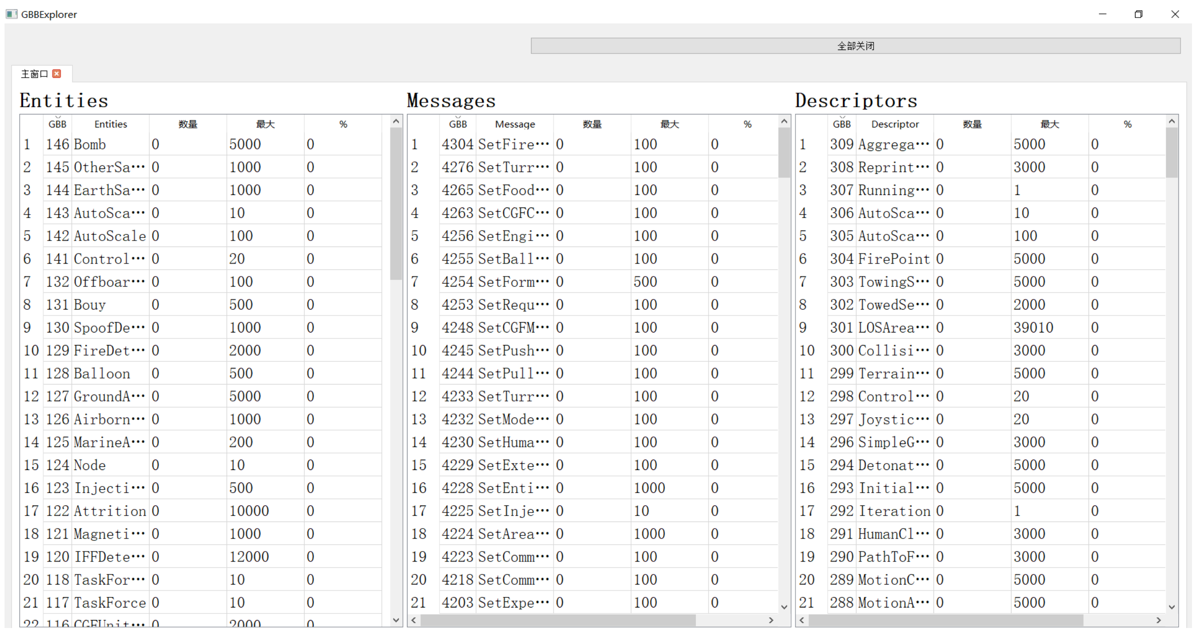
GBBExplorer记录阅读文档

Widget类，主要是绘制界面的类，界面展示如下：



Entities

	GBB	Entities	数量	最大	%
1	146	Bomb	0	5000	0
2	145	OtherSa…	0	1000	0
3	144	EarthSa…	0	1000	0
4	143	AutoSca…	0	10	0
5	142	AutoScale	0	100	0
6	141	Control…	0	20	0
7	132	Offboar…	0	100	0
8	131	Bouy	0	500	0
9	130	SpoofDe…	0	1000	0
10	129	FireDet…	0	2000	0
11	128	Balloon	0	500	0
12	127	GroundA…	0	5000	0
13	126	Airborn…	0	1000	0
14	125	MarineA…	0	200	0
15	124	Node	0	10	0
16	123	Injecti…	0	500	0
17	122	Attrition	0	10000	0
18	121	Magneti…	0	1000	0
19	120	IFFDete…	0	12000	0
20	118	TaskFor…	0	10	0
21	117	TaskForce	0	10	0
22	116	CGFUnit…	0	2000	0

