二次开发接口使用说明

文档编写说明

此文档记录了开发过程中使用的大部分 SAM 二次开发接口。

部分在使用中查找到的接口依赖库 (可能不准确) 会以 *调用库：**SAMLib.lib*** 的形式附加在头文件下。

文档忽略 **inline** 修饰符，包含 **static**、**const** 修饰符

文档中类型和对象使用样式规则为:

- 实例化的对象：*SAMObject*
- 类型名：`SAMDataType`

如:

- *SAMObject* 对象的 **Get()** 方法返回类型为 `SAMDataType` 。

提示

- 请注意，返回 const & 常引用类型的函数与返回 & 引用的函数混合使用很可能导致内存问题。如:

```
//此段代码会在内核中引发内存访问冲突异常

asoKAssembly assm = asoKConstGetAssembly(target.TargetModel());
ftrFeatureList fl = assm.ConstGetFeatureList();
cowListInt instIDs = asoFPartInstanceAC::GetActiveIds(fl);

//去掉 FeatureList 的 const 引用即可
asoKAssembly assm = asoKConstGetAssembly(target.TargetModel());
ftrFeatureList* fl = assm.GetFeatureList();
cowListInt instIDs = asoFPartInstanceAC::GetActiveIds(*fl);
```

内核 (**Kernel**) 和 **Gui** 两侧接口分离，体现在头文件的中 `K` 和 `G` 字母标识，不可混用。

basBasis.h

调用库：**SAMModelDatabase.lib**

- static basBasis* **basBasis::Instance()**

  返回 SAM 进程中全局唯一控制 *Mdb*、*Ddb* 的 `basBasis` 单例父对象。

- const basMdb &**basBasis::Fetch()**

  返回 *basBasis* 控制的 *basMdb* 对象

  示例:

```
basBasis* bas = basBasis::Instance();

//通过basBasis获取mdb对象
basMdb mdb = bas->Fetch();
```

basMdb.h

调用库：**SAMModelDatabase.lib**

- basModelMap **basMdb::GetModels()**

  返回*Mdb*下的模型`basNewModel`-名字映射表。

  示例：

  ```
  basModelMap modelsMap = mdb.GetModels();
  ```

basModelMap.h

调用库：**SAMModelDatabase.lib**

- cowListString **basModelMap::Keys()**

  返回*basModelMap*中`cowListString`类型的模型名字的列表。

  示例：

  ```
  //通过basModelMap获取模型名称列表
  cowListString modelList = modelsMap.Keys();
  ```

ptoKUtils.h , ptoKPartRepository.h

调用库：SAMGeometryComponent.lib

- const ptoKPartRepository& **ptoKConstGetPartRepos(const basMdb&, const QString &modelName)**

  返回存储有模型下 *Parts* 对象的`ptoKPartRepository`表。

  示例：

  ```
  const ptoKPartRepository& parts = ptoKConstGetPartRepos(mdb, s);

  //利用返回的parts表获取Part名称
  for (int p = 1; p <= parts.Size(); ++p) {
      QString partname = parts.GetKey(p);
  }
  ```

- const QString& **ptoKPartRepository::GetKey(uint)**

  返回`ptoKPartRepository`中序号下的Part名称。

- const ptoKPart& **ptoKPartRepository::ConstGet(uint)**

  返回 *ptoKPartRepository* 中特定编号下 `ptoKPart` 类型的 *Part* 。

ptoKPart.h

调用库：SAMPrimaryObjects.lib，SAMCoreAttributes.lib

- ftrFeatureList* **ptoKPart::GetFeatureList()** const

  该方法继承自`ftrPrimaryObject`，返回 *Part* 对象的 `FeatureList` 列表，其中包括 Part 对象的 Mesh 网格等数据。

  示例：

```
ftrFeatureList* flpart = part.GetFeatureList();

//利用FeatureList获取网格数据
const bmeMesh* objectMesh = flpart->ConstGetMesh(bdoDefaultInstId);
```

ftrFeatureList.h

- bool **ftrFeatureList::MeshExists(uint instId)** const

  检查编号下是否存在 *Mesh* 对象。

- bmeMesh* **ftrFeatureList::GetMesh(uint instId)**

  获取编号下 *Mesh* 对象。

- const bmeMesh* **ftrFeatureList::ConstGetMesh(uint instId)** const

  获取编号下 *Mesh* const对象。

  示例：

```
//检查 Part 对象是否存在 Mesh 对象并获取

if (!fl->MeshExists(meshInstID))
return -1;
const bmeMesh* mesh = fl->ConstGetMesh(meshInstID);
```

bmeMesh.h

调用库：SAMCoreMeshDefs.lib

- const bmeNodeData& **bmeMesh::NodeData()** const

  获取 *bmeMesh* 对象的 `Node` 节点数据。

- const bmeElementData& **bmeMesh::ElementData()** const

  获取 *bmeMesh* 对象的 `Element` 单元数据。

- uint **bmeMesh::NumNodes()** const

  获取 *Mesh* 中节点的数量。

omeMesh.h

- **omeMesh::omeMesh(**
  uint Id,
  uint num_nodes,
  utiCoordCont3D& coords,
  uint num_classes,
  bmeElementClass classes,
  uint numDim = 3,
  const gslMatrix* tform = NULL
)

