编　　制	沈胜文
审　　核	
批　　准	
实施责任人	

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	2008-3-19	
102	修改		沈胜文	2008-4-11	

Table Of Contents 目录

1. Introduction 简介

1.1. Objective 编写目的

本文档是在《XML 生成概要设计》的基础上，就 XML 生成进行的详细设计说明。本文档将尽可能地详细，全面地说明创建 XML 的方式和开发步骤，实现流程。

但是设计与实际开发之间的不符之处，以实际代码为准，但是所作修改，需经过讨论，并记入本文档。

[注意]：
1、本文档在第一次修改时，已经不再按照《XML 生成概要设计》进行；

1.2. Background 背景

本程序是 MuFTAD 项目的一部分，MuFTAD 为法电项目的软件实现部分，本扩展库主要为项目提供自动下载过程中所必须的 XML（信令）创建功能。

开发的软件（扩展库），局限于本项目，不可被其他项目所使用。

本软件的提出者：沈胜文
本软件的开发者：沈胜文
本软件的用　户：MuFTAD 项目

1.3. Terms and Abbreviation 术语与缩写解析

Terms&Abbreviation 术语&缩写	Description 解释
XML	XML 即可扩展标记语言（eXtensible Markup Language）。标记是指计算机所能理解的信息符号，通过此种标记，　计算机之间可以处理包含各种信息的文章等。
MuFTAD	软件名称 Microunit France-Telecom Auto-Download

1.4. Reference Material 参考资料

《XML 数据结构详细设计》

2. Rules 规则

2.1. Name Rules 命名规则

本程序的设计考虑使用扩展库的形式来实现。

该程序中，所有的函数均以 Mu_的形式开头，而不是 MuFTAD_。因此在该程序内，所有的函数形如：Mu_XXXX()；

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:
*
*         XMLBuild.c
*
* Description:
*
*         These functions build a XML in the buffer or write it into a file.
*
* Revision History:
*
*         19-3-2008 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。

可参考模板

```
/*****************************************************************
*Description:
*       This Function is Build the XML, return the Informations to caller use the
*       Value pointer;
*Input:
*       fileDescriptor: the file fd , which will be used to stored the XML contents
*Output:
*       Pointer: which is a pointer, point to the buffer stored the XML
*LOCK:
*       NONE
*Modify:
*       ssw (fzqing@gmail.com   10-3-2008)
*****************************************************************/
```

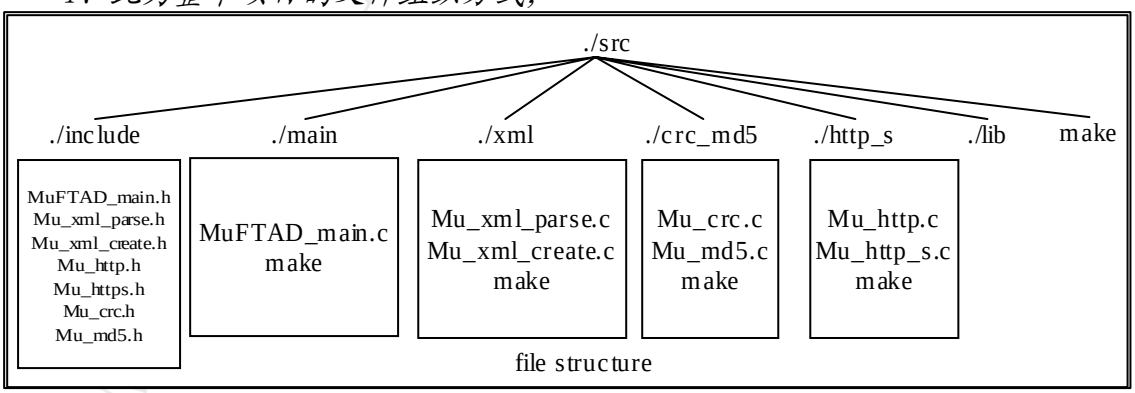
2.4. Library 库

待开发的扩展库是并不基于其他特殊库，在普通 Linux 主机上均可运行。

2.5. File Structure 文件结构

[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

调用者根据不同的 XML 类型，传递不同的参数给创建函数接口，函数选择不同的创建模块来完成 XML 创建;

