

# 1 Reference for unit 'Thorium'

## 1.1 Used units

Table 1.1: Used units by unit 'Thorium'

Name	Page
Classes	<a href="#">1</a>
contnrs	<a href="#">1</a>
md5	<a href="#">1</a>
SysUtils	<a href="#">1</a>
thorium_globals	<a href="#">1</a>
thorium_utils	<a href="#">1</a>
typinfo	<a href="#">1</a>
Variants	<a href="#">1</a>
zstream	<a href="#">1</a>

## 1.2 Constants, types and variables

### 1.2.1 Types

`PThoriumType = ^PThoriumType`

`PThoriumExternalFunctionVarType = ^TThoriumExternalFunctionVarType`

`PThoriumHostObjectTypeValue = ^TThoriumHostObjectTypeValue`

`PThoriumPersistent = ^TThoriumPersistent`

`PThoriumRelocation = ^TThoriumRelocation`

`PThoriumSimpleVarargs = ^TThoriumSimpleVarargs`

`PThoriumStackEntry = ^TThoriumStackEntry`

`PThoriumTableEntry = ^TThoriumTableEntry`

## 1.2. CONSTANTS, TYPES AND VARIABLES

---

```
PThoriumType = ^TThoriumType
```

```
PThoriumValue = ^TThoriumValue
```

```
TThoriumBuiltInValue = record  
end
```

An instance of this record represents a built-in type value in Thorium. This includes integers (Int64-based), floats (Double-based) and strings (UTF8String-based).

```
TThoriumClassMethod = procedure(const Input: Array of Pointer;  
                                Result: PThoriumValue) of object
```

This is the only kind of method which can be called without NativeCall.

```
TThoriumDebugCallbackObject = procedure  
                                (Sender: TThoriumDebuggingVirtualMachine;  
                                Kind: TThoriumDebugEvent;  
                                Obj: TObject) of object
```

```
TThoriumDebugCallbackValue = procedure  
                                (Sender: TThoriumDebuggingVirtualMachine;  
                                Kind: TThoriumDebugEvent;  
                                Value: LongInt) of object
```

```
TThoriumExternalFunctionVarType = record  
    HostType : TThoriumHostType;  
    Extended : TThoriumHostObjectType;  
    Storing : Boolean;  
end
```

This fully defines a type from the host environment, including a reference to an host object type object if it is a class type. It also tells whether it is Storing. For more information about the storing-flag see `TThoriumHostObjectType.GetPropertyStoring` (41).

```
TThoriumHostFunctionBaseClass = Class of TThoriumHostFunctionBase
```

```
TThoriumHostObjectTypeArray = Array of TThoriumHostObjectType
```

```
TThoriumHostObjectTypeClass = Class of TThoriumHostObjectType
```

```
TThoriumHostObjectTypeValue = record  
    Value : TThoriumHostObject;  
    TypeClass : TThoriumHostObjectType;  
    Size : TThoriumSizeInt;  
end
```

An instance of this record reflects a host object type value in Thorium. It contains the value, a pointer to the type and the size of the data allocated in the pointer if it should be automatically handled by Thorium.

```
TThoriumInstructionFunc1R = function(RI: Word) : TThoriumInstruction
```

```
TThoriumLibraryClass = Class of TThoriumLibrary
```

```
TThoriumLibraryPropertyArray = Array of TThoriumLibraryProperty
```

```
TThoriumLibraryPropertyClass = Class of TThoriumLibraryProperty
```

```
TThoriumOnCompilerOutput = procedure(Sender: TThorium;  
                                     const Module: TThoriumModule;  
                                     const Msg: String) of object
```

See `TThorium.OnCompilerOutput` (15).

```
TThoriumOnOpenModule = procedure(Sender: TThorium;  
                                const ModuleName: String;  
                                var Stream: TStream) of object
```

See `TThorium.OnOpenModule` (15)

```
TThoriumOnPropertyGet = procedure(Sender: TThoriumLibraryProperty;  
                                var AThoriumValue: TThoriumValue)  
                                of object
```

An event of this kind is needed when an event based library property is read.

```
TThoriumOnPropertySet = procedure(Sender: TThoriumLibraryProperty;  
                                const AThoriumValue: TThoriumValue)  
                                of object
```

An event of this kind is called when an event based library property is written.

```
TThoriumOnPropertySetCallback = procedure  
                                (Sender: TThoriumLibraryProperty;  
                                const AThoriumValue: TThoriumValue;  
                                var AllowSet: Boolean)  
                                of object
```

This kind of event gets called when a non-event based library with write-hook gets written.

```
TThoriumOnRequireModule = procedure(Sender: TThorium;const Name: String;  
                                var ANewModule: TThoriumModule)  
                                of object
```

See `TThorium.OnRequireModule` (15).

## 1.2. CONSTANTS, TYPES AND VARIABLES

---

```
TThoriumPersistentClass = Class of TThoriumPersistent
```

```
TThoriumQualifiedIdentifier = record
  FullStr : String;
  Kind : TThoriumQualifiedIdentifierKind;
  IsStatic : Boolean;
  FinalType : TThoriumType;
  Value : TThoriumValue;
  GetJumpMarks : TThoriumIntArray;
  GetCode : TThoriumInstructionArray;
  SetJumpMarks : TThoriumIntArray;
  SetCode : TThoriumInstructionArray;
  UsedExtendedTypes : Array of TThoriumHostObjectType;
  UsedLibraryProps : Array of TThoriumLibraryProperty;
end
```

This record represents a fully qualified identifier, which means that enough parsing has been done to find out what kind of identifier it is and how to access it (both read and write). It also keeps track about which types and library properties are accessed.

```
TThoriumRegisters = Array[0..THORIUM_REGISTER_COUNT-1] of TThoriumValue
```

An array which reflects the whole set of registers a Thorium virtual machine contains.

```
TThoriumRelocation = record
  ByteOffset : Cardinal;
  ObjectIndex : Cardinal;
end
```

Information about a relocation which has to be done when loading a module.

```
TThoriumRTTIMethods = Array of TThoriumHostMethodBase
```

An array of host methods for RTTI based host object types.

```
TThoriumRTTIMethodsCallback = procedure(Sender: TThoriumRTTIObjectType;
                                         var Methods: TThoriumRTTIMethods)
                                         of object
```

```
TThoriumRTTIStaticMethods = Array of TThoriumHostFunctionBase
```

