDF XML 文件地址；

FDDF 查询

取得 FDDF XML 文件，并解析其内容于内存空间；

下载

下载 XML 文件所描述的各个分段；

状态更新

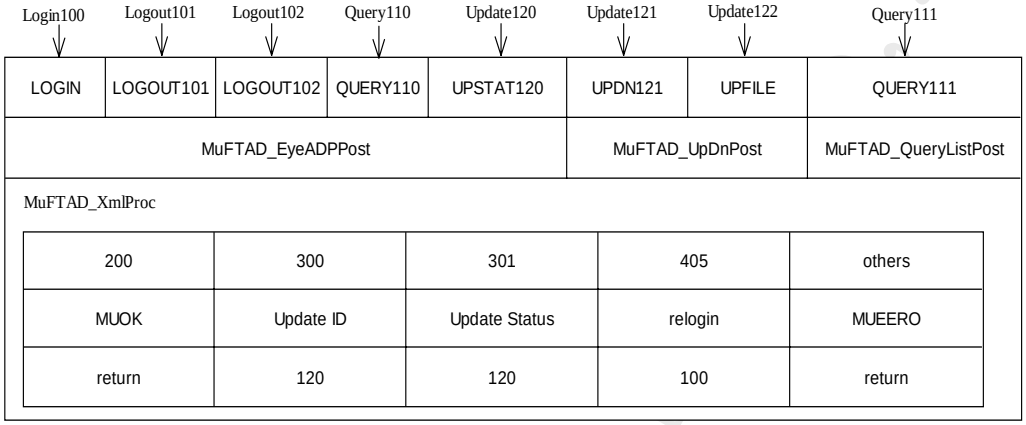
针对每一类节目，下载完成后，向服务器发起请求，以完成状态更新；

登出

登出服务器

3.2. Routines List 函数列表

3.3. Structure of Routines 函数结构



4. Global Description 全局描述

4.1. Global Type 全局类型

4.1.1. Macro 宏定义

```
#define LOGIN 100
#define LOGOUT101 101
#define LOGOUT102 102
#define QUERY110 110
#define QUERY111 111
#define UPSTAT120 120
#define UPDN121 121
#define UPFILE122 122

#define Mu_Close(fd)\
do{\
    if(fd > 0){\
```

```
            close(fd);\
            fd = 0;\
        }\
    }while(0)
```

参考《MuFTAD 需求说明书》及各详细设计文档

4.1.2. Structure 结构体

参考《MuFTAD 需求说明书》及各详细设计文档

4.2. Global Error 全局错误码

参考《MuFTAD 需求说明书》及各详细设计文档

5. Routines Details 函数细节

5.1. MuFTAD_DataInit

5.1.1. Name 函数名称

int MuFTAD_DataInit(void)

5.1.2. Description 函数描述

初始化项目中各模块所使用的内存空间和全局变量

5.1.3. Input 输入

无

5.1.4. Output 输出

函数的运行状态值，包括：
1、无错误，以 **MUOK** 返回；

2、初始化各结构错误，以 **MUNBUF** 返回；
2、其他错误，以 **MUEERO** 返回；

5.1.5. Arithmetic 算法

初始化内容包括：
1、FDDF status 头初始化：
Mu_FddfStatusHeadCreate
[注意]:
1、在函数 Mu_FddfStatusHeadCreate 中已包括 Mu_FddfStatusHeadInit 函数来对所申请的内存空间进行初始化;

2、DIR Status 头初始化：
Mu_DirStatusHeadCreate
在函数 Mu_DirStatusHeadCreate 中已包括 Mu_DirStatusHeadInit 函数来对所申请的内存空间进行初始化；

3、Server Information 初始化
Mu_ServiceInfoCreate
在函数 Mu_ServiceInfoCreate 中已包括 Mu_ServiceInfoInit 函数来对所申请的内存空间进行初始化；

4、FDDF List Header 初始化
Mu_FddfListHeaderCreate

5、FDDF File 结点初始化
Mu_FddfFileCreate
Mu_SegmentHeadCreate

6、EyewearADP 结构初始化
Mu_EyewearADPCreate

7、Update Info 结构初始化
Mu_UpdateInfoCreate

8、Device Info 结构初始化
Mu_DeviceInfoCreate

9、Resume 结构初始化
Mu_ResumeInfoInit

10、File Info 结构初始化

Mu_FileInfoCreate

11、网络 IO 初始化
Mu_InitNetIO

5.1.6. Process 处理流程

略

5.1.7. Pseudo-code 伪代码

```
    FddfStatusHeadPtr MuFddfStatHead;
    DirStatusHeadPtr MuDirStatHead;
    ServiceInfoPtr MuServiceInfo;
    FddfListHeadPtr MuFddfListHead;
    FddfFilePtr MuFddfFile;
    SegmentHeadPtr MuSegmentHead;
    EyewearADPPtr MuEyewearADPrcv;      //preserver the signal of current
    EyewearADPPtr MuEyewearADPsnd;      //preserver the signal
    DeviceInfoPtr MuDeviceInfo;
    ResumeInfo MuResume[10];
    FileInfoPtr MuFileInfo;
    MuNetIOPtr MuIO;

    int rcvfd;
    int budfd;
    int statfd;
    int listfd;
```

[注意]:

1、MuEyewearADPsnd 是一个全局值，其存储最新的 EyewearADP 值;

2、本函数并没有完全包括所有的需要初始化的结构和类型，在有新的结构需要添加到其中时，请添加初始化函数至该函数中;

```
int MuFTAD_DataInit(void)
{
        if((Mu_FddfStatusHeadCreate(&MuFddfStatHead) < 0)
            ||(Mu_DirStatusHeadCreate(&MuDirStatHead) < 0)
            ||(Mu_ServiceInfoCreate(&MuServiceInfo) < 0)
            ||(Mu_FddfListHeadCreate(&MuFddfListHead) < 0)
            ||(Mu_FddfFileCreate(&MuFddfFile) < 0)
            ||(Mu_SegmentHeadCreate(&MuSegmentHead) < 0)
            ||(Mu_EyewearADPCreate(&MuEyewearADPrcv) < 0)
```

批注 [ssw1]: 存储从服务器接收的解析后的信令

```
        ||(Mu_EyewearADPCreate(&MuEyewearADPsnd) < 0)
        ||(Mu_DeviceInfoCreate(&MuDeviceInfo) < 0)
        ||(Mu_FileInfoCreate(&MuFileInfo) < 0)){

        fprintf(stdout, "can't got the memory for work!\n");
        return MUNBUF;
    }

    /*init the array of the resumeifo*/
    for(int i = 0; i < 10; i ++)
        if(Mu_ResumeInfoInit(MuResume, i) < 0)
            return MUEERO;

    /*init the IO operations */
    if(Mu_InitNetIO(&MuIO) < 0){
        fprintf(stdout, "can't init the net work IO\n");
        return MUEERO;
    }

    //build the fd for to receive the contents
    if((rcvfd = open("rcv", O_RDWR | O_CREATE)) < 0)
        ||(sndfd = open("build", O_RDWR | O_CREATE)) < 0)
        ||(statfd = open("stat", O_RDWR | O_CREATE)) < 0)
        ||(listfd = open("list", O_RDWR | O_CREATE)) < 0)){
        fprintf(stdout, "can't create the file!\n");
        return MUEERO;
    }

    //when close the file or power down,
    //we need to delete the file
    unlink("rcv");
    unlink("build");
    unlink("stat");
    unlink("list");

    return MUOK;
}
```

批注 [ssw2]: 存储有最新的信令信息，在向服务器发起请求时，保证各信令始终是最新的

批注 [ssw3]: 初始化保存续传信息的结构，在登录后，设备会解析信息至该结构体内

批注 [ssw4]: 初始化项目中用到的各种网络 IO 操作函数接口

批注 [ssw5]: 存储从服务器返回的一连串 FDDF list 地址信息，若一次性全部解析，不用该 listfd 文件句柄

批注 [ssw6]: 当掉电或是程序退出时，必须自动删除以上文件，以避免产生大量的碎片文件

5.2. MuFTAD_DeviceInit

5.2.1. Name 函数名称

int MuFTAD_DeviceInit(const char *serverinfo, const char *deviceinfo)

5.2.2. Description 函数描述

解析 Device 端存储的 XML 文件，获取必要的参数，包括设备 ID 号和硬件版本号，服务器地址和请求 API 等信息；

检测磁盘余量；

5.2.3. Function 功能

初始化设备、服务器相关信息

5.2.4. Capability 性能

略

5.2.5. Input 输入

serverinfo：服务器信息描述 XML 文件名；
deviceinfo：设备信息描述 XML 文件名；

5.2.6. Output 输出

函数运行状态码，包括：
1、无错误，以 **MUOK** 返回；
2、空间不足，以 **MUENSP** 返回；
3、文件系统出错，以 **MUEFSY** 返回；
4、解析 XML 文件出错，以 **MUNXML** 返回；

5.2.7. Arithmetic 算法

略

5.2.8. Process 处理流程

略

5.2.9. Pseudo-code 伪代码

```
int MuFTAD_DeviceInit(const char *serverinfo, const char *deviceinfo)
{
    struct statfd fs_info;
    int space_rate;

    //check the space of the disk
    if(statfd("/", &fd_info) != 0){
        frpintf(stdout, "unable to stat the file system for downloading\n");
        return MUEFSY;
    }

    if((space_rate =
        (int)(100 *((double)fs_info.f_bavail / (double)fs_info.f_blocks))) < 2))
        return MUENSP;
    else
        printf(stdout, "the space avail is %d percent\n", space_rate);

    //parse the server information XML
    int fd = 0;
    if((fd = open(serverinfo, O_RDONLY)) < 0
        ||(Mu_Parse(PARSE_SERVICE_INFO, fd, NULL, &MuServiceInfo) <
0)){
        fprintf(stdout, "can't open the serverinfo XML\n");
        if(fd > 0)
            Mu_Close(fd);
        return MUNXML;
    }
    Mu_Close(fd);

    //parse the device information
    fd = 0;
    if((fd = open(deviceinfo, O_RDONLY)) < 0
        ||(Mu_Parse(PARSE_DEVICE_INFO, fd, NULL, &MuDeviceInfo) < 0)){
        fprintf(stdout, "can't open the deviceinfo XML\n");
        if(fd > 0)
            Mu_Close(fd);
        return MUNXML;
    }

    Mu_Close(fd);
    return MUOK;
```

}

5.2.10. Interface 函数接口

函数在实现时，需要用到模块中的接口函数：
Mu_Parse
用于解析 XML 文件，在本函数中，只用于解析 Service Information XML 和 Device information XML 文件；

5.2.11. Malloc 存储分配

本函数在调用解析函数时，会申请相应的内存空间来存储所解析内容；

5.3. MuFTAD_UpdateFireware

5.3.1. Name 函数名称

int MuFTAD_UpdateFireware(const char *path, const char *xml)

5.3.2. Description 函数描述

该函数由调用者在执行下载功能前，检测固件的更新，该函数并不包含至 MuFTAD 库中；

该函数由应该移至调用者代码中，在执行下载前调用；

5.3.3. Function 功能

更新固件

5.3.4. Capability 性能

略

5.3.5. Input 输入

path：下载过程中，用于存储更新的固件的文件夹名称；
[注意]:
1、只包括 path 路径名，不包括/字符;

xml：记录下载更新状态的 XML 文件名；

5.3.6. Output 输出

函数运行状态码，包括：
1、无错误，以 **MUOK** 返回；
2、无更新记录 XML 文件，以 **MUNUPD** 返回；
3、调用命令出错，以 **MUECOM** 返回；

5.3.7. Arithmetic 算法

更新的固件包括 Service Information XML 和程序库，所有从服务器端下载来的更新固件均存放于./updatefile 文件夹中，update_SRV XML 记录所有下载固件的属性；

