TDA Kernel DLL for native calls

Generated by Doxygen 1.8.14

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Module Documentation

3.1 TDA Kernel DLL RAAPI/IRepository Functions

Functions

- TDAEXTERN void TDACALL TDA_Close (void *tdaKernel)
- TDAEXTERN bool TDACALL TDA_Exists (void *tdaKernel, const char *location)
- TDAEXTERN bool TDACALL TDA_Open (void *tdaKernel, const char *location)
- TDAEXTERN bool TDACALL TDA_StartSave (void *tdaKernel)
- TDAEXTERN bool TDACALL TDA_CancelSave (void *tdaKernel)
- TDAEXTERN bool TDACALL TDA_FinishSave (void *tdaKernel)
- TDAEXTERN bool TDACALL TDA_Drop (void *tdaKernel, const char *location)
- TDAEXTERN __int64 TDACALL TDA_FindClass (void *tdaKernel, const char *name)
- TDAEXTERN const char *TDACALL TDA_GetClassName (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_CreateObject (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_CreateClass (void *tdaKernel, const char *name)
- TDAEXTERN bool TDACALL TDA_DeleteClass (void *tdaKernel, __int64 rClass)
- TDAEXTERN bool TDACALL TDA_IsClass (void *tdaKernel, __int64 r)
- TDAEXTERN bool TDACALL TDA_IsDirectSubClass (void *tdaKernel, __int64 rSubClass, __int64 rSuper↩ Class)
- TDAEXTERN bool TDACALL TDA_DeleteObject (void *tdaKernel, __int64 rObject)
- TDAEXTERN bool TDACALL TDA_MoveObject (void *tdaKernel, __int64 rObject, __int64 rToClass)
- TDAEXTERN bool TDACALL TDA_IsTypeOf (void *tdaKernel, __int64 rObject, __int64 rClass)
- TDAEXTERN bool TDACALL TDA_IsKindOf (void *tdaKernel, __int64 rObject, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_CreateAttribute (void *tdaKernel, __int64 rClass, const char *name, __int64 rPrimitiveType)
- TDAEXTERN bool TDACALL TDA_IsDerivedClass (void *tdaKernel, __int64 rDirectlyOrIndirectlyDerived↩ Class, __int64 rSuperClass)
- TDAEXTERN __int64 TDACALL TDA_FindAttribute (void *tdaKernel, __int64 rClass, const char *name)
- TDAEXTERN bool TDACALL TDA_DeleteAttribute (void *tdaKernel, __int64 rAttribute)
- TDAEXTERN const char *TDACALL TDA_GetAttributeName (void *tdaKernel, __int64 rAttribute)
- TDAEXTERN const char *TDACALL TDA_GetPrimitiveDataTypeName (void *tdaKernel, __int64 rDataType)
- TDAEXTERN __int64 TDACALL TDA_FindPrimitiveDataType (void *tdaKernel, const char *name)
- TDAEXTERN bool TDACALL TDA_IsPrimitiveDataType (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForClasses (void *tdaKernel)
- TDAEXTERN const char *TDACALL TDA_GetAttributeValue (void *tdaKernel, __int64 rObject, __int64 r↩ Attribute)

- TDAEXTERN __int64 TDACALL TDA_CreateAssociation (void *tdaKernel, __int64 rSourceClass, __int64 rTargetClass, const char *sourceRoleName, const char *targetRoleName, bool isComposition)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectSubClasses (void *tdaKernel, __int64 rSuper↩ Class)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllClassObjects (void *tdaKernel, __int64 rClassOr↩ AdvancedAssociation)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectAttributes (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetAttributeDomain (void *tdaKernel, __int64 rAttribute)
- TDAEXTERN bool TDACALL TDA_DeleteGeneralization (void *tdaKernel, __int64 rSubClass, __int64 r↩ SuperClass)
- TDAEXTERN bool TDACALL TDA_CreateGeneralization (void *tdaKernel, __int64 rSubClass, __int64 r↩ SuperClass)
- TDAEXTERN bool TDACALL TDA_ExcludeObjectFromClass (void *tdaKernel, __int64 rObject, __int64 r↩ Class)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllAttributes (void *tdaKernel, __int64 rClass)
- TDAEXTERN bool TDACALL TDA_IncludeObjectInClass (void *tdaKernel, __int64 rObject, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectSuperClasses (void *tdaKernel, __int64 rSub↩ Class)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectClassObjects (void *tdaKernel, __int64 rClass↩ OrAdvancedAssociation)
- TDAEXTERN bool TDACALL TDA_SetAttributeValue (void *tdaKernel, __int64 rObject, __int64 rAttribute, const char *value)
- TDAEXTERN bool TDACALL TDA_DeleteAttributeValue (void *tdaKernel, __int64 rObject, __int64 rAttribute)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectLinguisticInstances (void *tdaKernel, __int64 r↩ Class)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectObjectClasses (void *tdaKernel, __int64 r↩ ObjectOrAdvancedLink)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForObjectsByAttributeValue (void *tdaKernel, __int64 r↩ Attribute, const char *value)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllOutgoingAssociationEnds (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectIngoingAssociationEnds (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllLinguisticInstances (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectOutgoingAssociationEnds (void *tdaKernel, __↩ int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllIngoingAssociationEnds (void *tdaKernel, __int64 r↩ Class)
- TDAEXTERN __int64 TDACALL TDA_ResolveIterator (void *tdaKernel, __int64 it, int position)
- TDAEXTERN const char *TDACALL TDA_GetRoleName (void *tdaKernel, __int64 rAssociationEnd)
- TDAEXTERN bool TDACALL TDA_DeleteLink (void *tdaKernel, __int64 rSourceObject, __int64 rTarget↩ Object, __int64 rAssociationEnd)
- TDAEXTERN bool TDACALL TDA_CreateLink (void *tdaKernel, __int64 rSourceObject, __int64 rTarget↩ Object, __int64 rAssociationEnd)
- TDAEXTERN bool TDACALL TDA_IsLinguistic (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_GetAttributeType (void *tdaKernel, __int64 rAttribute)
- TDAEXTERN __int64 TDACALL TDA_GetTargetClass (void *tdaKernel, __int64 rTargetAssociationEnd)
- TDAEXTERN bool TDACALL TDA_IsAssociationEnd (void *tdaKernel, __int64 r)
- TDAEXTERN void TDACALL TDA_FreeReference (void *tdaKernel, __int64 r)
- TDAEXTERN bool TDACALL TDA_IsAttribute (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_GetSourceClass (void *tdaKernel, __int64 rTargetAssociationEnd)
- TDAEXTERN bool TDACALL TDA_LinkExists (void *tdaKernel, __int64 rSourceObject, __int64 rTarget↩ Object, __int64 rAssociationEnd)
- TDAEXTERN bool TDACALL TDA_IsComposition (void *tdaKernel, __int64 rTargetAssociationEnd)
- TDAEXTERN void TDACALL TDA_FreeIterator (void *tdaKernel, __int64 it)
- TDAEXTERN __int64 TDACALL TDA_GetLinguisticClassFor (void *tdaKernel, __int64 r)

- TDAEXTERN bool TDACALL TDA_DeleteAssociation (void *tdaKernel, __int64 rAssociationEndOr↩ AdvancedAssociation)
- TDAEXTERN bool TDACALL TDA_CreateOrderedLink (void *tdaKernel, __int64 rSourceObject, __int64 r↩ TargetObject, __int64 rAssociationEnd, int targetPosition)
- TDAEXTERN __int64 TDACALL TDA_DeserializeReference (void *tdaKernel, const char *r)
- TDAEXTERN __int64 TDACALL TDA_CreateDirectedAssociation (void *tdaKernel, __int64 rSourceClass, __int64 rTargetClass, const char *targetRoleName, bool isComposition)
- TDAEXTERN bool TDACALL TDA_IsAdvancedAssociation (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_ResolveIteratorFirst (void *tdaKernel, __int64 it)
- TDAEXTERN const char *TDACALL TDA_CallSpecificOperation (void *tdaKernel, const char *operation↩ Name, const char *arguments)
- TDAEXTERN __int64 TDACALL TDA_GetInverseAssociationEnd (void *tdaKernel, __int64 rAssociationEnd)
- TDAEXTERN int TDACALL TDA_GetLinkedObjectPosition (void *tdaKernel, __int64 rSourceObject, __int64 rTargetObject, __int64 rAssociationEnd)
- TDAEXTERN int TDACALL TDA_GetIteratorLength (void *tdaKernel, __int64 it)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForLinkedObjects (void *tdaKernel, __int64 rObject, __↩ int64 rAssociationEnd)
- TDAEXTERN __int64 TDACALL TDA_CreateAdvancedAssociation (void *tdaKernel, const char *name, bool nAry, bool associationClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForLinguisticClasses (void *tdaKernel)
- TDAEXTERN __int64 TDACALL TDA_FindAssociationEnd (void *tdaKernel, __int64 rSourceClass, const char *targetRoleName)
- TDAEXTERN const char *TDACALL TDA_SerializeReference (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_ResolveIteratorNext (void *tdaKernel, __int64 it)

3.1.1 Detailed Description

TDA Kernel DLL RAAPI/IRepository Functions

3.1.2 Function Documentation

3.1.2.1 TDA_Close()

```
TDAEXTERN void TDACALL TDA_Close (
            void * tdaKernel )
```

Closes the repository without save.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference

See also

IRepositoryManagement::startSave
IRepositoryManagement::finishSave
IRepositoryManagement::cancelSave

3.1.2.2 TDA_Exists()

```
TDAEXTERN bool TDACALL TDA_Exists (
            void * tdaKernel,
            const char * location )
```

Checks whether the given location is already occupied by some repository of the same type. This can be used to ask for the user confirmation to drop an existing repository, when creating a new one at the same location.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
location	a string denoting the location to check. The location string is is specific to the type of the repository, e.g., for ECore this is the .xmi file name, for JR this is the folder name, etc. TDA Kernel requires a URI, containing the repository name followed by a colon followed by a repository-specific location, e.g., "jr:/path/to/repository".

Returns

whether the given location is already occupied by some repository of the same type.