An array of static methods for RTTI based host object types.

```
TThoriumRTTIStaticMethodsCallback = procedure
                                         (Sender: TThoriumRTTIObjectType;
                                         var Methods: TThoriumRTTIStaticMethods)
                                         of object
```

```
TThoriumSimpleMethod = procedure(const Input: Array of Pointer;
                                Result: PThoriumValue) of object
```

This is the only kind of function which can be called without NativeCall.

```
TThoriumSimpleVarargs = record
  Count : SizeUInt;
  Data : Pointer;
end
```

This record is used in simple (i.e. not NativeCall based) calls to host environment functions to represent an array passed to the function.

```
TThoriumStackEntry = record
end
```

Each stack entry is represented by an instance of this record. It can contain either a value, a stack frame or a varargs-container.

```
TThoriumStackEntryType = (etValue, etStackFrame, etVarargs, etNull)
```

Table 1.2: Enumeration values for type TThoriumStackEntryType

Value	Explanation
etNull	
etStackFrame	
etValue	
etVarargs	

```
TThoriumTableEntry = record
  Name : PString;
  Scope : Integer;
  _Type : TThoriumTableEntryType;
  Offset : Integer;
  TypeSpec : TThoriumType;
  Value : TThoriumValue;
  Ptr : Pointer;
end
```

Each entry in an identifier table of Thorium is represented by an instance of this record. It contains information about the name, which type of object it is and where and how to access it.

```
TThoriumType = record
end
```

This record reflects a type which can be processed by Thorium.

### 1.3. PROCEDURES AND FUNCTIONS

---

```
TThoriumValue = record  
end
```

A value which can be processed by Thorium. This can either be a host object based or a built-in value. Functions are not supported during runtime.

```
TThoriumValues = Array of TThoriumValue
```

## 1.3 Procedures and functions

### 1.3.1 ThoriumCreateExtendedTypeValue

Declaration: 

```
function ThoriumCreateExtendedTypeValue  
                                (const TypeClass: TThoriumHostObjectType  
                                : TThoriumValue
```

Visibility: default

### 1.3.2 ThoriumCreateFloatValue

Declaration: 

```
function ThoriumCreateFloatValue(const Value: TThoriumFloat)  
                                : TThoriumValue
```

Visibility: default

### 1.3.3 ThoriumCreateIntegerValue

Declaration: 

```
function ThoriumCreateIntegerValue(const Value: TThoriumInteger)  
                                : TThoriumValue
```

Visibility: default

### 1.3.4 ThoriumCreateStringValue

Declaration: 

```
function ThoriumCreateStringValue(const Value: TThoriumString)  
                                : TThoriumValue
```

Visibility: default

### 1.3.5 ThoriumCreateValue

Declaration: 

```
function ThoriumCreateValue(const ATypeSpec: TThoriumType)  
                                : TThoriumValue
```

Visibility: default

### 1.3.6 ThoriumExternalVarTypeToTypeSpec

Declaration: 

```
procedure ThoriumExternalVarTypeToTypeSpec  
                                (VarType: PThoriumExternalFunctionVar  
                                out TypeSpec: TThoriumType)
```

Visibility: default

### 1.3.7 ThoriumInstructionToStr

Declaration: `function ThoriumInstructionToStr(AInstruction: TThoriumInstruction)  
: String`

Visibility: default

### 1.3.8 ThoriumMakeOOPEvent

Declaration: `function ThoriumMakeOOPEvent(ACode: Pointer; Userdata: Pointer) : TMethod`

Visibility: default

### 1.3.9 ThoriumRegisterToStr

Declaration: `function ThoriumRegisterToStr(ARegisterID: TThoriumRegisterID) : String`

Visibility: default

### 1.3.10 ThoriumValueToStr

Declaration: `function ThoriumValueToStr(const Value: TThoriumValue) : String`

Visibility: default

### 1.3.11 ThoriumVarTypeToTypeSpec

Declaration: `procedure ThoriumVarTypeToTypeSpec(VarType: TThoriumHostType;  
var TypeSpec: TThoriumType)`

Visibility: default

## 1.4 EThoriumCompilerException

### 1.4.1 Description

This exception is thrown whenever the compiler enters a state which was not expected to happen by the author. You should inform the author of Thorium about any exception of this kind you catch and provide a sample script to produce it.

## 1.5 EThoriumDebuggerException

## 1.6 EThoriumDependencyException

### 1.6.1 Description

This exception gets thrown for example by the `LoadModuleFromStream` ([13](#)) method when a required module or library cannot be found.

## 1.7 EThoriumException

### 1.7.1 Description

An exception of this class is only thrown when no other of the specialized exceptions matches the situation. Useful to catch any exception thrown by Thorium.

## 1.8 EThoriumHashException

### 1.8.1 Description

This kind of exception is thrown when a hash check for a module, library, function, property or class type fails.

## 1.9 EThoriumRuntimeException

### 1.9.1 Description

The virtual machine and other runtime parts of Thorium throw this kind of exception when anything is wrong. This includes mismatched parameter types or counts.

## 1.10 EThoriumRuntimeExecutionException

### 1.10.1 Description

The virtual machine catches any exception thrown by any instruction and encapsulates it in a new exception of this class. It contains additional information like the module in which the exception occurred, the line, the instruction address and which exact instruction caused the exception. A reference to the original exception is supplied too.

### 1.10.2 Method overview

Page	Property	Description
<a href="#">8</a>	Create	
<a href="#">9</a>	Destroy	

### 1.10.3 Property overview

Page	Property	Access	Description
<a href="#">9</a>	OriginalException	r	Access the original exception.

### 1.10.4 EThoriumRuntimeExecutionException.Create

**Declaration:** constructor `Create(Module: TThoriumModule;  
InstructionAddr: TThoriumInstructionAddress;  
Instruction: PThoriumInstruction;  
OriginalException: Exception)`

**Visibility:** public



### 1.10.5 EThoriumRuntimeExecutionException.Destroy

Declaration: `destructor Destroy; Override`

Visibility: `public`

### 1.10.6 EThoriumRuntimeExecutionException.OriginalException

Synopsis: Access the original exception.

Declaration: `Property OriginalException : Exception`

Visibility: `public`

Access: `Read`

Description: This property provides access to the original exception thrown by the instruction.

## 1.11 EThoriumVerificationException

### 1.11.1 Description

During the various `LoadFromStream` methods a lot of effort is done to ensure that any reference to anything is resolved correctly and does not result in some weird errors. If anything cannot be resolved for sure, an exception of this class or a descendant is thrown.

