

GBBExplorer记录阅读文档

Widget类，主要是绘制界面的类，界面展示如下：

GBBExplorer

全部关闭

主窗口

Entities					Messages					Descriptors							
	GBB	Entities	数量	最大	%		GBB	Message	数量	最大	%		GBB	Descriptor	数量	最大	%
1	146	Bomb	0	5000	0	1	4304	SetFire...	0	100	0	1	309	Aggrega...	0	5000	0
2	145	OtherSa...	0	1000	0	2	4276	SetTurr...	0	100	0	2	308	Reprint...	0	3000	0
3	144	EarthSa...	0	1000	0	3	4265	SetFood...	0	100	0	3	307	Running...	0	1	0
4	143	AutoSca...	0	10	0	4	4263	SetCGFC...	0	100	0	4	306	AutoSca...	0	10	0
5	142	AutoScale	0	100	0	5	4256	SetEngi...	0	100	0	5	305	AutoSca...	0	100	0
6	141	Control...	0	20	0	6	4255	SetBall...	0	100	0	6	304	FirePoint	0	5000	0
7	132	Offboar...	0	100	0	7	4254	SetForm...	0	500	0	7	303	TowingS...	0	5000	0
8	131	Bouy	0	500	0	8	4253	SetRequ...	0	100	0	8	302	TowedSe...	0	2000	0
9	130	SpoofDe...	0	1000	0	9	4248	SetCGFM...	0	100	0	9	301	LOSArea...	0	39010	0
10	129	FireDet...	0	2000	0	10	4245	SetPush...	0	100	0	10	300	Collisi...	0	3000	0
11	128	Balloon	0	500	0	11	4244	SetPull...	0	100	0	11	299	Terrain...	0	5000	0
12	127	GroundA...	0	5000	0	12	4233	SetTurr...	0	100	0	12	298	Control...	0	20	0
13	126	Airborn...	0	1000	0	13	4232	SetMode...	0	100	0	13	297	Joystic...	0	20	0
14	125	MarineA...	0	200	0	14	4230	SetHuma...	0	100	0	14	296	SimpleG...	0	3000	0
15	124	Node	0	10	0	15	4229	SetExte...	0	100	0	15	294	Detonat...	0	5000	0
16	123	Injecti...	0	500	0	16	4228	SetEnti...	0	1000	0	16	293	Initial...	0	5000	0
17	122	Attritio	0	10000	0	17	4225	SetInje...	0	10	0	17	292	Iteratio	0	1	0
18	121	Magneti...	0	1000	0	18	4224	SetArea...	0	1000	0	18	291	HumanCl...	0	3000	0
19	120	IFFDete...	0	12000	0	19	4223	SetComm...	0	100	0	19	290	PathToF...	0	3000	0
20	118	TaskFor...	0	10	0	20	4218	SetComm...	0	100	0	20	289	MotionC...	0	5000	0
21	117	TaskForce	0	10	0	21	4203	SetExpe...	0	100	0	21	288	MotionA...	0	5000	0
22	116	CGFUnit...	0	2000	0												