3.1.2.3 TDA_Open()

```
TDAEXTERN bool TDACALL TDA_Open (
            void * tdaKernel,
            const char * location )
```

Opens or creates (if the repository does not exist yet) the repository at the given location. This can be used to ask for the user confirmation to drop an existing repository, when creating a new one at the same location.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
location	a string denoting the location of the repository. The location string is is specific to the type of the repository, e.g., for ECore this is the .xmi file name, for JR this is the folder name, etc. TDA Kernel requires a URI, containing the repository name followed by a colon followed by a repository-specific location, e.g., "jr:/path/to/repository".

Returns

whether the repository has been opened or created.

3.1.2.4 TDA_StartSave()

```
TDAEXTERN bool TDACALL TDA_StartSave (
            void * tdaKernel )
```

Starts the two-phase save process of the repository. The save process can be rolled back by calling `cancelSave` or commited by calling `finishSave`.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference

Returns

whether the operation succeeded. If `false` is returned, neither `cancelSave`, nor `finishSave` must be called.

See also

IRepositoryManagement::finishSave
IRepositoryManagement::cancelSave

3.1.2.5 TDA_CancelSave()

```
TDAEXTERN bool TDACALL TDA_CancelSave (
            void * tdaKernel )
```

Rolls back the started save process. The repository content on the disk (or other media) is returned to the previous state. The repository content currently loaded in memory is not changed.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference

Returns

whether the operation succeeded.

See also

IRepositoryManagement::startSave
IRepositoryManagement::finishSave

3.1.2.6 TDA_FinishSave()

```
TDAEXTERN bool TDACALL TDA_FinishSave (
            void * tdaKernel )
```

Finishes the two-phase save process of the repository. After finishing, the save process cannot be rolled back anymore.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference

Returns

whether the operation succeeded.

See also

IRepositoryManagement::startSave
IRepositoryManagement::cancelSave

3.1.2.7 TDA_Drop()

```
TDAEXTERN bool TDACALL TDA_Drop (
            void * tdaKernel,
            const char * location )
```

Deletes the repository at the given location. The repository must be closed.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
location	a string denoting the location of the repository. The location string is specific to the type of the repository, e.g., for ECore this is the .xmi file name, for JR this is the folder name, etc. TDA Kernel requires a URI, containing the repository name followed by a colon followed by a repository-specific location, e.g., "jr:/path/to/repository".

Returns

whether the operation succeeded.

3.1.2.8 TDA_FindClass()

```
TDAEXTERN __int64 TDACALL TDA_FindClass (
            void * tdaKernel,
            const char * name )
```

Obtains a reference to an existing class with the given fully qualified name.
Note (M3): If the underlying repository provides access to its quasi-linguistic meta-metamodel, quasi-linguistic classes can be accessed by using the prefix "M3::", e.g., `SomePath::MountPoint::M3::SomeMetaType`
Note (adapters): This function is optional for repository adapters. If not implemented in an adapter, TDA Kernel implements it through `getIteratorForClasses` [TODO] and `getIteratorForLinguisticClasses`.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
name	the fully qualified name of the class (for quasi-linguistic classes, use prefix "M3::")

Returns

a reference to a (quasi-ontological or quasi-linguistic) class with the given fully qualified name.

See also

on meta-levels

3.1.2.9 TDA_GetClassName()

```
TDAEXTERN const char* TDACALL TDA_GetClassName (
            void * tdaKernel,
            __int64 rClass )
```

Returns the fully qualified name of the given class.
[TODO] ***Note (M3):*** If the reference points to a quasi-linguistic class, then the prefix "M3::" is also included in the return value, e.g., `MountPointForTheCorrespondingRepository::M3::ClassName`.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to the class, for which the class name has to be obtained

Returns

the fully qualified name of the given class, or `null` on error.

See also

on meta-levels

3.1.2.10 TDA_CreateObject()

```
TDAEXTERN __int64 TDACALL TDA_CreateObject (
            void * tdaKernel,
            __int64 rClass )
```

Creates an instance of the given class.
Note (M3): If the given class is a quasi-linguistic class, then its quasi-linguistic instance at Level M_Omega is being created.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to a class (either quasi-ontological, or quasi-linguistic)

Returns

whether the operation succeeded.

See also

on meta-levels

3.1.2.11 TDA_CreateClass()

```
TDAEXTERN __int64 TDACALL TDA_CreateClass (
            void * tdaKernel,
            const char * name )
```

Creates a class with the given fully qualified name.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
name	the fully qualified name of the class (packages are delimited by double colon "::"); this fully qualified name must be unique

Returns

a reference to the class just created, or 0 on error.

3.1.2.12 TDA_DeleteClass()

```
TDAEXTERN bool TDACALL TDA_DeleteClass (
            void * tdaKernel,
            __int64 rClass )
```

Deletes the class and frees the reference.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to the class to be deleted

Returns

whether the operation succeeded.

3.1.2.13 TDA_IsClass()

```
TDAEXTERN bool TDACALL TDA_IsClass (
            void * tdaKernel,
            __int64 r )
```

Checks whether the given reference is associated with a class.
Note (M3): A reference at Level M3 can also be passed.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
r	a reference in question

Returns

whether the given reference is associated with a class. On error, `false` is returned.

See also

on meta-levels

3.1.2.14 TDA_IsDirectSubClass()

```
TDAEXTERN bool TDACALL TDA_IsDirectSubClass (
            void * tdaKernel,
            __int64 rSubClass,
            __int64 rSuperClass )
```

Checks whether the generalization relation between the two given classes holds.
Note (M3): Both classes may be either quasi-ontological, or quasi-linguistic.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSubClass	a reference to a potential subclass
rSuperClass	a reference to a potential superclass

Returns

whether the generalization relation holds. On error, `false` is returned.

See also

`on meta-levels`

3.1.2.15 TDA_DeleteObject()

```
TDAEXTERN bool TDACALL TDA_DeleteObject (
            void * tdaKernel,
            __int64 rObject )
```

Creates the given object.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	a reference to the object to be deleted

Returns

whether the operation succeeded.

3.1.2.16 TDA_MoveObject()

```
TDAEXTERN bool TDACALL TDA_MoveObject (
            void * tdaKernel,
            __int64 rObject,
            __int64 rToClass )
```

Moves (reclassifies) the given object into the given (quasi-ontological) class, removing it from its current class (classes).
The function is similar to calling
`includeObjectInClass(rObject, rToClass);` followed by calling
`excludeObjectInClass(rObject, c)` for all other current classifiers *c* of the given object.
The distinction is that it may be possible to implement this function even when multiple classification is not supported.

Note (adapters): This function is optional for repository adapters. If not implemented in an adapter, TDA Kernel implements it by [TODO] recreating the object (with the new type), while also recreating all attributes and links.
Note (M3): It is assumed that an element from a quasi-ontological level cannot dynamically change its quasi-linguistic type (quasi-linguistic class), thus, `moveObject` is meaningless in this case.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	a reference to the object to be reclassified
rToClass	a reference to the class, to which the object will belong

Returns

whether the operation succeeded.

See also

on meta-levels

3.1.2.17 TDA_IsTypeOf()

```
TDAEXTERN bool TDACALL TDA_IsTypeOf (
            void * tdaKernel,
            __int64 rObject,
            __int64 rClass )
```

Checks whether the given object is a direct (quasi-ontological or quasi-linguistic) instance of the given class.
Note (M3): The function works also when one or both of `rObject` and `rClass` is/are quasi-linguistic. If the object is at a quasi-ontological meta-level, but the class is quasi-linguistic, then the function checks whether the object is a direct quasi-linguistic instance of the given class.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	a reference to an object
rClass	a reference to a class

Returns

whether the given object is a direct instance of the given class. On error, `false` is returned.

See also

RAAPI::isKindOf
on meta-levels

3.1.2.18 TDA_IsKindOf()

```
TDAEXTERN bool TDACALL TDA_IsKindOf (
            void * tdaKernel,
```

```
            __int64 rObject,
            __int64 rClass )
```

Checks whether the given object is a direct or indirect, quasi-ontological or quasi-linguistic, instance of the given class.
Note (M3): The function works also when one or both of `rObject` and `rClass` is/are quasi-linguistic. If the object is at a quasi-ontological meta-level, but the class is quasi-linguistic, then the function checks whether the object is a quasi-linguistic instance of the given class or one of its subclasses.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	a reference to an object
rClass	a reference to a class

Returns

whether the given object is a (direct or indirect) instance of the given class. On error, `false` is returned.

See also

RAAPI::isTypeOf
on meta-levels

3.1.2.19 TDA_CreateAttribute()

```
TDAEXTERN __int64 TDACALL TDA_CreateAttribute (
            void * tdaKernel,
            __int64 rClass,
            const char * name,
            __int64 rPrimitiveType )
```

Creates (defines) a new attribute for the given class. The default cardinality is the widest cardinality supported by the repository (e.g., "0..*", if multi-valued attributes are supported; or "0..1", otherwise). The cardinality can be looked up and changed by using the quasi-linguistic meta-metalevel.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to an existing class, for which to define the attribute
name	the name of the attribute being created; it must be unique within all the attributes defined for this class, including derived ones
rPrimitiveType	a reference to a primitive data type for attribute values

Returns

a reference to the attribute just created, or 0 on error.

See also

on meta-levels

3.1.2.20 TDA_IsDerivedClass()

```
TDAEXTERN bool TDACALL TDA_IsDerivedClass (
            void * tdaKernel,
            __int64 rDirectlyOrIndirectlyDerivedClass,
            __int64 rSuperClass )
```

Checks whether one class is a direct or indirect subclass of another.
Note (M3): Both classes may be either quasi-ontological, or quasi-linguistic.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rDirectlyOrIndirectlyDerivedClass	a reference to a potential subclass or derived class
rSuperClass	a reference to a potential (direct or indirect) superclass

Returns

whether the first class derives from the second. On error, `false` is returned.