创建完成的 XML 文件被存储于内存中，并且向调用者返回存储 XML 的地址和长度;

若需要将创建的 XML 内容写入文件，调用者必须提供一个待写入文件的文件句柄;

[注意]:

1、具体的说明介绍可以参考《XML 生成概要设计》;

3.2. Routine List 函数列表

```
int Mu_BuildXML(int type, FILE *fp, void *Ptr1, void *Ptr2)
int Mu_BuildDeviceInfo(FILE *fp, DeviceInfoPtr Ptr1)
int Mu_BuildUpdateInfo(FILE *fp, UpdateInfoPtr Ptr1)
int Mu_BuildFileDes(FILE *fp, FddfFilePtr Ptr1);
int Mu_BuildResume(FILE *fp, ResumeInfoPtr Ptr1);
int Mu_BuildSignal(int type, FILE *fp, void *Ptr1, void *Ptr2);
int Mu_BuildSignalQuery(int type, FILE *fp, void *Ptr1, void *Ptr2)
```

```
int Mu_BuildSignalUpDn(int type, FILE *fp, void *Ptr1, void *Ptr2)
 int Mu_BuildSignalUpDn902(FILE *fp,EyewearADPPtr Ptr1, DBIInfoPtr Ptr2)
int Mu_BuildSignalUpDn903(FILE *fp,
                          EyewearADPPtr Ptr1, DBUDirtPtr Ptr2);
int Mu_BuildSignalQuery20X(FILE *fp,
                          EyewearADPPtr Ptr1, DBUInfoPtr Ptr2);
int Mu_BuildSignalQuery902(FILE *fp, EyewearADPPtr Ptr1,
                          DBIInfoPtr Ptr2);
```

4. Global Description 全局描述

4.1. Global Type 全局描述

[注意]:
1、该类型与 XML 解析所需要的定义一样，被定义在 xml.h 中;

4.1.1. Macro 宏

```
#define BUILD_SERV_INFO 1
#define BUILD_UPDATE_INFO 2
#define BUILD_DEVICE_INFO 3
#define BUILD_FILE_DES 4

#define BUILD_SIGNAL_ADP 15
```

用于标识构建普通无 payload 信令;

```
#define BUILD_SIGUP_FILE 23
#define BUILD_SIGUP_DN902 159
#define BUILD_SIGUP_DN903 223
```

用于标识构建状态更新信令;

```
#define BUILD_SIGQUE_20X 295
#define BUILD_SIGQUE_902 167
```

用于标识构建查询信令;

```
#define IS_SERVINFO(x) (((x)&0x0F)==1)
#define IS_UPDATEINFO(x) (((x)&0x0F)==2)
#define IS_DEVICEINFO(x) (((x)&0x0F)==3)
#define IS_FILEDES(x) (((x)&0x0F)==4)
```

用于检测出调用者想要创建的 XML 类型

```
#define IS_SIGNAL(x) (((x)&0x0F)==7)
#define IS_SIGADP(x) ((((x)&0x0F0)>>4)==1)
#define IS_SIGUPFILE(x) ((((x)&0x0F0)>>4))==2)
#define IS_SIGUPDN(x) ((((x)&0x0F0)>>4)==3)
#define IS_SIGQUE(x) ((((x)&0x0F0)>>4)==4)
```

用于检测调用者想要创建的信令类型

```
#define IS_SIG901(x) ((((x)&0x0F00)>>8)==1)
#define IS_SIG902(x) ((((x)&0x0F00)>>8)==2)
#define IS_SIG903(x) ((((x)&0x0F00)>>8)==3)
#define IS_SIG20X(x) ((((x)&0x0F00)>>8)==4)
```

用于检测出调用者想要创建的信令中的 payload 字段类型

```
#define MU_WRITE_ELEMENT(fp, string)\
do{\
    fprintf(fp, “%s”, string);\
    fflush(fp);\
}while(0)
```

在 XML 中创建一个空元素标记

```
#define WRITE_HEAD(fp) \
do{\
    fprintf(fp, “<?xml version="1.0" encoding="UTF-8" ?>\n”);\
    fflush(fp);\
}while(0)
```