detail类，实体跳转界面的具体展示

GBBExplorer

全部关闭

主窗口 Entity-Chair

Descriptors

描述符

>

EntityInfo

>

EntityIdentification

>

GeoPosAndVel

>

Orientation

>

ParentId

>

CommanderData

>

Entity ID

>

ActionData

>

Source

>

Execution Time

>

Action ID

>

Params

>

CommandedId

刷新

PushButton

全选

清除所有

Entities

实体名称

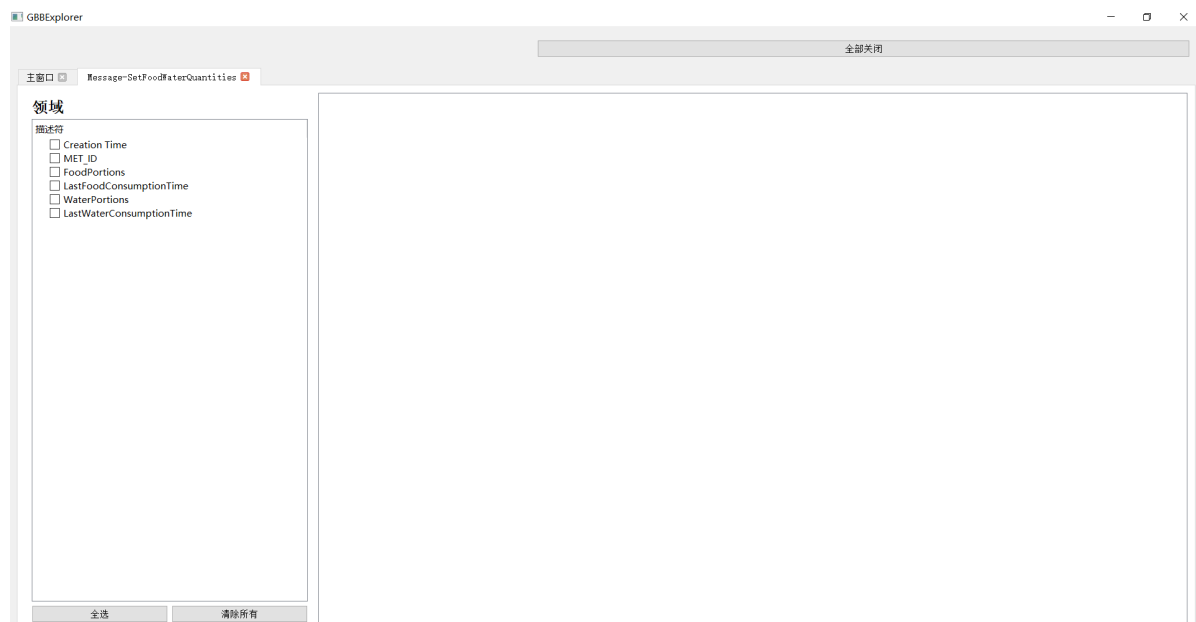
全选

清除所有

刷新

过滤器

detailmessage类，消息跳转界面展示



在Widget.h中定义的函数和槽函数如下，实现在Widget.cpp里

具体的一些声明和注释如下

```
1 private:
2     void initForm(); //初始化主窗口
3
4
5 private slots:
6     void on_tableviewdoubleClicked(const QModelIndex &index); //双击主页
    tableview上的名称跳转显示详情
7     void on_removebtn_clicked(int index); //删除标签
8     void on_pushButton_8_clicked(); //详情页全选子项目按钮的实现
9     void on_pushButton_7_clicked(); //清除按钮功能实现
10    void on_treewidget_2_clicked(QTreeWidgetItem *item); //treewidget_2选中/不
    选中触发事件，模拟GBBExplorer中选择与取消
11    void on_treewidget_clicked(QTreeWidgetItem *item); //treewidget选中进行全部
    的行显示
```

StaticData类，主要是初始化静态数据

主要参照实现了CHSim-TKE_GBBExplorer\Infra\GBBExplorer\GBBExplorer文件夹中staticData.cs和entity.cs等定义的结构，从SerializedBuffer中获取静态数据

```
1 void InitStructures(); // Create the Structures Static List
2 void InitDescriptors(); // Create the Descriptors Static List
3 void InitEntities(); // Create the Entities Static List
4 void InitMessages(); // Create the Messages Static List
5 int SetStringFromPtr(char* CurrentIntPtr, std::string &StringName);
```

main函数中

```

1 //用于日志记录和链接初始化GBB平台
2 LoggerUtil::Init();
3     if (theConfigManager.Load("UI"))
4     {
5         theConfigManager.SetApplicationArgs(argc, argv);
6         if (theProcessHelper->startNotificationEngine() == SUCCESS
7             &&theProcessHelper->waitForBlackboardToStart() == SUCCESS
8             &&theMonitorManager.Init())