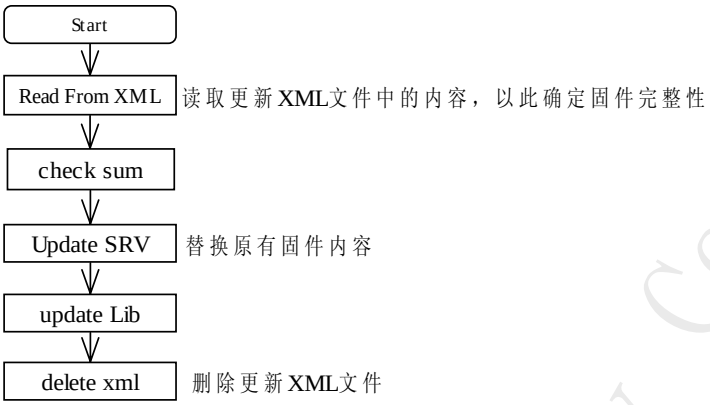
update_SRV XML 文件在被创建时间上，先于开始下载固件。因此，如果没 update_SRV 不正确或是不存在，一定不会存在下载的待更新的固件；

update_SRV XML 文件中包括有待更新固件的 MD5值，更新（替换原有固件）前需要校验固件的完整性；

更新流程见《MuFTAD 需求说明书》；

GUI 或是触发程序必须首先调用该代码，以对各内容进行更新；

5.3.8. Process 处理流程



5.3.9. Pseudo-code 伪代码

```
UpdateInfoPtr MuUpdateInfo;
#define SRVNAME “serviceinfo.xml”
#define FIRNAME “lib”
int MuFTAD_UpdateFireware(const char *path, const char *xml)
{
    char temp[250];
    snprintf(temp, strlen(path)+strlen(xml)+1, “%s%s\0”, path, xml);

    //exist the update info
    if(access(temp, F_OK|R_OK) == -1){
        fprintf(stdout, “have no fireware to update!\n”);
        return MUNUPD;
    }

    Mu_UpdateInfoCreate(&MuUpdateInfo);

    //if the update xml is exist, then read it , parse it’s contents to structure!
    int fd;
    if((fd = open(temp, RD_ONLY)) < 0
        ||Mu_Parse(PARSE_UPDATE_INFO, fd, NULL, &MuUpdateInfo) < 0){
        fprintf(stdout, “can’t open the xml! shall clear the directory!\n”);
        sprintf(temp, “rm –f ./%s/*\0”, path);
        system(temp);
        return MUNUPD;
    }
```

```
    Mu_Close(fd);
    sprintf(temp, “%s%s\0”, path, SRVNAME);
    if(MuUpdateInfo->srvsvrinfo->version
        && MuUpdateInfo->srvsvrinfo->checksum
        && (fd = open(temp, RD_ONLY)) > 0
        &&    (Mu_Md5file(fd,    MuUpdateInfo->srvsvrinfo->checksum)    ==
MUOK){
        sprintf(temp, “cp ./%s/%s ./ServerInfo.xml\0”, path, SRVNAME);
        if(system(temp) == 127)
            return MUECOM;
    }

    Mu_Close(fd);
    sprintf(temp, “%s%s\0”, path, FIRNAME);
    if(MuUpdateInfo->srvfiresf->firewareversion
        && MuUpdateInfo->srvfiresf->checksum
        && (fd = open(temp, RD_ONLY)) > 0
        && (Mu_Md5file(fd, MuUpdateInfo->srvfiresf->checksum) == MUOK){
        sprintf(temp, “cp ./%s/%s ./lib\0”, path, FIRNAME);
        if( system(temp) == 127)
            return MUECOM;
    }

    Mu_Close(fd);
    sprintf(temp, “rm –rf ./%s/*\0”, path);
    system(temp);
    return MUOK;
}
```

5.3.10. Interface 接口

该函数的实现需要 XML 解析库与 MD5校验库的相关函数进行运算；

5.3.11. Malloc 存储分配

函数中使用数组存储相关的文件名和指令；

5.3.12. Restrict 限制

略

5.3.13. Test 测试

略

5.3.14. Unsolved 未解决情况

法电协议及早期约定并没有对下载的更新固件名进行说明，本函数中对更新的固件名采用宏定义的形式约定；

函数在实现时，将所有待更新的固件，进行校验后，替换原有文件，拷贝完成后清空 updatefile 文件夹内的所有内容；

对于指令运行出错，调用者应该停止运行下载功能；

5.4. MuFTAD_PreLogin

5.4.1. Name 函数名称

int MuFTAD_PreLogin(void)

5.4.2. Description 函数描述

请求服务器，从服务器处取得登录方式；

服务器返回的消息，被存储于 rcvfd 文件中，待接收完成后，解析其内容至 MuEyewearADPrcv，从其中可以获得 EncryptionMethod 和 Challenge 二个字段；

5.4.3. Function 功能

取得登录方式

5.4.4. Capability 性能

略

5.4.5. Input 输入

无

5.4.6. Output 输出

函数运行状态码，包括：
1、无错，以 **MUOK** 返回；
2、MUECON 和 MUOK 外错误，以 **MUEERO** 返回；
3、解析 XML 错误，以解析函数错误码返回；

5.4.7. Arithmetic 算法

在 MuServiceInfo 中，有三个指针指向服务器地址，一个是链表首地址；另一个是链表的尾地址；第三个是当前正在使用的服务器地址；

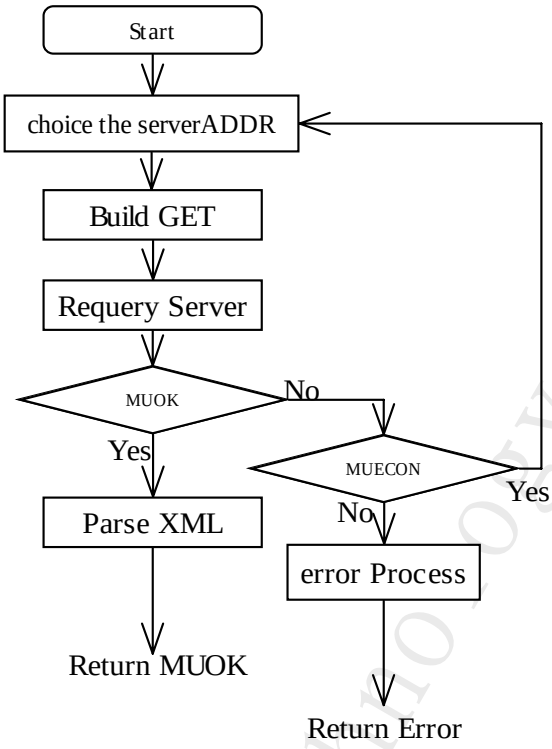
在使用过程中，需要对当前链表地址进行更新，其地址从首地址开始向尾地址增长，并且循环工作；

[Note]:

1、Currently, <EncryptionMethod> will be 0 and Device_ID will send to server in clear text and <Chanllge> is ignored.

解析存储于 rcvfd 文件句柄中内容，全部信息解析于 MuEyewearADPrcv 中；

5.4.8. Process 处理流程



5.4.9. Pseudo-code 伪代码

```
Mu_DnStatPtr MuDnStat;
Mu_DnStatInit(Mu_DnStatPtr);
int MuFTAD_PreLogin(void)
{
    char *addr;
    int querylen;
    char *query;
    int ret = MUOK;
    ServerHeadPtr server = MuServiceInfo->server;

    if(server->cur == NULL)
        return MUEERO;

again:
    if(NULL == (addr = server->cur->addr))
```

```
        return MUEERO;

    //build the query addr
    querylen = snprintf(NULL, 0,
        "https://%s/%s",addr, MuServiceInfo->apilist->prelogin);

    query = malloc(querylen + 1);
    snprintf(query, querylen,
            "https://%s/%s\0", addr, MuServiceInfo->apilist->prelogin);

    //query server!
    Mu_DnStatInit(MuDnStat);
    MuDnStat->type = MU_GET;
    MuDnStat->fd = rcvfd;
    ret = Mu_HttpPlus(query, MU_QUERY, MuDnStat);
    free(query);
    query = NULL;

    //if connect error, we must reconnect to the server!
    if(ret == MUECON){
        if(server->cur->next != NULL)
            server->cur = server->cur->next;
        else
            server->cur = server->head;
        goto again;
    }

    if(ret != MUOK){
        MuFTAD_ErrorProc(ret);
        return MUERRO;
    }

    //parse the Information to MuEyewearADPrcv
    if((ret = Mu_Parse(PARSE_EYEWEARE_ADP,    rcvfd,
                        NULL, &MuEyewearADPrcv) )< 0){
        //do error
        return ret;
    }

    return MUOK;
}
```

5.4.10. Interface 接口

函数在实现时，需要用到的函数有：
Mu_DnStatInit
Mu_HttpPlus
MuFTAD_Error
Mu_Parse

5.4.11. Malloc 存储分配

申请内存空间存储请求信令

5.5. MuFTAD_EyeADPPost

5.5.1. Name 函数名称

int MuFTAD_EyeADPPost(int opccode)

5.5.2. Description 函数描述

函数只处理**仅仅包含 EyewearADP 的 XML 信令**，函数构建该信令，并发送到服务器端；

设备将服务器返回的信令存储在对于于 opccode 的临时文件夹中，以便于解析；

在本函数中，对服务器返回的信令，不解析也不进行更进一步的处理；

5.5.3. Input 输入

opccode：信令操作码，在项目中，可以使用的值包括
1、LOGIN
2、LOGOUT101
3、LOGOUT102
4、QUERY110
5、UPSTAT120
[注意]:

1、详细请参考本文档4.1.1.中说明;

5.5.6. Output 输出

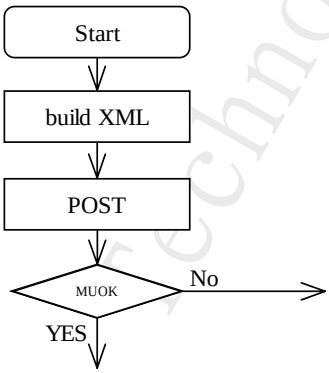
函数运行状态码，包括：
1、无错，以 **MUOK** 返回；
2、出错，以 **MUEERO** 返回，在这种错误情况下，函数无法继续处理；

5.5.7. Arithmetic 算法

函数需要根据 MuEyewearADPsnd 中的字段，构建 POST 信令；

以 POST 方式发送到服务器，服务器解析相关信令后，返回 XML，不解析该 XML 内容；
向服务器发送信令由函数 MuFTAD_QueryPost 函数完成；

5.5.8. Process 处理流程



5.5.9. Pseudo-code 伪代码

```
int MuFTAD_EyeADPPost(int opccode)
{
    int ret = MUOK;
    MuEyewearADPsnd->OPC = opccode;

    //Build the XML
    if(Mu_BuildXML(BUILD_SIGNAL_ADP, sndfd,
```

批注 [ssw7]: Mu_BuildXML 在创建 XML 时，必须重新定位到文件首，然后创建节点；写完后，必须写入 EOF；

```
                                MuEyewearADPsnd, NULL) < 0
        || MuFTAD_QueryPost(opccode) < 0 ){

        MuFTAD_ErrorProc(ret); //process the error ,which us defined
        return MUEERO;
    }

    return MUOK;
}
```

批注 [ssw8]: 该函数仅仅对各种错误码输出打印信息

5.5.10. Interface 接口

函数在实现的时候需要用到的函数有：
Mu_buildXML：创建请求信令；
Mu_QueryPost：将信令发送到服务器；

5.6. MuFTAD_QueryListPost

5.6.1. Name 函数名称

int MuFTAD_QueryListPost(int type)

5.6.2. Description 函数描述

在向服务器请求 FDDF List 时，需要能够提交二种不同类型的 payload，分别为902和20X；

函数按照请求类型，创建不同类型的请求信令，并发送到服务器端；

从服务器端接收服务器返回的信令，但不处理该返回信令；

5.6.3. Input 输入

type：表明待传送到服务器的 XML 信令类型，以 payload 名称区别；
type 包括二种类型的值：
1、902；
2、201-204;

5.6.4. Output 输出

函数运行状态码，包括：
1、无错，以 **MUOK** 返回；
2、无类型，以 **MUETYP** 返回；
3、向服务器请求出错，以 **MUEERO** 返回；

5.6.5. Arithmetic 算法

以 type 类型确定待创建的 XML 信令，用来存储该信令的文件为 sndfd；

创建信令无错误返回后，将该信令发送到服务器；

5.6.6. Process 处理流程

略

5.6.7. Pseudo-code 伪代码

参考 MuFTAD_EyeADPPost

```
int MuFTAD_QueryListPost(int type)
{
    int ret = MUOK;
    MuEyewearADPsnd->OPC = 111;

    //build Post XML
    if(type == 902)
        Mu_BuildXML(BUILD_QUERY_902, sndfd,
                    MuEyewearADPsnd, MuDBIInfo);
    else if(type == 20X) //use macro to check the value of    201-204
        Mu_BuildXML(BUILD_QUERY_20X, sndfd,
                    MuEyewearADPsnd, MuDBUInfo);
    else
        return MUETYP;

    //post to server
    if(MuFTAD_QueryPost(QUERY111) < 0 ){
        MuFTAD_ErrorProc(ret); //process the error ,which us defined
        return MUEERO;
    }
```

批注 [ssw9]: 用宏实现

return **MUOK**;
}

5.6.8. Interface 函数接口

函数在实现上述功能时，必须调用以下函数接口；
1、Mu_BuildXML
2、MuFTAD_QueryPost

5.7. MuFTAD_UpDnPost

5.7.1. Name 函数名称

int MuFTAD_UpDnPost(FddfStatNodePtr Ptr, int opccode, int type)

5.7.2. Description 函数描述

函数用于向服务器发送更新信令，待发送的信令包括 payload 字段；

5.7.3. Input 输入

opccode：操作码，包括 UPDN121和 UPFILE 二种类型；
type：标明待创建的信令的类型，包括902和903；

5.7.4. Output 输出

函数运行的返回状态值，包括：
1、无错，以 **MUOK** 返回；
2、类型值错，或是操作码错，以 **MUETYP** 返回；
3、其他错误，以 **MUEERO** 返回；

5.7.5. Arithmetic 算法

同5.6.5.中说明

5.7.6. Process 处理流程

略

5.7.7. Pseudo-code 伪代码

参考 MuFTAD_EyeADPPost
使用前，MuDBIIno, MuDBUDirt, MuDBURate 均需要被初始化

```
int MuFTAD_UpDnPost(FddfStatNodePtr Ptr, int opccode， int type)
{
    int ret = MUOK;
    MuEyewearADPsnd->OPC = opccode;

    //Init the XML
    if(opccode == UPDN121){
        if(type = 902){
            MuFTAD_CopyFddfToDBII(Ptr, MuDBIInfo);
            Mu_BuildXML(BUILD_SIGUP_DN902,
                        sndfd, MuEyewearADPsnd, MuDBIInfo);
        }
        else if(type = 903){
            MuFTAD_CopyFddfToDBID(Ptr, MuDBIDirt);
            Mu_BuildXML(BUILD_SIGUP_DN903,
                        sndfd, MuEyewearADPsnd, MuDBIDirt);
        }
        else
            return MUETYP;
    }else if(opccode == UPFILE){
        MuFTAD_CopyFddfToDBUR(Ptr, MuDBURate);
        Mu_BuildXML(BUILD_SIGUP_FILE,
                        sndfd, MuEyewearADPsnd, MuDBURate);
    }
    else
        return MUETYP;

    if(MuFTAD_QueryPost(opccode) < 0 ){
        MuFTAD_ErrorProc(ret); //process the error ,which us defined
        return MUEERO;
    }

    return MUOK;
}
```