写 XML 文件的声明部分

4.1.2. Structure 结构体

```
typedef struct _DBURate{
    int filerate;
    int index;
    char *program;
    char *group;
    char *owener;
}DBURate, *DBURatePtr;

typedef struct _DBUInfo{
    int newer;
    int older;
    char *program;
    char *group;
```

```
    char *owner;
    int status;
}DBUInfo, *DBUInfoPtr;

typedef struct _DBUDirt{
    char *download;
    char *dat1;
    char *dat2;
    char *dat3;
    char *dat4;
    char *dat5;
    int status;
}DBUDirt, *DBUDirtPtr;
```

[注意]:
1、项目中用到的其他数据结构请参考《MuFTAD 数据结构详细设计》;

4.2. Global Error 全局错误码

参考《XML 解析详细设计》中4.2.章节中说明

5. Routines Details 函数细节

5.1. Mu_BuildXML

5.1.1. Name 函数名称

int Mu_BuildXML(int type, FILE *fp, void *Ptr1, void *Ptr2)

5.1.2. Description 函数描述

该函数是创建 XML 函数的接口，也是本扩展库向调用者提供的唯一函数接口；

函数按照调用者传递的 type 值，从四个参数中选择合适的，可用的值传递给真正意义上的 XML 创建函数；

创建完成的 XML 被存放于文件内，该接口函数同时需要调用者提供二个参数，以便提供创建的 XML 信息；

5.1.3. Function 功能

将调用者想要创建的 XML 分配给不同的创建函数，函数本身不做任何初始化，创建等工作；

5.1.4. Capability 性能

5.1.5. Input 输入

该函数接受四个参数，包括一个整型，一个文件句柄，二个指针；

type：用于表明调用者将要创建的 XML 文件，调用者在调用本函数时，必须准确地填充该参数值；

[注意]:

1、该参数值的具体内容，可参考4.1.1. 章节所描述;

fp：创建的 XML 文件需要写入文件，因此，需要提供一个可写入的文件句柄；

[注意]:

1、fp 为一个文件流，而不是一个 file Descriptor !

Ptr1，Ptr2：该参数是二个无类型指针参数，分别指向二个结构体，用于向创建函数传递 XML 元素各项的值；

这二个参数具体的类型依 type 而确定；

5.1.6. Output 输出

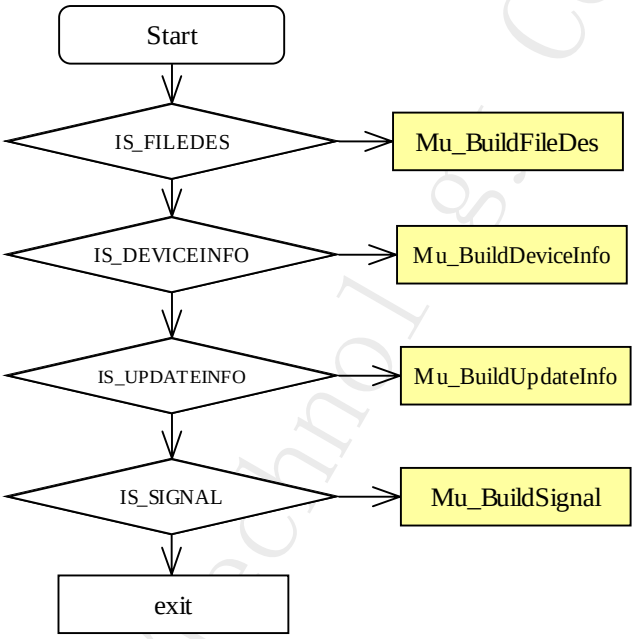
函数运行状态码；

要求处理的错误，包括**拷贝错，写文件错，超过长度和参数不合法**等等情况；

5.1.7. Arithmetic 算法

略

5.1.8. Process 处理流程



[注意]:
1、黄色部分由其他函数完成;

IS_XXX 是4.1.1.中定义的宏;
Mu_BuildXXX 是本函数调用的 XML 创建函数;