  `omeMesh` 继承于 `bmeMesh`，可用于构造 `bmeMesh` 类型的 *Mesh* 网格数据对象，必须的输入参数按顺序为：网格位置号 (构造 *Part* 对象的网格数据时为 *bdoDefaultInstId = 1000U*)，节点数量，节点坐标容器，`bmeElementClass*` 类型的数量及其二级指针数组。

mesUtils.h

- void **mesSetMesh(ftrFeatureList& flist, int instId, bmeMesh mesh)**

  设置 *FeatureList* 中 *instId* 下的 `bmeMesh` 网格数据。

bmeNodeData.h

调用库：SAMCoreMeshDefs.lib

- const utiCoordCont3D& **bmeNodeData::CoordContainer()** const

  获取 *NodeData* 中点的 `utiCoordCont3D` 类型的坐标数据容器。两者结合使用获取坐标见章节 utiCoordCont3D.h 的示例

bmeElementData.h

调用库：SAMCoreMeshDefs.lib

- const bmeElementClassList& **bmeElementData::ConstGetClasses()** const

  获取 *ElementData* 对象的 `bmeElementClassList` 单元容器的列表对象。

bmeElementClassList.h

- const int **bmeElementClassList::Size()** const

  获取 *ElementClassList* 中的 `bmeElementClass` 单元容器数量。

- const bmeElementClass& **bmeElementClassList::ConstGet(int i)** const

  获取对应编号位置下 *ElementClassList* 中的 `bmeElementClass&` 单元容器。

bmeElementClass.h

- uint **bmeElementClass::NumElements()** const

  获取 *ElementClass* 中的单元数量。

- const int* const **bmeElementClass::Connectivity()** const

  获取 *ElementClass* 中单元的节点编号 (内部编号，非 *UserLabel*) 数组，数组长度由其单元数量和 *ElementClass* 标签单元类型决定。

- const QString& **bmeElementClass::ElemTypeLabel()** const

  获取 *ElementClass* 的 `QString` 标签单元类型，如 **"B31"** 等。

- const shpShape* **bmeElementClass::Shape()**

  获取单元形状。其中包含其几何维数和实际空间维数。注意几何形状和实际空间不同，如二维单元 **S4** 在空间变形后成为三维单元。

- static bmeElementClass* **bmeElementClass::ConstructObject(**
  const int numElements,
  const QString& elTypLabel,
  int* connectivity
)

  构造 `bmeElementClass` 的方法，输入参数分别为：单元数量，`QString` 单元类型，单元节点编号数组。

- void *bmeElementClass::SetUserElementLabel(int l)**

  设置向用户展示的单元的编号，传入编号数组。

utiCoordCont3D.h

- bool **utiCoordCont3D::GetCoord(int index, float &x, float &y, float &z)** const

  示例：

```
//获取容器
utiCoordCont3D nodeContainer = nodeData.CoordContainer();
```

```
//使用utiCoordCont3D容器获取点的坐标，index为点的内部序号而非UserLabel
nodeContainer.GetCoord(index, x, y, z);
```

- void **utiCoordCont3D::Append(float x, float y, float z)**

  向 *nodeContainer* 中插入坐标数据。

cmdCWIP.h

- static cmdCWIP& **cmdCWIP::Instance()**

  获取 `cmdCWIP` 的 **Singleton** 实例。

- void **cmdCWIP::Print(const QString&)**

  在 **Message** 窗口打印消息。

- void **cmdCWIP::Warning(const QString&)**

  在 **Message** 窗口打印红色警告消息。

asoKUtils.h

- const asoKAssembly& **asoKConstGetAssembly(const QString& model)**

  获取模型的 `asoKAssembly` 装配对象。

- ftrFeatureList* **asoKAssembly::GetFeatureList()**

  获取 `asoKAssembly` 的 `ftrFeatureList*` 特征列表。

asoFPartInstance.h

- static cowListInt **asoFPartInstanceAC::GetActiveIds(const ftrFeatureList& fl)**

  获取 `asoKAssembly` 装配中存放 `bmeMesh` 网格信息位置 *instId* 的 **ID** 列表。装配中的 *Mesh* 数据不再存放在 *Part* 网格信息对应的 *bdoDefaultInstId = 1000U* 位置。

  示例：

```
//获取模型的 Assembly 装配，并读取其所有 Mesh 信息

asoKAssembly assm = asoKConstGetAssembly(target.TargetModel());
ftrFeatureList* fl = assm.GetFeatureList();
cowListInt instIDs = asoFPartInstanceAC::GetActiveIds(*fl);
for (int i = 0; i < instIDs.Length(); ++i) {
    int meshInstID = instIDs.Get(i);
    if (!fl->MeshExists(meshInstID))
        return -1;
    const bmeMesh* mesh = fl->ConstGetMesh(meshInstID);
}
```

cmdKCommandDeliveryRole.h

- static cmdKCommandDeliveryRole& **cmdKCommandDeliveryRole::Instance()**

  返回 `cmdKCommandDeliveryRole` 的 **Singleton** 实例。

- void **cmdKCommandDeliveryRole::ProcessCommand(const QString&)**

  从内核向 **Gui** 发送 *Python* 命令。

  示例:

```
//选择 Part , 刷新视图

QString pyt =
    QString("session.viewports['Viewport: 1'].setValues(displayedObject =
mdb.models['Model-1'].parts['Part-1'])");

cmdKCommandDeliveryRole::Instance().ProcessCommand(pyt);
```