批注 [ssw10]: 进行相应的结构间的转换

5.8. MuFTAD_XmlProc

5.8.1. Name 函数名称

int MuFTAD_XmlProc(int statcode)

5.8.2. Description 函数描述

该函数负责对信令中的所有信令操作码进行处理；

批注 [ssw11]: 在该项目的第一开发阶段，有一些操作码是不用处理的，直接以错误返回；

函数根据解析出的 XML 信令信息，对各种操作码做出适当的处理，具体的处理函数由各子函数完成；

待处理的信令操作码包括：

Class	Data	Description
Reply: Service Done	200	OK
Reply: Temporary	300	Need device update its Device ID
	301	Need device update its status
Reply: Error comes from Device	401	Device ID Error
	402	Device ID not found on this server
	403	Syntax error in parameters or arguments
	404	Bad command sequence
	405	Conversation Timeout
Reply: Error comes from Server	501	Database error
	502	Requested action aborted
	503	Requested action Not taken
	504	Server timeout

5.8.3. Function 功能

根据各返回类型操作码，选择合适的处理函数进行错误处理；

5.8.4. Capability 性能

略

5.8.5. Input 输入

statcode：访问服务器的请求操作码，可以使用的值包括

1、LOGIN
2、LOGOUT101
3、LOGOUT102
4、QUERY110
5、QUERY111
6、UPSTAT120

[注意]:
1、详细请参考本文档4. 1. 1. 中说明;

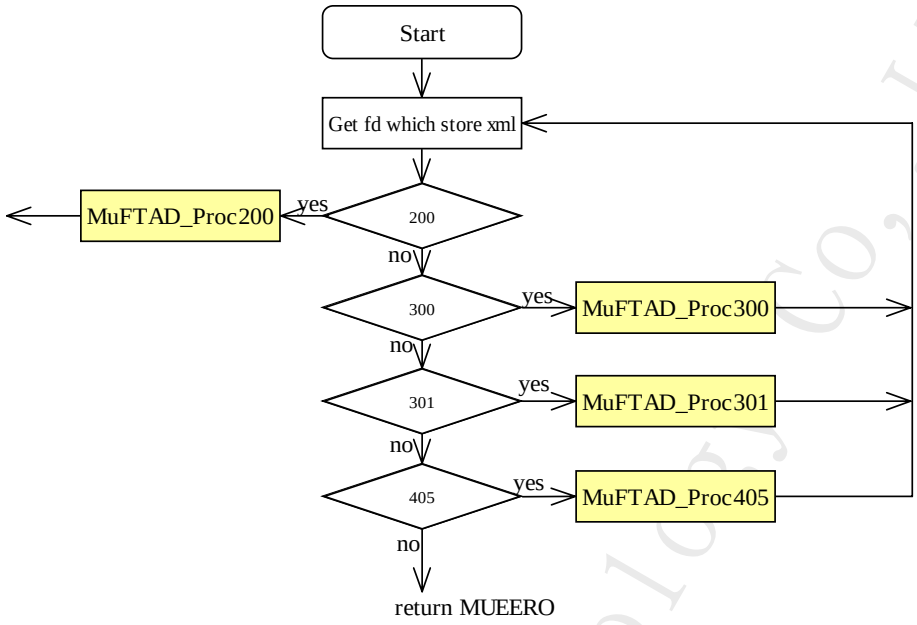
5.8.6. Output 输出

函数处理状态码，包括：
1、无错误，以 **MUOK** 返回；

5.8.7. Arithmetic 算法

略

5.8.8. Process 处理流程



5.8.9. Pseudo-code 伪代码

```
int need_retry = 0;
int MuFTAD_XmlProc(int statcode)
{
    int ret = MUOK;
    int fd;

    //the first time, we must parse the file stored the XML,
    //which maybe rcvfd, listfd, or rcvfd;
    //when the statcode if 30X or 405,the statcode is ONLY rcvfd;
    if(0 ==  （fd = MuFTAD_GetStoreFd(statcode)）){
        return MUEERO;

    while(1){
        if((ret = Mu_Parse(PARSE_EYEWEARE_ADP,
                        fd, NULL, &MuEyewearADPrcv)) < 0)
            break;

        switch(MuEyewearADPrcv->ProtocolOPC){
```

批注 [ssw12]: 解析从服务器接收到的信令，并将 EyewearADP 解析至 MuEyeWearADPrcv 结构体中

```
            case 200:
                if((ret = MuFTAD_Proc200(statcode) ) < 0)
                    do error;
                need_retry = 0;
                break;

            case 300:
                if((ret = MuFTAD_Proc300()) < 0)
                    break;
                fd = rcvfd;
                need_retry = 1;
                continue;

            case 301:
                if((ret = MuFTAD_Proc301()) < 0)
                    break;
                fd = rcvfd;
                need_retry = 1;
                continue;

            case 405:
                if((ret = MuFTAD_EyeADPPost(LOGIN)) < 0)
                    break;
                fd = rcvfd;
                need_retry = 1;
                continue;

            default:
                return MUEERO;
        }
    }

    return ret;
}
```

批注 [ssw13]: 用于重新发起请求

5.8.10. Interface 接口

处理函数在处理过程中，仅对200，300，301进行更进一步的处理，其他类型的操作码，均以出错码直接返回；

5.9. MuFTAD_ProcLogin200

5.9.1. Name 函数名称

int MuFTAD_ProcLogin200(void)

5.9.2. Description 函数描述

该函数对服务器返回操作码为200的信令进行处理

5.9.3. Function 功能

略

5.9.4. Capability 性能

略

5.9.5. Input 输入

参考5.6.5.中说明；

5.9.6. Output 输出

函数运行状态码，包括：
1、无错，以 **MUOK** 返回；

5.9.7. Arithmetic 算法

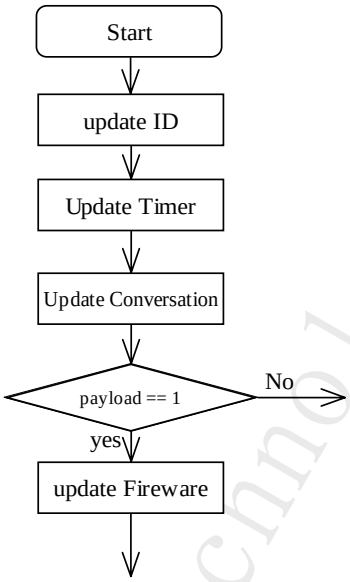
当服务器返回的操作码为200时，需要处理的情况包括：
1、更新 ID 字段
ID 字段为 MBC，因此，若 Device_ID_Int 和 Device_ID_App 字段非空，则需要更新 DeviceInfo 结构中的值，并且写入 DeviceInfo.xml 文件中；

2、更新时间

若 SRV_Date_Time 非空，更新设备时间值；

3、更新 Conversation 字段
若原有 Conversation 为空，或是与现有内容相同，则放弃更新；
否则更新该字段内的内容；

5.9.8. Process 处理流程



5.9.9. Pseudo-code 伪代码

```
int MuFTAD_ProcLogin200(void)
{
    DevieInfoPtr Ptr = MuEyewearADPrcv->device_info;

    //updateTimer
    if(Ptr->SRVDateTime)
        MuFTAD_UpdateTime(Ptr->SRVDateTime);

    //update ID
    if(Ptr->device_info->deviceIdInit
        || Ptr->device_info->deviceIdApp)
```

批注 [ssw14]: 根据该字段值，更新设备的时间

```
        MuFTAD_UpdateID(Ptr->device_info->DeviceIdInit,
            Ptr->device_info->DeviceIdApp);

    //update conversation
    if(Ptr->conversation->ConversationID
        || Ptr->conversation->AuthUsername
        || Ptr->conversation->AuthPassword)
        MuFTAD_UpdateConv (Ptr->conversation->ConversationID,
                    Ptr->conversation->AuthUsername,
                    Ptr->conversation->AuthPassword);

    return MUOK;
}
```

批注 [ssw15]: 更新设备的 ID_Int 和 ID_App 号，并且将这些信息更新至 **MuEyewearADPsnd** 和 **MuDeviceInfo** 中，同时，将上述信息更新至 **deviceinfo.xml** 文件中

批注 [ssw16]: 更新服务器为设备临时分配的会话 ID，会话名称和会话密码；
该部分必须更新至 **MuEyewearADPsnd** 和 **MuConversation** 结构中

5.9.10. Interface 接口

函数在处理服务器返回的操作码时，调用三个函数对其进行操作，分别为：

1、**MuFTAD_UpdateTime**

更新设备的时间，注意从服务器返回的时间为一个字串，需要进行一些处理；

2、 **MuFTAD_UpdateID**

当 ID_Int 和 ID_App 二者之中，只要有一个不为 NULL，即更新 MuDeviceInfoPtr 全局结构及 MuEyeWearADPsnd 结构；

更新后的内容，必须修改 DeviceInfo.xml 文件；

3、**MuFTAD_UpdateConv**

函数有三个参数，只要其中之一不为空，即调用函数更新上述内容，包括 MuEyeWearADPsnd 结构；

上述更新的内容，应该用三个全局值单独保存；

5.10. MuFTAD_ProcQueryStat200

参考5.8.中说明，但是只需要更新 Username 和 Password

```
int MuFTAD_ProcQueryStat200()
{
    ConversationPtr Ptr = MuEyewearADPrcv-> conversation;

    if(Ptr->AuthUsername|| Ptr ->AuthPassword)
        MuFTAD_UpdateConv (NULL, Ptr-> AuthUsername,
                                Ptr-> AuthPassword);
```

return **MUOK**;
}

5.11. MuFTAD_Proc200

5.11.1. Name 函数名称

int MuFTAD_Proc200(int statcode)

5.11.2. Description 函数描述

函数对协议中所有的200状态进行处理，由于不同状态下，服务器返回的200操作码，需要进行不同的更新，因此需要 statcode 进行选择适当的处理函数；

5.11.3. Input 输入

Statcode：服务器接收到操作码为200时的处理状态，可以用请求状态表示，现阶段仅仅需要特别处理 LOGIN 和 QUERY110二种状态下的200操作码；
可以使用的值包括：
1、LOGIN
2、LOGOUT101
3、LOGOUT102
4、QUERY110
5、QUERY111
6、UPSTAT120
[注意]:
1、详细请参考本文档4.1.1.中说明;

对于其他阶段的200操作码，直接以 **MUOK** 返回；

5.11.4. Output 输出

函数运行的状态码，包括：
1、无错，以 **MUOK** 返回；

2、类型值不符合，以 **MUETYP** 返回；
3、其他错误，以各函数处理的错误码返回；

5.11.5. Arithmetic 算法

略

5.11.6. Process 处理流程

略

5.11.7. Pseudo-code 伪代码

```
int MuFTAD_Proc200(int statcode)
{
    int ret = MUOK;

    switch(statcode){
        case LOGIN:
            ret = MuFTAD_ProcLogin200();
            break;

        case QUERY110:
            ret = MuFTAD_ProcQueryStat200();
            break;

        default:
            return MUETYP;
    }
    return ret;
}
```

5.11.8. Interface 接口

略

5.12. MuFTAD_Proc300

5.12.1. Name 函数名称

int MuFTAD_Proc30X(void)

5.12.2. Description 函数描述

当从服务器接收到的信令中的操作码为300，表明设备需要更新 App 字段，在设备写成更新后，需要以操作码120向服务器更新状态；

5.12.3. Input 输入

无

5.12.4. Output 输出

函数运行状态码，包括：
1、无错，以 **MUOK** 返回；

5.12.5. Arithmetic 算法

当操作码为300时，设备需要更新保存设备 ID 信息的 XML 文件，同时也必须更新 MuEyewearADPSnd 结构；

当更新完成后，设备必须以操作码120向服务器更新状态。调用 MuFTAD_EyeADPPost 完成；

5.12.6. Process 处理流程

略

5.12.7. Pseudo-code 伪代码

int MuFTAD_Proc300(void)

```
{
    DevieInfoPtr Ptr = MuEyewearADPrcv->device_info;
    //update the device information
    if(Ptr-> deviceIdInit || Ptr ->deviceIdApp)
        MuFTAD_UpdateID(Ptr ->DeviceIdInit,  Ptr-> DeviceIdApp);

    //post to server for the update!
    if(MuFTAD_EyeADPPost(UPSTAT120) < 0)
        return MUEERO;

    return MUOK;
}
```

批注 [ssw17]: 该函数需要更新设备的 ID_Int 和 ID_App 号，在更新时，对于 **MuEyewearADPsnd**, **MuDeviceInfo** 结构中均需要更新（重新申请内存存储），并且将上述字段写入 **deviceinfo.xml** 中

5.13. MuFTAD_Proc301

同 MuFTAD_Proc300

5.14. MuFTAD_Login

5.14.1. Name 函数名称

int MuFTAD_Login(void)

5.14.2. Description 函数描述

函数利用 MuEyeWearADPsnd 结构中的值，构建请求信令，并发送到服务器；

[注意]:

1、MuEyeWearADPsnd 结构中存储的是当前最新的信令成员，在构建请求信令时，需要用该结构中的成员创建请求 XML 文件（信令）;