## 1.12 IThoriumPersistent

### 1.12.1 Description

Any class which is to be published to Thorium using `TThoriumRTTIObjectType` (75) must implement this interface. It is used to notify the class about copies on the stack and in the registers of Thorium to avoid it from being freed.

### 1.12.2 Method overview

Page	Property	Description
<a href="#">10</a>	<code>DisableHostControl</code>	Enable freeing by reference count.
<a href="#">9</a>	<code>EnableHostControl</code>	Disable free by reference counting.
<a href="#">10</a>	<code>FreeReference</code>	Release a reference.
<a href="#">10</a>	<code>GetReference</code>	Increase reference counter and return reference.

### 1.12.3 IThoriumPersistent.EnableHostControl

Synopsis: Disable free by reference counting.

Declaration: `procedure EnableHostControl`

Visibility: `default`

Description: An implementation of this method is expected to set a flag which disables freeing the object when it runs out of references.

#### 1.12.4 IThoriumPersistent.DisableHostControl

Synopsis: Enable freeing by reference count.

Declaration: `procedure DisableHostControl`

Visibility: default

Description: An implementation of this method is expected to set a flag which enables freeing the object when it runs out of references. It should also immediately free the object if the reference counter is already at zero.

#### 1.12.5 IThoriumPersistent.FreeReference

Synopsis: Release a reference.

Declaration: `procedure FreeReference`

Visibility: default

Description: An implementation of this method is expected to decrease the reference counter of the instance by one and, if applicable, free the instance.

#### 1.12.6 IThoriumPersistent.GetReference

Synopsis: Increase reference counter and return reference.

Declaration: `function GetReference : TObject`

Visibility: default

Description: An implementation of this method is expected to increase the reference counter of the object by one and return the object itself too.

### 1.13 TThorium

#### 1.13.1 Description

This class manages all the modules, libraries and the virtual machine and thus is the class you probably want to use first. It represents a whole Thorium context. There may even exist several instances of this class representing different Thorium contexts. Please note that this class is not threadsafe and each instance should only be used by exactly one thread or the usage must be carefully synchronized since accessing modules or even the virtual machine at the same time from two different threads, maybe while an execution is running in a third thread will cause at least interesting errors.

### 1.13.2 Method overview

Page	Property	Description
<a href="#">12</a>	ClearLibraries	Unload all loaded libraries.
<a href="#">12</a>	ClearModules	Delete all loaded modules.
<a href="#">11</a>	Create	
<a href="#">11</a>	Destroy	
<a href="#">11</a>	DoCompilerOutput	
<a href="#">12</a>	DoOpenModule	
<a href="#">11</a>	DoRequireModule	
<a href="#">12</a>	FindLibrary	Return a library by name.
<a href="#">12</a>	FindModule	Look up a module by name.
<a href="#">12</a>	InitializeVirtualMachine	Attach and initialize a virtual machine.
<a href="#">13</a>	LoadLibrary	Load a host library.
<a href="#">13</a>	LoadModuleFromFile	Load a module from file.
<a href="#">13</a>	LoadModuleFromStream	Load a module from stream.
<a href="#">13</a>	NewModule	Create a new empty module.
<a href="#">13</a>	ReleaseVirtualMachine	Free the virtual machine.

### 1.13.3 Property overview

Page	Property	Access	Description
<a href="#">14</a>	HostLibrary	r	Access to loaded libraries
<a href="#">14</a>	HostLibraryCount	r	Amount of loaded libraries
<a href="#">14</a>	Locked	r	Whether the context is locked.
<a href="#">14</a>	Module	r	Access to modules in the context.
<a href="#">14</a>	ModuleCount	r	Count of loaded modules.
<a href="#">15</a>	OnCompilerOutput	rw	Event for compiler output.
<a href="#">15</a>	OnOpenModule	rw	Event when a file needs to be opened.
<a href="#">15</a>	OnRequireModule	rw	Event before a module is loaded from file.
<a href="#">15</a>	VirtualMachine	r	Access to the attached virtual machine.

### 1.13.4 TThorium.Create

Declaration: constructor Create; Virtual

Visibility: default

### 1.13.5 TThorium.Destroy

Declaration: destructor Destroy; Override

Visibility: default

### 1.13.6 TThorium.DoCompilerOutput

Declaration: procedure DoCompilerOutput(const Module: TThoriumModule;  
const Msg: String); Virtual

Visibility: protected

### 1.13.7 TThorium.DoRequireModule

Declaration: function DoRequireModule(const Name: String; NeededHash: PThoriumHash)

: TThoriumModule; Virtual

Visibility: protected

### 1.13.8 TThorium.DoOpenModule

Declaration: `function DoOpenModule(const ModuleName: String) : TStream; Virtual`

Visibility: protected

### 1.13.9 TThorium.ClearLibraries

Synopsis: Unload all loaded libraries.

Declaration: `procedure ClearLibraries`

Visibility: public

Description: Unloads all libraries loaded in the current context. Since modules may depend on these libraries they are cleared too.

### 1.13.10 TThorium.ClearModules

Synopsis: Delete all loaded modules.

Declaration: `procedure ClearModules`

Visibility: public

Description: Deletes all modules which are currently loaded in the context. Libraries stay unchanged though.

### 1.13.11 TThorium.FindLibrary

Synopsis: Return a library by name.

Declaration: `function FindLibrary(const Name: String) : TThoriumLibrary`

Visibility: public

Description: Looks up the instance of a library whose name is equal to the one passed in *Name* and returns it if any is found. Otherwise returns nil.

### 1.13.12 TThorium.FindModule

Synopsis: Look up a module by name.

Declaration: `function FindModule(const Name: String; AllowLoad: Boolean)  
: TThoriumModule`

Visibility: public

Description: This method searches for a module in the context which is called like *Name*. If no module is found and *AllowLoad* is true, an attempt to load the module using the `LoadModuleFromFile` (13) method is started and the loaded module is returned if it is successful. Otherwise nil is returned.

### 1.13.13 TThorium.InitializeVirtualMachine

Synopsis: Attach and initialize a virtual machine.

Declaration: `procedure InitializeVirtualMachine`

Visibility: public

Description: Makes sure a virtual machine is initialized and attached to the context. This also brings the context in a locked state which disallows loading of modules and libraries to keep the virtual machine in a consistent state.