See also

on meta-levels

3.1.2.21 TDA_FindAttribute()

```
TDAEXTERN __int64 TDACALL TDA_FindAttribute (
            void * tdaKernel,
            __int64 rClass,
            const char * name )
```

Obtains a reference to an existing attribute with the given name of the given class.
Note (M3): The class reference may point also to a quasi-linguistic class.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to a class, where the attribute in question belongs; `rClass` may be also one of its subclasses, since the attribute is available for subclasses, too
name	the name of the attribute

Returns

a reference to the desired attribute, or 0 on error; the reference returned is the same reference for the class, for which the attribute was defined, as well as for derived classes.

See also

on meta-levels

3.1.2.22 TDA_DeleteAttribute()

```
TDAEXTERN bool TDACALL TDA_DeleteAttribute (
            void * tdaKernel,
            __int64 rAttribute )
```

Deletes the given attribute.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rAttribute	a reference to the attribute to be deleted

Returns

whether the operation succeeded.

3.1.2.23 TDA_GetAttributeName()

```
TDAEXTERN const char* TDACALL TDA_GetAttributeName (
            void * tdaKernel,
            __int64 rAttribute )
```

Returns the name of the given attribute.
Note (M3): The function works also for attributes of quasi-linguistic classes.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rAttribute	a reference to the attribute in question

Returns

the name of the given attribute, or `null` on error.

See also

on meta-levels

3.1.2.24 TDA_GetPrimitiveDataTypeName()

```
TDAEXTERN const char* TDACALL TDA_GetPrimitiveDataTypeName (
            void * tdaKernel,
            __int64 rDataType )
```

Returns the name of the given primitive data type.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rDataType	a reference to a primitive data type, for which the name has to be obtained

Returns

the name of the given primitive data type, or `null` on error.

See also

RAAPI::findPrimitiveDataType

3.1.2.25 TDA_FindPrimitiveDataType()

```
TDAEXTERN __int64 TDACALL TDA_FindPrimitiveDataType (
            void * tdaKernel,
            const char * name )
```

Obtains a reference to a primitive data type with the given name.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
name	the type name. Each repository must support at least four standard primitive data types: "Integer", "Real", "Boolean", and "String". [TODO] Certain repositories may introduce additional primitive types. To denote a repository-specific additional primitive data type, prepend the mount point of that repository, e.g., `MountPoint::PeculiarDataType`.

Returns

a reference to a primitive data type with the given name, or 0 on error.
Note (TDA Kernel): TDA Kernel returns a proxy reference, which is usable even when there are multiple repositories mounted or non-standard primitive data types are used.

3.1.2.26 TDA_IsPrimitiveDataType()

```
TDAEXTERN bool TDACALL TDA_IsPrimitiveDataType (
            void * tdaKernel,
            __int64 r )
```

Checks whether the given reference is associated with a primitive data type.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
r	a reference in question

Returns

whether the given reference is associated with a primitive data type. On error, `false` is returned.

See also

RAAPI::findPrimitiveDataType
RAAPI::getPrimitiveDataTypeName

3.1.2.27 TDA_GetIteratorForClasses()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForClasses (
            void * tdaKernel )
```

Obtains an iterator for all classes (all quasi-ontological classes at all quasi-ontological meta-levels).
Note (M3): Linguistic classes are not traversed by this iterator. Use `getIteratorForLinguisticClasses` instead.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference

Returns

an iterator for all classes, or 0 on error.

See also

RAAPI::getIteratorForLinguisticClasses
on iterators
on meta-levels

3.1.2.28 TDA_GetAttributeValue()

```
TDAEXTERN const char* TDACALL TDA_GetAttributeValue (
            void * tdaKernel,
            __int64 rObject,
            __int64 rAttribute )
```

Gets the value or the ordered collection of values (encoded as a string) of the given attribute for the given object.
Note (M3): The attribute reference can be a reference at the M3 level.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	the object, for which to get the attribute value (values)
rAttribute	the attribute, for which to obtain the value; this attribute must be associated either with a quasi-ontological class or the quasi-linguistic class of the given object

Returns

the attribute value (values) encoded as a string (for the decimal point the dot symbol "." is used), or `null` on error.

See also

on encoding values
on meta-levels

3.1.2.29 TDA_CreateAssociation()

```
TDAEXTERN __int64 TDACALL TDA_CreateAssociation (
            void * tdaKernel,
            __int64 rSourceClass,
            __int64 rTargetClass,
            const char * sourceRoleName,
            const char * targetRoleName,
            bool isComposition )
```

Creates a bidirectional association (or two directed associations, where each is an inverse of the other). The default value for the source and target cardinalities should be "∗".
Note (M3): The M3 level can be used to get/set the cardinality, if the repository supports constraints and the M3 level operations. Cardinality constraints must be accessible via M3 for that.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSourceClass	the class, where the association starts
rTargetClass	the class, where the association ends
sourceRoleName	the name of the association end near the source class
targetRoleName	the name of the association end near the target class
isComposition	whether the association is a composition, i.e., the source class objects are containers for the target class objects

Returns

a reference for the target association end of the association just created, or 0 on error.

See also

on meta-levels

3.1.2.30 TDA_GetIteratorForDirectSubClasses()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectSubClasses (
            void * tdaKernel,
            __int64 rSuperClass )
```

Obtains an iterator for all direct subclasses of the given superclass.
Note (M3): If the given superclass is a quasi-linguistic class, then an iterator for direct quasi-linguistic subclasses is returned.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSuperClass	a superclass for which to obtain direct subclasses

Returns

an iterator for all direct subclasses of the given superclass, or 0 on error.

See also

on iterators
on meta-levels

3.1.2.31 TDA_GetIteratorForAllClassObjects()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllClassObjects (
            void * tdaKernel,
            __int64 rClassOrAdvancedAssociation )
```

Obtains an iterator for all quasi-ontological instances of the given class or advanced association.
Note (adapters): A repository adapter may implement only one of the functions `getIteratorForAllClass↩Objects` and `getIteratorForDirectClassObjects`. The unimplemented function will be implemented via another by TDA Kernel.
Note (M3): If the given class or advanced association is quasi-linguistic, then an iterator for the quasi-linguistic elements it describes is returned, e.g., for the EMOF class "Class", an iterator for all classes found in EMOF is returned; for the EMOF class "Property", an iterator for all properties found in EMOF is returned, etc.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClassOrAdvancedAssociation	a reference to a class or an advanced association

Returns

an iterator for all quasi-ontological instances (objects) of the given class or advanced association. On error, 0 is returned.

See also

RAAPI::getIteratorForDirectClassObjects
on iterators
on advanced associations
on meta-levels

3.1.2.32 TDA_GetIteratorForDirectAttributes()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectAttributes (
            void * tdaKernel,
            __int64 rClass )
```

Obtains an iterator for direct (without inherited) attributes of the given class.
Note (adapters): A repository adapter may implement only one of the functions `getIteratorForAll↩Attributes` and `getIteratorForDirectAttributes`. The unimplemented function will be implemented via another by TDA Kernel.
Note (M3): The function works also for quasi-linguistic classes.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to a class, whose attributes we are interested in

Returns

an iterator for direct attributes of the given class. On error, 0 is returned.

See also

RAAPI::getIteratorForAllAttributes
on iterators
on meta-levels

3.1.2.33 TDA_GetAttributeDomain()

```
TDAEXTERN __int64 TDACALL TDA_GetAttributeDomain (
            void * tdaKernel,
            __int64 rAttribute )
```

Obtains a class, for which the given attribute was defined.
Note (M3): The function works also for attributes of quasi-linguistic classes.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rAttribute	a reference to the attribute in question

Returns

a reference to a class, for which the given attribute belongs, or 0 on error.

See also

on meta-levels

3.1.2.34 TDA_DeleteGeneralization()

```
TDAEXTERN bool TDACALL TDA_DeleteGeneralization (
            void * tdaKernel,
            __int64 rSubClass,
            __int64 rSuperClass )
```

Deletes the generalization between the given two classes.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSubClass	a class that was a subclass
rSuperClass	a class that was a superclass

Returns

whether the operation succeeded.

3.1.2.35 TDA_CreateGeneralization()

```
TDAEXTERN bool TDACALL TDA_CreateGeneralization (
            void * tdaKernel,
            __int64 rSubClass,
            __int64 rSuperClass )
```

Creates a generalization between the two given classes.
The given subclass can be a derived class of the given superclass, but the direct generalization between them must not exist.
The generalization relation being created must not introduce inheritance loops.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSubClass	a class that becomes a subclass
rSuperClass	a class that becomes a superclass

Returns

whether the operation succeeded.

3.1.2.36 TDA_ExcludeObjectFromClass()

```
TDAEXTERN bool TDACALL TDA_ExcludeObjectFromClass (
            void * tdaKernel,
            __int64 rObject,
            __int64 rClass )
```

Takes out the given object from the given (quasi-ontological) class.
The function works, if the underlying repository supports multiple classification and dynamic reclassification. If the object currently is only in one class, then the operation fails (it is assumed that each object must be at least in one class).
Note (M3): It is assumed that an element from a quasi-ontological level can be associated with only one quasi-linguistic type (quasi-linguistic class), thus, `excludeObjectInClass` as well as `includeObjectInClass` are meaningless in this case.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	a reference to the object to be excluded from the given class
rClass	a reference to the class, which to exclude from the classifiers of the given object

Returns

whether the operation succeeded.

See also

on meta-levels

3.1.2.37 TDA_GetIteratorForAllAttributes()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllAttributes (
            void * tdaKernel,
            __int64 rClass )
```

Obtains an iterator for all (including inherited) attributes of the given class.
Note (adapters): A repository adapter may implement only one of the functions `getIteratorForAll↩Attributes` and `getIteratorForDirectAttributes`. The unimplemented function will be implemented via another by TDA Kernel.
Note (M3): The function works also for quasi-linguistic classes.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to a class, whose attributes we are interested in

Returns

an iterator for all attributes (including inherited) of the given class. On error, 0 is returned.

See also

RAAPI::getIteratorForDirectAttributes
on iterators
on meta-levels

3.1.2.38 TDA_IncludeObjectInClass()

```
TDAEXTERN bool TDACALL TDA_IncludeObjectInClass (
            void * tdaKernel,
            __int64 rObject,
            __int64 rClass )
```

Adds the given object to the given (quasi-ontological) class. The function works, if the underlying repository supports multiple classification and dynamic reclassification.
Note (M3): It is assumed that an element from a quasi-ontological level can be associated with only one quasi-linguistic type (quasi-linguistic class), thus, `includeObjectInClass` is meaningless in this case.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	a reference to the object to be included in the given class
rClass	a reference to the class, where to put the object (in addition to classes, where the object already belongs)

Returns

whether the operation succeeded.

See also

on meta-levels

3.1.2.39 TDA_GetIteratorForDirectSuperClasses()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectSuperClasses (
            void * tdaKernel,
            __int64 rSubClass )
```

Obtains an iterator for all direct superclasses of the given subclass.
Note (M3): If the given subclass is a quasi-linguistic class, then an iterator for its direct quasi-linguistic superclasses is returned.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSubClass	a subclass for which to obtain direct superclasses

Returns

an iterator for all direct superclasses of the given subclass, or 0 on error.