函数退出时，只有二种可能：200和出错；
其他错误，包括300、301、405等错误都将在函数内部完成处理；

5.14.3. Input 输入

略

5.14.4. Output 输出

函数运行状态码，包括：
1、无错，以 **MUOK** 返回；
2、出错时，返回各调用函数的错误代码；

5.14.5. Arithmetic 算法

函数的处理包括三个部分，主要由三个函数来完成，包括：
1、向服务器发送请求信令，由函数 MuFTAD_EyeADPPost 函数完成；

2、解析服务器返回信令，并对200外的操作码，如超时，更新固件等信令进行处理，最终以200成功登录或是出错返回；
该部分由 MuFTAD_XmlProc 函数完成；
[注意]:
1、当程序处理200外的操作码后，仍必须以200返回，当正常返回后，程序必须重新进行登录;
2、MuFTAD_XmlProc 在处理200外的操作码时，会对全局值 need_retry 进行设置，以便于重新操作请求;

3、更新固件，该部分由 MuFTAD_DnLoadFireware 完成；

5.14.6. Process 处理流程

略

5.14.7. Pseudo-code 伪代码

```
volatile int need_retry = 0;
int MuFTAD_Login(void)
{
    int ret;

    do{
        if((ret = MuFTAD_EyeADPPost(LOGIN)) < 0
            || (ret = MuFTAD_XmlProc(LOGIN)) < 0){

            //do error;
            return ret;
```

```
        }
    }while(need_retry)

    //download the update fireware, if it's exist!
    //when download fireware failed, can't return with error;
    if(MuFTAD_DnLoadFireware() < 0){
        fprintf(stdout, "can't download the update fireware!\n");

    return MUOK;
}
```

批注 [ssw18]: need_retry 用于标记重新请求，在解析服务器返回的信令时，若不是200会更改其值，此时，请求函数需要重新请求服务器；

批注 [ssw19]: 用于更新文件，若服务器返回中包含payload，则需要相应地更新固件
根据法电协议，100与110**最多只可能同时出现一个**

5.15. MuFTAD_UpSRV120

参考 MuFTAD_Login 实现

```
int MuFTAD_UpSRV120Post(void)
{
    int ret ;

    do{
        if((ret = MuFTAD_EyeADPPost(UPSTAT120)) < 0
            ||(ret = MuFTAD_XmlProc(UPSTAT120)) < 0){

            //do error;
            return ret;
        }
    }while(need_retry)

    return MUOK;
}
```

5.16. MuFTAD_UpSRV121

参考 MuFTAD_Login 实现

```
int MuFTAD_UpSRV121(int type)
{
    int ret;

    do{
        if((ret = MuFTAD_QueryListPost(type) < 0
            ||(ret = MuFTAD_XmlProc(UPSTAT121)) < 0){
            do error;
            return ret;
        }
```

```
    }while(need_retry)

    return MUOK;
}
```

5.17. MuFTAD_UpSRV122

参考 MuFTAD_Login 实现

```
int MuFTAD_UpSRV122(int type)
{
    int ret;

    do{
        if((ret = MuFTAD_UpDnPost(UPFILE, type)) < 0
            || (ret = MuFTAD_XmlProc(UPSTAT122)) < 0){
            do error;
            return ret;
        }
    }while(need_retry)

    return MUOK;
}
```

5.18. MuFTAD_Logout101

参考 MuFTAD_Login 实现

```
int MuFTAD_Logout101(void)
{
    int ret;

    do{
        if((ret = MuFTAD_EyeADPPost(LOGOUT101)) < 0
            ||((ret =  MuFTAD_XmlProc(LOGOUT101)) < 0){
            do error;
            return ret;
        }
    }while(need_retry)

    return MUOK;
}
```

5.19. MuFTAD_Logout102

参考 MuFTAD_Login 实现

```
int MuFTAD_Logout102(void)
{
    int ret;

    do{
        if((ret = MuFTAD_EyeADPPost(LOGOUT102)) < 0
            ||((ret =  MuFTAD_XmlProc(LOGOUT102) < 0){
            do error;
            return ret;
        }
    }while(need_retry)

    return MUOK;
}
```

5.20. MuFTAD_Query110

参考 MuFTAD_Login 实现

```
int MuFTAD_Query110(void)
{
    int ret;

    do{
        if((MuFTAD_EyeADPPost(LOGOUT110)) < 0
            ||((ret =  MuFTAD_XmlProc(QUERY110)) < 0){
            do error;
            return ret;
        }
    }while(need_retry)

    return MUOK;
}
```

5.21. MuFTAD_Query111

5.21.1. Name 函数名称

int MuFTAD_Query111(FddfStatNodePtr Ptr, int type, int resume)

5.21.2. Description 函数描述

本函数根据 MuFTAD_Query110或是续传时的直接调用，查询服务器，以取得所查询的节目的 FDDF 地址；

[注意]:

1、FDDF 地址是进行2XX 节目下载的前提，一个 FDDF 地址对应一个节目;

5.21.3 Function 功能

查询服务器,以获得节目相关的 FDDF 地址,并且调用 MuFTAD_QueryFddf 函数

5.21.4. Input 输入

Ptr：FddfStatNode 结构，该结构存储了查询服务器的相关参数；

[注意]:

1、详细的结构体，请参考《MuFTAD 数据结构详细设计》文档中说明;

type：用于区别正常的 FDDF 查询和指定查找 FDDF；

[注意]:

1、正常的 FDDF 查询，payload 字段中的 type 值一般为2XX;

2、指定查找时，payloadpb 字段中的 type 值一般为902;

resume：用于标识对于该2XX 节目的下载是否是重传；

[注意]:

1、续传时，需要调用该函数接口续传节目，若设置 resume 值，在下载前会首先进行分段校验，再进行下载;

5.21.5. Output 输出

函数运行的状态码，包括：
1、无错，以 MUOK 返回；
2、其他错误，包括解析出错，删除节点出错，以 **MUEERO** 返回；

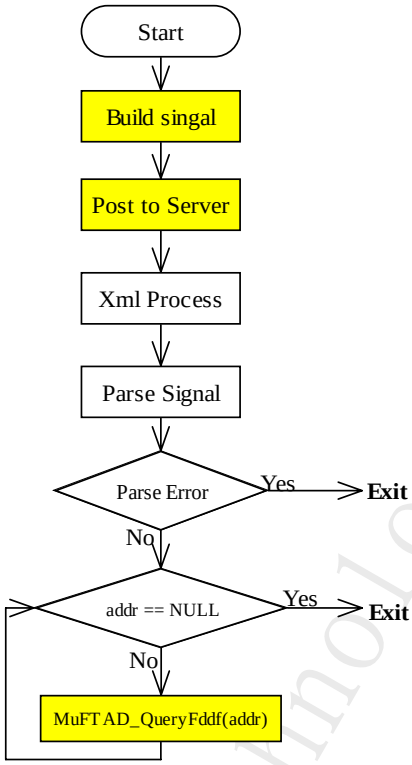
5.21.6. Arithmetic 算法

本函数首先会利用各信息，创建请求信令，并且 POST 到服务器，接收从服务器返回的信令；

对各信令，进行错误处理后，进行 XML 解析，获得各 FDDF 地址内容，并且存储于 FddfListHead 所指结构体中；
[注意]:
1、在对 FDDF 信令解析时，所有的 payload 均一次性解析出来，存储在 FddfListHead 所指向的链表中;

对每一个 Fddf 地址调用 MuFTAD_QueryFddf 函数，下载 FDDF 内容，进一步处理；

5.21.7. Process 处理流程



5.21.8. Pseudo-code 伪代码

```
volatile int need_retry = 0;
int MuFTAD_Query111(FddfStatNodePtr Ptr, int type, int resume)
{
    int ret;
    char *addr = NULL;
    int method;

    //build signal and post to server!!
    do{
        if((MuFTAD_UpDnPost(Ptr, QUERY111, type) < 0
            ||MuFTAD_XmlProc(QUERY111) < 0 ){
            do error;
            return ret;
        }
```

```
    }while(need_retry)

    if(Mu_Parse(PARSE_MULTI_LIST, listfd,
                        "//payload", MuFddfListHead) < 0)
        return MUEERO;

    while(MuFddfListHead->total){
        addr = MuFddfListHead->next;
        method = MuFddflustHead->authmethod;
        MuFTAD_QueryFddf(addr, method, resume);

        //delete one of the node, NOT the list of ALL
        if(Mu_FddfListNodeDelete(MuFddfListHead) < 0)
            return MUEERO;

        MuFddfListHead->total -- ;
    }
    return MUOK;
}
```

批注 [ssw20]: Mu_Parse 在解析 PARSE_MULTI_LIST 时，是一次性全部解析出所有 FDDF XML 的地址的

批注 [ssw21]: 函数循环处理的条件是存在着 FddF XML 地址，即 Fddf listNode 存在，也即 MuFddfListHead->next != NULL

但是代码中，还是使用 MuFddfListHead->total!= 0来判断是否继续；

批注 [ssw22]: 该函数只删除一个节点，而不会删除整条链表

5.21.9. Interface 接口

函数在实现上述功能时，需要用到的函数接口有：
Mu_Parse
MuFTAD_QueryFddf
Mu_FddfListNodeDelete
MuFTAD_UpDnPost
MuFTAD_XmlProc
[注意]:
1、Mu_FddfListNodeDelete 函数仅仅删除一个节点 NODE，而不是该链表所有的结点;

5.21.10. Unsolved 未解决问题

略

5.22. MuFTAD_QueryPost

5.22.1. Name 函数名称

int MuFTAD_QueryPost(int querytype)

5.22.2 Description 函数描述

函数构建服务器请求地址，并且根据请求方式，选择合适的请求参数，向服务器发起请求；

对于服务器返回的信息，函数将其存储在 MuDnStat->fd 中，该结构可能包括 rcvfd, statfd, listfd；

如果在请求服务器返回的错误码为 MUECON，取一个新的服务器地址发起连接，直到发送取得成功；

5.22.3. Input 输入

querytype：请求类型，用以确定请求的 API 和存储服务器返回信息存储的文件句柄；

5.22.4. Output 输出

1、无错，以 **MUOK** 返回；
2、不能取得合法 API，以 **MUNAPI** 返回；
3、不能取得合法的存储文件句柄，以 **MUEERO** 返回；

5.22.5. Arithmetic 算法

函数首先从 MuServiceInfo 结构中取得服务器的地址，然后取得 API，组合服务器请求地址；

通过 Mu_HttpPlus 向服务器发起请求时，需要传递一个 MuDnStat 结构，在调用该函数前必须初始化该结构；

对于请求服务器时返回的错误，只对 **MUECON** 进行取理：选用下一个可用的服务器地址，继续上面的操作；

若请求服务器时，返回的是其他类型的错误，将以错误码直接返回；

5.22.6. Process 处理流程

同5.4中说明

5.22.7. Pseudo-code 伪代码

```
ServiceInfoPtr MuServiceInfo
int MuFTAD_QueryPost(int querytype)
{
    char *addr;
    int querylen;
    char *query = NULL;
    char *api = NULL;
    int ret = MUOK;
    ServerHeadPtr server = MuServiceInfo->server;

    if(NULL == (api =( MuFTAD_GetAPI(querytype)) )
        return MUNAPI;

    if(NULL == server->cur)
        return MUEERO;
again:
    if(NULL == (addr = server->cur->addr))
        return MUEERO;

    //build the query addr
    querylen = snprintf(NULL, 0, “https://%s/%s”,addr, api);
    if(NULL == (query = malloc(querylen + 1))
        return MUNBUF;
    snprintf(query, querylen, “https://%s/%s\0”, addr, api);

    //query server!
    Mu_DnStatInit(MuDnStat);
    MuDnStat->type = MU_POST;
    if( 0==(MuDnStat->fd = MuFTAD_GetStoreFd(querytype)))
            return MUEERO;
    MuDNStat->xmlfd = sndfd;
    ret = Mu_HttpPlus(query, MU_QUERY, MuDnStat);
    free(query);
```

批注 [ssw23]: 根据函数的请求类型，确定下载时写入的文件句柄；
主要是用于请求和查询；

批注 [ssw24]: 如果 MuDnStat->startops = 0，Mu_HttpPlus 函数在下载前都需要定位，直接使用 lseek(MuDnStat->fd, MuDnstat->startops, SEEK_SET)
同样，该函数对每个写完的部分都需要写入 EOF

```
    //reconnect to next server
    if(ret == MUECON){
        if(server->cur->next != NULL)
            server->cur = server->cur->next;
        else
            server->cur = server->head;
        goto again;
    }

    return ret;
}
```

5.22.8. Interface 接口

在本函数完成操作时，需要用到 Mu_DnStatInit 函数初始化 MuDnStat 结构体；

对服务器的请求查询，需要用到 MuHttpPlus 函数；

5.23. MuFTAD_GetStoreFd

通过 opccode 确定用于存储服务器返回信息的文件句柄。伪代码如下：

```
int MuFTAD_GetStoreFd(int opccode)
{
    int fd = 0;

    //get the api method
    switch(opccode){
        case QUERY110:
            fd = statfd;
            break;
        case QUERY111:
            fd = listfd;
            break;
        case LOGIN:
        case UPSTAT120:
        case UPSTAT121:
        case UPSTAT122:
        case LOGOUT101:
        case LOGOUT102:
            fd = rcvfd;
            break;
```

批注 [ssw25]: 存储 Query Status 后，服务器返回的信令，该信令中包含所有已订阅内容；
用该文件存储是便于多次解析

批注 [ssw26]: 存储 Query Fddf List 后，服务器返回的信令，其中包含 FDDF XML 文件的下载地址；
该文件句柄专用于存储上述内容；

批注 [ssw27]: 所有从服务器返回的信令，若不需要多次解析，均存储于该文件句柄中；

```
        default:
            return fd;
    }

    return fd;
}
```