### 1.13.14 TThorium.LoadLibrary

Synopsis: Load a host library.

Declaration: `function LoadLibrary(const ALibrary: TThoriumLibraryClass)  
: TThoriumLibrary`

Visibility: public

Description: Creates an instance of the library class *ALibrary*, loads it into the context and returns it.

### 1.13.15 TThorium.LoadModuleFromFile

Synopsis: Load a module from file.

Declaration: `function LoadModuleFromFile(AModuleName: String;  
NeededHash: PThoriumHash) : TThoriumModule`

Visibility: public

Description: This method tries to load a module from file using any callbacks assigned to the context. If *NeededHash* is not nil, the hash of the module is compared against it and if they do not match, an *ETHoriumVerificationException* (9) exception is thrown.

### 1.13.16 TThorium.LoadModuleFromStream

Synopsis: Load a module from stream.

Declaration: `function LoadModuleFromStream(AStream: TStream; AName: String;  
NeededHash: PThoriumHash) : TThoriumModule`

Visibility: public

Description: This method will attempt to load a module from the stream given. If no name is passed via *AName*, the module will be assigned a generated anonymous name. If a hash is supplied via *NeededHash*, the hash of the loaded module is compared against it and, in case of a mismatch, an *ETHoriumVerificationException* (9) is thrown.

### 1.13.17 TThorium.NewModule

Synopsis: Create a new empty module.

Declaration: `function NewModule(AName: String) : TThoriumModule`

Visibility: public

Description: This method creates a new empty module, registers it with the context and returns it. If *AName* is empty, a anonymous name is generated and assigned to the module.

### 1.13.18 TThorium.ReleaseVirtualMachine

Synopsis: Free the virtual machine.

Declaration: `procedure ReleaseVirtualMachine`

Visibility: public

Description: If a virtual machine is attached to the Thorium context, it will get freed by this call. This also reverts the locked state of the context.

#### 1.13.19 TThorium.HostLibrary

Synopsis: Access to loaded libraries

Declaration: `Property HostLibrary[Index: Integer]: TThoriumLibrary`

Visibility: public

Access: Read

Description: This property provides access to the host libraries loaded in the current context.

#### 1.13.20 TThorium.HostLibraryCount

Synopsis: Amount of loaded libraries

Declaration: `Property HostLibraryCount : Integer`

Visibility: public

Access: Read

Description: Property which reflects the amount of libraries loaded into the context.

#### 1.13.21 TThorium.Locked

Synopsis: Whether the context is locked.

Declaration: `Property Locked : Boolean`

Visibility: public

Access: Read

Description: Reflects whether the context is locked (i.e. a virtual machine is attached).

#### 1.13.22 TThorium.Module

Synopsis: Access to modules in the context.

Declaration: `Property Module[Index: Integer]: TThoriumModule`

Visibility: public

Access: Read

Description: This property provides access to the modules loaded into the current context.

#### 1.13.23 TThorium.ModuleCount

Synopsis: Count of loaded modules.

Declaration: `Property ModuleCount : Integer`

Visibility: public

Access: Read

Description: This property reflects the amount of loaded modules.

### 1.13.24 TThorium.OnCompilerOutput

Synopsis: Event for compiler output.

Declaration: `Property OnCompilerOutput : TThoriumOnCompilerOutput`

Visibility: public

Access: Read,Write

Description: This event is called whenever a module which gets compiled in the context produces compiler output. Useful for logging and keeping track of compilations.

### 1.13.25 TThorium.OnOpenModule

Synopsis: Event when a file needs to be opened.

Declaration: `Property OnOpenModule : TThoriumOnOpenModule`

Visibility: public

Access: Read,Write

Description: This event gets called when the context needs to open a file. If this event is not assigned, the default `TFileStream` mechanism will be used. Otherwise it is possible to redirect file request into other directories or a virtual file system.

### 1.13.26 TThorium.OnRequireModule

Synopsis: Event before a module is loaded from file.

Declaration: `Property OnRequireModule : TThoriumOnRequireModule`

Visibility: public

Access: Read,Write

Description: This event is called before a module is loaded from a file. You may hook it and replace it with a module of your choice (which may be already loaded, but which must not be in any context).

### 1.13.27 TThorium.VirtualMachine

Synopsis: Access to the attached virtual machine.

Declaration: `Property VirtualMachine : TThoriumVirtualMachine`

Visibility: public

Access: Read

Description: If a virtual machine is attached, this property provides access. Otherwise it is nil.

## 1.14 TThoriumDebuggingVirtualMachine

### 1.14.1 Description

To be implemented and thus to be described later.

### 1.14.2 Method overview

Page	Property	Description
<a href="#">16</a>	Create	
<a href="#">16</a>	Destroy	
<a href="#">16</a>	Execute	
<a href="#">16</a>	StepInto	
<a href="#">16</a>	StepOver	

---

### 1.14.3 Property overview

Page	Property	Access	Description
<a href="#">16</a>	BreakpointInstructions	r	
<a href="#">17</a>	BreakpointLines	r	
<a href="#">17</a>	Registers	r	
<a href="#">17</a>	Stack	r	
<a href="#">17</a>	StepMode	rw	

---

### 1.14.4 TThoriumDebuggingVirtualMachine.Create

Declaration: `constructor Create(AThorium: TThorium)`

Visibility: `default`

### 1.14.5 TThoriumDebuggingVirtualMachine.Destroy

Declaration: `destructor Destroy; Override`

Visibility: `default`

### 1.14.6 TThoriumDebuggingVirtualMachine.Execute

Declaration: `procedure Execute(StartModuleIndex: Integer; StartInstruction: Integer; CreateDefaultStackFrame: Boolean); Override`

Visibility: `public`

### 1.14.7 TThoriumDebuggingVirtualMachine.StepInto

Declaration: `procedure StepInto`

Visibility: `public`

### 1.14.8 TThoriumDebuggingVirtualMachine.StepOver

Declaration: `procedure StepOver`

Visibility: `public`

### 1.14.9 TThoriumDebuggingVirtualMachine.BreakpointInstructions

Declaration: `Property BreakpointInstructions : TThoriumIntList`

Visibility: `public`

Access: `Read`



### 1.14.10 TThoriumDebuggingVirtualMachine.BreakpointLines

Declaration: Property BreakpointLines : TThoriumIntList

Visibility: public

Access: Read

### 1.14.11 TThoriumDebuggingVirtualMachine.Registers

Declaration: Property Registers[ARegID: TThoriumRegisterID]: PThoriumValue

Visibility: public

Access: Read

### 1.14.12 TThoriumDebuggingVirtualMachine.Stack

Declaration: Property Stack : TThoriumStack

Visibility: public

Access: Read

### 1.14.13 TThoriumDebuggingVirtualMachine.StepMode

Declaration: Property StepMode : TThoriumDebuggerStepMode

Visibility: public

Access: Read,Write

## 1.15 TThoriumFunction

### 1.15.1 Description

This class represents a function declared in a Thorium script, probably published by a module. It is also used as a temporary object by the compiler to store information about the current function.