5.1.9. Pseudocode 伪代码

```
int Mu_BuildXML(int type, FILE *fp, void *Ptr1, void *Ptr2)
{

    retval = MUOK;

    if(IS_FILEDES(type)){
        retval = Mu_BuildFileDes(fp, (FddfFilePtr)Ptr1);
        return retval;
```

```
}else if(IS_DEVICEINFO(type)){
    retval = Mu_BuildDeviceInfo(fp, (DeviceInfoPtr)Ptr1);
    return retval;
}else if(IS_UPDATEINFO(type)){
    retval = Mu_BuildUpdateInfo(fp, (UpdateInfoPtr)Ptr1);
    return retval;
}else if(IS_SIGNAL(type)){
    retval = Mu_BuildSignal(type, fp, void *Ptr1, void *Ptr2);
    return retval;
}else
    return MUNBLD;

}
```

5.1.10. Interface 接口

本函数为主体创建函数接口，在创建 XML 时，还利用到其他函数：Mu_BuildXXX 系统 XML 创建函数；

5.2. Mu_BuildDeviceInfo

5.2.1. Name 函数名称

int Mu_BuildDeviceInfo(FILE *fp, DeviceInfoPtr Ptr1)

5.2.2. Description 函数描述

该函数是创建设备信息描述 XML 文件的函数接口；
函数根据 Ptr1中的参数值，填充该 XML 文件中的各元素信息；

5.2.3. Function 功能

创建 Device Information Description XML 文件；

5.2.4. Capability 性能

同5.1.4. 中说明；

5.2.5. Input 输入

同5.1.5. 中说明

5.2.6. Output 输出

同5.1.6. 中说明

5.2.7. Arithmetic 算法

略

5.2.8. Process 处理流程

按该 XML 中各元素字段顺序添加;

5.2.9. Pseudocode 伪代码

```
int Mu_BuildDeviceInfo(FILE *fp, DeviceInfoPtr Ptr1)
{
    if(NULL == Ptr1){
        do error;
        return MUEERO;
    }

    //seek to the start of the file
    lseek(fd, 0, SEEK_SET);
    WRITE_HEAD(fp);

    MU_WRITE_ELEMENT( fp, "<DeviceInfo>\n");
    if(Mu_WriteStrToXML(2, fp, "Device_ID_Int", Ptr1->Int) < 0
        ||Mu_WriteStrToXML(2, fp, "Device_ID_App", Ptr1->App) < 0
        ||Mu_WriteIntToXML(2, fp,
                    "Device_Fireversion", Ptr1->Hardwareversion)< 0){

        fprintf(stdout, "build device info error!\n");
        return MUERRO;
    }
```

```
    MU_WRITE_ELEMENT( fd, "</DeviceInfo>");
    fflush(fd);
    return MUOK;
}
```

5.2.10. Interface 接口

该函数在提供创建 Device Information Description XML 文件功能的同时，利用了其他宏定义；

WRITE_HEAD；
MU_WRITE_ELEMENT；
Mu_WriteIntToXML；
Mu_WriteStrToXML；

5.2.11. Malloc 存储分配

5.2.12. Restrict 限制

该函数使用了 Mu_WriteIntToXML 和 Mu_WriteStrToXML 二个函数来写 XML，上述二个函数使用了局部数组变量，默认为51字节；

5.2.13. Test 测试

创建生成的 XML 文件必须是 UTF-8码的，对这部分的校验可以使用我们开发的 XML 解析扩展库，解析本扩展库生成的 XML 文件；

必须测试 XML 元素字段值的长度对整个函数的影响；
由于字段过长，可能会导致内存溢出。

5.2.14. Unsolve 未解决情况

略

5.3. Mu_BuildUpdateInfo

同5.2. 中说明

需要创建函数
int Mu_BuildUpdateInfo(FILE *fp, UpdateInfoPtr Ptr1)

5.4. Mu_BuildFileDes

同5.2. 中说明
需要创建函数
int Mu_BuildFileDes(FILE *fp, FddfFilePtr Ptr1);

5.5. Mu_BuildResume

同5.2.中说明
需要创建函数
int Mu_BuildResume(FILE *fp, ResumeInfoPtr Ptr1);

5.6. Mu_BuildSignal

5.6.1. Name 函数名称

int Mu_BuildSignal(int type, FILE *fp, void *Ptr1, void *Ptr2);