5.24. MuFTAD_GetAPI

通过 opccode 确定服务器的 API，类似于 MuFTAD_GetStoreFd。伪代码如下：

```
char *MuFTAD_GetAPI(int opccode)
{
    char *api = NULL;

    //get the api method
    switch(querytype){
        case LOGIN:
            api = MuServiceInfo->apilist->login;
            break;
        case QUERY110:
            api = MuServiceInfo->apilist->querystatus;
            break;
        case QUERY111:
            api = MuServiceInfo->apilist->queryfddflist;
            break;
        case UPSTAT120:
        case UPSTAT121:
        case UPSTAT122:
            api = MuServiceInfo->apilist->upstatus;
            break;
        case LOGOUT101:
        case LOGOUT102:
            api = MuServiceInfo->apilist->logout;
            break;
        default:
            return api;
    }

    return api;
}
```

5.25. MuFTAD_MakeURL

5.25.1. Name 函数名称

int MuFTAD_MakeURL(char **query, int method, char *path)

5.25.2. Description 函数描述

在下载固件或是节目内容时，有四种不同的方式，方式包括 http, https，分别包括用用户名和密码的形式；
本函数用于构建这类 URL 请求地址；

5.25.3. Input 输入

query：用以返回构建的请求 URL 内存地址；

method：用以确定构建的请求 URL 类型：
包括以下四种：
0、以 Https 请求服务器，但是不用用户名和密码；
1、以 Https 请求服务器，用用户名和密码；
2、以 Http 请求服务器，但是不用用户名和密码；
3、以 Http 请求服务器，用用户名和密码；
[注意]:
1、详细请参考法电协议3.2.3.中说明;

path：除协议名称外的路径名；

批注 [ssw28]: 该 path 包括路径名和文件名，加上服务器名，协议名称，用户名和密码后就是完整的 URL

5.25.4. Output 输出

返回函数的运行状态，包括：
1、无错，以 **MUOK** 返回；
2、申请内存空间，以 **MUNBUF** 返回；
3、构建 URL 方式出错，以 **MUENEH** 返回；

5.25.5. Arithmetic 算法

构建服务器请求 URL，该 URL 使用当前服务器地址，构建 URL；

5.25.6. Process 处理流程

略

5.25.7 Pseudo-code 伪代码

```
MuUsername, MuPassword,
int MuFTAD_MakeURL(char **query, int method, char *path)
{
    int querylen;
    ServerHeadPtr server = MuServiceInfo->server;
    char *addr = server->cur->addr;

    //build the query addr
    switch(method){
        case 0:
            querylen = snprintf(NULL, 0,
                "https://%s/%s",addr, path);

            if(NULL == (*query = malloc(querylen + 1)))
                return MUNBUF;
            snprintf(query, querylen,
                "https://%s/%s\0", addr, path);
            break;
        case 1:
            querylen = snprintf(NULL, 0,
                "https://%s:%s@%s/%s",
                MuUsername, MuPassword,
                addr, path);

            if(NULL == (*query = malloc(querylen + 1)))
                return MUNBUF;
            snprintf(query, querylen,
                "https://%s:%s@%s/%s\0",
                MuUsername, MuPassword,
                addr, path);
            break;
        case 2:
```

```
                querylen = snprintf(NULL, 0,
                        “https://%s/%s”,addr, path);

                if(NULL == (*query = malloc(querylen + 1)))
                    return MUNBUF;
                snprintf(query, querylen,
                        “https://%s/%s\0”, addr, path);
                break;
        case 3:
                querylen = snprintf(NULL, 0,
                        “https://%s:%s@%s/%s”,
                        MuUsername, MuPassword,
                        addr, path);

                if(NULL == (*query = malloc(querylen + 1)))
                    return MUNBUF;
                snprintf(query, querylen,
                        “https://%s:%s@%s/%s\0”,
                        MuUsername, MuPassword,
                        addr, path);
                break;
        default:
                return MUEMEH;
    }

    return MUOK;
}
```

5.26. MuFTAD_EyeADPSndInit

5.26.1. Name 函数名称

int MuFTAD_EyeADPSndInit(void)

5.26.2. Description 函数描述

该函数**在预登录完成后**，被调用。用于初始化 MuEyewearADPsnd 结构，而 MuEyewearADPsnd 将用于创建请求信令；

[注意]:

1、MuEyewearADPsnd 结构体中保存的始终是各字段中最新的信息;

2、从服务器返回的信令中解析各参数至 MuEyewearADPrcv 结构中，若需要更新可以直接将 MuEyewearADPrcv 结构中的字符串成员空间赋值给 MuEyewearADPsnd，并且 MuEyewearADPrcv 相关结构被置 NULL;

5.26.3. Function 功能

初始化 MuEyewearADPsnd 结构

5.26.4. Input 输入

无

5.26.5. Output 输出

函数的运行状态码，包括：
1、无错，以 **MUOK** 返回；
2、无内存空间，以 **MUNBUF** 返回；

5.26.6. Arithmetic 算法

初始化 MuEyewearADPsnd 结构；设置结构中的协议版本号和操作码；对法电协议中在 Login 中应该具备的选项赋初值；

对应于 MuEyewearADPsnd 结构中，若 MuEyewearADPecv 中有相应的最新字串，函数直接将 MuEyewearADPrcv 中的字串地址直接赋值给 MuEyewearADPsnd，并且将 MuEyewearADPrcv 结构中的该字串地址置 NULL；

但是，若 MuEyewearADPsnd 中的字段对应于其他结构体中，比如 DeviceInfo 时，需要将其拷贝过来，而不可以按上面操作进行；

5.26.7. Process 处理流程

略

5.26.8. Pseudo-cdoe 伪代码

```
#define PROCVER 1
#define FIREWEAREVERSION 1
EyewearADPPtr MuEyewearADPsnd
int MuFTAD_EyeADPSndInit()
{
    char *ID = NULL;

    //init the structure of MuEyewearADPsnd
    Mu_EyewearADPInit(MuEyewearADPsnd);

    //Init the protocol version & OPC of the XML
    MuEyewearADPsnd->protocolver = PROCVER;
    MuEyewearADPsnd->OPC = 0;

    //create the login method
    //in the function, we make the Challenge's Content of the MuEyewearADPrcv
    //Be pointed to MuEyewearADPsnd
    //set the MuEyewearADPrcv is NULL;
    MuEyewearADPsnd->prelogin->encriptionmethod =
                    MuEyewearADPrcv->prelogin->encriptionmethod;
    MuEyewearADPsnd->prelogin->Challenge =
                    MuEyewearADPrcv->prelogin-> Challenge;
    MuEyewearADPrcv->prelogin-> Challenge = NULL;

    //create the IDInit and App
    //NOT same TO Above
    if(MuDeviceInfo->IDInit){
        if((ID = malloc(strlen(MuDeviceInfo->IDInit) + 1)) == NULL)
            return MUNBUF;
        sprintf(ID, "%s\0", MuDeviceInfo->IDInit);
        MuEyewearADPsnd->device_info->IDInt = ID;
    }
    if(MuDeviceInfo->IDApp){
        if((ID = malloc(strlen(MuDeviceInfo->IDApp) + 1)) == NULL)
            return MUNBUF;
        sprintf(ID, "%s\0", MuDeviceInfo->IDApp);
        MuEyewearADPsnd->device_info->IDApp = ID;
    }

    MuEyewearADPsnd->device_info->firewareversion =
                                        FIREWEAREVERSION;
    MuEyewearADPsnd->device_info->hardwareversion =
```

```
                                        MuDeviceInfo->hardwareversion;
    MuEyewearADPsnd->device_info->serverinfostep =
                                        MuServiceInfo->stepping;

    return MUOK;
}
```

5.26.9. Interface 接口

略

5.26.10. Unsolved 未解决问题

略

5.27. MuFTAD_DnUpdate

5.27.1. Name 函数名称

int MuFTAD_DnUpdate(void)

5.27.2. Description 函数描述

该函数处理登录时，服务器标明的更新设备固件；

根据法电协议说明，同一时间只可能更新一个固件，或是没有固件需要更新，详见《doc00_SoftProtocol_0.1.0_RC1.doc》5.2节中说明；

5.27.3. Input 输入

无

5.27.4. Output 输出

函数运行状态码，包括：
1、无错误，以 **MUOK** 返回；

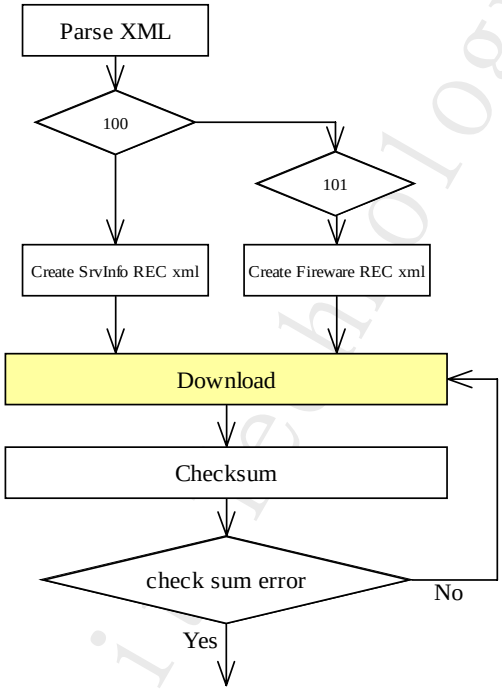
5.27.5. Arithmetic 算法

在下载各固件前，设备必须先解析服务器返回的 XML 信令，从中获得下载类型和下载的条件；

在成功解析各内容后，构建固件更新 XML：update_SRV.xml 文件，其中需要用到 UpdateInfo 结构；

等成功创建记录 XML 后，构建固件下载地址，下载更新固件，下载完成后，需要对固件进行校验；

5.27.6. Process 处理流程



5.27.7. Pseudo-code 伪代码

```
int MuFTAD_DnUpdate(void)
{
    int dnfd = 0;
    char *query;
```

```
    SRVUpdateInfoPtr MuSRVUpdateInfo;

    //parse the XML
    if(Mu_Parse(PARSE_UPDATE_INFO,
                rcvfd, "//payload", &MuSRVUpdateInfo) < 0)
        return MUEERO;

    //create the update recording XML
    int fd = fopen("/updatefile/update_SRV.xml", O_RDWR |O_CREATE);

    //if update service information XML
    if(MuSRVUpdateInfo->type == 100){
        MuUpdateInfo->srvsvrinfo->version = MuSRVUpdateInfo->version;
        MuUpdateInfo->srvsvrinfo->checksum =
                                    MuSRVUpdateInfo->checksum;
        Mu_BuildXML(BUILD_UPDATE_INFO, fd, MuUpdateInfo, NULL);
        dnfd = open("./updatefile/ServerInfo.xml", O_RDWR|O_CREATE);
    }else if(MuSRVUpdateInfo->type == 101){
        MuUpdateInfo->srvfiresf->version = MuSRVUpdateInfo->version;
        MuUpdateInfo->srvfiresf->checksum =
                                    MuSRVUpdateInfo->checksum;
        MuUpdateInfo->srvfiresf->hardversion =
                                MuSRVUpdateInfo->hardwareversion;
        Mu_BuildXML(BUILD_UPDATE_INFO, fd, MuUpdateInfo, NULL);
        dnfd = open("./updatefile/lib", O_RDWR|O_CREATE);
    }else
        return MUETYP;

    //protect the segment error,when free a buffer;
    MuUpdateInfo->srvfiresf->checksum = NULL;

again:
    addr = server->cur->addr;
    int ret = MUOK;

    MuFTAD_MakeURL(&query, MuSRVUpdateInfo->method,
                        MuSRVUpdateInfo->file);
    //query server!
    Mu_DnStatInit(MuDnStat);
    MuDnStat->type = MU_GET;
    MuDnStat->fd = dnfd;

    ret = Mu_HttpPlus(query, MU_QUERY, MuDnStat);

    free(query);
```

```
    if(ret == MUECON){
        if(server->cur->next != NULL)
            server->cur = server->cur->next;
        else
            server->cur = server->head;

        goto again;
    }

    if(ret != MUOK){
        MuFTAD_Error(ret);
        return MUERRO;
    }

    if(Mu_Md5File(dnfd, MuSRVUpdateInfo->checksum) < 0)
        goto again;

    close(dnfd);
    Mu_ SRVUpInfoDelete(MuSRVUpdateInfo);
    return MUOK;
}
```