### 1.15.2 Method overview

Page	Property	Description
<a href="#">19</a>	AsEvent	Not implemented yet.
<a href="#">18</a>	Call	Call the function
<a href="#">18</a>	Create	Create an instance.
<a href="#">18</a>	Destroy	
<a href="#">18</a>	Duplicate	Duplicate this instance.
<a href="#">19</a>	LoadFromStream	Load specification from stream.
<a href="#">19</a>	SafeCall	Call the function with additional checks
<a href="#">19</a>	SaveToStream	Saves the specification to stream.

### 1.15.3 Property overview

Page	Property	Access	Description
<a href="#">19</a>	EntryPoint	r	Entry point address.
<a href="#">20</a>	NestingLevel	r	Deprecated.
<a href="#">20</a>	Parameters	r	Parameter specification.
<a href="#">20</a>	Prototyped	r	Whether the function is still prototyped.
<a href="#">20</a>	ReturnValues	r	Return value specification.
<a href="#">21</a>	VisibilityLevel	r	The level of visibility.

### 1.15.4 TThoriumFunction.Create

Synopsis: Create an instance.

Declaration: `constructor Create(AModule: TThoriumModule);` Override

Visibility: default

Description: This function creates a new function specification. It expects the module which owns this declaration as the first and only parameter. Normally there is no need for you to create an instance of this class, since the compiler does it for you.

See also: `TThoriumFunction.Duplicate` ([18](#))

### 1.15.5 TThoriumFunction.Destroy

Declaration: `destructor Destroy;` Override

Visibility: default

### 1.15.6 TThoriumFunction.Call

Synopsis: Call the function

Declaration: `function Call(AParameters: Array of TThoriumValue) : TThoriumValue`

Visibility: public

Description: Calls the function using the virtual machine assigned to the Thorium engine which owns the module owning this function. The contents of the array of `TThoriumValue` ([6](#)) being the first parameter are passed as parameters to the function when calling it. There are no checks made whether the type or amount of parameters is correct for this function. If you need this, use `SafeCall` ([19](#)) instead. This method returns the value which has been returned by the function. If the function does not supply any return value, the result is unspecified.

Errors: Throws an exception if no module is assigned, the assigned module does not have a Thorium engine assigned or the virtual machine has not been initialized.

See also: `TThoriumFunction.SafeCall` ([19](#))

### 1.15.7 TThoriumFunction.Duplicate

Synopsis: Duplicate this instance.

Declaration: `function Duplicate : TThoriumFunction`

Visibility: public

Description: Creates a new instance of `TThoriumFunction` and fills it with the same data this instance has and returns it. This is mostly used by the compiler when publishing functions.

### 1.15.8 TThoriumFunction.AsEvent

Synopsis: Not implemented yet.

Declaration: `function AsEvent (AParameters: Array of TThoriumHostType;  
                                  ReturnType: TThoriumHostType)  
                                  : TThoriumFunctionCallbackCapsule; Overload  
function AsEvent (AParameters: Array of TThoriumHostType;  
                                  ReturnType: TThoriumHostType;  
                                  ExtParameters: Array of TThoriumHostObjectType;  
                                  ExtReturnType: TThoriumHostObjectType)  
                                  : TThoriumFunctionCallbackCapsule; Overload`

Visibility: public

### 1.15.9 TThoriumFunction.LoadFromStream

Synopsis: Load specification from stream.

Declaration: `procedure LoadFromStream (Stream: TStream); Override`

Visibility: public

Description: Loads the specification of a function from the given stream and assigns it to this instance. There are not many checks made for valid values, so you should make sure the data is not corrupted. Hashes and identifier names are used to check for the validity of identifier references.

See also: [TThoriumFunction.SaveToStream \(19\)](#)

### 1.15.10 TThoriumFunction.SafeCall

Synopsis: Call the function with additional checks

Declaration: `function SafeCall (AParameters: Array of TThoriumValue) : TThoriumValue`

Visibility: public

Description: Other than the [Call \(18\)](#) method this checks whether the types of the passed parameters and the parameter count matches those specified in this instance. If this is not the case, an exception is thrown. After that the default [Call \(18\)](#) method is called.

Errors: Throws an exception when the types or count of parameters do not match.

See also: [TThoriumFunction.Call \(18\)](#)

### 1.15.11 TThoriumFunction.SaveToStream

Synopsis: Saves the specification to stream.

Declaration: `procedure SaveToStream (Stream: TStream); Override`

Visibility: public

Description: Saves the specification of this instance to a stream. References to identifiers are encoded as their name and a hash to make sure they can be validated on loading.

See also: [TThoriumFunction.LoadFromStream \(19\)](#)

### 1.15.12 TThoriumFunction.EntryPoint

Synopsis: Entry point address.

Declaration: `Property EntryPoint : Integer`

Visibility: public

Access: Read

Description: Address in the script byte code where the function begins. This information is crucial to call the method in the virtual machine.

#### 1.15.13 TThoriumFunction.NestingLevel

Synopsis: Deprecated.

Declaration: `Property NestingLevel : Integer`

Visibility: public

Access: Read

#### 1.15.14 TThoriumFunction.Parameters

Synopsis: Parameter specification.

Declaration: `Property Parameters : TThoriumParameters`

Visibility: public

Access: Read

Description: Pointer to an instance of TThoriumParameters (70) representing the parameter list of the function. The names of the parameters are not saved.

See also: TThoriumFunction.ReturnValue (17)

#### 1.15.15 TThoriumFunction.Prototyped

Synopsis: Whether the function is still prototyped.

Declaration: `Property Prototyped : Boolean`

Visibility: public

Access: Read

Description: This is true when the function has not been implemented, only prototyped. You must not call a function which is only prototyped and normally the compiler should post errors about any function being only prototyped after the compilation has been finished.