5.6.2. Description 函数描述

函数不进行任何 XML 创建操作，它仅仅根据 type 的类型，选择正确的创建函数；
功能同5.1.章节 Mu_BuildXML 函数；

5.6.3. Function 功能

按类型选择合适的构建函数；

5.6.4. Capability 性能

同5.1.4. 中说明；

5.6.5. Input 输入

同5.1.5. 中说明

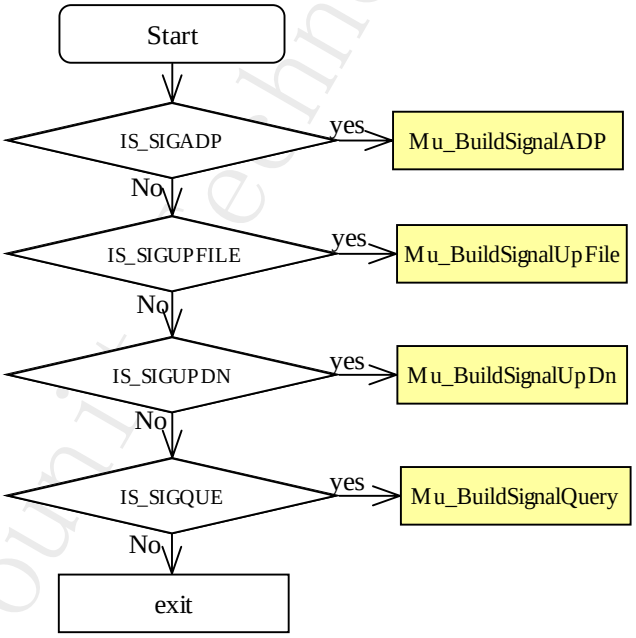
5.6.6. Output 输出

同5.1.6. 中说明

5.6.7. Arithmetic 算法

略

5.6.8. Process 处理流程



[注意]:
1、黄色部分需要其他函数完成;

IS_XXX 是4.1.1.中定义的宏;
Mu_BuildXXX 是本函数调用的 XML 创建函数;

5.6.9. Psuedocode 伪代码

```
int Mu_BuildSignal(int type, FILE *fp, void *Ptr1, void *Ptr2)
{
    int retval = MUOK;

    if(IS_SIGQDP(type)){
        retval = Mu_BuildSignalADP(fp, (EyewearADPPtr)Ptr1);
        return retval;
    }else if(IS_SIGUPFILE(type)){
                                    retval = Mu_BuildSignalUpFile(fp,
                    (EyewearADPPtr)Ptr1, (DBURatePtr)Ptr2);
        return retval;
    }else if(IS_SIGUPDN(type)){
        retval = Mu_BuildSignalUpDn(type, fp, Ptr1, Ptr2);
        return retval;
    }else if(IS_SIGQUE(type)){
        retval = Mu_BuildSignalQuery(type, fp, Ptr1, Ptr2);
        return retval;
    }else
        return MUEERO;

}
```

5.6.10. Interface 接口

本函数为主体创建函数接口，在创建 XML 时，还利用到其他函数：Mu_BuildSignalXXX 系统 XML 创建函数；

5.6.11.Malloc 存储分配

同5.1.11.中说明；

5.6.12. Restrict 限制

略

5.6.13.Test 测试

略

5.6.14. Unsolve 未解决情况

略

5.7. Mu_BuildSignalQuery

5.7.1. Name 函数名称

int Mu_BuildSignalQuery(int type, FILE *fp, void *Ptr1, void *Ptr2)