5.27.8. Interface 接口

函数在实现时，利用的接口有：
Mu_BuildXML
Mu_Parse
MuFTAD_MakeURL
Mu_HttpPlus
Mu_Md5File

5.28. MuFTAD_Query110Proc

5.28.1. Name 函数名称

```
int MuFTAD_Query110Proc(void)
```

5.28.2. Description 函数描述

函数解析请求码为 OPC = 110所返回的信息，根据所解析的内容和信息决定进行更进一步的处理；

当设备用 OPC=110请求服务器时，返回的 payload 中包括2XX 和4XX 类型的节目，本函数一并处理上述二种类型；

5.28.3. Function 功能

解析服务器返回信令，选择进一步的处理；

5.28.4. Input 输入

无

5.28.5. Output 输出

函数运行的状态码，包括：
1、无错，以 **MUOK** 返回；
2、解析 XML 出错，或是删除节点出错，均以 **MUEERO** 返回；

5.28.6. Arithmetic 算法

解析服务器返回的信令时，一次只解析有限个（MUMULTI_LIST_MAX）或是解析完所有的2XX 类型节目。在进行处理时，需要采用循环解析的方式处理信令；

针对已解析出来的 payload，均需要调用 MuFTAD_Query111函数进行处理；

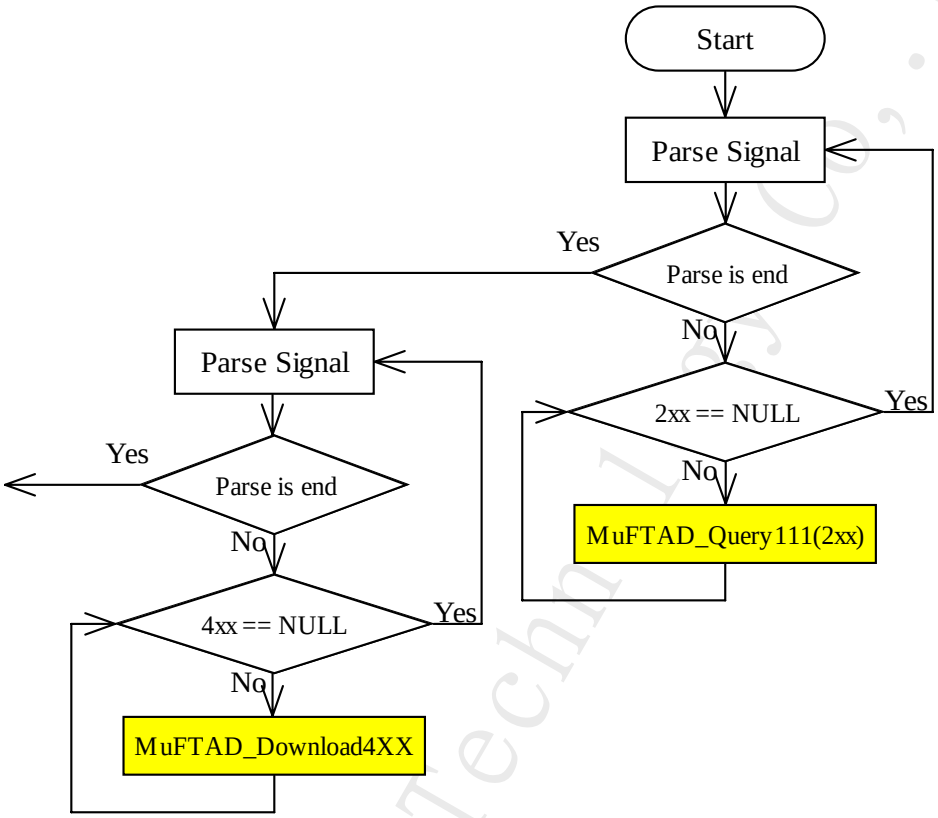
解析出错时，应该跳出循环，以表示没有可解析的2XX 节目内容；

在调用 OPC = 111查询前，必须首先从 FddfStatusHead 结构数组中查找出优先级最高的，有节目的2XX 节目链表，并且调用 MuFTAD_Query111进行处理；

[注意]:
1、在上述结构数组中，各成员均是按优先级进行排序过的;

对4XX 节目的处理类似于2XX；

5.28.7. Process 处理流程



5.28.8. Pseudo-code 伪代码

```
int MuFTAD_Query110Proc(void)
{
    int ret;
    FddfStatusNodePtr Ptr = NULL;
    DirNodePtr MuNodePtr;
    DirDownloadAddrPtr addrPtr;

    do{
        if(ret = Mu_Parse(PARSE_MULTI_2XX, statfd,
```

批注 [ssw29]: 对于 PARSE_MULTI_2XX 类型解析时，是采用分次解析的方式进行的，每次最多只解析有限个 FddfNode；
代码应该能区分2XX 解析完，和解析异常；

```
                    "//payload", MuFddfStatHead) < 0){
            if(ret == MUN2XX)
                break;
            else
                return MUEERO;
        }

        //find the newest list
        for(int i = 0; i < 4; i ++)
            if( MuFddfStatHead[i].total)
                break;

        //have no 2XX to get ride of
        if(i == 4 && MuFddfStatHead[i].total == 0)
            break;

        while(MuFddfStatHead[i].total){
            Ptr = MuFddfStatHead[i].next;
            MuFTAD_Query111(Ptr,  2xx,  0);

            if(Mu_FddfStatusNodeDelete(Ptr) < 0)
                return MUEERO;
            MuFddfStatHead[i].total -- ;
        }

    }while(1);

    do{
        if(ret = Mu_Parse(PARSE_MULTI_4XX, statfd,
                    "//payload", MuDirStatHead) < 0){
            if(ret == MUN4XX)
                break;
            else
                return MUEERO;
        }

        //find the newest list
        for(int i = 0; i < 4; i ++)
            if( MuDirStatHead[i].total)
                break;

        //have no 4XX to get ride of
        if(i == 4 && MuDirStatHead[i].total == 0)
            break;
```

批注 [ssw30]: 第三个参数用0，表明在下载时，不使用续传实现；

批注 [ssw31]: 仅删除一个节点，而不是整个链表

```
        while(MuDirStatHead[i].total){
            MuNodePtr = MuDirStatHead[i].next;
            method = MuNodePtr->method;
            addrPtr = MuNodePtr->addrnext;
            while(addrPtr){
                    MuFTAD_Download4XX(addrPtr->addr， method,  0);
                    addrPtr = addrPtr->next;
                    Mu_DirDownloadAddrDeleteone(MuNodePtr);
            }

            if(Mu_DirStatusNodeDelete(MuDirStatHead[i], MuNodePtr) < 0)
                return MUEERO;
            MuFddfStatHead[i].total -- ;
        }

    }while(1);

    return MUOK;
}
```

批注 [ssw32]: 删除一个地址节点

批注 [ssw33]: 删除一个 Item 节点

5.28.9. Interface 接口

函数在运行时，需要利用 MuFTAD_Query111查询服务器，OPC=110访问信令时，一个 payload 对应一个查询服务；

5.28.10. Unsolved 未解决问题

略

5.29. MuFTAD_QueryFddf

5.29.1. Name 函数名称

int MuFTAD_QueryFddf(const char *url, int authmethod, int resume)

5.29.2. Description 函数描述

本函数根据参数 url，请求服务器，从服务器端获得与一节目相对应的 FDDF XML 文件，该文件包含了一个节目的描述及该节目的所有分段信息；

对于节目的描述信息，解析至 MuFddfFile 结构中，并且根据节目描述信息创建节目描述 XML；

对分段信息，解析所有分段至以 MuSegmentHead 为头结点的链表中；

对每一个分段，调用 MuFTAD_Download2XX 进行节目下载；

5.29.3. Function 功能

请求服务器，取得 FDDF XML 文件，获取节目信息，并且调用函数下载节目；

5.29.4. Input 输入

url：const char 类型，用以存储访问服务器地址的信息，通过该地址可以取得一个节目的 FDDF 文件；

[注意]:

1、url 不包括服务器地址，该 URL 仅仅是除去服务器地址后的 path 和 file;

2、在请求服务器时，必须与当前有效服务器地址进行拼接;

authmethod：验证方式，函数由该参数决定请求服务器的方式和方法；

[注意]:

1、详情可以参考 MuFTAD_MakeURL 函数;

resume：判断下载是否需要续传；

5.29.5. Output 输出

函数的运行状态，包括：

1、无错，以 MUOK 返回；

5.29.6. Arithmetic 算法

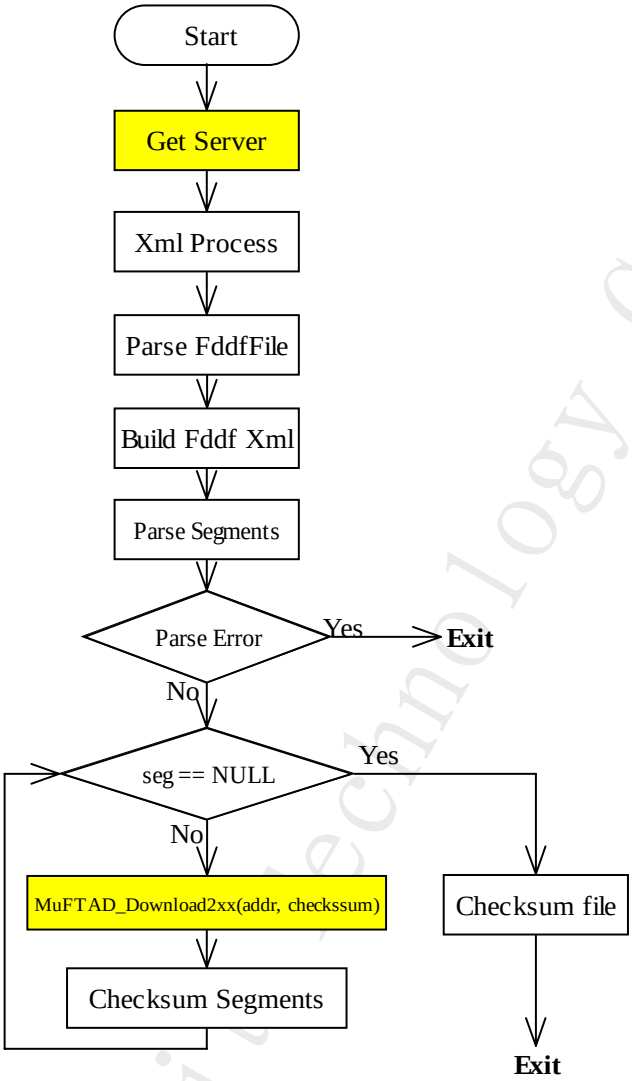
函数必须首先以参数 url, method 创建请求 URL，设备以 GET 方式访问服务器，从服务器端获得 FDDF XML 文件；

解析该 FDDF XML 文件，取得 FDDFFile 结构体内所有信息，并创建节目描述文件；

解析分段信息（全部）至 MuSegmentHead 为表头的链表中；

对每一个链表均调用下载函数 MuFTAD_Download2XX 进行处理；

5.29.7. Process 处理流程



8.29.8. Pseudo-code 伪代码

```
int MuFTAD_QueryFddf(const char *url, int authmethod, int resume)
{
    char *query;
    int xmlfd;
```

```
    int dnfd;
    temp[256];
    temp1[256];
    char *addr;
    ServerHeadPtr server = MuServiceInfo->server;

    sprintf(temp, "%s.mu\0", MuFddfFile->name);

    //check the file name.mv is exist or not
    if(access(MuFddfFile->name, O_ROK|O_WOK) < 0)
        return MUOK;

    //if the name.mv.mu is exist, then resume the program
    if(access(temp, O_ROK|O_WOK) == 0)
        resume = 1;

    if((dnfd = open(temp, O_RDWR|O_CREATE)) < 0){
        do error;
        return MUEERO;
    }

again:
    if(MuFTAD_MakeURL(&query, authmethod, url) < 0)
        return MUEERO;

    //query server!
    Mu_DnStatInit(MuDnStat);
    MuDnStat->type = MU_GET;
    MuDnStat->fd = rcvfd;
    ret = Mu_HttpPlus(query, MU_QUERY, MuDnStat);
    free(query);

    //if need to reconnect ot another servers,
    //we must change to the next server point
    if(ret == MUECON){
        if(server->cur->next != NULL)
            server->cur = server->cur->next;
        else
            server->cur = server->head;
        goto again;
    }

    if(ret != MUOK){
        MuFTAD_ErrorProc(ret);
        return MUERRO;
```

```
    }

    //parse Fddf content about the file desctiption
    if((ret = Mu_Parse(PARSE_EYEWEARE_FDDF, rcvfd, NULL,
&MuFddfFile) )< 0){
        //do error
        return ret;
    }

    sprintf(temp, "%s.xml\0", MuFddfFile->name);
    //build the file description of a Program
    if(((xmlfd = open(temp, O_RDWR|O_CREATE)) < 0)
        || Mu_BuildXML(BUILD_FILE_DES, xmlfd, MuFddfFile, NULL) <
0){
        do error;
        return MUEERO;
    }
    Mu_Close(xmlfd);

    //parse Fddf content about the Segments
    if((ret = Mu_Parse(PARSE_EYEWEARE_FDDF, rcvfd,
                        "//payload", &MuSegmentHead) )< 0){
        //do error
        return ret;
    }

    MuFTAD_RecordResum(MuResume, MuFddfFile->type, Ptr +1, &i)
    SegmentNode Ptr_1 = MuSegmentHead->next;
    SegmentNode Ptr_2 = MuSegmentHead->next;
    off_t offset = 0;
    while(Ptr_1){
        do{
            //check out the offset of the segment
            while(Ptr_2){
                if(Ptr_2->index < Ptr_1->index)
                    offset += Ptr_2->size;
                Ptr_2 = Ptr_2->next;
            }

            if(MuFTAD_Download2XX(dnfd, offset, Ptr_1, resume)< 0)
                resume =1;
            else
                break;
        }while(1);
```

批注 [ssw34]: 对应于一个节目的 FDDF，其内容包括文件描述信息和分段信息

批注 [ssw35]: 该函数的使用似乎有问题，该部分需要解析出 payload 中的分段信息

批注 [ssw36]: 在下载节目前，需要将节目的名称及类型写入结构数组 MuResume 中，同时写入文件 resume.xml 中

```
        MuSegmentHead->total --;
        Ptr_1 = Ptr_1->next;
    }

    if(Mu_Md5File(dnfd, MuSegmentHead->md5) < 0){
        return MUERRO;
    }

    Mu_Close(dnfd);
    rename(temp1, MuFddfFile->name);
    MuFTAD_UpSRV120();
    MuFTAD_DeleteResum(MuResume, i);
    return MUOK;
}
```