#### 1.15.16 TThoriumFunction.ReturnValues

Synopsis: Return value specification.

Declaration: `Property ReturnValues : TThoriumParameters`

Visibility: public

Access: Read

Description: Originally planned as list, now only the first element of the TThoriumParameters (70) structure is used. This represents the type(s) of the return value the function gives. If this is empty, the function does not have any return value.

See also: TThoriumFunction.Parameters (20)

### 1.15.17 TThoriumFunction.VisibilityLevel

Synopsis: The level of visibility.

Declaration: `Property VisibilityLevel : TThoriumVisibilityLevel`

Visibility: public

Access: Read

Description: This property represents the visibility of the function. Normally you will only find functions which have this set to vsPublic, since private functions are not shown.

## 1.16 TThoriumFunctionCallbackCapsule

### 1.16.1 Description

For future use.

### 1.16.2 Method overview

Page	Property	Description
<a href="#">21</a>	Create	

### 1.16.3 TThoriumFunctionCallbackCapsule.Create

Declaration: `constructor Create(AFunction: TThoriumFunction;  
                                  Parameters: Array of TThoriumHostType;  
                                  ReturnType: TThoriumHostType;  
                                  ExtParameters: Array of TThoriumHostObjectType;  
                                  ExtReturnType: TThoriumHostObjectType)`

Visibility: public

## 1.17 TThoriumHashableObject

### 1.17.1 Description

This class is used in Thorium as a base class to implement hashing which is used to compare different runtimes while loading modules.

### 1.17.2 Method overview

Page	Property	Description
<a href="#">22</a>	CalcHash	Calculate the hash.
<a href="#">21</a>	Create	Initialize the class.
<a href="#">22</a>	GetHash	Returns the hash of the object.
<a href="#">22</a>	InvalidateHash	Invalidate any generated hash.

### 1.17.3 TThoriumHashableObject.Create

Synopsis: Initialize the class.

Declaration: `constructor Create`

Visibility: public

Description: Creates the instance and prepares hashing.

### 1.17.4 TThoriumHashableObject.CalcHash

Synopsis: Calculate the hash.

Declaration: `procedure CalcHash; Virtual; Abstract`

Visibility: protected

Description: This method must be overridden by descendant classes. It should generate a 16 byte hash which is "unique" to the contents of the class and save it to FHash.

See also: TThoriumHashableObject.GetHash ([22](#)), TThoriumHashableObject.InvalidateHash ([22](#))

### 1.17.5 TThoriumHashableObject.InvalidateHash

Synopsis: Invalidate any generated hash.

Declaration: `procedure InvalidateHash`

Visibility: protected

Description: If an hash has been generated, it is invaildated. This wants to say, if an hash is requested after invalidation, it will be regenerated and stored after that.

See also: TThoriumHashableObject.GetHash ([22](#)), TThoriumHashableObject.CalcHash ([22](#))

### 1.17.6 TThoriumHashableObject.GetHash

Synopsis: Returns the hash of the object.

Declaration: `function GetHash : TThoriumHash`

Visibility: public

Description: This function returns the hash of the object. If an hash has been generated before, it will be reused.

See also: TThoriumHashableObject.CalcHash ([22](#)), TThoriumHashableObject.InvalidateHash ([22](#))

## 1.18 TThoriumHostCallableBase

### 1.18.1 Description

This class is an abstract class whose descendants are used to represent a callable entity placed in the host environment (i.e. functions and methods). This class implements the hashing functionality and declares some shared properties.

### 1.18.2 Method overview

Page	Property	Description
<a href="#">23</a>	CalcHash	Calculate callable hash.
<a href="#">23</a>	Create	
<a href="#">23</a>	Destroy	

### 1.18.3 Property overview

Page	Property	Access	Description
<a href="#">24</a>	Name	rw	Identifier of the callable.
<a href="#">23</a>	Parameters	r	Parameter specification.
<a href="#">23</a>	ReturnType	rw	Return type of the callable.

### 1.18.4 TThoriumHostCallableBase.Create

Declaration: `constructor Create; Virtual`

Visibility: `public`

### 1.18.5 TThoriumHostCallableBase.Destroy

Declaration: `destructor Destroy; Override`

Visibility: `public`

### 1.18.6 TThoriumHostCallableBase.CalcHash

Synopsis: Calculate callable hash.

Declaration: `procedure CalcHash; Override`

Visibility: `protected`

Description: Calculates the hash of this callable instance. It includes the function signature like parameters, name and return value type.

See also: `TThoriumHashableObject` ([21](#))

### 1.18.7 TThoriumHostCallableBase.Parameters

Synopsis: Parameter specification.

Declaration: `Property Parameters : TThoriumHostFunctionParameterSpec`

Visibility: `public`

Access: `Read`

Description: Pointing to a `TThoriumHostFunctionParameterSpec` ([25](#)) instance representing the parameters of the callable.

### 1.18.8 TThoriumHostCallableBase.ReturnType

Synopsis: Return type of the callable.

Declaration: `Property ReturnType : TThoriumExternalFunctionVarType`

Visibility: `public`

Access: `Read,Write`

Description: Specifies the return type of the callable.

### 1.18.9 TThoriumHostCallableBase.Name

**Synopsis:** Identifier of the callable.

```
Declaration: Property Name : String
```

Visibility: public

Access: Read, Write

**Description:** Name identifier of the callable.

## 1.19 TThoriumHostFunctionBase

### 1.19.1 Method overview

Page	Property	Description
24	CallFromVirtualMachine	

### 1.19.2 TThoriumHostFunctionBase.CallFromVirtualMachine

```
Declaration: procedure CallFromVirtualMachine
                                     (AVirtualMachine: TThoriumVirtualMachine)
                                     ; Virtual; Abstract
```

Visibility: protected

## 1.20 TThoriumHostFunctionNativeCall

### 1.20.1 Description

Implements a host call as native call, meaning that you do not need a wrapper to get the parameters in the format you want. They will be converted and passed in computer native formats to your function, without having you to change anything.

### 1.20.2 Method overview

Page	Property	Description
<a href="#">25</a>	CallFromVirtualMachine	
<a href="#">24</a>	Create	
<a href="#">25</a>	Destroy	
<a href="#">25</a>	Precompile	Precompile the native call subscript.