See also

on iterators
on meta-levels

3.1.2.40 TDA_GetIteratorForDirectClassObjects()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectClassObjects (
            void * tdaKernel,
            __int64 rClassOrAdvancedAssociation )
```

Obtains an iterator for direct quasi-ontological instances of the given class or advanced association.
Note (adapters): A repository adapter may implement only one of the functions `getIteratorForAllClass↩Objects` and `getIteratorForDirectClassObjects`. The unimplemented function will be implemented via another by TDA Kernel.
Note (M3): If the given class or advanced association is quasi-linguistic, then an iterator for the quasi-linguistic elements it describes is returned, e.g., for the EMOF class "Class", an iterator for all classes found in EMOF is returned; for the EMOF class "Property", an iterator for all properties found in EMOF is returned, etc.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClassOrAdvancedAssociation	a reference to a class or an advanced association

Returns

an iterator for direct quasi-ontological instances (objects) of the given class or advanced association. On error, 0 is returned.

See also

RAAPI::getIteratorForAllClassObjects
on iterators
on advanced associations
on meta-levels

3.1.2.41 TDA_SetAttributeValue()

```
TDAEXTERN bool TDACALL TDA_SetAttributeValue (
            void * tdaKernel,
            __int64 rObject,
            __int64 rAttribute,
            const char * value )
```

Sets the value or the ordered collection of values (encoded as a string) of the given attribute for the given object.
Note (adapters): Repository adapters may assume that the value is not `null` and not a string encoding `null`, since for those cases TDA Kernel forwards the call to `deleteAttributeValue`.
Note (M3): The attribute reference can be a reference at the M3 level. In this case the object can be any element at the M_Omega level.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	the object, for which to set the attribute value (values)
rAttribute	the attribute, for which to set the value; this attribute must be associated either with a quasi-ontological class or the quasi-linguistic class of the given object
value	the attribute value (values) encoded as a string (use "." for the decimal point)

Returns

whether the value(s) has (have) been set. On error, `false` is returned.

See also

on encoding values
on meta-levels

3.1.2.42 TDA_DeleteAttributeValue()

```
TDAEXTERN bool TDACALL TDA_DeleteAttributeValue (
            void * tdaKernel,
            __int64 rObject,
            __int64 rAttribute )
```

Deletes the value (all the values) of the given attribute for the given object.
Note (M3): The attribute reference can be a reference at the M3 level. In this case the object can be any element at the M_Omega level.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	the object, for which to get the attribute value (values)
rAttribute	the attribute, for which to obtain the value; this attribute must be associated either with a quasi-ontological class or the quasi-linguistic class of the given object

Returns

whether the operation succeeded.

See also

on encoding values
on meta-levels

3.1.2.43 TDA_GetIteratorForDirectLinguisticInstances()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectLinguisticInstances (
            void * tdaKernel,
            __int64 rClass )
```

Returns an iterator for direct quasi-linguistic instances (not including instances of subclasses) at Level M_Omega of the given class at Level M3.
Note (M3): This function takes a class at Level M3 and returns an iterator for elements at Level M_Omega.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a Level M3 class

Returns

an iterator for direct quasi-linguistic instances at Level M_Omega of the given class at Level M3, or 0 on error.

See also

on meta-levels

3.1.2.44 TDA_GetIteratorForDirectObjectClasses()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectObjectClasses (
            void * tdaKernel,
            __int64 rObjectOrAdvancedLink )
```

Obtains an iterator for direct quasi-ontological classes of the given object or advanced link.
Note (M3): The function works also if the given object or advanced link is quasi-linguistic and the underlying repository provides access to quasi-linguistic elements. To get the quasi-linguistic class for the given element at some quasi-ontological level, use `getLinguisticClassFor`.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObjectOrAdvancedLink	a reference to an object or advanced link

Returns

an iterator for direct quasi-ontological classes of the given object or advanced link. On error, 0 is returned.

See also

RAAPI::getLinguisticClassFor
on iterators
on advanced associations
on meta-levels

3.1.2.45 TDA_GetIteratorForObjectsByAttributeValue()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForObjectsByAttributeValue (
            void * tdaKernel,
```

```
__int64 rAttribute,
const char * value )
```

Obtains an iterator for objects, for whose the value of the given attribute equals to the given value. The value has to be encoded as a string (it may encode an ordered collection of multiple values).
Note (M3): The attribute reference can be a reference at the M3 level. In this case the objects traversed by the returned iterator are elements at the M_Omega level.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rAttribute	the attribute to check
value	the value to check

Returns

the iterator for objects with the given attribute value, or 0 on error.

See also

on iterators
on encoding values
on meta-levels

3.1.2.46 TDA_GetIteratorForAllOutgoingAssociationEnds()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllOutgoingAssociationEnds (
            void * tdaKernel,
            __int64 rClass )
```

Obtains an iterator for all (including inherited) outgoing association ends of the given class.
Note (adapters): A repository adapter may implement only one of the functions `getIteratorForAll↩OutgoingAssociationEnds` and `getIteratorForDirectOutgoingAssociationEnds`. The unimplemented function will be implemented via another by TDA Kernel.
Note (M3): The function works also for associations at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to a class, whose outgoing associations (including inherited) have to be traversed

Returns

an iterator for all (including inherited) outgoing association ends of the given class. On error, 0 is returned.

See also

RAAPI::getIteratorForDirectOutgoingAssociationEnds
on iterators
on meta-levels

3.1.2.47 TDA_GetIteratorForDirectIngoingAssociationEnds()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectIngoingAssociationEnds (
            void * tdaKernel,
            __int64 rClass )
```

Obtains an iterator for direct (without inherited) ingoing association ends of the given class.
Note (adapters): A repository adapter may implement only one of the functions `getIteratorForAll↩IngoingAssociationEnds` and `getIteratorForDirectIngoingAssociationEnds`. The unimplemented function will be implemented via another by TDA Kernel.
Note (M3): The function works also for associations at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to a class, whose direct ingoing associations have to be traversed

Returns

an iterator for direct (without inherited) ingoing association ends of the given class. On error, 0 is returned.

See also

RAAPI::getIteratorForAllIngoingAssociationEnds
on iterators
on meta-levels

3.1.2.48 TDA_GetIteratorForAllLinguisticInstances()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllLinguisticInstances (
            void * tdaKernel,
            __int64 rClass )
```

Returns an iterator for all quasi-linguistic instances (including instances of subclasses) at Level M_Omega of the given class at Level M3.
Note (M3): This function takes a class at Level M3 and returns an iterator for elements at Level M_Omega.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a Level M3 class

Returns

an iterator for direct quasi-linguistic instances at Level M_Omega of the given class (and its subclasses) at Level M3, or 0 on error.

See also

on meta-levels

3.1.2.49 TDA_GetIteratorForDirectOutgoingAssociationEnds()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectOutgoingAssociationEnds (
            void * tdaKernel,
            __int64 rClass )
```

Obtains an iterator for direct (without inherited) outgoing association ends of the given class.
Note (adapters): A repository adapter may implement only one of the functions `getIteratorForAll↩OutgoingAssociationEnds` and `getIteratorForDirectOutgoingAssociationEnds`. The unimplemented function will be implemented via another by TDA Kernel.
Note (M3): The function works also for associations at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to a class, whose direct outgoing associations have to be traversed

Returns

an iterator for direct (without inherited) outgoing association ends of the given class. On error, 0 is returned.

See also

RAAPI::getIteratorForAllOutgoingAssociationEnds
on iterators
on meta-levels

3.1.2.50 TDA_GetIteratorForAllIngoingAssociationEnds()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllIngoingAssociationEnds (
            void * tdaKernel,
            __int64 rClass )
```

Obtains an iterator for all (including inherited) ingoing association ends of the given class.
Note (adapters): A repository adapter may implement only one of the functions `getIteratorForAll↩IngoingAssociationEnds` and `getIteratorForDirectIngoingAssociationEnds`. The unimplemented function will be implemented via another by TDA Kernel.
Note (M3): The function works also for associations at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rClass	a reference to a class, whose ingoing associations (including inherited) have to be traversed

Returns

an iterator for all (including inherited) ingoing association ends of the given class. On error, 0 is returned.

See also

RAAPI::getIteratorForDirectIngoingAssociationEnds
on iterators
on meta-levels

3.1.2.51 TDA_ResolveIterator()

```
TDAEXTERN __int64 TDACALL TDA_ResolveIterator (
            void * tdaKernel,
            __int64 it,
            int position )
```

Returns a reference to the element at the given `position` (numeration starts from 0) and forwards the iterator to `position+1`.
Note (adapters): If not implemented in a repository adapter, TDA Kernel traverses all the elements and stores them in a temporary list. Thus, the first call will take the linear execution time, while all subsequent calls will take the constant time. The same refers to the `getIteratorLength` function. If both `getIteratorLength` and `resolveIterator` are used, the temporary list is created only once.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
it	an iterator reference
position	the position in the iterable list, where the interested element is located

Returns

a reference to the element at the given position, or 0 if the position is out of bounds, or if an error occurred.

See also

RAAPI::getIteratorLength
RAAPI::freeIterator
on iterators

3.1.2.52 TDA_GetRoleName()