批注 [ssw37]: 对于每一个完成下载的节目，下载前使用 name.mv.mu，即使用.mu 后缀名，在下载完成后需要更名，去掉.mu 后缀

批注 [ssw38]: 在一个节目下载完成后，需要从 MuResume 结构数组中清除已下载完成节目信息，同时更新 rssume.xml 文件

5.29.9. Interface 接口

本函数在实现的时候，需要用到的函数接口有：

Mu_Parse
Mu_BuildXML
Mu_Md5File
Mu_Md5Segment
MuFTAD_UpSRV120
MuFTAD_Download2XX

MuFTAD_RecordResum：将节目相关信息记录到 MuResume 结构数组中，并且同时更新 resume.xml 文件；

[注意]:

1、该函数在查找数组空间存储节目信息时，必须首先遍历该数组，以确保数组中无同名，同类型节目;

2、若数组中存在同名，同类型节目，则以续传方式下载，并且返回同名同类型数组的下标号;

MuFTAD_DeleteResum：下载完成后，需要更新 MuResume 结构数组，并且同时更新 resume.xml 文件；

5.29.10. Unsolved 未解决问题

略

5.30. MuFTAD_Download2XX

5.30.1. Name 函数名称

int MuFTAD_Download2XX(int fd, off_t offset, SegmnetNode Ptr, int resume)

5.30.2. Description 函数描述

根据解析分段信息，下载分段内容至本地，在下载前，需要传入相关的参数，以使得下载的分段能准确地定位到文件中的相应位置；

对于每一个下载的分段，都应该进行 MD5校验，对不能校验正确的分段，函数以出错返回；

当一个分段下载完成后，不可关闭文件句柄；

5.30.3. Function 功能

略

5.30.4. Input 输入

fd：待写入的文件句柄，该文件句柄为待下载的节目；

offset：偏移量，对于一个分段，将标记了下载内容写入一个文件的起始部分；

[注意]:
1、该偏移量是通过计算所有小于该分段的 index 字段获得的;
2、因第1点，所以 SegmentHead 不能删除节点;

Ptr：分段信息的节点；

resume：标记是否续传；

5.30.5. Output 输出

函数的处理状态，包括：
1、无错，以 **MUOK** 返回；
2、构建 URL 出错，或是连接服务器时出现 MUECON 外的错误，以 **MUEERO** 返回；
3、校验分段的完整性出错，以 **MUECKS** 返回；

5.30.6. Arithmetic 算法

根据相关信息，创建请求地址。由于下载的节目是按分段组织的，所以下载前还必须设置文件的偏移量信息；

对于下载完成的内容，需要进行分段校验；

5.30.7. Process 处理流程

略

5.30.8.Pseudo-code 伪代码

```
int MuFTAD_Download2XX(int fd, off_t offset, SegmnetNode Ptr, int resume)
{

    char *query;
    temp[256];
    char *addr;
    ServerHeadPtr server = MuServiceInfo->server;

    //if need to resume, we must checksum the segment
    if(resume&&Mu_Md5Segment(fd, offset, Ptr->length, Ptr->md5) == MUOK)
        return MUOK;

again:
    sprintf(temp, “%s/%s\0”, MuSegmentHead->path, Ptr->name);
    if(MuFTAD_MakeURL(&query, MuSegmentHead->method, temp) < 0)
        return MUEERO;

    //query server!
    Mu_DnStatInit(MuDnStat);
```

```
    MuDnStat->type = MU_GET;
    MuDnStat->fd = fd;
    MuDnStat->resume = resume;
    MuDnStat->startops = offset;
    MuDnStat->length = Ptr->size;
    ret = Mu_HttpPlus(query, MU_DOWNLOAD, MuDnStat);
    free(query);

    if(ret == MUECON){
        if(server->cur->next != NULL)
            server->cur = server->cur->next;
        else
            server->cur = server->head;
        goto again;
    }

    if(ret != MUOK){
        MuFTAD_ErrorProc(ret);
        return MUERRO;
    }

    if(Mu_Md5Segment(fd, offset, Ptr->length, Ptr->md5) < 0)
        return MUECKS;

    return MUOK;
}
```

5.30.9. Interface 接口

略

5.30.10. Unsolved 未解决问题

略

5.31. MuFTAD_Download4XX

5.31.1. Name 函数名称

int MuFTAD_Download4XX(const char *url, int authmethod, int resume)

5.31.2. Description 函数描述

本函数下载4XX 节目内容，对于该部分内容，采取直接下载的方式；

5.31.3. Function 功能

略

5.31.4. Input 输入

url：地址，不包括服务器地址；
authmethod：验证方式；
resume：同5.30.中说明；

5.31.5. Output 输出

函数的运行状态，包括：
1、无错，以 MUOK 返回；

5.31.6. Arithmetic 算法

参考5.30.中说明

5.31.7. Process 处理流程

略

5.31.8. Pseudo-code 伪代码

```
int MuFTAD_Download4XX(const char *url, int authmethod, int resume)
{
    char *query;
    temp[256];
    char *addr;
    int i;
    int fd ;
    ServerHeadPtr server = MuServiceInfo->server;

    if(MuFTAD_MakeURL(&query, authmethod, url) < 0)
        return MUEERO;

    char *ptr = strrchar(url, '/');
    sprintf(temp, "%s.mu\0", Ptr + 1);
    if((fd = open(temp, O_RDWR|O_CREATE)) < 0)
        return MUEERO;
    MuFTAD_RecordResum(MuResume, type, Ptr +1, &i);

again:
    //query server!
    Mu_DnStatInit(MuDnStat);
    MuDnStat->type = MU_GET;
    MuDnStat->fd = fd;
    MuDnStat->resume = resume;
    ret = Mu_HttpPlus(query, MU_DOWNLOAD, MuDnStat);
    free(query);

    if(ret == MUECON){
        if(server->cur->next != NULL)
            server->cur = server->cur->next;
        else
            server->cur = server->head;
        goto again;
    }
    Mu_Close(fd);

    if(ret != MUOK){
        MuFTAD_ErrorProc(ret);
        return MUERRO;
    }

    rename(temp, Ptr + 1);
```

```
    MuFTAD_DeleteResume(MuResume, i);
    return MUOK;
}
```

5.31.9. Interface 接口

本函数在实现上述功能时，还用到了其他函数，例如：
MuFTAD_RecordResum：该函数首先在 MuResume 数组中记录相关信息，然后写入 resume.xml 文件中；

MuFTAD_DeleteResum：该函数首先清除在 MuResume 数组中记录的相关信息，然后更新 resume.xml 文件；

5.31.10. Unsolved 未解决问题

略

5.32. MuFTAD_Resume

5.32.1. Name 函数名称

int MuFTAD_Resume(void)

5.32.2. Description 函数描述

在存在需要续传的文件时，需要分类处理，一类是2XX 类型节目，另一类是4XX 类型节目；

函数解析 resume.xml 文件内的内容，得到尚未下载完全的节目列表；函数根据未下载完全节目的类型，选择合适的续传函数处理各个节目；

5.32.3. Function 功能

略

5.32.4. Input 输入

无

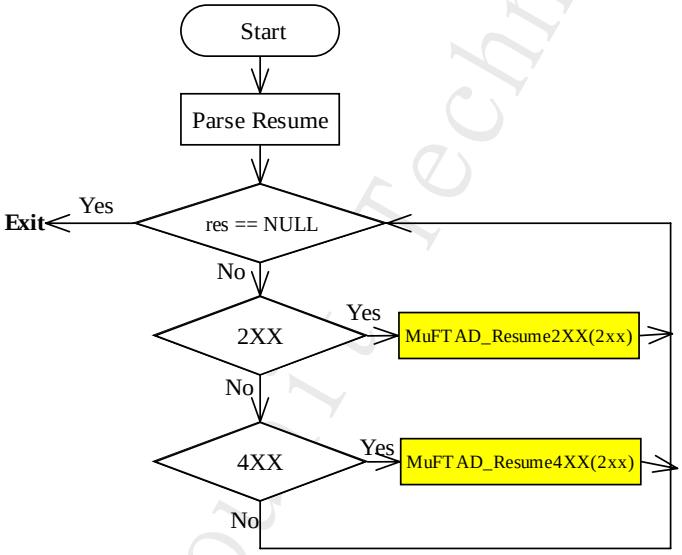
5.32.5. Output 输出

函数的运行状态，包括：
1、无错，以 MUOK 返回；

5.32.6.Arithmetic 算法

1、函数解析 resume.xml 文件，从中取出未下载完全的节目类型和节目名称；
2、根据节目类型，选择不同的续传函数处理续传；

5.32.7. Process 处理流程



5.32.8. Pseudo-code 伪代码

```
int MuFTAD_Resume(void)
{
    int fd;

    if((fd = open(RESUME_FILE, O_RDONLY)) < 0)
        return MUEERO;

    Mu_Parse(PARSE_RESUME_INFO, fd, NULL, MuResume);

    for(int i = 0; i < 10; i++){
        if(CHECK_2XX(MuResume[i].type))
            MuFTAD_Resume2XX(i);
        else if(CHECK_4XX(MuResume[i].type))
            MuFTAD_Resume4XX(i);
    }

    return MUOK;
}
```

5.32.9. Interface 接口

略

5.32.10. Unsolved 未解决问题

略

5.33. MuFTAD_Resume2XX

5.33.1. Name 函数名称

int MuFTAD_Resume2XX(int i)

5.33.2. Description 函数描述

函数根据文件名，处理节目的续传；

在续传处理时，需要对2XX 类型和4XX 类型的节目区别对待，在2XX 节目下载时，会首先创建该节目的描述文件，再下载；

所以，在处理2XX 节目续传时，需要根据节目名，找出节目的描述文件，根据描述文件内的校验字段来检验文件的完整性；

若已下载部分不完整，需要续传未完成部分；

完成更新后，向服务器更新已完成节目；

5.33.3. Function 功能

处理未完成的2XX 节目的续传；

5.33.4. Input 输入

i：用于在 MuResume 中找到未下载完全文件名；

5.33.5. Output 输出

函数处理的状态码，包括：
1、无错，以 MUOK 返回；
2、

5.33.6. Arithmetic 算法

参考《MuFTAD 需求说明书》文档中关于 Resume 部分的说明介绍

5.33.7. Process 处理流程

略

5.33.8. Pseudo-code 伪代码

```
int MuFTAD_Resume2XX(int i)
{
```

```
    int ret;
    char temp[256];
    FddfNodePtr Ptr;
    Mu_FddfStatusNodeCreate(&Ptr);

    ret = MuFTAD_Resume2XXCheck(Ptr, MuResume[i].name);
    if(ret == MUEXML){
        goto error;
    }else if(ret == MUECNT){
        memset(temp, 0, 256);
        srpintf(temp, "%s.mv\0", name);
    }else if(ret == MUOK)
        return ret;

    if((fd = open(temp, RDWR)) < 0)
        goto error;

    MuFTAD_Query111(Ptr, 902, 1);

    Mu_FddfStatusNodeDelete(Ptr);
    return MUOK;

error:
    sprintf(temp, "rm –rf %s.*\0", name);
    system(temp);
    Mu_FddfStatusNodeDelete(Ptr);
    return MUOK;
}
```

批注 [ssw39]: 第三个参数说明续传

5.33.9. Interface 接口

函数在实现时，需要用到 MuFTAD_Resume2XXCheck 函数解析 resume.xml 文件，并且对文件内容进行校验；

5.33.10. Unsolved 未解决问题

略

5.34. MuFTAD_Resume2XXCheck

5.34.1. Name 函数名称

int MuFTAD_Resume2XXCheck(FddfStatusNodePtr Ptr, const char *name)

5.34.2. Description 函数描述

检测未下载文件是否不完整，存在几种不同的情况：
1、存在 name.mv.mu，对文件以不完整对待；
2、不存在 name.mv.mu，但是存在 name.mv，进行 MD5校验，以结果判断是否完整；

在本函数中还必须对 name.xml 函数进行解析，以获得其中的相关参数；

5.34.3. Function 功能

确定不完整节目的文件名及确认不完整性；

5.34.4. Input 输入

Ptr ： FddfStatusNode 结构指针，用于存储相关信息，以便于调用 MuFTAD_Query111续传节目；

name：文件名

5.34.5. Output 输出

函数运行状态码，包括：
1、解析 XML 错，以 **MUEXML** 返回；
2、name.mv 不完整，以 **MUEXML** 返回；
3、name.mv.mu，以 **MUETMP** 返回；

5.34.6. Arithmetic 算法

略

5.34.7. Process 处理流程

略

5.34.8. Pseudo-code 伪代码

```
int MuFTAD_Resume2XXCheck (const char *name)
{
    temp[256];
    int ret ;
    int fd;

    sprintf(temp, "%s.xml\0", name);
    if(access(temp, F_OK|R_OK) == -1){
        fprintf(stdout, "have no fireware to update!\n");
        return MUEXML;
    }

    Mu_FileInfoInit(&MuFileInfo);

    //if the update xml is exist, then read it , parse it's contents to structure!
    if((fd = open(temp, RD_ONLY)) < 0
        ||Mu_Parse(PARSE_FILE_INFO, fd, NULL, &MuFileInfo) < 0){
        fprintf(stdout, "can't open the xml! shall clear the directory!\n");
        return MUEXML;
    }

    Mu_Close(fd);

    MuFTAD_FileInfoToFddf(MuFileInfo, Ptr);
    memset(temp, 0, 256);
    sprintf(temp, "%s.mv\0", name);
    if((ret = access(temp, F_OK|R_OK)) == -1){
        memset(temp, 0, 256);
        sprintf(temp, "%s.mv.mu\0", name);
        if((fd = open(temp, RD_ONLY)) < 0
            ||Mu_Md5File(fd, MuFileInfo->md5) < 0){
            fprintf(stdout, "can't open the xml! shall clear the directory!\n");
```