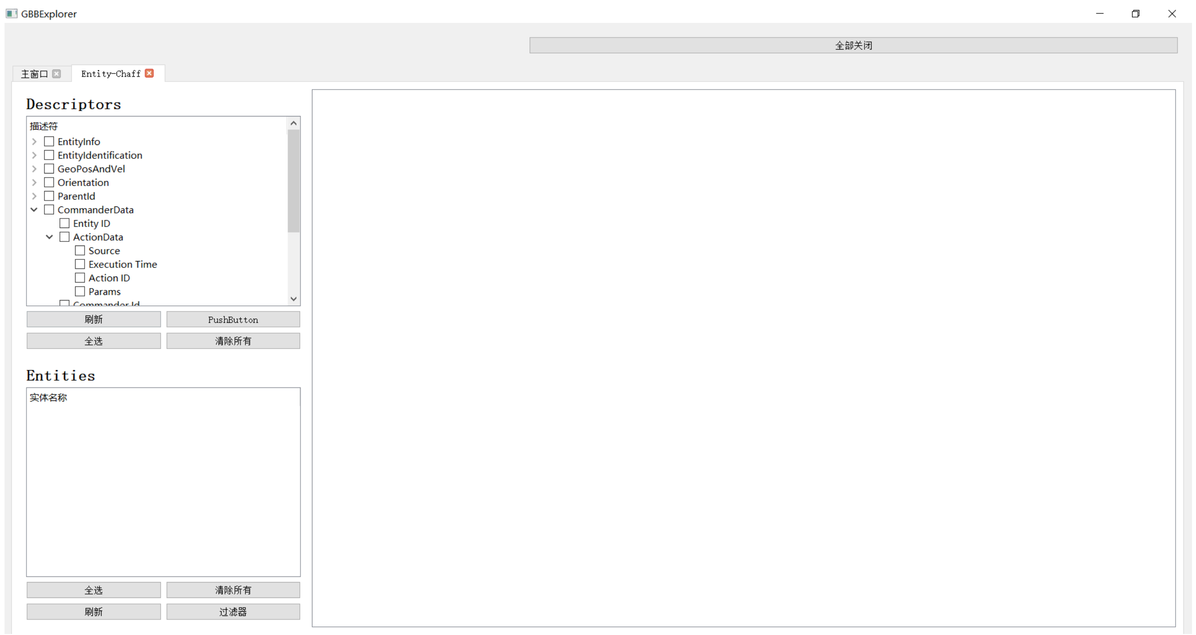
Messages

	GBB	Message	数量	最大	%
1	4304	SetFire…	0	100	0
2	4276	SetTurr…	0	100	0
3	4265	SetFood…	0	100	0
4	4263	SetCGFC…	0	100	0
5	4256	SetEngi…	0	100	0
6	4255	SetBall…	0	100	0
7	4254	SetForm…	0	500	0
8	4253	SetRequ…	0	100	0
9	4248	SetCGFM…	0	100	0
10	4245	SetPush…	0	100	0
11	4244	SetPull…	0	100	0
12	4233	SetTurr…	0	100	0
13	4232	SetMode…	0	100	0
14	4230	SetHuma…	0	100	0
15	4229	SetExte…	0	100	0
16	4228	SetEnti…	0	1000	0
17	4225	SetInje…	0	10	0
18	4224	SetArea…	0	1000	0
19	4223	SetComm…	0	100	0
20	4218	SetComm…	0	100	0
21	4203	SetExpe…	0	100	0

Descriptors

	GBB	Descriptor	数量	最大	%
1	309	Aggrega…	0	5000	0
2	308	Reprint…	0	3000	0
3	307	Running…	0	1	0
4	306	AutoSca…	0	10	0
5	305	AutoSca…	0	100	0
6	304	FirePoint	0	5000	0
7	303	TowingS…	0	5000	0
8	302	TowedSe…	0	2000	0
9	301	LOSArea…	0	39010	0
10	300	Collisi…	0	3000	0
11	299	Terrain…	0	5000	0
12	298	Control…	0	20	0
13	297	Joystic…	0	20	0
14	296	SimpleG…	0	3000	0
15	294	Detonat…	0	5000	0
16	293	Initial…	0	5000	0
17	292	Iteration	0	1	0
18	291	HumanCl…	0	3000	0
19	290	PathToF…	0	3000	0
20	289	MotionC…	0	5000	0
21	288	MotionA…	0	5000	0

detail类，实体跳转界面的具体展示



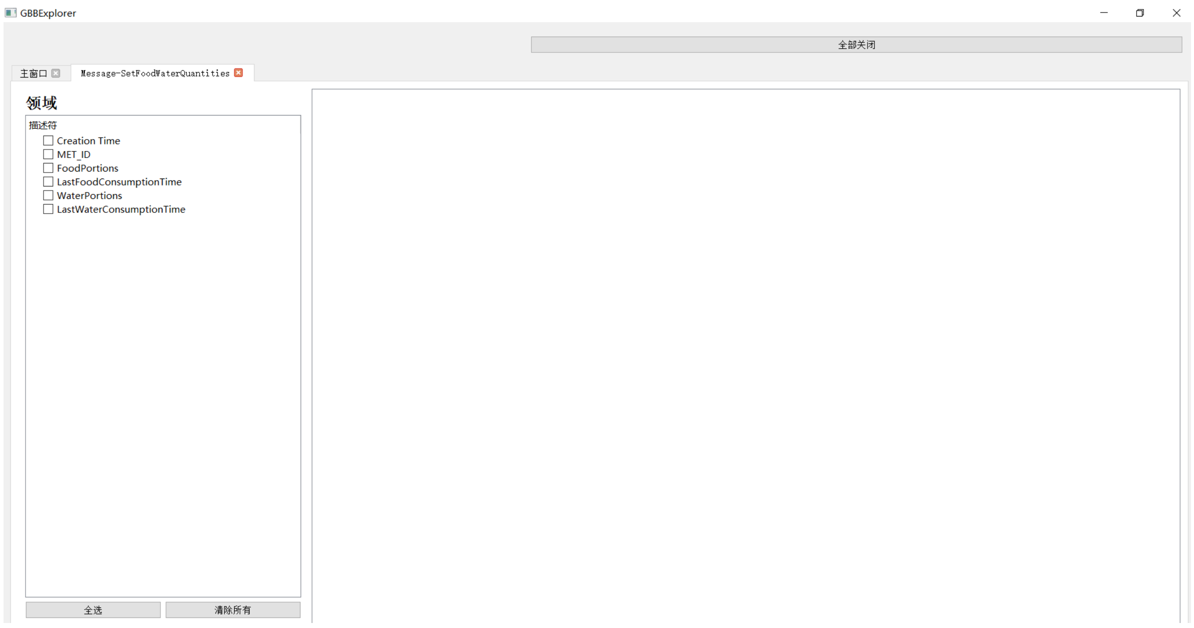
Descriptors

描述符
- ☐ EntityInfo
- ☐ EntityIdentification
- ☐ GeoPosAndVel
- ☐ Orientation
- ☐ ParentId
- ☐ CommanderData
  - ☐ Entity ID
  - ☐ ActionData
    - ☐ Source
    - ☐ Execution Time
    - ☐ Action ID
    - ☐ Params
  - ☐ Commander Id