```
TDAEXTERN const char* TDACALL TDA_GetRoleName (
            void * tdaKernel,
            __int64 rAssociationEnd )
```

Returns the role name of the given association end.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rAssociationEnd	an association end of some directed, bidirectional, or advanced association

Returns

the role name of the given association end, or `null` on error.

See also

`on advanced associations`

3.1.2.53 TDA_DeleteLink()

```
TDAEXTERN bool TDACALL TDA_DeleteLink (
            void * tdaKernel,
            __int64 rSourceObject,
            __int64 rTargetObject,
            __int64 rAssociationEnd )
```

Deletes a link of the given type (specified by `rTargetAssociationEnd`) between the given two objects.
Note (M3): An association end at Level M3 can also be passed. In this case, at least one of the source and target objects must be an element at the M_Omega level. The semantics of such link then depends on a particular quasi-linguistic metamodel at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSourceObject	a start object of the link; this object must be an instance of the source class for the given association end
rTargetObject	an end object of the link; this object must be an instance of the target class for the given association end
rAssociationEnd	a target association end that specifies the link type

Returns

whether the operation succeeded.

See also

on advanced associations
on meta-levels

3.1.2.54 TDA_CreateLink()

```
TDAEXTERN bool TDACALL TDA_CreateLink (
            void * tdaKernel,
            __int64 rSourceObject,
            __int64 rTargetObject,
            __int64 rAssociationEnd )
```

Creates a link of the given type (specified by `rAssociationEnd`) between two objects.
Note (M3): An association end at Level M3 can also be passed. In this case, at least one of the source and target objects must be an element at the M_Omega level. The semantics of such link then depends on a particular quasi-linguistic metamodel at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSourceObject	a start object of the link; this object must be an instance of the source class for the given association end
rTargetObject	an end object of the link; this object must be an instance of the target class for the given association end
rAssociationEnd	a target association end that specifies the link type

Returns

whether the operation succeeded.

See also

on advanced associations
on meta-levels

3.1.2.55 TDA_IsLinguistic()

```
TDAEXTERN bool TDACALL TDA_IsLinguistic (
            void * tdaKernel,
            __int64 r )
```

Checks, whether the given reference is associated with a Level M3 element. Can be used together with `isClass`, `isAssociationEnd`, etc. to get more details about the element.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
r	a reference in question

Returns

whether the given reference is associated with a Level M3 element. On error, `false` is returned.

See also

on meta-levels

3.1.2.56 TDA_GetAttributeType()

```
TDAEXTERN __int64 TDACALL TDA_GetAttributeType (
            void * tdaKernel,
            __int64 rAttribute )
```

Returns the (primitive) type for values of the given attribute.
Note (M3): The function works also for attributes of quasi-linguistic classes.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rAttribute	a reference to the attribute in question

Returns

a reference to a primitive data type for values of the given attribute.

See also

on meta-levels

3.1.2.57 TDA_GetTargetClass()

```
TDAEXTERN __int64 TDACALL TDA_GetTargetClass (
            void * tdaKernel,
            __int64 rTargetAssociationEnd )
```

Obtains a reference to the class corresponding to the given association end of some directed, bidirectional, or advanced association. For bidirectional and advanced associations, any of the two association ends can be considered a target end, when calling this function.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rTargetAssociationEnd	an association end of some association; this association end will be considered a target end

Returns

a reference to the class corresponding to the given association end.

See also

on advanced associations

3.1.2.58 TDA_IsAssociationEnd()

```
TDAEXTERN bool TDACALL TDA_IsAssociationEnd (
            void * tdaKernel,
            __int64 r )
```

Checks, whether the given reference corresponds to an association end.
Note (adapters): If not implemented in a repository adapter, TDA Kernel will implement it by means of `getSourceClass`, `getRoleName` and `findAssociationEnd`.
Note (M3): A reference at Level M3 can also be passed.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
r	a reference in question

Returns

whether the given reference corresponds to an association end. On error, `false` is returned.

See also

on advanced associations
on meta-levels

3.1.2.59 TDA_FreeReference()

```
TDAEXTERN void TDACALL TDA_FreeReference (
            void * tdaKernel,
            __int64 r )
```

Frees the memory associated with the given reference (if necessary).

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference

3.1.2.60 TDA_IsAttribute()

```
TDAEXTERN bool TDACALL TDA_IsAttribute (
            void * tdaKernel,
            __int64 r )
```

Checks whether the given reference is associated with an attribute.
Note (adapters): If a repository adapter does not implement this function, TDA Kernel will implement it by means of `getAttributeDomain`, `getAttributeName` and `findAttribute`.
Note (M3): A reference at Level M3 can also be passed.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
r	a reference in question

Returns

whether the given reference is associated with an attribute. On error, `false` is returned.

See also

on meta-levels

3.1.2.61 TDA_GetSourceClass()

```
TDAEXTERN __int64 TDACALL TDA_GetSourceClass (
            void * tdaKernel,
            __int64 rTargetAssociationEnd )
```

Obtains a reference to the source class of the given directed or bidirectional association (or part of an advanced association) specified by its target end. Any of the association ends can be considered a target end, when calling this function.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rTargetAssociationEnd	an association end of some association; this association end will be considered a target end

Returns

a reference to the source class of the given association specified by its target end.

See also

on advanced associations

3.1.2.62 TDA_LinkExists()

```
TDAEXTERN bool TDACALL TDA_LinkExists (
            void * tdaKernel,
            __int64 rSourceObject,
            __int64 rTargetObject,
            __int64 rAssociationEnd )
```

Checks whether the link of the given type (specified by `rTargetAssociationEnd`) between the given two objects exists.
Note (adapters): If not implemented in a repository adapter, TDA Kernel will implement this function through `getIteratorForLinkedObjects`.
Note (M3): An association end at Level M3 can also be passed. In this case, at least one of the source and target objects must be an element at the M_Omega level. The semantics of such link then depends on a particular quasi-linguistic metamodel at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSourceObject	a start object of the link; this object must be an instance of the source class for the given association end
rTargetObject	an end object of the link; this object must be an instance of the target class for the given association end
rAssociationEnd	a target association end that specifies the link type

Returns

whether the link exists. On error, `false` is returned.

See also

on advanced associations
on meta-levels

3.1.2.63 TDA_IsComposition()

```
TDAEXTERN bool TDACALL TDA_IsComposition (
            void * tdaKernel,
            __int64 rTargetAssociationEnd )
```

Returns, whether the directed or bidirectional association (or a part of an advanced association) specified by its target association end is a composition (i.e., whether the source class objects are containers for the target class objects).
Note (M3): A reference at Level M3 can also be passed.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rTargetAssociationEnd	an association end of some association; this association end will be considered a target end

Returns

whether the directed or bidirectional association (or a part of an advanced association) is a composition.

See also

on advanced associations
on meta-levels

3.1.2.64 TDA_FreeIterator()

```
TDAEXTERN void TDACALL TDA_FreeIterator (
            void * tdaKernel,
            __int64 it )
```

Frees the memory associated with the given iterator reference.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
it	an iterator reference

See also

on iterators

3.1.2.65 TDA_GetLinguisticClassFor()

```
TDAEXTERN __int64 TDACALL TDA_GetLinguisticClassFor (
            void * tdaKernel,
            __int64 r )
```

Returns a reference to the Level M3 class of the given quasi-ontological (Level M_Omega) element. It is assumed that there may be at most one quasi-linguistic class at M3 for each quasi-ontological element at M_Omega.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
r	a quasi-ontological (Level M_Omega) element

Returns

a reference to the Level M3 class of the given quasi-ontological (Level M_Omega) element. On error or if M3 is not supported by the underlying repository, 0 is returned.

See also

on meta-levels

3.1.2.66 TDA_DeleteAssociation()

```
TDAEXTERN bool TDACALL TDA_DeleteAssociation (
            void * tdaKernel,
            __int64 rAssociationEndOrAdvancedAssociation )
```

Deletes the given association. Directed and bidirectional associations are specified by (one of) their ends. Advanced associations have their own references. If the association is bidirectional, the inverse association end is deleted as well. For advanced associations, all association parts are deleted.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rAssociationEndOrAdvancedAssociation	a reference to an association end (if the association is directed or bidirectional) or a reference to an advanced association

Returns

whether the operation succeeded.

See also

on advanced associations

3.1.2.67 TDA_CreateOrderedLink()

```
TDAEXTERN bool TDACALL TDA_CreateOrderedLink (
            void * tdaKernel,
            __int64 rSourceObject,
            __int64 rTargetObject,
            __int64 rAssociationEnd,
            int targetPosition )
```

Creates a link of the given type (specified by `rAssociationEnd`) between two objects at the given position. The target position normally should be from 0 to n, where n is the number of currently linked objects at positions from 0 to n-1. If the target position is outside [0..n], then the link is appended to the end.
Note (M3): An association end at Level M3 can also be passed. In this case, at least one of the source and target objects must be an element at the M_Omega level. The semantics of such link then depends on a particular quasi-linguistic metamodel at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSourceObject	a start object of the link; this object must be an instance of the source class for the given association end
rTargetObject	an end object of the link; this object must be an instance of the target class for the given association end
rAssociationEnd	a target association end that specifies the link type
targetPosition	the position (starting from 0) of the target object in the list of linked objects of the source object;

Returns

whether the operation succeeded.

See also

on advanced associations
on meta-levels

3.1.2.68 TDA_DeserializeReference()

```
TDAEXTERN __int64 TDACALL TDA_DeserializeReference (
            void * tdaKernel,
            const char * r )
```

Obtains a reference to a serialized element from the given serialization. This is essential for loading inter-repository relations.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
r	the serialization of an element, for which to obtain a reference

Returns

a reference corresponding for the given serialization, or 0 on error.

See also

RAAPI::serializeReference

3.1.2.69 TDA_CreateDirectedAssociation()