### 1.20.3 Property overview

Page	Property	Access	Description
<a href="#">25</a>	CallingConvention	rw	Calling convention to be used.
<a href="#">25</a>	CodePointer	rw	Pointer to the function.

### 1.20.4 TThoriumHostFunctionNativeCall.Create

**Declaration:** constructor Create; Override

Visibility: public



### 1.20.5 TThoriumHostFunctionNativeCall.Destroy

Declaration: destructor Destroy; Override

Visibility: public

### 1.20.6 TThoriumHostFunctionNativeCall.CallFromVirtualMachine

Declaration: procedure CallFromVirtualMachine  
(AVirtualMachine: TThoriumVirtualMachine)  
; Override

Visibility: protected

### 1.20.7 TThoriumHostFunctionNativeCall.Precompile

Synopsis: Precompile the native call subscript.

Declaration: procedure Precompile; Virtual

Visibility: public

Description: This method performs precompilation of the NativeCall subscript which is needed to perform the call. You must call this before the first attempt of a native call but after you have configured the parameters.

### 1.20.8 TThoriumHostFunctionNativeCall.CallingConvention

Synopsis: Calling convention to be used.

Declaration: Property CallingConvention : TThoriumNativeCallingConvention

Visibility: public

Access: Read,Write

Description: This must describe the calling convention the function has you want to be called. The default calling convention of FreePascal is *register*, so you probably want to use ncRegister.

See also: TThoriumHostFunctionSimpleMethod ([30](#)), TThoriumHostFunctionNativeCall ([24](#))

### 1.20.9 TThoriumHostFunctionNativeCall.CodePointer

Synopsis: Pointer to the function.

Declaration: Property CodePointer : Pointer

Visibility: public

Access: Read,Write

Description: This must point to the function you want to call.

## 1.21 TThoriumHostFunctionParameterSpec

### 1.21.1 Description

This class is able to store and represent the parameter list of a function of the host environment. Types are represented by either a TThoriumHostType ([1](#)) or an TThoriumHostObjectType ([35](#)), if it is an object/class type.

### 1.21.2 Method overview

Page	Property	Description
<a href="#">28</a>	AddExtendedType	Adds an host object type to the list.
<a href="#">28</a>	AddType	Adds a new basic entry.
<a href="#">28</a>	AllTypes	Access to all elements through a pointer.
<a href="#">29</a>	Clear	Clears the whole list.
<a href="#">26</a>	Create	
<a href="#">29</a>	DeleteType	Delete a type from the list.
<a href="#">26</a>	Destroy	
<a href="#">26</a>	Expand	Enlarge the buffer.
<a href="#">27</a>	GetCompleteType	Get a complete type representation.
<a href="#">27</a>	GetExtendedType	Get host object type
<a href="#">27</a>	GetParamType	Get host type
<a href="#">28</a>	IndexOfType	Find an occurrence of the given type.
<a href="#">29</a>	InsertExtendedType	Insert an host object type.
<a href="#">29</a>	InsertType	Insert a type.
<a href="#">27</a>	SetCapacity	Set the capacity of the list.
<a href="#">27</a>	SetExtendedType	Set the host object type of an entry.
<a href="#">28</a>	SetParamType	Set the host type of an entry

### 1.21.3 Property overview

Page	Property	Access	Description
<a href="#">30</a>	Capacity	rw	Access the capacity of the list.
<a href="#">30</a>	CompleteTypes	r	Access to complete specifications.
<a href="#">30</a>	Count	r	Access the amount of items.
<a href="#">30</a>	ExtendedTypes	rw	Access to host object type.
<a href="#">29</a>	Types	rw	Access to the host types.

### 1.21.4 TThoriumHostFunctionParameterSpec.Create

Declaration: `constructor Create`

Visibility: `default`

### 1.21.5 TThoriumHostFunctionParameterSpec.Destroy

Declaration: `destructor Destroy; Override`

Visibility: `default`

### 1.21.6 TThoriumHostFunctionParameterSpec.Expand

Synopsis: Enlarge the buffer.

Declaration: `procedure Expand`

Visibility: `protected`

Description: The list is optimized for best performance. So each time the list grows, multiple elements are allocated. This function automatically grows the list, depending on the already present count of elements.

See also: `TThoriumHostFunctionParameterSpec.SetCapacity` ([27](#))

### 1.21.7 TThoriumHostFunctionParameterSpec.GetCompleteType

Synopsis: Get a complete type representation.

Declaration: `function GetCompleteType(AIndex: Integer)  
: PThoriumExternalFunctionVarType`

Visibility: protected

Description: Returns a pointer to the complete type representation containing both host type (1) and object type (35) at the location in the list specified by AIndex. This can also used to modify the specification.

See also: TThoriumHostFunctionParameterSpec.GetExtendedType (27), TThoriumHostFunctionParameterSpec.GetParamType (27), TThoriumHostFunctionParameterSpec.CompleteTypes (30)

### 1.21.8 TThoriumHostFunctionParameterSpec.GetExtendedType

Synopsis: Get host object type

Declaration: `function GetExtendedType(AIndex: Integer) : TThoriumHostObjectType`

Visibility: protected

Description: Returns the host object type (35) of the parameter at AIndex or nil, if it is not an extended type parameter.

See also: TThoriumHostFunctionParameterSpec.GetCompleteType (27), TThoriumHostFunctionParameterSpec.SetExtendedType (27), TThoriumHostFunctionParameterSpec.GetParamType (27), TThoriumHostFunctionParameterSpec.ExtendedTypes (30)

### 1.21.9 TThoriumHostFunctionParameterSpec.GetParamType

Synopsis: Get host type

Declaration: `function GetParamType(AIndex: Integer) : TThoriumHostType`

Visibility: protected

Description: Returns the host type (1) of the parameter at AIndex.

See also: TThoriumHostFunctionParameterSpec.GetCompleteType (27), TThoriumHostFunctionParameterSpec.GetExtendedType (27), TThoriumHostFunctionParameterSpec.Types (29), TThoriumHostFunctionParameterSpec.SetParamType (28)

### 1.21.10 TThoriumHostFunctionParameterSpec.SetCapacity

Synopsis: Set the capacity of the list.

Declaration: `procedure SetCapacity(AValue: Integer)`

Visibility: protected

Description: Sets the capacity to the list to the given value. This does not work if the value is smaller than the amount of elements already in the list. Used to preallocate entries if you add many to the list.