Entities

实体名称

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

```
private:
    void initForm(); //初始化主窗口


private slots:
    void on_tableViewdoubleClicked(const QModelIndex &index); //双击主页tableview上的名称跳转显示详情
    void on_removetabbtn(int index); //删除标签
    void on_pushButton_8_clicked(); //详情页全选子项目按钮的实现
    void on_pushButton_7_clicked();//清除按钮功能实现
    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
    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

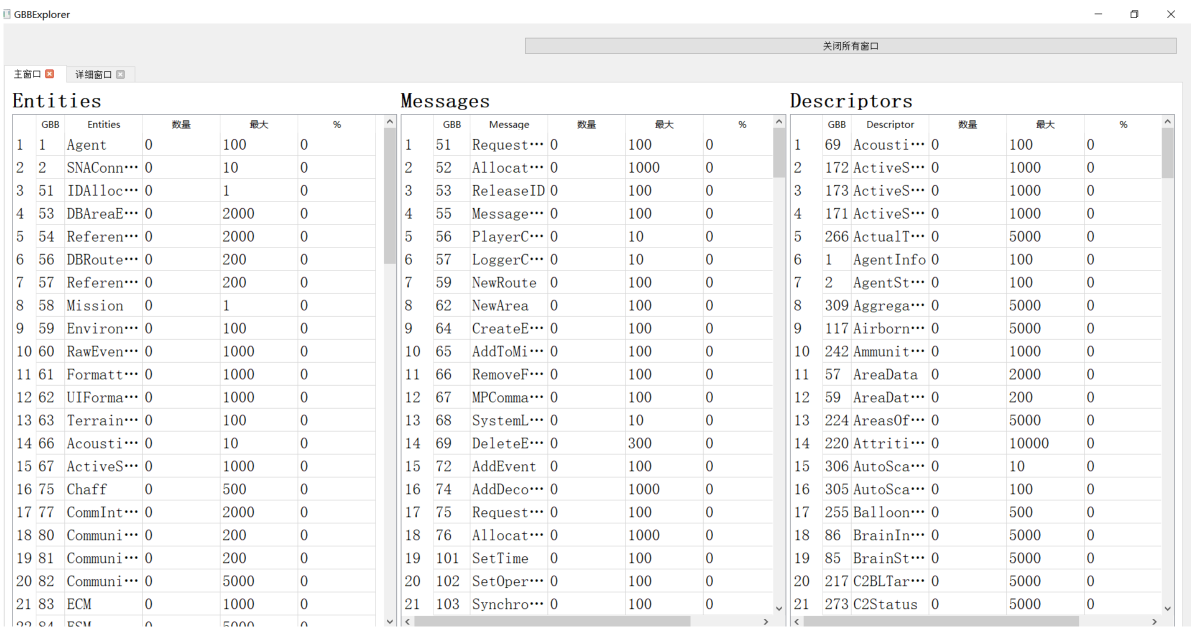
主要参照实现了CHSim-TKE_GBBExplorer\Infra\GBBExplorer\GBBExplorer文件夹中staticData.cs和entity.cs等定义的结构，从SerializedBuffer中获取静态数据

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

main函数中

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

已进行静态页面的展示



Entities

	GBB	Entities	数量	最大	%
1	1	Agent	0	100	0
2	2	SNAConn···	0	10	0
3	51	IDAlloc···	0	1	0
4	53	DBAreaE···	0	2000	0
5	54	Referen···	0	2000	0
6	56	DBRoute···	0	200	0
7	57	Referen···	0	200	0
8	58	Mission	0	1	0
9	59	Environ···	0	100	0
10	60	RawEven···	0	1000	0
11	61	Formatt···	0	1000	0
12	62	UIForma···	0	1000	0
13	63	Terrain···	0	100	0
14	66	Acousti···	0	10	0
15	67	ActiveS···	0	1000	0
16	75	Chaff	0	500	0
17	77	CommInt···	0	2000	0
18	80	Communi···	0	200	0
19	81	Communi···	0	200	0
20	82	Communi···	0	5000	0
21	83	ECM	0	1000	0
22	84	ECM	0	5000	0

Messages

	GBB	Message	数量	最大	%
1	51	Request···	0	100	0
2	52	Allocat···	0	1000	0
3	53	ReleaseID	0	100	0
4	55	Message···	0	100	0
5	56	PlayerC···	0	10	0
6	57	LoggerC···	0	10	0
7	59	NewRoute	0	100	0
8	62	NewArea	0	100	0
9	64	CreateE···	0	100	0
10	65	AddToMi···	0	100	0
11	66	RemoveF···	0	100	0
12	67	MPComma···	0	100	0
13	68	SystemL···	0	10	0
14	69	DeleteE···	0	300	0
15	72	AddEvent	0	100	0
16	74	AddDeco···	0	1000	0
17	75	Request···	0	100	0
18	76	Allocat···	0	1000	0
19	101	SetTime	0	100	0
20	102	SetOper···	0	100	0
21	103	Synchro···	0	100	0

Descriptors

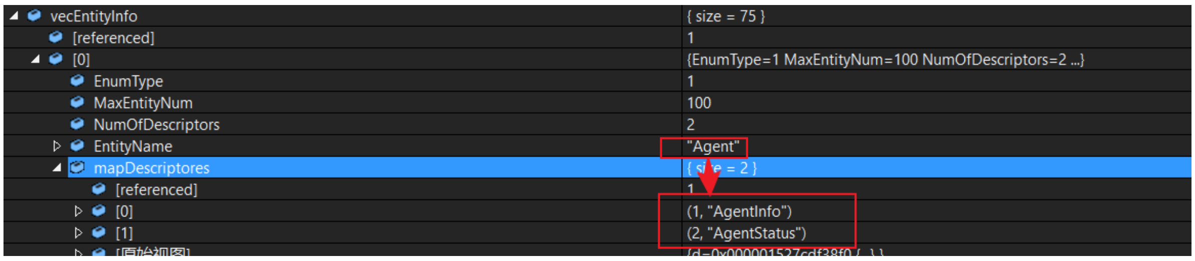
	GBB	Descriptor	数量	最大	%
1	69	Acousti···	0	100	0
2	172	ActiveS···	0	1000	0
3	173	ActiveS···	0	1000	0
4	171	ActiveS···	0	1000	0
5	266	ActualT···	0	5000	0
6	1	AgentInfo	0	100	0
7	2	AgentSt···	0	100	0
8	309	Aggrega···	0	5000	0
9	117	Airborn···	0	5000	0
10	242	Ammunit···	0	1000	0
11	57	AreaData	0	2000	0
12	59	AreaDat···	0	200	0
13	224	AreasOf···	0	5000	0
14	220	Attriti···	0	10000	0
15	306	AutoSca···	0	10	0
16	305	AutoSca···	0	100	0
17	255	Balloon···	0	500	0
18	86	BrainIn···	0	5000	0
19	85	BrainSt···	0	5000	0
20	217	C2BLTar···	0	5000	0
21	273	C2Status	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