```
TDAEXTERN __int64 TDACALL TDA_CreateDirectedAssociation (
            void * tdaKernel,
            __int64 rSourceClass,
            __int64 rTargetClass,
            const char * targetRoleName,
            bool isComposition )
```

Creates a directed association. The default value for the source and target cardinalities should be "∗".
Note (adapters): If a repository adapter does not implement this function, TDA kernel will simulate it by means of `createAssociation` (a stub inverse role will be generated).
Note (M3): The M3 level can be used to get/set the cardinality, if the repository supports constraints and the M3 level operations. Cardinality constraints must be accessible via M3 for that.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSourceClass	the class, where the association starts
rTargetClass	the class, where the association ends
targetRoleName	the name of the association end near the target class
isComposition	whether the association is a composition, i.e., the source class objects are containers for the target class objects

Returns

a reference for the target association end of the association just created, or 0 on error.

See also

on meta-levels

3.1.2.70 TDA_IsAdvancedAssociation()

```
TDAEXTERN bool TDACALL TDA_IsAdvancedAssociation (
            void * tdaKernel,
            __int64 r )
```

Checks, whether the given reference corresponds to an advanced association.
Note (M3): A reference at Level M3 can also be passed.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
r	a reference in question

Returns

whether the given reference corresponds to an advanced association. On error, `false` is returned.

See also

on advanced associations
on meta-levels

3.1.2.71 TDA_ResolveIteratorFirst()

```
TDAEXTERN __int64 TDACALL TDA_ResolveIteratorFirst (
            void * tdaKernel,
            __int64 it )
```

Places the iterator to the position 0 and gets the element there.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
it	an iterator reference

Returns

the element at position 0 in the iterable list. If there are no elements or if an error occurred, 0 is returned.

See also

RAAPI::resolveIteratorNext
RAAPI::freeIterator
on iterators

3.1.2.72 TDA_CallSpecificOperation()

```
TDAEXTERN const char* TDACALL TDA_CallSpecificOperation (
            void * tdaKernel,
            const char * operationName,
            const char * arguments )
```

Calls a repository-specific operation (e.g., or MOF/ECore-like operation, an SQL statement, or a SPARQL statement). Arguments (if any) are encoded as a string delimited by means of the Unicode character U+001E (INFORMATION SEPARATOR TWO). For no-argument methods `arguments` must be `null`.
For instance, a repository may accept the following calls:

```
callSpecificOperation("SQL", "SELECT * FROM MY TABLE");
callSpecificOperation("myMethod", "[object-reference][u001E][argument1]...");

callSpecificOperation("", null);
```

For static MOF/ECore-like operations, the first argument should point to a class. For non-static operations the first argument should point to an object (that resembles `this` pointer in Java).

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
operationName	a repository-specific operation name
arguments	operation-specific arguments encoded as a string

Returns

the return value of the call encoded as a string, or `null` on error.

See also

on encoding values

3.1.2.73 TDA_GetInverseAssociationEnd()

```
TDAEXTERN __int64 TDACALL TDA_GetInverseAssociationEnd (
            void * tdaKernel,
            __int64 rAssociationEnd )
```

Obtains a reference to the inverse association end of the given association end (if association is bidirectional or a bidirectional part of an advanced association).
Note (M3): The function works also for association ends at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rAssociationEnd	a reference to a known association end, for which the inverse end has to be obtained

Returns

a reference to the inverse association end. On error or if the association end does not have the inverse, 0 is returned.

See also

on advanced associations
on meta-levels

3.1.2.74 TDA_GetLinkedObjectPosition()

```
TDAEXTERN int TDACALL TDA_GetLinkedObjectPosition (
            void * tdaKernel,
            __int64 rSourceObject,
```

```
            __int64 rTargetObject,
            __int64 rAssociationEnd )
```

Returns the index (numeration starts from 0) of the target object in the list of objects linked to the source object by links of the given type.
Note (M3): The type of links may also be an association end at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSourceObject	a source object; this object must be an instance of the source class for the given association end
rTargetObject	a target object; this object must be an instance of the target class for the given association end
rAssociationEnd	a target association end that specifies the type of links

Returns

the index (numeration starts from 0) of the given target object in the list of objects linked to the source object. On error or when the source and target objects are not linked by the given association, -1 is returned.

See also

on iterators
on advanced associations
on meta-levels

3.1.2.75 TDA_GetIteratorLength()

```
TDAEXTERN int TDACALL TDA_GetIteratorLength (
            void * tdaKernel,
            __int64 it )
```

Places the iterator to the position 0 and returns the total number of elements to iterate through. Call `resolveIteratorFirst` or `resolveIterator` to move the iterator.
Note (adapters): If not implemented in a repository adapter, TDA Kernel traverses all the elements and stores them in a temporary list. Thus, the first call will take the linear execution time, while all subsequent calls will take the constant time. The same refers to the `resolveIterator` function. If both `getIteratorLength` and `resolveIterator` are used, the temporary list is created only once.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
it	an iterator reference

Returns

the total number of elements to iterate through. On error returns 0 (thus, the return value still represents the number of iterations, which can be performed with this iterator).

See also

RAAPI::resolveIterator
RAAPI::freeIterator
on iterators

3.1.2.76 TDA_GetIteratorForLinkedObjects()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForLinkedObjects (
            void * tdaKernel,
            __int64 rObject,
            __int64 rAssociationEnd )
```

Returns an iterator for objects linked to the given start object by links of the given type.
Note (M3): The type of links may also be an association end at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rObject	a start object, for which the iterable objects are linked this object must be an instance of the source class for the given association end
rAssociationEnd	a target association end that specifies the type of links

Returns

an iterator for objects, linked to the given object by links of the given type, or 0 on error.

See also

on iterators
on advanced associations
on meta-levels

3.1.2.77 TDA_CreateAdvancedAssociation()

```
TDAEXTERN __int64 TDACALL TDA_CreateAdvancedAssociation (
            void * tdaKernel,
            const char * name,
            bool nAry,
            bool associationClass )
```

Creates an n-ary association, an association class, or an n-ary association class.
An advanced association behaves likes a class (although it might not be a class internally) with n bidirectional associations attached to it. To specify all n association ends, call `createAssociation` n times, where a reference to the n-ary association has to be passed instead of one of the class references. N-ary association links can be created by means of `createObject`, and n-ary link ends can be created by calling `createLink` n times and passing a reference to the n-ary link instead of one of the object references.

Note (adapters): The underlying repository is allowed to create an n-ary association class, even when nAry or associationClass is `false`.
Note (adapters): If a repository adapter does not implement this function, TDA kernel will implement this function by introducing an additional class.
Note (M3): The M3 level can be used to get/set the cardinality, if the repository supports constraints and the M3 level operations. Cardinality constraints must be accessible via M3 for that.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
name	the name of the advanced association (the class name in case of an association class)
nAry	whether the association is an n-ary association
associationClass	whether the association is an association class

Returns

a reference to the n-ary association just created (not the association end, since no association ends are created yet), or 0 on error.

See also

on meta-levels
on advanced associations

3.1.2.78 TDA_GetIteratorForLinguisticClasses()

```
TDAEXTERN __int64 TDACALL TDA_GetIteratorForLinguisticClasses (
            void * tdaKernel )
```

Returns an iterator for all quasi-linguistic classes at Level M3.
Note (M3): This function works only at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference

Returns

an iterator for all quasi-linguistic classes at Level M3.

See also

on meta-levels

3.1.2.79 TDA_FindAssociationEnd()

```
TDAEXTERN __int64 TDACALL TDA_FindAssociationEnd (
            void * tdaKernel,
            __int64 rSourceClass,
            const char * targetRoleName )
```

Obtains a reference to an association end (by its role name) starting at the given class.

Note (adapters): If not implemented in the adapter, TDA kernel will implement it by means of `getIterator↩ForAllOutgoingAssociationEnds`.
Note (M3): The function works also, when searching for association ends at Level M3.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
rSourceClass	a class that is a source class for the association, or one of its subclasses
targetRoleName	a role name associated with the target association end

Returns

a reference to an association end corresponding to the given target role name.

See also

on meta-levels

3.1.2.80 TDA_SerializeReference()

```
TDAEXTERN const char* TDACALL TDA_SerializeReference (
            void * tdaKernel,
            __int64 r )
```

Creates a string representation of the given reference, which survives the current session. For the next session, TDA kernel will use this string to get another reference to the same element by means of `deserializeReference`. This is essential for storing inter-repository relations.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
r	the reference to serialize

Returns

a string representation of the given reference, which survives the current session, or `null` on error.

See also

RAAPI::deserializeReference

3.1.2.81 TDA_ResolveIteratorNext()

```
TDAEXTERN __int64 TDACALL TDA_ResolveIteratorNext (
            void * tdaKernel,
            __int64 it )
```

Moves the iterator forward and gets the element at that position.

Parameters

tdaKernel	a pointer to TDA Kernel returned by TDA_GetTDAKernelReference
it	an iterator reference

Returns

the element the iterator points to, after the iterator has been moved one step forward. If there are no elements or if an error occurred, 0 is returned.

See also

RAAPI::resolveIteratorFirst
RAAPI::freeIterator
on iterators

3.2 TDA Kernel DLL Additional Functions

Functions

- const char * TDA_CreateReturnString (const char *s)
- unsigned int TDA_GetCurrentProcessID ()
- unsigned int TDA_GetParentProcessID ()
- TDAEXTERN const char *TDACALL TDA_GetTDAKernelLibraryPath ()
- TDAEXTERN const wchar_t *TDACALL TDA_GetTDAKernelLibraryPathW ()
- TDAEXTERN void *TDACALL TDA_GetTDAKernelReference (const char *protocolOrUri)
- TDAEXTERN bool TDACALL TDA_FreeTDAKernelReference (void *tdaKernel)

3.2.1 Detailed Description

TDA Kernel DLL Additional Functions

3.2.2 Function Documentation

3.2.2.1 TDA_CreateReturnString()

```
const char* TDA_CreateReturnString (
            const char * s )
```

3.2.2.2 TDA_GetCurrentProcessID()

```
unsigned int TDA_GetCurrentProcessID ( )
```

3.2.2.3 TDA_GetParentProcessID()

```
unsigned int TDA_GetParentProcessID ( )
```

3.2.2.4 TDA_GetTDAKernelLibraryPath()

```
TDAEXTERN const char* TDACALL TDA_GetTDAKernelLibraryPath ( )
```

3.2.2.5 TDA_GetTDAKernelLibraryPathW()

```
TDAEXTERN const wchar_t* TDACALL TDA_GetTDAKernelLibraryPathW ( )
```

3.2.2.6 TDA_GetTDAKernelReference()

```
TDAEXTERN void* TDACALL TDA_GetTDAKernelReference (
            const char * protocolOrUri )
```

Either creates a new TDA Kernel instance and returns its reference (if just the name of a communication protocol is specified), or obtains a reference to an existing TDA Kernel (if an URI is specified).