5.7.2. Description 函数描述

本函数并不进行创建 XML 的任何操作，它仅仅根据 type 类型，选择合适的创建函数；

5.7.3. Function 功能

选择适当的函数创建 XML 文件；

5.7.4. Capability 性能

同5.1.4. 中说明；

5.7.5. Input 输入

同5.1.5. 中说明

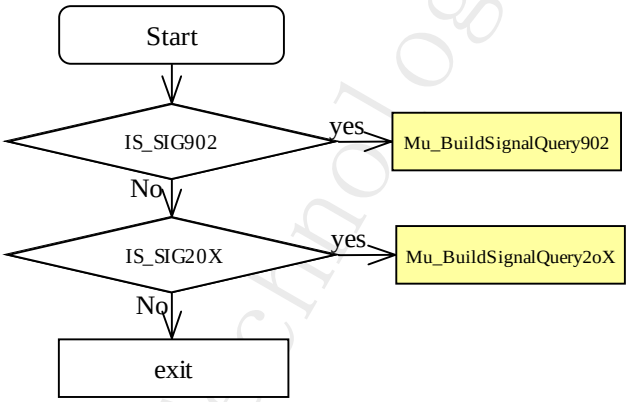
5.7.6. Output 输出

同5.1.6. 中说明

5.7.7. Arithmetic 算法

略

5.7.8. Process 处理流程



[注意]:
1、黄色部分需要其他函数完成;

IS_XXX 是4.1.1.中定义的宏;
Mu_BuildSignalQueryXXX 是本函数调用的 XML 创建函数;

5.7.9. psuedocode 伪代码

```
int Mu_BuidlSignalQuery(int type, FILE *fp, void *Ptr1, void *Ptr2)
{
    int retval = MUOK;

    if(IS_SIG902(type)){
        retval = Mu_BuildSignalQuery902(fp, (EyewearADPPtr)Ptr1,
                                        (DBIInfoPtr)Ptr2);
        return retval;
    }else if(IS_SIG20X(type)){
        retval = Mu_BuildSignalQuery20X(fp, (EyewearADPPtr)Ptr1,
```

```
                                    (DBUInfoPtr)Ptr2);
        return retval;
    }else
        return MUEERO;

}
```

5.7.10. Interface 接口

本函数为主体创建函数接口，在创建 XML 时，还利用到其他函数：Mu_BuildSingalQueryXXX 系统 XML 创建函数；

5.7.11.Malloc 存储分配

同5.1.11.中说明；

5.7.12. Restrict 限制

略

5.7.13.Test 测试

略

5.7.14. Unsolve 未解决情况

略

5.8. Mu_BuildSignalUpDn

5.8.1. Name 函数名称

int Mu_BuildSignalUpDn(int type, FILE *fp, void *Ptr1, void *Ptr2)

5.8.2. Description 函数描述

本函数并不进行创建 XML 的任何操作，它仅仅根据 type 类型，选择合适的创建函数；

5.8.3. Function 功能

选择适当的函数创建 XML 文件；

5.8.4. Capability 性能

同5.1.4. 中说明；

5.8.5. Input 输入

同5.1.5. 中说明

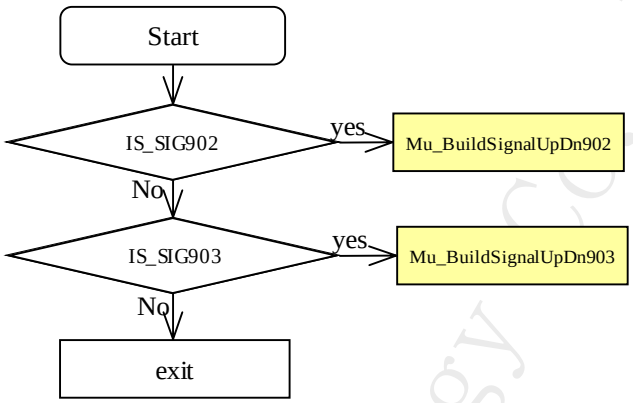
5.8.6. Output 输出

同5.1.6. 中说明

5.8.7. Arithmetic 算法

略

5.8.8. Process 处理流程



[注意]:
1、黄色部分需要其他函数完成;

IS_XXX 是4.1.1.中定义的宏;
Mu_BuildSignalUpDnXXX 是本函数调用的 XML 创建函数;

5.8.9. psuedocode 伪代码

```
int Mu_BuidlSignaUpDn(int type, FILE *fp, void *Ptr1, void *Ptr2)
{
    int retval = MUOK;

    if(IS_SIG902(type)){
        retval = Mu_BuildSignalUpDn902(fp, (EyewearADPPtr)Ptr1,
                                DBIInfoPtr Ptr2);
        return retval;
    }else if(IS_SIG2903(type)){
        retval = Mu_BuildSignalUpDn903(fp, (EyewearADPPtr)Ptr1,
                                (DBUDirtPtr)Ptr2);
        return retval;
    }else
        return MUEERO;

}
```