```
//初始化实例
        StaticData staticdata;

        staticdata.InitStructures();

        //读取静态实体数据显示
        staticdata.InitEntities();
        for (int i = 0; i < staticdata.vecEntityInfo.size(); i++)
        {
            QString EnumType = QString::number(staticdata.vecEntityInfo[i].EnumType);
            QString EntityName = QString::fromStdString(staticdata.vecEntityInfo[i].EntityName);
            QString MaxEntityNum = QString::number(staticdata.vecEntityInfo[i].MaxEntityNum);
            model->setItem(i, 0, new QStandardItem(EnumType));
            model->setItem(i, 1, new QStandardItem(EntityName));
            model->setItem(i, 2, new QStandardItem("0"));
            model->setItem(i, 3, new QStandardItem(MaxEntityNum));
            //double rate = 0;
            model->setItem(i, 4, new QStandardItem("0"));
        }
```

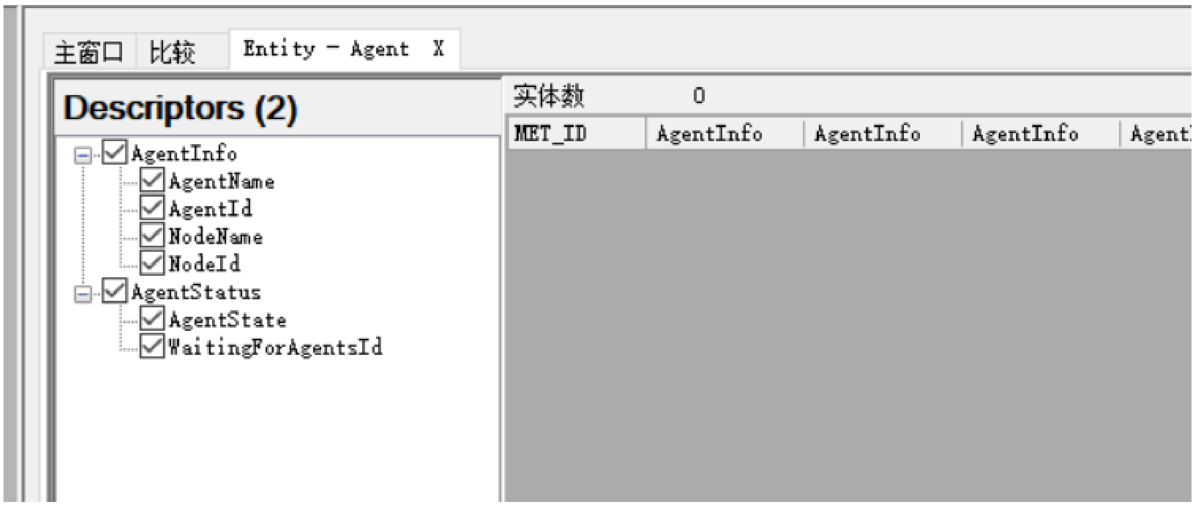
各个数据关联情况

- Entity点击详情，以agent为例：



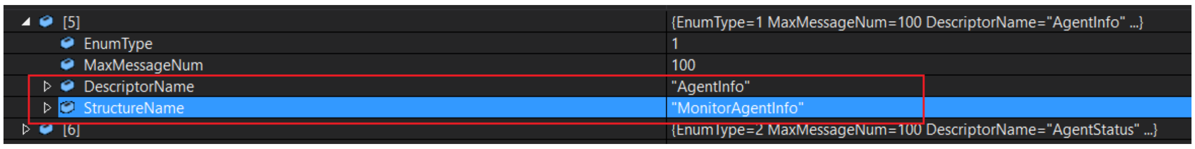
```
◢ vecEntityInfo                 { size = 75 }
    [referenced]                1
  ◢ [0]                         {EnumType=1 MaxEntityNum=100 NumOfDescriptors=2 ...}
      EnumType                  1
      MaxEntityNum              100
      NumOfDescriptors          2
    ▷ EntityName                "Agent"
    ◢ mapDescriptores           { size = 2 }
        [referenced]            1
      ▷ [0]                     (1, "AgentInfo")
      ▷ [1]                     (2, "AgentStatus")
```

mapdescriptors是对应的2个根节点：



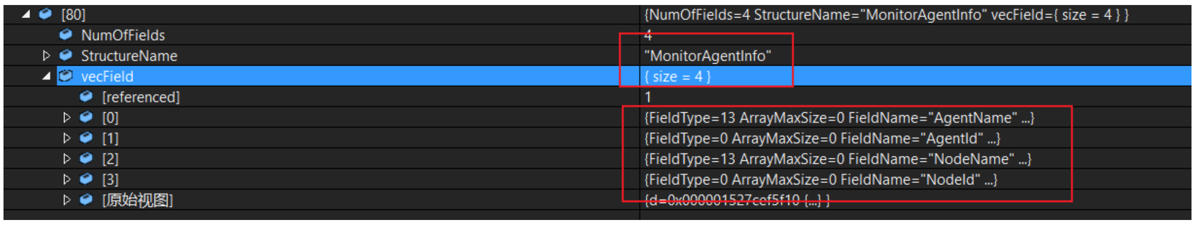
```
主窗口 | 比较 | Entity - Agent  X