```

已进行静态页面的展示

GBB Explorer

主窗口 详细窗口 关闭所有窗口

Entities						Messages						Descriptors					
	GBB	Entities	数量	最大	%		GBB	Message	数量	最大	%		GBB	Descriptor	数量	最大	%
1	1	Agent	0	100	0	1	51	Request...	0	100	0	1	69	Acousti...	0	100	0
2	2	SNAConn...	0	10	0	2	52	Allocat...	0	1000	0	2	172	ActiveS...	0	1000	0
3	51	IDAlloc...	0	1	0	3	53	ReleaseID	0	100	0	3	173	ActiveS...	0	1000	0
4	53	DBAreaE...	0	2000	0	4	55	Message...	0	100	0	4	171	ActiveS...	0	1000	0
5	54	Referen...	0	2000	0	5	56	PlayerC...	0	10	0	5	266	ActualI...	0	5000	0
6	56	DBRoute...	0	200	0	6	57	LoggerC...	0	10	0	6	1	AgentInfo	0	100	0
7	57	Referen...	0	200	0	7	59	NewRoute	0	100	0	7	2	AgentSt...	0	100	0
8	58	Mission	0	1	0	8	62	NewArea	0	100	0	8	309	Aggrega...	0	5000	0
9	59	Environ...	0	100	0	9	64	CreateE...	0	100	0	9	117	Airborn...	0	5000	0
10	60	RawEven...	0	1000	0	10	65	AddToMi...	0	100	0	10	242	Ammunit...	0	1000	0
11	61	Format...	0	1000	0	11	66	RemoveF...	0	100	0	11	57	AreaData	0	2000	0
12	62	UIForma...	0	1000	0	12	67	MPComma...	0	100	0	12	59	AreaDat...	0	200	0
13	63	Terrain...	0	100	0	13	68	SystemL...	0	10	0	13	224	AreasOf...	0	5000	0
14	66	Acousti...	0	10	0	14	69	DeleteE...	0	300	0	14	220	Attriti...	0	10000	0
15	67	ActiveS...	0	1000	0	15	72	AddEvent	0	100	0	15	306	AutoSca...	0	10	0
16	75	Chaff	0	500	0	16	74	AddDeco...	0	1000	0	16	305	AutoSca...	0	100	0
17	77	CommInt...	0	2000	0	17	75	Request...	0	100	0	17	255	Balloon...	0	500	0
18	80	Communi...	0	200	0	18	76	Allocat...	0	1000	0	18	86	BrainIn...	0	5000	0
19	81	Communi...	0	200	0	19	101	SetTime	0	100	0	19	85	BrainSt...	0	5000	0
20	82	Communi...	0	5000	0	20	102	SetOper...	0	100	0	20	217	C2BLTar...	0	5000	0
21	83	ECM	0	1000	0	21	103	Synchro...	0	100	0	21	273	C2Status	0	5000	0
22	84	ECM	0	5000	0												

在widget.cpp里对获得的数组vecinfo进行迭代展示代码如下，以entity为例

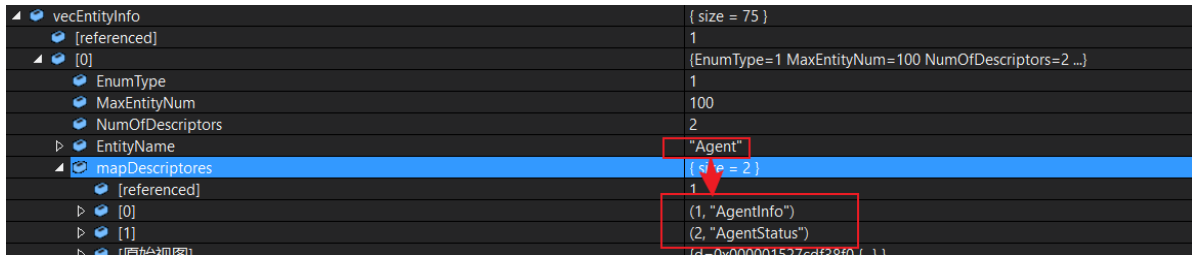
```

1 //初始化实例
2     StaticData staticdata;
3
4     staticdata.InitStructures();
5
6     //读取静态实体数据显示
7     staticdata.InitEntities();
8     for (int i = 0; i < staticdata.vecEntityInfo.size(); i++)
9     {
10         QString EnumType =
11         QString::number(staticdata.vecEntityInfo[i].EnumType);
12         QString EntityName =
13         QString::fromStdString(staticdata.vecEntityInfo[i].EntityName);
14         QString MaxEntityNum =
15         QString::number(staticdata.vecEntityInfo[i].MaxEntityNum);
16         model->setItem(i, 0, new QStandardItem(EnumType));
17         model->setItem(i, 1, new QStandardItem(EntityName));
18         model->setItem(i, 2, new QStandardItem("0"));
19         model->setItem(i, 3, new QStandardItem(MaxEntityNum));
20         //double rate = 0;
21         model->setItem(i, 4, new QStandardItem("0"));
22     }