5.8.10. Interface 接口

本函数为主体创建函数接口，在创建 XML 时，还利用到其他函数：Mu_BuildSingalUpDnXXX 系统 XML 创建函数；

5.8.11.Malloc 存储分配

同5.1.11.中说明；

5.8.12. Restrict 限制

略

5.8.13.Test 测试

略

5.8.14. Unsolve 未解决情况

略

5.9. Mu_BuildSignalUpDn902

5.9.1. Name 函数名称

int Mu_BuildSignalUpDn902(FILE *fp, EyewearADPPtr Ptr1, DBIInfoPtr Ptr2)

5.9.2. Description 函数描述

本函数用于生成带有 payload 字段的更新信令，Device 用该信令向服务器更新2XX 类节目的下载状态；

5.9.3. Function 功能

函数按 CMD 格式，并且利用 Ptr1和 Ptr2二个指针的参数，逐字段创建 XML 文件；

5.9.4. Capability 性能

同5.1.4. 中说明；

5.9.5. Input 输入

同5.1.5. 中说明

5.9.6. Output 输出

同5.1.6. 中说明

5.9.7. Arithmetic 算法

略

5.9.8. Process 处理流程

按该 XML 中各元素字段顺序添加；

5.9.9. Pseudocode 伪代码

```
int Mu_BuildSignalUpDn902(FILE *fp, EyewearADPPtr Ptr1, DBIInfoPtr Ptr2)
{
    if((NULL ==Ptr1) || (NULL == Ptr2)){
        do error;
        return MUEERO;
    }

    lseek(fd, 0, SEEK_SET);
    WRITE_HEAD(fp);
```

```
MU_WRITE_ELEMENT(fp, "<Eyewear_ADP>\n");
if(Mu_WriteIntToXML(2, fp, "Protocol_Ver", Ptr1->Protocolver) < 0
    ||Mu_WriteIntToXML(2,fp,  "Protocol_OPC", Ptr1->protocolOPC) < 0)
    goto error;

MU_WRITE_ELEMENT(fp, "\t<PreLogin>\n");
if(Mu_WriteIntToXML(3, fp,
                "EncryptionMethod", Ptr1->prelogin->encrptionmethod)< 0
   ||Mu_WriteStrToXML(3,fp,  "Challenge", Ptr1->prelogin->challenge)< 0)
    goto error;

 MU_WRITE_ELEMENT(fp, "\t<PreLogin>\n");
MU_WRITE_ELEMENT(fp, "\t<Device_Info>\n");

if(Mu_WriteStrToXML(3, fp,
              "Device_ID_Int", Ptr1->Device_info->DeviceIDInt) < 0
    ||Mu_WriteStrToXML(3, fp,
                "Device_ID_APP", Ptr1->Device_info->DeviceIDApp) < 0
    ||Mu_WriteIntToXML(3, fp,
                "Device_Status", Ptr1->Device_info->Status)< 0
    ||Mu_WriteIntToXML(3, fp,
              "Trig_Resource", Ptr1->Device_info->trigresource) < 0
    ||Mu_WriteStrToXML(3, fp,
                    "SRV_Date_time", Ptr1->Device_info->srvdatetime)< 0
    ||Mu_WriteIntToXML(3, fp,
                "Fireware_Version", Ptr1->Device_info->firewareversion) < 0
     ||Mu_WriteIntToXML(3, fp,
              "Hardware_Version", Ptr1->Device_info->hardwareversion) < 0)
     goto error;

MU_WRITE_ELEMENT( fp, "\t</Device_Info>\n");
MU_WRITE_ELEMENT(fp, "\t<Conversation>\n");

if(Mu_WriteStrToXML(3, fp,
             "ConversationID", Ptr1->Conversation->conversationID)< 0
    ||Mu_WriteStrToXML(3, fp,
             "AuthUsername", Ptr1->Conversation->authusername)< 0
    ||Mu_WriteStrToXML(3, fp,
             "AuthPassword", Ptr1->Conversation->authpassword)< 0)
    goto error;

MU_WRITE_ELEMENT(fp, "\t</Conversation>\n");
MU_WRITE_ELEMENT( fp, "\t<Payload>\n");
MU_WRITE_ELEMENT(fp, "\t\t<Item>\n");
```

```
    if(Mu_WriteIntToXML(4, fp, “Type”, 902)< 0
        ||Mu_WriteStrToXML(4, fp, “Text”, “DBU_Info”) < 0
        ||Mu_WriteIntToXML(4, fp, “Size”, Ptr2->size)<0
        ||Mu_WriteIntToXML(4, fp,  “Dat1”, Ptr2->Newer)<0
        ||Mu_WriteIntToXML(4, fp, “Dat2”, Ptr2->Older)<0
        ||Mu_WriteStrToXML(4, fp, “Dat3”, Ptr2->program)<0
        ||Mu_WriteStrToXML(4, fp,  “Dat4”, Ptr2->group)<0
        ||Mu_WriteStrToXML(4, fp, “Dat5”, Ptr2->owner)<0
        ||Mu_WriteIntToXML(4, fp, “Dat6”, Ptr2->status)<0)
        goto error;

    MU_WRITE_ELEMENT(fp, “\t\t</Item>\n”);
    MU_WRITE_ELEMENT(fp, “\t</Payload>\n”);
    MU_WRITE_ELEMENT(fd,  “</Eyewear_ADP>\n”);

    fflush(fp);
    return MUOK;

error:
    fpritnf(stdout, “when build signal upload error!\n”);
    return MUEERO;
}
```