Descriptors (2)                 实体数    0
⊟ ☑ AgentInfo                   MET_ID | AgentInfo | AgentInfo | AgentInfo | Agent
     ☑ AgentName
     ☑ AgentId
     ☑ NodeName
     ☑ NodeId
⊟ ☑ AgentStatus
     ☑ AgentState
     ☑ WaitingForAgentsId
```

再去vecdescriptors中找到对应name的structure



```
◢ [5]                     {EnumType=1 MaxMessageNum=100 DescriptorName="AgentInfo" ...}
    EnumType              1
    MaxMessageNum         100
  ▷ DescriptorName        "AgentInfo"
  ▷ StructureName         "MonitorAgentInfo"
▷ [6]                     {EnumType=2 MaxMessageNum=100 DescriptorName="AgentStatus" ...}
```

根据`monitoragebtinfo`在vecstruct中找子节点

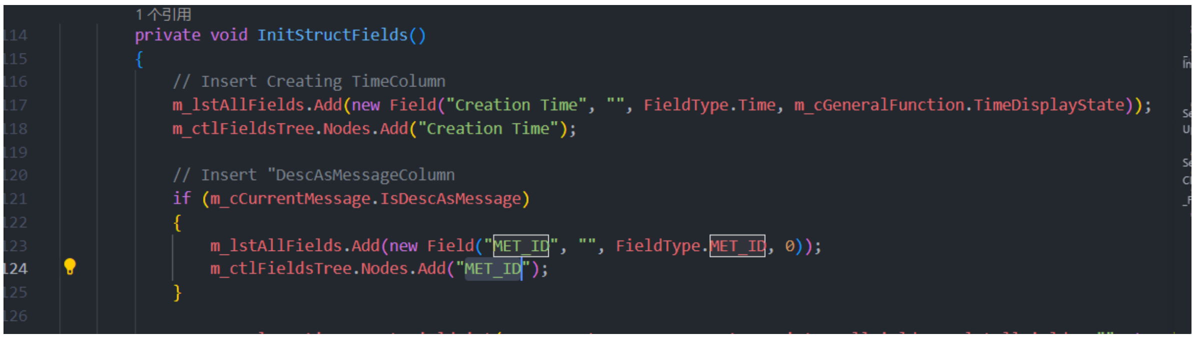


```
◢ [80]                    {NumOfFields=4 StructureName="MonitorAgentInfo" vecField={ size = 4 } }
    NumOfFields           4
  ▷ StructureName         "MonitorAgentInfo"
  ◢ vecField              { size = 4 }
      [referenced]        1
    ▷ [0]                 {FieldType=13 ArrayMaxSize=0 FieldName="AgentName" ...}
    ▷ [1]                 {FieldType=0 ArrayMaxSize=0 FieldName="AgentId" ...}
    ▷ [2]                 {FieldType=13 ArrayMaxSize=0 FieldName="NodeName" ...}
    ▷ [3]                 {FieldType=0 ArrayMaxSize=0 FieldName="NodeId" ...}
    ▷ [原始视图]           {d=0x000001527ccf5f10 {...} }
```

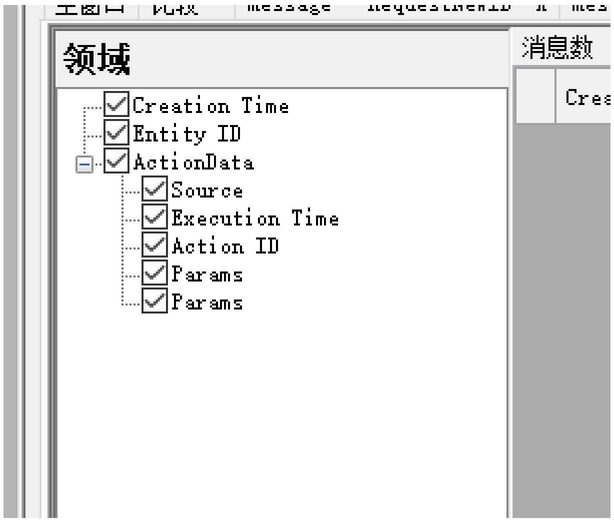
- message详情，以requestnewid为例：

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