Parameters

protocolOrURI	a string denoting either a protocol name ("jni", "pipe", "shared_memory", "corba"), which specifies the communication mechanism for a TDA Kernel instance to be created, or an URI ("jni:UUID", "shared_memory:MEMORY_NAME", "corba:IOR") for obtaining a reference to an existing TDA Kernel.

Returns

a reference to a TDA Kernel, or null on error.

3.2.2.7 TDA_FreeTDAKernelReference()

```
TDAEXTERN bool TDACALL TDA_FreeTDAKernelReference (
            void * tdaKernel )
```

3.3 TDA Kernel DLL Functions For Accessing Java VM and Piped Processes

Functions

- TDAEXTERN void *TDACALL TDA_LaunchPipedProcess (IN const char *program, IN const char **args)
- TDAEXTERN void *TDACALL TDA_GetParentPipedProcess ()
- TDAEXTERN bool TDACALL TDA_ReadProcessOutputStream (void *hProcess, void *buffer, IN unsigned int size, OUT unsigned int &read)
- TDAEXTERN bool TDACALL TDA_WriteProcessInputStream (void *hProcess, void *buffer, IN unsigned int size, OUT unsigned int &written)
- TDAEXTERN bool TDACALL TDA_IsPipedProcessTerminated (void *hProcess)
- TDAEXTERN void TDACALL TDA_ReleasePipedProcess (void *hProcess, bool terminate)
- TDAEXTERN const char *TDACALL TDA_GetJavaHomeAnyBits ()
- TDAEXTERN const char *TDACALL TDA_GetJavaHomeSameBits ()
- TDAEXTERN char **TDACALL TDA_UpdateJVMOptions (IN const char **options)
- TDAEXTERN void TDACALL TDA_FreeUpdatedJVMOptions (IN char **options)
- TDAEXTERN void *TDACALL TDA_LaunchPipedJavaProcess (IN const char **jvmOptions, IN const char *mainClassName, IN const char **mainArgs)
- TDAEXTERN void **TDACALL TDA_GetExistingJavaVMs ()
- TDAEXTERN void TDACALL TDA_FreeArrayOfExistingJavaVMs (IN void **array)
- TDAEXTERN bool TDACALL TDA_CreateNewJavaVM (IN const char **jvmOptions, OUT void **jvm, OUT void **jvmLibHandle)
- TDAEXTERN void TDACALL TDA_DestroyJavaVM (IN void *jvm, IN void *jvmLibHandle)
- TDAEXTERN bool TDACALL TDA_LaunchJavaClass (IN void *jvm, IN const char *mainClassName, IN const char **mainArgs)
- TDAEXTERN const char *TDACALL TDA_LaunchJavaStringToStringClassMethod (IN void *jvm, IN const char *className, IN const char *methodName, IN const char *arg)
- TDAEXTERN void *TDACALL TDA_CreateSharedMemory (IN const char *memoryName, IN const unsigned int size)
- TDAEXTERN unsigned char *TDACALL TDA_GetSharedMemoryByteArray (void *sharedMemory)
- TDAEXTERN void TDACALL TDA_CloseSharedMemory (void *sharedMemory)
- TDAEXTERN void TDACALL TDA_Sleep (IN const unsigned int ms)

3.3.1 Detailed Description

TDA Kernel DLL Functions For Accessing Java VM and Piped Processes

3.3.2 Function Documentation

3.3.2.1 TDA_LaunchPipedProcess()

```
TDAEXTERN void* TDACALL TDA_LaunchPipedProcess (
            IN const char * program,
            IN const char ** args )
```

3.3.2.2 TDA_GetParentPipedProcess()

```
TDAEXTERN void* TDACALL TDA_GetParentPipedProcess ( )
```

3.3.2.3 TDA_ReadProcessOutputStream()

```
TDAEXTERN bool TDACALL TDA_ReadProcessOutputStream (
            void * hProcess,
            void * buffer,
            IN unsigned int size,
            OUT unsigned int & read )
```

3.3.2.4 TDA_WriteProcessInputStream()

```
TDAEXTERN bool TDACALL TDA_WriteProcessInputStream (
            void * hProcess,
            void * buffer,
            IN unsigned int size,
            OUT unsigned int & written )
```

3.3.2.5 TDA_IsPipedProcessTerminated()

```
TDAEXTERN bool TDACALL TDA_IsPipedProcessTerminated (
            void * hProcess )
```

3.3.2.6 TDA_ReleasePipedProcess()

```
TDAEXTERN void TDACALL TDA_ReleasePipedProcess (
            void * hProcess,
            bool terminate )
```

3.3.2.7 TDA_GetJavaHomeAnyBits()

```
TDAEXTERN const char* TDACALL TDA_GetJavaHomeAnyBits ( )
```

3.3.2.8 TDA_GetJavaHomeSameBits()

```
TDAEXTERN const char* TDACALL TDA_GetJavaHomeSameBits ( )
```

3.3.2.9 TDA_UpdateJVMOptions()

```
TDAEXTERN char** TDACALL TDA_UpdateJVMOptions (
            IN const char ** options )
```

3.3.2.10 TDA_FreeUpdatedJVMOptions()

```
TDAEXTERN void TDACALL TDA_FreeUpdatedJVMOptions (
            IN char ** options )
```

3.3.2.11 TDA_LaunchPipedJavaProcess()

```
TDAEXTERN void* TDACALL TDA_LaunchPipedJavaProcess (
            IN const char ** jvmOptions,
            IN const char * mainClassName,
            IN const char ** mainArgs )
```

3.3.2.12 TDA_GetExistingJavaVMs()

```
TDAEXTERN void** TDACALL TDA_GetExistingJavaVMs ( )
```

3.3.2.13 TDA_FreeArrayOfExistingJavaVMs()

```
TDAEXTERN void TDACALL TDA_FreeArrayOfExistingJavaVMs (
            IN void ** array )
```

3.3.2.14 TDA_CreateNewJavaVM()

```
TDAEXTERN bool TDACALL TDA_CreateNewJavaVM (
            IN const char ** jvmOptions,
            OUT void ** jvm,
            OUT void ** jvmLibHandle )
```

3.3.2.15 TDA_DestroyJavaVM()

```
TDAEXTERN void TDACALL TDA_DestroyJavaVM (
            IN void * jvm,
            IN void * jvmLibHandle )
```

3.3.2.16 TDA_LaunchJavaClass()

```
TDAEXTERN bool TDACALL TDA_LaunchJavaClass (
            IN void * jvm,
            IN const char * mainClassName,
            IN const char ** mainArgs )
```

3.3.2.17 TDA_LaunchJavaStringToStringClassMethod()

```
TDAEXTERN const char* TDACALL TDA_LaunchJavaStringToStringClassMethod (
            IN void * jvm,
            IN const char * className,
            IN const char * methodName,
            IN const char * arg )
```

3.3.2.18 TDA_CreateSharedMemory()

```
TDAEXTERN void* TDACALL TDA_CreateSharedMemory (
            IN const char * memoryName,
            IN const unsigned int size )
```

3.3.2.19 TDA_GetSharedMemoryByteArray()

```
TDAEXTERN unsigned char* TDACALL TDA_GetSharedMemoryByteArray (
            void * sharedMemory )
```

3.3.2.20 TDA_CloseSharedMemory()

```
TDAEXTERN void TDACALL TDA_CloseSharedMemory (
            void * sharedMemory )
```

3.3.2.21 TDA_Sleep()

```
TDAEXTERN void TDACALL TDA_Sleep (
            IN const unsigned int ms )
```

Chapter 4

File Documentation

4.1 tda_pipes_and_java.h File Reference

Functions

- TDAEXTERN void *TDACALL TDA_LaunchPipedProcess (IN const char *program, IN const char **args)
- TDAEXTERN void *TDACALL TDA_GetParentPipedProcess ()
- TDAEXTERN bool TDACALL TDA_ReadProcessOutputStream (void *hProcess, void *buffer, IN unsigned int size, OUT unsigned int &read)
- TDAEXTERN bool TDACALL TDA_WriteProcessInputStream (void *hProcess, void *buffer, IN unsigned int size, OUT unsigned int &written)
- TDAEXTERN bool TDACALL TDA_IsPipedProcessTerminated (void *hProcess)
- TDAEXTERN void TDACALL TDA_ReleasePipedProcess (void *hProcess, bool terminate)
- TDAEXTERN const char *TDACALL TDA_GetJavaHomeAnyBits ()
- TDAEXTERN const char *TDACALL TDA_GetJavaHomeSameBits ()
- TDAEXTERN char **TDACALL TDA_UpdateJVMOptions (IN const char **options)
- TDAEXTERN void TDACALL TDA_FreeUpdatedJVMOptions (IN char **options)
- TDAEXTERN void *TDACALL TDA_LaunchPipedJavaProcess (IN const char **jvmOptions, IN const char *mainClassName, IN const char **mainArgs)
- TDAEXTERN void **TDACALL TDA_GetExistingJavaVMs ()
- TDAEXTERN void TDACALL TDA_FreeArrayOfExistingJavaVMs (IN void **array)
- TDAEXTERN bool TDACALL TDA_CreateNewJavaVM (IN const char **jvmOptions, OUT void **jvm, OUT void **jvmLibHandle)
- TDAEXTERN void TDACALL TDA_DestroyJavaVM (IN void *jvm, IN void *jvmLibHandle)
- TDAEXTERN bool TDACALL TDA_LaunchJavaClass (IN void *jvm, IN const char *mainClassName, IN const char **mainArgs)
- TDAEXTERN const char *TDACALL TDA_LaunchJavaStringToStringClassMethod (IN void *jvm, IN const char *className, IN const char *methodName, IN const char *arg)
- TDAEXTERN void *TDACALL TDA_CreateSharedMemory (IN const char *memoryName, IN const unsigned int size)
- TDAEXTERN unsigned char *TDACALL TDA_GetSharedMemoryByteArray (void *sharedMemory)
- TDAEXTERN void TDACALL TDA_CloseSharedMemory (void *sharedMemory)
- TDAEXTERN void TDACALL TDA_Sleep (IN const unsigned int ms)