批注 [ssw40]: 初始化 MuFileInfo 结构，该结构的初始化应该在解析中自动完成

批注 [ssw41]: 该函数用 MuFileInfo 结构体的成员构建 MuFddfListNode 结构，以便于调用 MuFTAD_Query111

```
            Mu_Close(fd);
            return MUEXML;
        }
    }else if(ret == 0){
        return MUOK;
    }

    return MUOK;
}
```

5.34.9.Interface 接口

函数在完成上述功能时，需要利用 MuFTAD_FileINfoToFddf 函数将解析地的文件内容信息拷贝到 FddfstatusNode 结构中，以调用 MuFTAD_Query111下载节目；

5.34.10. Unsolved 未解决问题

略

5.35. MuFTAD_Resume4XX

5.35.1. name 函数名称

int MuFTAD_Resume4XX(int i)

5.35.2. Description 函数描述

参考5.33.2.中说明

5.35.3. Function 功能

略

5.35.4. Input 输入

同5.33.2.中说明

5.35.5. Output 输出

函数运行状态，包括：
1、无错，以 MUOK 返回；

5.35.6. Arithmetic 算法

函数从下载函数为续传所保留的文件中读取 URL，向服务器发起续传请求；

5.35.7. Process 处理流程

略

5.35.8. Pseudo-code 伪代码

```
int MuFTAD_Resume4XX(int i)
{
    char temp[256];
    char url[1024];
    int fd;

    sprintf(temp, “%s.jw”, MuResume[i].name);
    if((ret = access(temp, F_OK|R_OK)) == -1)
        goto error;

    if((fd = open(temp, O_RDONLY)) < 0)
        goto error;
    if(read(fd, &urllen, sizeof(int)) < 0)
        goto error;

    //read the url from the file, and compare the url
    //make sure the address resuming is the same;
```

```
    if((read(fd , url, urllen) <= 0)
        fprintf(stdout, “wrong url\n”);
        goto error;
    }
    Mu_Close(fd);

    sprintf(temp, “%s.mv.mu\0”, MuResume[i].name);
    if((ret = access(temp, F_OK|R_OK)) == -1){
        memset(temp, 0, 256);
        sprintf(temp, “%s.mv\0”, MuResume[i].name);
        if((ret = access(temp, F_OK|R_OK)) == -1)
            goto error_end;

    }

    if((fd = open(temp, O_RDONLY)) < 0)
        goto error;

    Mu_DnStatInit(MuDnStat);
    MuDnStat->type = MU_GET;
    MuDnStat->fd = fd;
    MuDnStat->resume = resume;
    ret = Mu_HttpPlus(url, MU_DOWNLOAD, MuDnStat);

    if(ret < 0)
        goto error;

    MuFTAD_DeleteResume(MuResume, i);
    return MUOK;

error:
    sprintf(temp, “rm –rf %s.*”, MuResume[i].name);
    system(temp);

error_end:
    MuFTAD_DeleteResume(MuResume, i);
    return MUEERO;
}
```

5.35.9. Interface 接口

略

5.35.10.Unsolved 未解决问题

略

5.36. MuFTAD_DnLoadFireware

5.36.1. Name 函数名称

int MuFTAD_DnLoadFireware(void)

5.36.2. Description 函数描述

设备在登录服务器时，如果设备的固件版本过低，服务器会要求设备更新相应的固件；

在服务器返回的信令中，最多只可能包含一个更新内容；

对于新的固件，设备必须下载其到./update 文件夹中，并且创建相应的描述 XML 文件，以便于开机时更新固件；

5.36.3. Function 功能

略

5.36.4. Input 输入

无

5.36.5. Output 输出

函数的运行状态，包括：
1、无错，以 MUOK 返回；

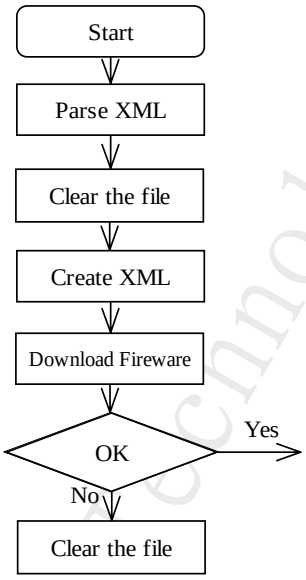
5.36.6. Arithmetic 算法

在下载新固件前，必须清空./update 文件夹内的所有内容，以避免产生冲突；另外，还必须首先创建固件描述 XML；

下载固件至本地，下载固件的名称，直接使用下载文件名称创建本地固件名；

若下载出错，清空./update 中的所有内容；

5.36.7. Process 处理流程



5.36.8. Pseudo-code 伪代码

```
#define UPDATESRV “update_srv.xml”
int MuFTAD_DnLoadFireware(void)
{
    int fd;
    char *query
    temp[256];
    ServerHeadPtr server = MuServiceInfo->server;
```

```
    if(Mu_Parse(PARSE_UPDATE_INFO, rcvfd, NULL, &MuUpdateInfo) < 0)
        return MUEERO;

    sprintf(temp, "rm –rf ./update/*\0");
    system(temp);

    sprintf(temp, "./update/%s\0", UPDATESRV);
    if((fd = open(temp, O_CREATE)) < 0
        || Mu_BuildXML(BUILD_UPDATE_FILE, fd,
                                        MuUpdateInfo, NULL) < 0)
        goto error;
    Mu_Close(fd);

    char *ptr = strrchar(MuUpdateInfo->name, '/');
    sprintf(temp, "./UPDATESRV/%s \0", Ptr + 1);
    if((fd = open(temp, O_CREATE)) < 0)
        goto error;

again:
    if(MuFTAD_MakeURL(&query, authmethod, url) < 0)
        goto error;

    //query server!
    Mu_DnStatInit(MuDnStat);
    MuDnStat->type = MU_GET;
    MuDnStat->fd = fd;
    ret = Mu_HttpPlus(query, MU_DOWNLOAD, MuDnStat);
    free(query);

    if(ret == MUECON){
        if(server->cur->next != NULL)
            server->cur = server->cur->next;
        else
            server->cur = server->head;
        goto again;
    }
    Mu_Close(fd);

    if(ret != MUOK){
        MuFTAD_ErrorProc(ret);
        goto error;
    }

    return MUOK;
```

批注 [ssw42]: 该结构针对100与101结构，同用一个结构体

```
error:
    sprintf(temp, “rm –rf ./update/*\0”);
    system(temp);
    return MUEERO;
}
```

5.37. main

5.37.1. Name 函数名称

int main(void)

5.37.2. Description 函数描述

本函数是整个项目的主函数，调用项目中所有的函数接口，按照《MuFTAD 需求说明书》文档中所描述的流程处理法电协议中所描述的所有任务；

5.37.3. Input 输入

无

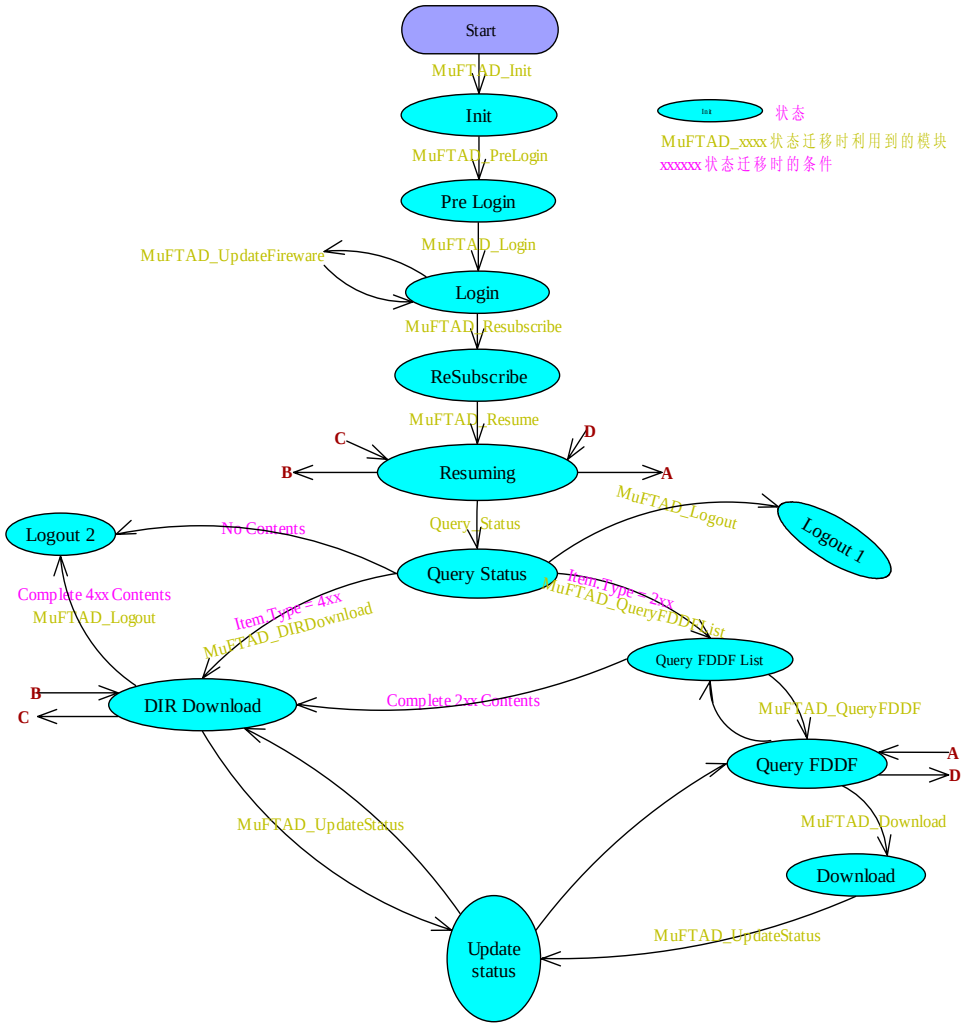
5.37.4. Output 输出

函数运行状态

5.37.5. Arithmetic 算法

参考《MuFTAD 需求说明书》

5.37.6. Process 处理流程



5.37.7. Pseudo-code 伪代码

```
int main()
{
    MuFTAD_DataInit();
    MuFTAD_DeviceInit();
    MuFTAD_PreLogin();
    MuFTAD_Login();
    MuFTAD_Resume();
```

批注 [ssw43]: 完成初始化

批注 [ssw44]: 预登录

批注 [ssw45]: 登录服务器

批注 [ssw46]: 续传，续传中包括解析 resume.xml 文件，并且重传未完成部分

```
    MuFTAD_Query110();
    MuFTAD_Query110Proc();
    return 0;
}
```

批注 [ssw47]: 待没有重传内容时，需要查询服务器

批注 [ssw48]: 对查询得到的结果进行处理，包括2XX 和4XX 处理，下载节目，更新状态等

6. Appendix 附录

6.1. Routines 函数

6.1.1. Mu_DnStatInit

```
int Mu_DnStatInit(MuDnStatPtr Ptr)
{
    //init structure of the MuDnStat
}
```

6.1.2. MuFTAD_ErrorProc

```
int MuFTAD_ErrorProc(int error)
{
    //Process the error code,which the function return
    //print all the information about the error code, except the 200,300, 301,405

}
```

6.1.3. MuFTAD_CopyFddfToDBII

```
int MuFTAD_CopyFddfToDBII(FddfStatNodePtr, DBIInfoPtr)
{
    //use FddfStatNode to create DBIInfo
}
```

6.1.4. MuFTAD_CopyFddfToDBID

```
int MuFTAD_CopyFddfToDBUD(FddfStatNodePtr, DBUDirtPtr)
{
    //use FddfStatNode to create DBUDirt
}
```

6.1.5. MuFTAD_CopyFddfToDBUR

```
int MuFTAD_CopyFddfToDBUR(FddfStatNode, DBURatePtr)
{
    //use FddfStatNode to create DBURate
}
```

6.1.6. MuFTAD_UpdateTime

```
int MuFTAD_UpdateTime(const char*time)
{
    //according the value of time
    //setting the time of the device time
}
```

6.1.7. MuFTAD_UpdateID

```
int MuFTAD_UpdateID(const char*, const char *)
{
    //update the device ID_Int &device ID_App
    //U must change the MuEyewearADPsnd, DeviceInfo and device.xml

}
```

6.1.8. MuFTAD_UpdateConv

```
int MuFTAD_UpdateConv(int,    const char, const char *)
{
    //Update the conversation in MuEyewearADPsnd
    //one of the value MAYBE set to GOLABLE value
}
```

6.1.9. MuFTAD_RecordResum

```
int MuFTAD_RecordResum(Resume, int , const char *,    int *)
{
    //Before download the program,
    //U must insert the information about the program(just a type value and name)
    //into the array of MuResume[]
    //and Update the Muresume into the file resume.xml
```

}

6.1.10. MuFTAD_DeleteResum

```
int MuFTAD_DeleteResum(Resume, int)
{
    //If download the programs successfully, We
    //should delete the information about the program
    //from the array of MuResum[]
    //and Update the Muresume into the file resume.xml
}
```