```
1 个引用
private void InitStructFields()
{
    // Insert Creating TimeColumn
    m_lstAllFields.Add(new Field("Creation Time", "", FieldType.Time, m_cGeneralFunction.TimeDisplayState));
    m_ctlFieldsTree.Nodes.Add("Creation Time");

    // Insert "DescAsMessageColumn
    if (m_cCurrentMessage.IsDescAsMessage)
    {
        m_lstAllFields.Add(new Field("MET_ID", "", FieldType.MET_ID, 0));
        m_ctlFieldsTree.Nodes.Add("MET_ID");
    }

```


领域

消息数

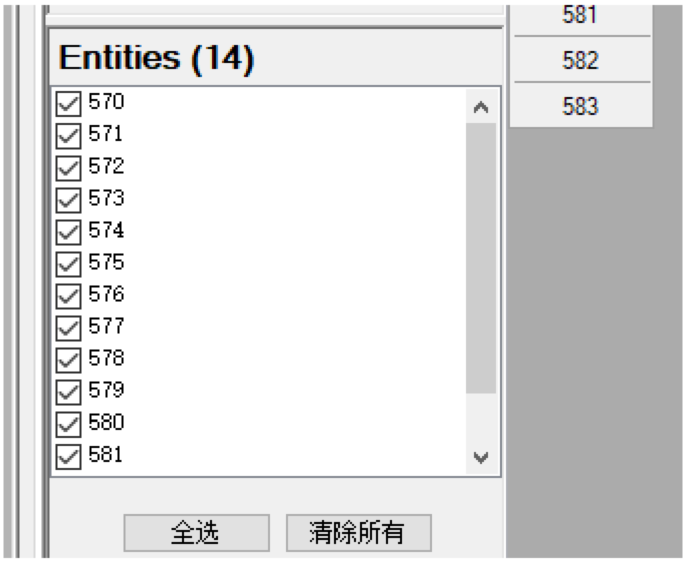
- ☑ Creation Time
- ☑ Entity ID
- ☑ ActionData
  - ☑ Source
  - ☑ Execution Time
  - ☑ Action ID
  - ☑ Params
  - ☑ Params

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**

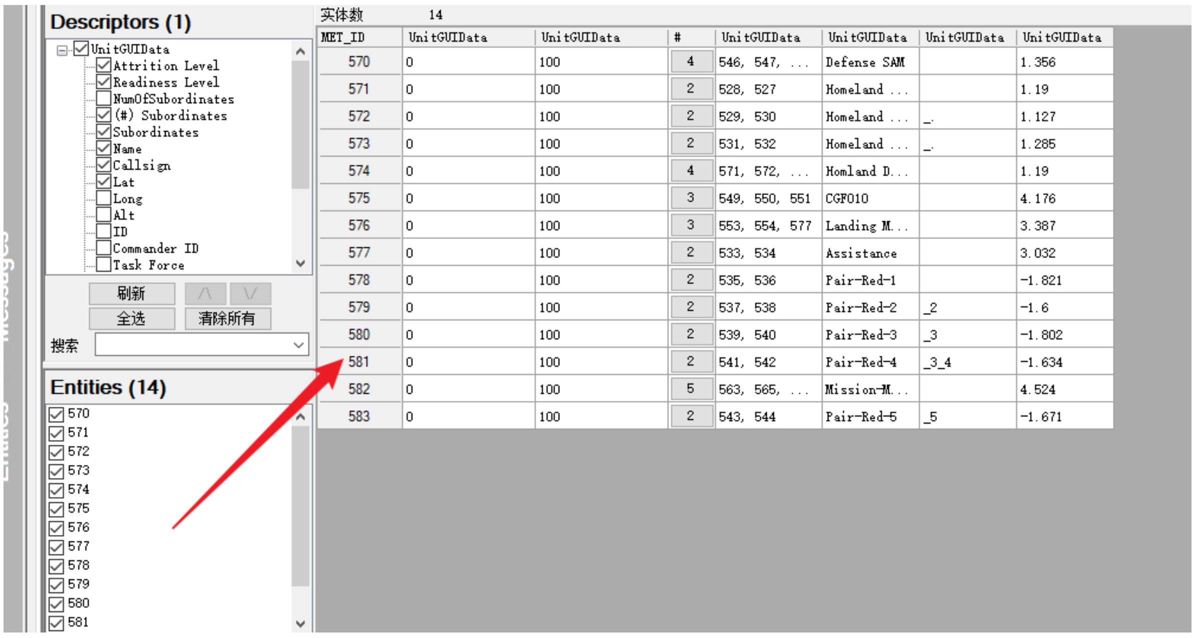
`bool GetEntityCount(enum_t eEntityType);` 每个周期，动态获取实体数量

用于获取entity里的详情，如下图


Entities (14)

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

581
582
583


Descriptors (1)

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

Entities (14)

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

实体数 14

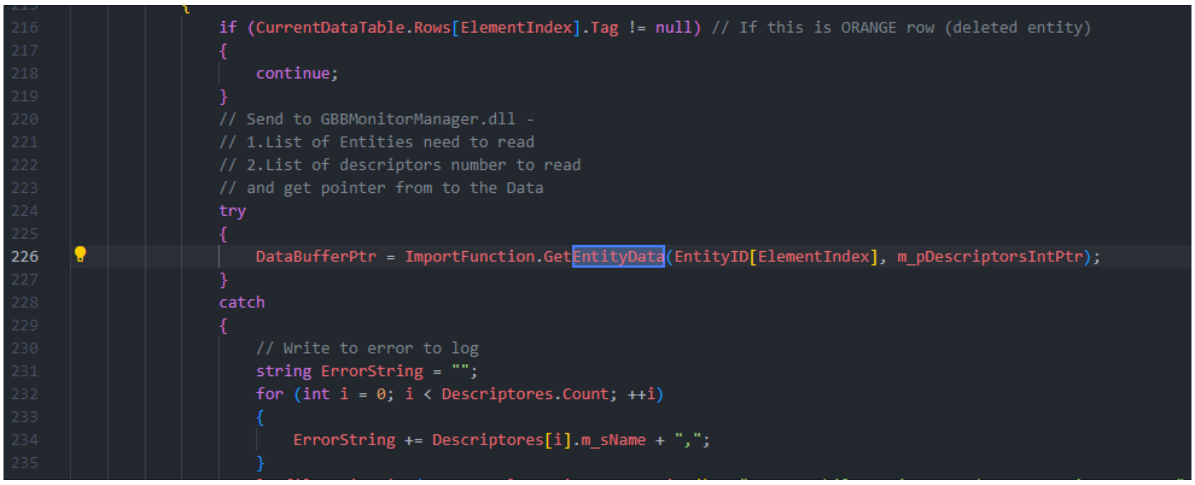
MET_ID	UnitGUIData	UnitGUIData	#	UnitGUIData	UnitGUIData	UnitGUIData	UnitGUIData
570	0	100	4	546, 547, ...	Defense SAM		1.356
571	0	100	2	528, 527	Homeland ...		1.19
572	0	100	2	529, 530	Homeland ...	_.	1.127
573	0	100	2	531, 532	Homeland ...	_.	1.285
574	0	100	4	571, 572, ...	Homland D...		1.19
575	0	100	3	549, 550, 551	CGF010		4.176