4.2 tdakernel.h File Reference

Functions

- const char * TDA_CreateReturnString (const char *s)
- unsigned int TDA_GetCurrentProcessID ()
- unsigned int TDA_GetParentProcessID ()
- TDAEXTERN const char *TDACALL TDA_GetTDAKernelLibraryPath ()
- TDAEXTERN const wchar_t *TDACALL TDA_GetTDAKernelLibraryPathW ()
- TDAEXTERN void *TDACALL TDA_GetTDAKernelReference (const char *protocolOrUri)
- TDAEXTERN bool TDACALL TDA_FreeTDAKernelReference (void *tdaKernel)

4.3 tdakernel_stub_c2base.h File Reference

Functions

- TDAEXTERN void TDACALL TDA_Close (void *tdaKernel)
- TDAEXTERN bool TDACALL TDA_Exists (void *tdaKernel, const char *location)
- TDAEXTERN bool TDACALL TDA_Open (void *tdaKernel, const char *location)
- TDAEXTERN bool TDACALL TDA_StartSave (void *tdaKernel)
- TDAEXTERN bool TDACALL TDA_CancelSave (void *tdaKernel)
- TDAEXTERN bool TDACALL TDA_FinishSave (void *tdaKernel)
- TDAEXTERN bool TDACALL TDA_Drop (void *tdaKernel, const char *location)
- TDAEXTERN __int64 TDACALL TDA_FindClass (void *tdaKernel, const char *name)
- TDAEXTERN const char *TDACALL TDA_GetClassName (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_CreateObject (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_CreateClass (void *tdaKernel, const char *name)
- TDAEXTERN bool TDACALL TDA_DeleteClass (void *tdaKernel, __int64 rClass)
- TDAEXTERN bool TDACALL TDA_IsClass (void *tdaKernel, __int64 r)
- TDAEXTERN bool TDACALL TDA_IsDirectSubClass (void *tdaKernel, __int64 rSubClass, __int64 rSuper↩ Class)
- TDAEXTERN bool TDACALL TDA_DeleteObject (void *tdaKernel, __int64 rObject)
- TDAEXTERN bool TDACALL TDA_MoveObject (void *tdaKernel, __int64 rObject, __int64 rToClass)
- TDAEXTERN bool TDACALL TDA_IsTypeOf (void *tdaKernel, __int64 rObject, __int64 rClass)
- TDAEXTERN bool TDACALL TDA_IsKindOf (void *tdaKernel, __int64 rObject, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_CreateAttribute (void *tdaKernel, __int64 rClass, const char *name, __int64 rPrimitiveType)
- TDAEXTERN bool TDACALL TDA_IsDerivedClass (void *tdaKernel, __int64 rDirectlyOrIndirectlyDerived↩ Class, __int64 rSuperClass)
- TDAEXTERN __int64 TDACALL TDA_FindAttribute (void *tdaKernel, __int64 rClass, const char *name)
- TDAEXTERN bool TDACALL TDA_DeleteAttribute (void *tdaKernel, __int64 rAttribute)
- TDAEXTERN const char *TDACALL TDA_GetAttributeName (void *tdaKernel, __int64 rAttribute)
- TDAEXTERN const char *TDACALL TDA_GetPrimitiveDataTypeName (void *tdaKernel, __int64 rDataType)
- TDAEXTERN __int64 TDACALL TDA_FindPrimitiveDataType (void *tdaKernel, const char *name)
- TDAEXTERN bool TDACALL TDA_IsPrimitiveDataType (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForClasses (void *tdaKernel)
- TDAEXTERN const char *TDACALL TDA_GetAttributeValue (void *tdaKernel, __int64 rObject, __int64 r↩ Attribute)
- TDAEXTERN __int64 TDACALL TDA_CreateAssociation (void *tdaKernel, __int64 rSourceClass, __int64 rTargetClass, const char *sourceRoleName, const char *targetRoleName, bool isComposition)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectSubClasses (void *tdaKernel, __int64 rSuper↩ Class)

- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllClassObjects (void *tdaKernel, __int64 rClassOr↩ AdvancedAssociation)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectAttributes (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetAttributeDomain (void *tdaKernel, __int64 rAttribute)
- TDAEXTERN bool TDACALL TDA_DeleteGeneralization (void *tdaKernel, __int64 rSubClass, __int64 r↩ SuperClass)
- TDAEXTERN bool TDACALL TDA_CreateGeneralization (void *tdaKernel, __int64 rSubClass, __int64 r↩ SuperClass)
- TDAEXTERN bool TDACALL TDA_ExcludeObjectFromClass (void *tdaKernel, __int64 rObject, __int64 r↩ Class)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllAttributes (void *tdaKernel, __int64 rClass)
- TDAEXTERN bool TDACALL TDA_IncludeObjectInClass (void *tdaKernel, __int64 rObject, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectSuperClasses (void *tdaKernel, __int64 rSub↩ Class)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectClassObjects (void *tdaKernel, __int64 rClass↩ OrAdvancedAssociation)
- TDAEXTERN bool TDACALL TDA_SetAttributeValue (void *tdaKernel, __int64 rObject, __int64 rAttribute, const char *value)
- TDAEXTERN bool TDACALL TDA_DeleteAttributeValue (void *tdaKernel, __int64 rObject, __int64 rAttribute)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectLinguisticInstances (void *tdaKernel, __int64 r↩ Class)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectObjectClasses (void *tdaKernel, __int64 r↩ ObjectOrAdvancedLink)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForObjectsByAttributeValue (void *tdaKernel, __int64 r↩ Attribute, const char *value)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllOutgoingAssociationEnds (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectIngoingAssociationEnds (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllLinguisticInstances (void *tdaKernel, __int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForDirectOutgoingAssociationEnds (void *tdaKernel, __↩ int64 rClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForAllIngoingAssociationEnds (void *tdaKernel, __int64 r↩ Class)
- TDAEXTERN __int64 TDACALL TDA_ResolveIterator (void *tdaKernel, __int64 it, int position)
- TDAEXTERN const char *TDACALL TDA_GetRoleName (void *tdaKernel, __int64 rAssociationEnd)
- TDAEXTERN bool TDACALL TDA_DeleteLink (void *tdaKernel, __int64 rSourceObject, __int64 rTarget↩ Object, __int64 rAssociationEnd)
- TDAEXTERN bool TDACALL TDA_CreateLink (void *tdaKernel, __int64 rSourceObject, __int64 rTarget↩ Object, __int64 rAssociationEnd)
- TDAEXTERN bool TDACALL TDA_IsLinguistic (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_GetAttributeType (void *tdaKernel, __int64 rAttribute)
- TDAEXTERN __int64 TDACALL TDA_GetTargetClass (void *tdaKernel, __int64 rTargetAssociationEnd)
- TDAEXTERN bool TDACALL TDA_IsAssociationEnd (void *tdaKernel, __int64 r)
- TDAEXTERN void TDACALL TDA_FreeReference (void *tdaKernel, __int64 r)
- TDAEXTERN bool TDACALL TDA_IsAttribute (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_GetSourceClass (void *tdaKernel, __int64 rTargetAssociationEnd)
- TDAEXTERN bool TDACALL TDA_LinkExists (void *tdaKernel, __int64 rSourceObject, __int64 rTarget↩ Object, __int64 rAssociationEnd)
- TDAEXTERN bool TDACALL TDA_IsComposition (void *tdaKernel, __int64 rTargetAssociationEnd)
- TDAEXTERN void TDACALL TDA_FreeIterator (void *tdaKernel, __int64 it)
- TDAEXTERN __int64 TDACALL TDA_GetLinguisticClassFor (void *tdaKernel, __int64 r)
- TDAEXTERN bool TDACALL TDA_DeleteAssociation (void *tdaKernel, __int64 rAssociationEndOr↩ AdvancedAssociation)
- TDAEXTERN bool TDACALL TDA_CreateOrderedLink (void *tdaKernel, __int64 rSourceObject, __int64 r↩ TargetObject, __int64 rAssociationEnd, int targetPosition)

- TDAEXTERN __int64 TDACALL TDA_DeserializeReference (void *tdaKernel, const char *r)
- TDAEXTERN __int64 TDACALL TDA_CreateDirectedAssociation (void *tdaKernel, __int64 rSourceClass, __int64 rTargetClass, const char *targetRoleName, bool isComposition)
- TDAEXTERN bool TDACALL TDA_IsAdvancedAssociation (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_ResolveIteratorFirst (void *tdaKernel, __int64 it)
- TDAEXTERN const char *TDACALL TDA_CallSpecificOperation (void *tdaKernel, const char *operation↩ Name, const char *arguments)
- TDAEXTERN __int64 TDACALL TDA_GetInverseAssociationEnd (void *tdaKernel, __int64 rAssociationEnd)
- TDAEXTERN int TDACALL TDA_GetLinkedObjectPosition (void *tdaKernel, __int64 rSourceObject, __int64 rTargetObject, __int64 rAssociationEnd)
- TDAEXTERN int TDACALL TDA_GetIteratorLength (void *tdaKernel, __int64 it)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForLinkedObjects (void *tdaKernel, __int64 rObject, __↩ int64 rAssociationEnd)
- TDAEXTERN __int64 TDACALL TDA_CreateAdvancedAssociation (void *tdaKernel, const char *name, bool nAry, bool associationClass)
- TDAEXTERN __int64 TDACALL TDA_GetIteratorForLinguisticClasses (void *tdaKernel)
- TDAEXTERN __int64 TDACALL TDA_FindAssociationEnd (void *tdaKernel, __int64 rSourceClass, const char *targetRoleName)
- TDAEXTERN const char *TDACALL TDA_SerializeReference (void *tdaKernel, __int64 r)
- TDAEXTERN __int64 TDACALL TDA_ResolveIteratorNext (void *tdaKernel, __int64 it)

Index