5.9.10. Interface 接口

本函数为主体创建函数接口，在创建 XML 时，还利用到其他函数：Mu_WriteIntToXML 和 Mu_WriteStrToXML；

5.9.11.Malloc 存储分配

同5.1.11.中说明；

5.9.12. Restrict 限制

略

5.9.13.Test 测试

同5.2.13中说明

5.9.14. Unsolve 未解决情况

略

5.10. Mu_BuildSignalUpDn903

创建函数
int Mu_BuildSignalUpDn903(FILE *fp, EyewearADPPtr Ptr1, DBUDirtPtr Ptr2);

详细参考5.9.

5.11. Mu_BuildSignalQuery20X

创建函数
int Mu_BuildSignalQuery20X(FILE *fp, EyewearADPPtr Ptr1, DBUInfoPtr Ptr2);

详细参考5.9.

5.12. Mu_BuildSignalQuery902

创建函数
int Mu_BuildSignalQuery902(FILE *fp, EyewearADPPtr Ptr1, DBIInfoPtr Ptr2);
详细参考5.9.

6. Appendix 附录

6.1. Assistant Routines 辅助函数

6.1.1. Mu_WriteIntToXML

```
int Mu_WriteIntToXML(int class, FILE *fp, const char *element, int val)
{
    char temp[51];
```

```
    memset(temp, 0, 51);

    //add tabstop
    while(--class)
        strncat(temp, "\t", 1);

    if(strlen(temp) >= 50)
        return MUNBUF;

    if(val)
        fprintf(fp, "%s<%s>%d</%s>\n", temp, element, val, element);
    else
        fprintf(fp, "%s<%s></%s>\n", temp, element, element);

    fflush(fp);
    return MUOK;
}
```

6.1.2 Mu_WriteStrToXML

```
int Mu_WriteStrToXML(int class, FILE *fp, const char *element, const char *val)
{
    char temp[51];
    memset(temp, 0, 51);

    //add tabstop
    while(--class)
        strncat(temp, "\t", 1);

    if((strlen(temp)) >= 50)
        return MUNBUF;

    fprintf(fp, "%s<%s>%s</%s>\n", temp, element, (val?val:""), element);

    fflush(fp);
    return MUOK;
}
```

6.2. XML

- 00_ServerInfo.xml
  服务器信息描述符，假设 Device 连接上的是 Localhost

- 02_Before_Login.xml
  登录信令

- 04_Before_Query.xml
  查询信令

- 06_Before_QueryFDDF.xml
  FDDF 查询信令（OPC=111）

- 08_Before_Update.xml
  状态更新信令

- device information.xml
  设备描述 XML

- file_info.xml
  节目描述 XML

- resume.xml
  未完成和正在下载节目信息

- update_SRV.xml
  更新的固件的信息