```

各个数据关联情况

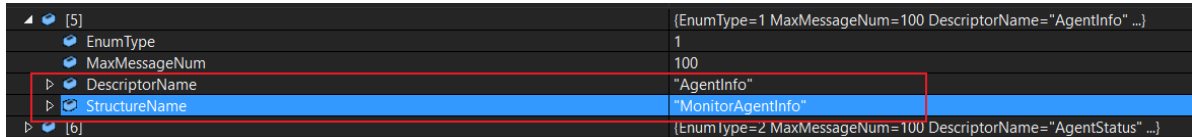
- Entity点击详情，以agent为例：



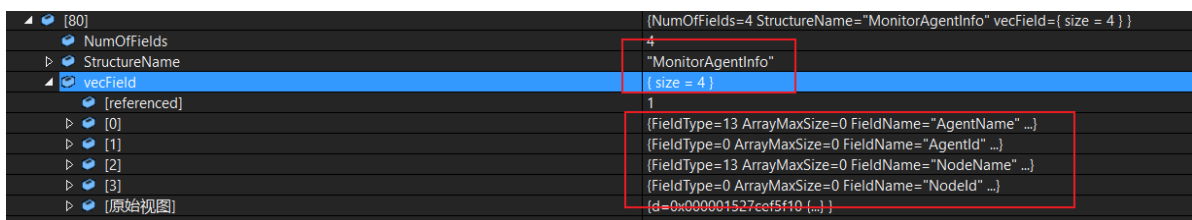
mapdescriptors是对应的2个根节点：



再去vecdescriptors中找到对应name的structure



根据 structname 在vecstruct中找子节点



- message详情，以requestnewid为例：

根据消息名称，在M_MessageInfo结构中找到descriptoname，再和上面一样去vecdescriptors中找到对应name的structure，做多重展示

注意，Creation time是手动添加的，逻辑在MessageView.cs中，MET_ID是根据m_bIsDescAsMessage添加的



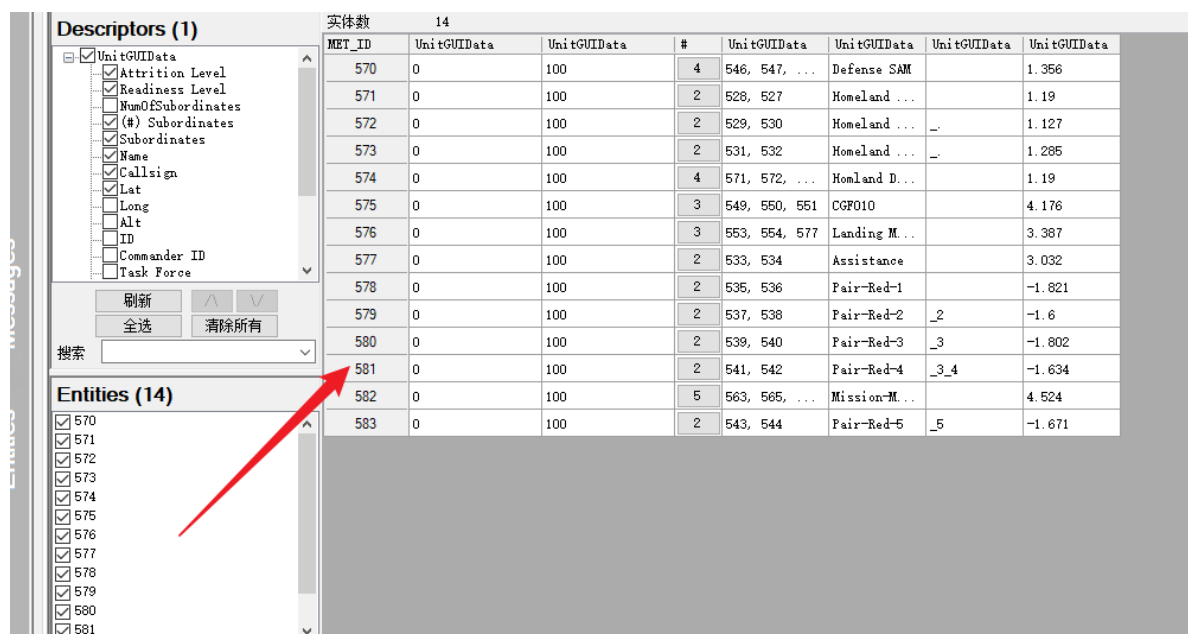


动态数据获取，在DynamicData类里

`bool GetEntitiesIDs(enum_t eEntityType);` 每个周期，获取实体的METID

`bool GetEntityCount(enum_t eEntityType);` 每个周期，动态获取实体数量

用于获取entity里的详情，如下图



`bool GetEntityDynamicData(id_t idEntity);`用于根据METID获取对应id的动态数据

`bool GetEntityDynamicData(id_t idEntity, void* pDesNumbersVoid);`这个pDesnumber不清楚是做什么的，应该传什么参呢