576	0	100	3	553, 554, 577	Landing M...		3.387
577	0	100	2	533, 534	Assistance		3.032
578	0	100	2	535, 536	Pair-Red-1		-1.821
579	0	100	2	537, 538	Pair-Red-2	_2	-1.6
580	0	100	2	539, 540	Pair-Red-3	_3	-1.802
581	0	100	2	541, 542	Pair-Red-4	_3_4	-1.634
582	0	100	5	563, 565, ...	Mission-M...		4.524
583	0	100	2	543, 544	Pair-Red-5	_5	-1.671

`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

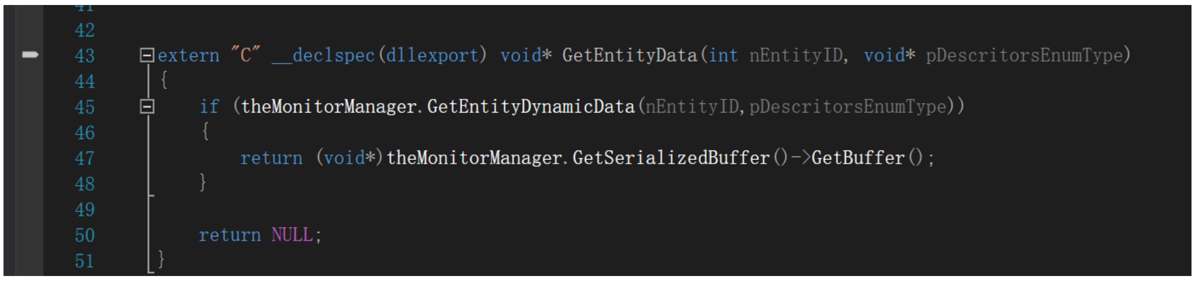
调用情况：

dynamicdata.cs中：



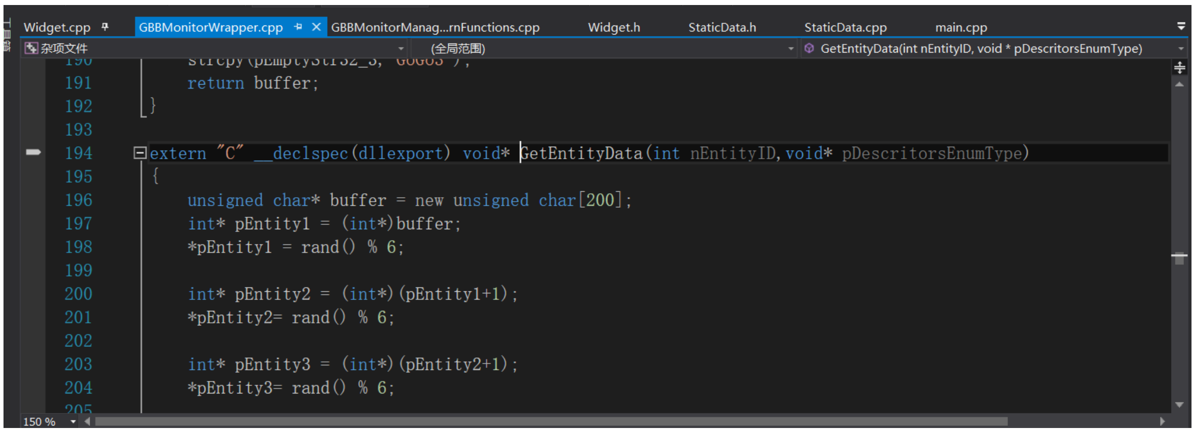
```
                if (CurrentDataTable.Rows[ElementIndex].Tag != null) // If this is ORANGE row (deleted entity)
                {
                    continue;
                }
                // Send to GBBMonitorManager.dll -
                // 1.List of Entities need to read
                // 2.List of descriptors number to read
                // and get pointer from to the Data
                try
                {
                    DataBufferPtr = ImportFunction.GetEntityData(EntityID[ElementIndex], m_pDescriptorsIntPtr);
                }
                catch
                {
                    // Write to error to log
                    string ErrorString = "";
                    for (int i = 0; i < Descriptores.Count; ++i)
                    {
                        ErrorString += Descriptores[i].m_sName + ",";
                    }
```

补充源码里的cpp定义



