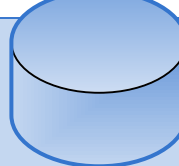


## StorageService



### IStorageService

- FSAdmin
- VertexTransformation
- AccessHandling
- VertexStore
  - IStorageGraphElement
  - TopologieService
  - (SerializationService)
  - (TransactionService)

### IStorageGraphElement

### IStorageAttribute

## VertexStore

ConcurrentHash<Long,LinkedList<IStorageGraphElement>>

Straight assignment of the id's  
Start @ Long.Min\_Value

IStorageGraphElement:  
Represents all Vertices within the  
graph. Each one got its unique id. So  
it is possible to gather all vertices  
without knowing there related  
VertexScheme(s).

## IStorageGraphElement

HashMap<Long,HashMap<Integer,IStorageAttribute>> structuredAttributes

AttributeID.  
Defined by the  
VertexScheme, this is  
the  
IStorageGraphElement's  
id of the related  
VertexScheme.

The id of the related  
VertexScheme  
unstructured's id.  
(Must be the same)

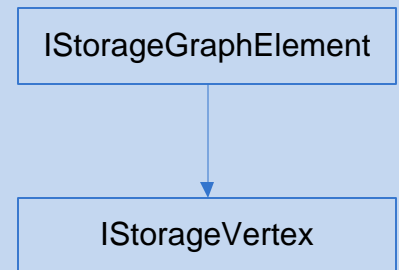
Stores a set of information.  
Property:<ObjectType,IAttributeType>  
Edge:<IStorageGraphElement>

HashMap<Integer,Tuple<String,IStorageAttribute>> unstructuredAttributes

The name of the attribute

## IVertexStore

```
boolean getVertexExists(  
    long myVertexID);  
  
boolean getCountOfVertices(  
    long myVertexID);  
  
IStorageVertex getVertexCurrentRevision(  
    long myVertexID);  
  
IStorageVertex getVertexFirstRevision(  
    long myVertexID);  
  
IStorageVertex getVertexByRevision(  
    long myVertexID,  
    int myRevision);  
  
Collection<IStorageVertex> getAllVertexRevisions(  
    long myVertexID);  
  
Collection<IStorageVertex> getVerticesByCurrentRevision(  
    Collection<long> myVertexIDs);  
  
int getCountOfVertexRevisions(  
    long myVertexID);  
  
void RemoveVertex(  
    long myVertexID);  
  
void RemoveVertexRevision(  
    long myVertexID,  
    int myToBeRemovedRevisionID);  
  
void RemoveVertices(  
    Collection<long> myToBeDeltedVertices = null);  
  
long addVertex(  
    IStorageVertex myToBeAddedVertex);  
  
long updateVertex(  
    IStorageVertex myToBeAddedVertexRevision);
```



## IStorageGraphElement

```
LinkedList<IStorageAttribute> getAllAttributesByVertexSchemeId(
    long myVertexSchemeId);

Collection<Long> getAllVertexSchemeIds();

IStorageAttribute getAttribute(
    long myVertexSchemeId,
    int myAttributeId);

IStorageAttribute getAttributeOfFirstVertexScheme(
    int myAttributeId);

LinkedList<IStorageAttribute> getAttributesOfFirstVertexScheme(
    Collection<Integer> myAttributeIds = null);

Collection<Long,<LinkedList<IStorageAttribute>>> getAllStructuredAttributes();

int getCountOfStructuredAttributes();

Collection<String,IStorageAttribute> getAllUnstructuredAttributes(
    Collection<Integer> myAttributeIds);

int getIdOfUnstructuredAttributes(
    String myAttributeName);

int getIdOfUnstructuredAttributes(
    IStorageAttribute myAttributeValue);

Collection<String> getAllUnstructuredAttributeNames(
    Collection<IStorageAttribute> myInterestingAttributes = null);

Collection<IStorageAttribute> getAllUnstructuredAttributes(
    Collection<String> myAttributeNames = null);

int addUnstructuredAttribute(
    String myAttributeName,
    IStorageAttribute myAttribute);

void removeUnstructureAttribute(
    String myToBeRemovedAttributeName);

void removeUnstructuredAttributes(
    Collection<Integer> myToBeRemovedAttributeId = null);

int getCountOfUnstructuredAttributes();
```