调用情况：

dynamicdata.cs中：

```
216         if (CurrentDataTable.Rows[ElementIndex].Tag != null) // If this is ORANGE row (deleted entity)
217         {
218             continue;
219         }
220         // Send to GBBMonitorManager.dll -
221         // 1.List of Entities need to read
222         // 2.List of descriptors number to read
223         // and get pointer from to the Data
224         try
225         {
226             DataBufferPtr = ImportFunction.GetEntityData(EntityID[ElementIndex], m_pDescriptorsIntPtr);
227         }
228         catch
229         {
230             // Write to error to log
231             string ErrorString = "";
232             for (int i = 0; i < Descriptors.Count; ++i)
233             {
234                 ErrorString += Descriptors[i].m_sName + ",";
235             }
236         }
237     }
```

补充源码里的cpp定义

```
41
42
43 extern "C" __declspec(dllexport) void* GetEntityData(int nEntityID, void* pDescriptorsEnumType)
44 {
45     if (theMonitorManager.GetEntityDynamicData(nEntityID, pDescriptorsEnumType))
46     {
47         return (void*)theMonitorManager.GetSerializedBuffer()->GetBuffer();
48     }
49
50     return NULL;
51 }
```

gbbmonitorwrapper.cpp里的定义

```
Widget.cpp * GBBMonitorWrapper.cpp * x GBBMonitorManag...nFunctions.cpp Widget.h StaticData.h StaticData.cpp main.cpp
190 strcpy(pEmptyStr12_3, "00003");
191 return buffer;
192 }
193
194 extern "C" __declspec(dllexport) void* GetEntityData(int nEntityID, void* pDescriptorsEnumType)
195 {
196     unsigned char* buffer = new unsigned char[200];
197     int* pEntity1 = (int*)buffer;
198     *pEntity1 = rand() % 6;
199
200     int* pEntity2 = (int*)(pEntity1+1);
201     *pEntity2 = rand() % 6;
202
203     int* pEntity3 = (int*)(pEntity2+1);
204     *pEntity3 = rand() % 6;
205 }
```

`bool GetEntityEnumType(long pEntityMet_ID);`获取enumtype

已经在DynamicData类里实现

```

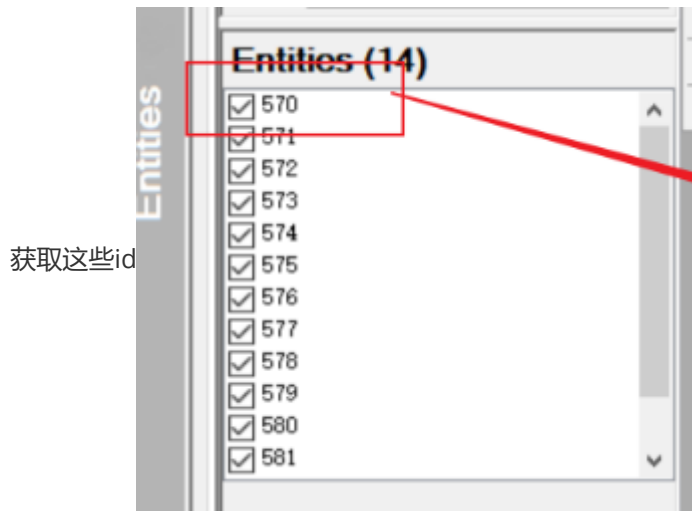
33
34 public:
35     DynamicData();
36     //DynamicData(QString configPath);
37     int GetEntityCount(int eEntityType); //每个周期，动态获取某个实体数量
38     void GetEntitiesIDs(int eEntityType); //每个周期，获取实体的所有metid
39     int GetEntityEnumType(long pEntityMet_ID); //每个周期，根据metid获取属于哪个实体enum
40     int GetMessageCount(int eMessageType); //每个周期获取某个消息的数量
41     int GetDescriptorCount(int eDescriptorType); //每个周期获取某个描述符的数量
42     //void GetEntityDynamicData(int nEntityID); //根据id获取动态数据
43     void GetEntityDynamicData(int eEntityType); //获取某个实体的全部动态数据
44     //bool ReadFieldFromPtr(char* allTheTablePtr, StaticData::M_FieldInfo currentField, )
45     ~DynamicData();
46