```

extern "C" __declspec(dllexport) void* GetEntityData(int nEntityID, void* pDescritorsEnumType)
{
    if (theMonitorManager.GetEntityDynamicData(nEntityID, pDescritorsEnumType))
    {
        return (void*)theMonitorManager.GetSerializedBuffer()->GetBuffer();
    }

    return NULL;
}
```

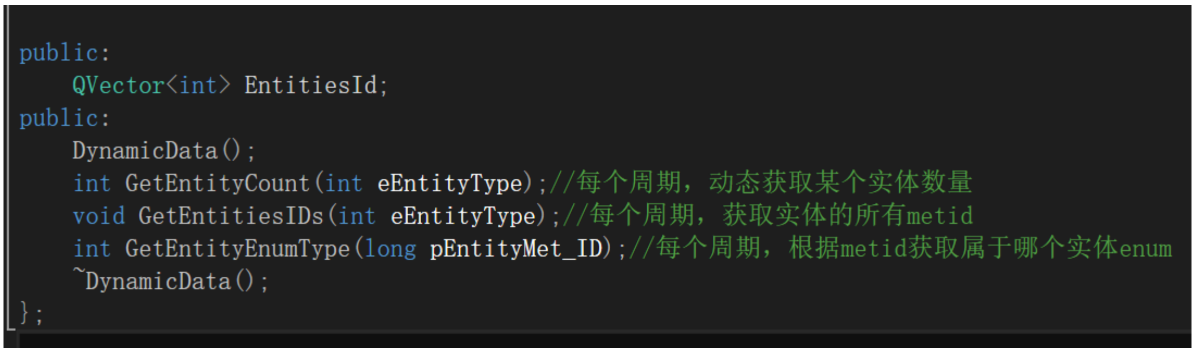
gbbmonitorwrapper.cpp里的定义



```
    return buffer;
}

extern "C" __declspec(dllexport) void* GetEntityData(int nEntityID, void* pDescritorsEnumType)
{
    unsigned char* buffer = new unsigned char[200];
    int* pEntity1 = (int*)buffer;
    *pEntity1 = rand() % 6;

    int* pEntity2 = (int*)(pEntity1+1);
    *pEntity2= rand() % 6;

    int* pEntity3 = (int*)(pEntity2+1);
    *pEntity3= rand() % 6;
```

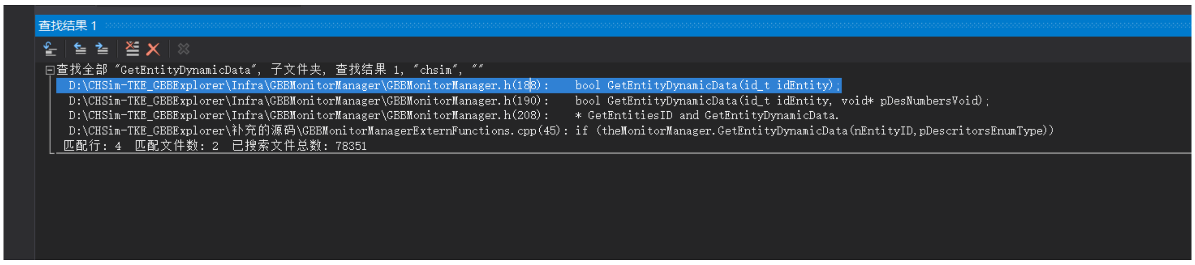
`bool GetEntityEnumType(long pEntityMet_ID);` 获取enumtype

已经在DynamicData类里实现



```
public:
    QVector<int> EntitiesId;
public:
    DynamicData();
    int GetEntityCount(int eEntityType);//每个周期，动态获取某个实体数量
    void GetEntitiesIDs(int eEntityType);//每个周期，获取实体的所有metid
    int GetEntityEnumType(long pEntityMet_ID);//每个周期，根据metid获取属于哪个实体enum
    ~DynamicData();
};
```

GetEntityDynamicData的结构?



```
⊟查找全部 "GetEntityDynamicData", 子文件夹, 查找结果 1, "chsim", ""
  D:\CHSim-TKE_GBEExplorer\Infra\GBEMonitorManager\GBEMonitorManager.h(188):    bool GetEntityDynamicData(id_t idEntity);
  D:\CHSim-TKE_GBEExplorer\Infra\GBEMonitorManager\GBEMonitorManager.h(190):    bool GetEntityDynamicData(id_t idEntity, void* pDesNumbersVoid);
  D:\CHSim-TKE_GBEExplorer\Infra\GBEMonitorManager\GBEMonitorManager.h(208):    * GetEntitiesID and GetEntityDynamicData.
  D:\CHSim-TKE_GBEExplorer\补充的源码\GBEMonitorManagerExternFunctions.cpp(45): if (theMonitorManager.GetEntityDynamicData(nEntityID,pDescritorsEnumType))
 匹配行: 4  匹配文件数: 2  已搜索文件总数: 78351
```