```

void GetEntitiesIDs(int eEntityType); //每个周期，获取实体的所有metid

比如：点击

116	CGFUnitGUIData	14	2000	0.
-----	----------------	----	------	----



GetEntityDynamicData的结构

```

DynamicData
GetEntityDynamicData(int eEn

int num = staticdata.vecDescriptorsInfo.size();
int m = 0;
//为每个描述符在分配的空间中占位，以enumtype
for (int i = 0; i < num; ++i) {
    *(int*)(m_descriptorPtr + m) = staticdata.vecDescriptorsInfo[i].EnumType;
    m += sizeof(int);
}
*(int*)(m_descriptorPtr + m) = -1;

GetEntitiesIDs(eEntityType); //每个周期，获取对应id实体的所有metid
int num2 = EntitiesId.size();
for (int i = 0; i < num2; i++) {
    if (theMonitorManager.GetEntityDynamicData(EntitiesId[i], m_descriptorPtr)) {
        GBBMonitor::SerializedBuffer* p = theMonitorManager.GetSerializedBuffer();
        char* ptr = (char*)p->GetBuffer();
        //对应的DataBufferPtr
    }
}

```

可以理解成下面，其中描述符指针按enumtype进行占位（空间由xml中读取）

Entity - CUnitGUIData A

Descriptors (1)

- ☒ UnitGUIData
- ☒ Attrition Level
- ☒ Readiness Level
- ☐ NumOfSubordinates
- ☐ (#) Subordinates
- ☐ Subordinates
- ☐ Name
- ☐ Callsign
- ☐ Lat
- ☐ Long
- ☐ Alt
- ☐ ID
- ☐ Commander ID
- ☐ Task Force

刷新 ^ v

全选 清除所有

搜索

Entities (14)

- ☒ 570
- ☒ 571
- ☒ 572
- ☒ 573
- ☒ 574
- ☒ 575
- ☒ 576
- ☒ 577
- ☒ 578
- ☒ 579
- ☒ 580
- ☒ 581

全选 清除所有

实体数 14

MET_ID	UnitGUIData
570	0
571	0
572	0
573	0
574	0
575	0
576	0
577	0
578	0
579	0
580	0
581	0
582	0
583	0

getentitydynamicdata, 根据前面获得的id+描述符指针, 获取数据, 展示到结构上

查找结果 1

查找全部 "GetEntityDynamicData", 子文件夹, 查找结果 1, "chsin", ""

D:\CHSIN-TKE_CBBExplorer\Infra\GEMonitorManager\GEMonitorManager.h(188): bool GetEntityDynamicData(id_t idEntity)

D:\CHSIN-TKE_CBBExplorer\Infra\GEMonitorManager\GEMonitorManager.h(190): bool GetEntityDynamicData(id_t idEntity, void* pDesNumbersVoid).

D:\CHSIN-TKE_CBBExplorer\Infra\GEMonitorManager\GEMonitorManager.h(208): * GetEntitiesID and GetEntityDynamicData.

D:\CHSIN-TKE_CBBExplorer\补充的源码\GEMonitorManager\ExternFunctions.cpp(45): if (theMonitorManager.GetEntityDynamicData(nEntityID, pDescriptorsEnumType))

匹配文件数: 2 已搜索文件总数: 78351

实现里的重要函数理解 ReadRowFromIntPtr

```

char* pBuffer = ptr;
int NumOfBuffer = *(int*)(ptr1); ptr1 += sizeof(int); //buffer长度
//TODO dynamicdata.cs中 GetEntityTableData等函数使用到的 ReadRowFromIntPtr函数功能
char a[10000];
for (int i = 0; i < 10000; i++) {
    a[i] = *ptr1;
    ptr1++;
}
qDebug() << "hello";

```

核心逻辑

```

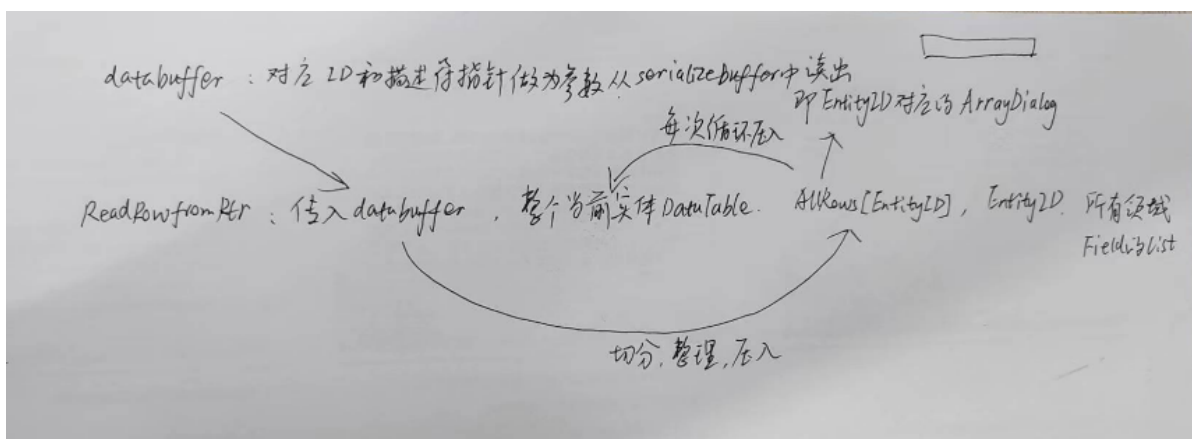
}

m_nCurrentPos = 0;
m_nBufferLength = Get32Bit(ref DataBufferPtr);

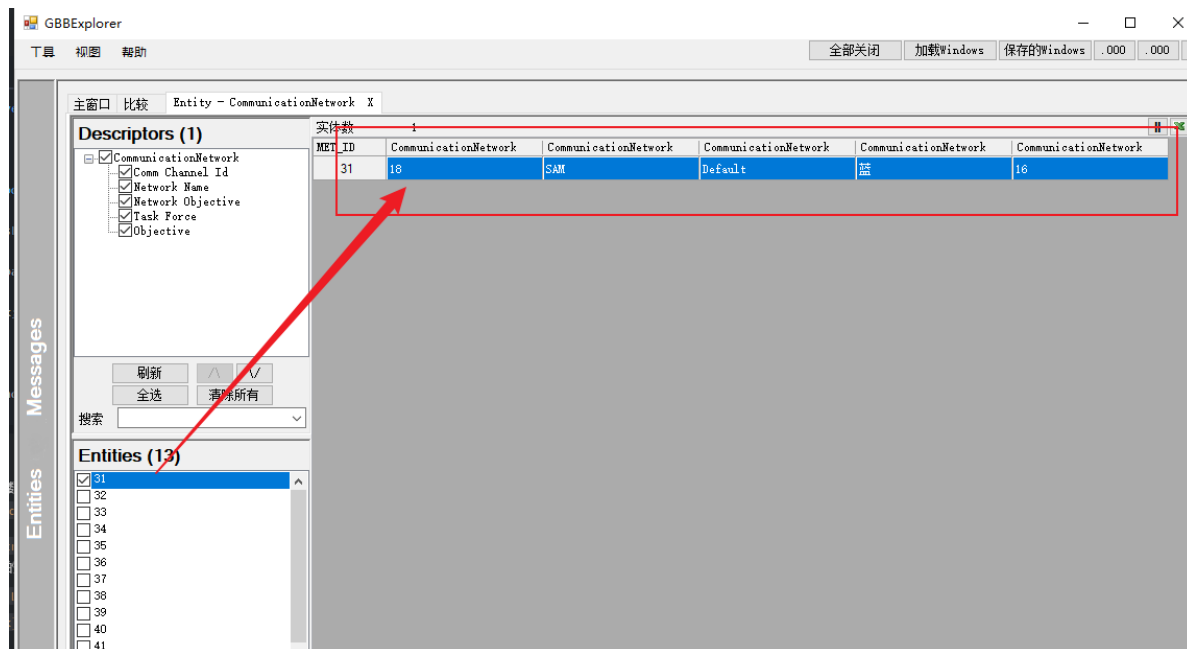
ReadRowFromIntPtr(ref DataBufferPtr, ref CurrentDataTable, AllRowsArrayColumnIndex[ElementIndex], ElementIndex, F

//Byte[] temp = new byte[10000];
//int index = 0;

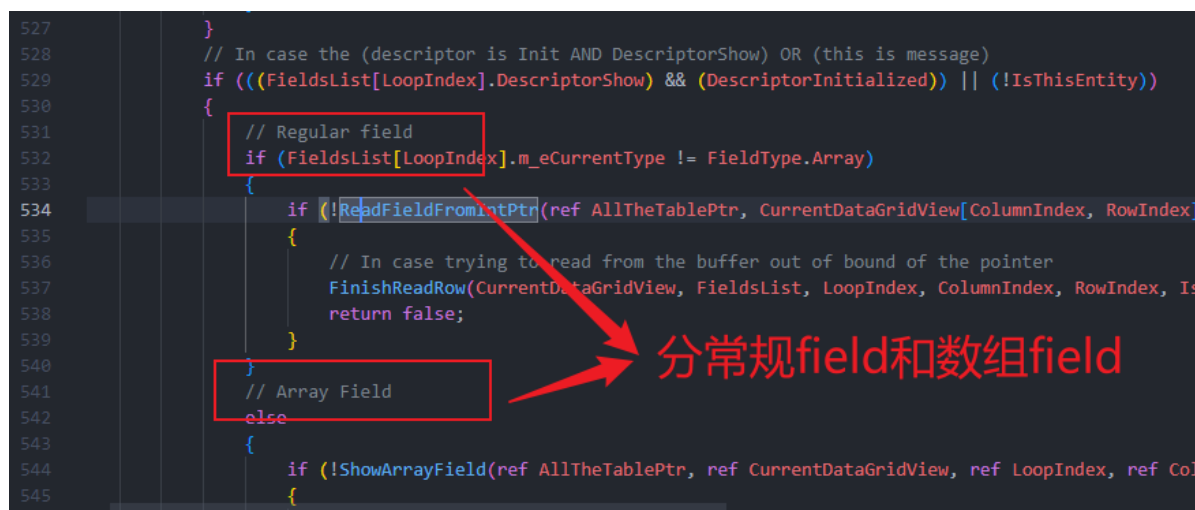
```



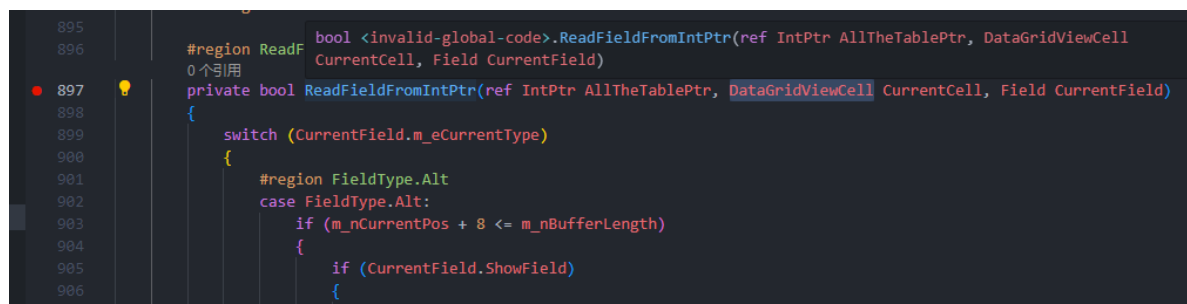
实际上是实现了切分获得下面这一行数据的功能



ReadRowFromIntPtr 中



对于常规field，readfieldFromPtr函数根据fieldtype进行切割划分，得到值，以enum=81为例，选中id=31的实体，得到一个bufferlength长度的buffer（每一行都是这样），第一个字节是bool值，标志是否是开始描述符，接着按照不同类型切分



核心：showArrayField函数解析

deshandler里定义了各个field的结构描述和添加

```
h DataAddingHelper.h 4 98
h GBBMonitorManagerExport.h 1 99
GBBMONITORMANAGER 100
> x64 101
h DataAddingHelper.h 4 102
h DescHandler.h 2 103
h-- DescHandler.inl 104
h GBBMonitorManager.h 7 105
h GBBMonitorManagerExport.h 1 106
h SerializedBuffer.h 4 107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123

std::vector<FieldData> m_vecFields;

/**
 * Reads the descriptor data, returning it as a char*, and write the status of the descriptor
 * to the serialized buffer (0 for invalid, 1 for valid)
 */
char* ReadData();

protected:

void AddAckMessage(char* pData);

void AddFieldInt(const std::string& sFieldName, int field);
void AddFieldInt(const std::string& sFieldName, int* field, size_t nMaxSize, size_t nCurSize);
void AddFieldDouble(const std::string& sFieldName, double field);
void AddFieldDouble(const std::string& sFieldName, double* field, size_t nMaxSize, size_t nCurSize);
void AddFieldLong(const std::string& sFieldName, long field);
void AddFieldLong(const std::string& sFieldName, long* field, size_t nMaxSize, size_t nCurSize);
void AddFieldLongLong(const std::string& sFieldName, long long field);
void AddFieldLongLong(const std::string& sFieldName, long long* field, size_t nMaxSize, size_t nCurSize);
void AddFieldChar(const std::string& sFieldName, char field);
void AddFieldChar(const std::string& sFieldName, char* field, size_t nMaxSize, size_t nCurSize);
void AddFieldShort(const std::string& sFieldName, short field);
void AddFieldShort(const std::string& sFieldName, short* field, size_t nMaxSize, size_t nCurSize);
void AddFieldBool(const std::string& sFieldName, bool field);
void AddFieldBool(const std::string& sFieldName, bool* field, size_t nMaxSize, size_t nCurSize);
```