See also: TThoriumHostFunctionParameterSpec.Expand (26)

### 1.21.11 TThoriumHostFunctionParameterSpec.SetExtendedType

Synopsis: Set the host object type of an entry.

Declaration: `procedure SetExtendedType(AIndex: Integer;  
AValue: TThoriumHostObjectType)`

Visibility: protected

Description: Sets the host object type (35) of the parameter at index AIndex.

See also: TThoriumHostFunctionParameterSpec.SetParamType (28), TThoriumHostFunctionParameterSpec.ExtendedTypes (30)

### 1.21.12 TThoriumHostFunctionParameterSpec.SetParamType

Synopsis: Set the host type of an entry

Declaration: `procedure SetParamType(AIndex: Integer; AValue: TThoriumHostType)`

Visibility: protected

Description: Sets the host type (1) of the parameter at index AIndex. Make sure you set an host object type (35) too if you set it as an extended type.

### 1.21.13 TThoriumHostFunctionParameterSpec.AddType

Synopsis: Adds a new basic entry.

Declaration: `function AddType(AType: TThoriumHostType) : Integer`

Visibility: public

Description: Adds a new parameter to the list as a non-extended type specified by AType and returns the index where the new entry is placed.

See also: TThoriumHostFunctionParameterSpec.AddExtendedType (28), TThoriumHostFunctionParameterSpec.InsetType (25)

### 1.21.14 TThoriumHostFunctionParameterSpec.AddExtendedType

Synopsis: Adds an host object type to the list.

Declaration: `function AddExtendedType(AType: TThoriumHostObjectType) : Integer`

Visibility: public

Description: Adds a new entry which contains an extended type (i.e. host object type) specified by AType and returns the index at which the entry is placed.

See also: TThoriumHostFunctionParameterSpec.AddType (28), TThoriumHostFunctionParameterSpec.InsertExtendedType (29)

### 1.21.15 TThoriumHostFunctionParameterSpec.AllTypes

Synopsis: Access to all elements through a pointer.

Declaration: `function AllTypes : PThoriumExternalFunctionVarType`

Visibility: public

Description: Returns a pointer to the first element in the list. This allows faster access of any element in the list.

### 1.21.16 TThoriumHostFunctionParameterSpec.IndexOfType

Synopsis: Find an occurrence of the given type.

Declaration: `function IndexOfType(AType: TThoriumHostType; Nth: Integer) : Integer`

Visibility: public

Description: Looks for the given type in the list and returns the index of the Nth occurrence.

### 1.21.17 TThoriumHostFunctionParameterSpec.InsertType

Synopsis: Insert a type.

Declaration: `procedure InsertType(AType: TThoriumHostType; AIndex: Integer)`

Visibility: public

Description: Inserts the given type in the list so that it has the given index afterwards.

See also: `TThoriumHostFunctionParameterSpec.InsertExtendedType` (29), `TThoriumHostFunctionParameterSpec.AddType` (28)

### 1.21.18 TThoriumHostFunctionParameterSpec.InsertExtendedType

Synopsis: Insert an host object type.

Declaration: `procedure InsertExtendedType(AType: TThoriumHostObjectType;  
AIndex: Integer)`

Visibility: public

Description: Inserts the given host objec type into the list so that it has the specified index afterwards.

See also: `TThoriumHostFunctionParameterSpec.InsertType` (29), `TThoriumHostFunctionParameterSpec.AddExtendedType` (28)

### 1.21.19 TThoriumHostFunctionParameterSpec.DeleteType

Synopsis: Delete a type from the list.

Declaration: `procedure DeleteType(AIndex: Integer)`

Visibility: public

Description: Deletes the type at the specified location from the list.

See also: `TThoriumHostFunctionParameterSpec.Clear` (29)

### 1.21.20 TThoriumHostFunctionParameterSpec.Clear

Synopsis: Clears the whole list.

Declaration: `procedure Clear`

Visibility: public

Description: Deletes all entries from the list.

See also: `TThoriumHostFunctionParameterSpec.DeleteType` (29)

### 1.21.21 TThoriumHostFunctionParameterSpec.Types

Synopsis: Access to the host types.

Declaration: `Property Types[Index: Integer]: TThoriumHostType`

Visibility: public

Access: Read, Write

Description: Provides access to the host type (1) part of a parameter.

See also: `TThoriumHostFunctionParameterSpec.GetParamType` (27), `TThoriumHostFunctionParameterSpec.SetParamType` (28), `TThoriumHostFunctionParameterSpec.ExtendedTypes` (30), `TThoriumHostFunctionParameterSpec.CompleteTypes` (30)

### 1.21.22 TThoriumHostFunctionParameterSpec.ExtendedTypes

Synopsis: Access to host object type.

Declaration: `Property ExtendedTypes[Index: Integer]: TThoriumHostObjectType`

Visibility: public

Access: Read,Write

Description: Provides access to the host object type (35) part of a parameter.

See also: `TThoriumHostFunctionParameterSpec.GetExtendedType` (27), `TThoriumHostFunctionParameterSpec.SetExtendedType` (27), `TThoriumHostFunctionParameterSpec.Types` (29), `TThoriumHostFunctionParameterSpec.CompleteTypes` (30)

### 1.21.23 TThoriumHostFunctionParameterSpec.CompleteTypes

Synopsis: Access to complete specifications.

Declaration: `Property CompleteTypes[Index: Integer]: PThoriumExternalFunctionVarType`

Visibility: public

Access: Read

Description: Provides access to the complete specification of a parameter.

See also: `TThoriumHostFunctionParameterSpec.GetCompleteType` (27), `TThoriumHostFunctionParameterSpec.Types` (29), `TThoriumHostFunctionParameterSpec.ExtendedTypes` (30)

### 1.21.24 TThoriumHostFunctionParameterSpec.Capacity

Synopsis: Access the capacity of the list.

Declaration: `Property Capacity : Integer`

Visibility: public

Access: Read,Write

Description: Provides read-write access to the capacity of the list. The constraints of the `SetCapacity` (27) method apply here too.

See also: `TThoriumHostFunctionParameterSpec.SetCapacity` (27)

### 1.21.25 TThoriumHostFunctionParameterSpec.Count

Synopsis: Access the amount of items.

Declaration: `Property Count : Integer`

Visibility: public

Access: Read

Description: Provides read access to the amount of items placed in the list.

## 1.22 TThoriumHostFunctionSimpleMethod

### 1.22.1 Description

This class implements a simple call to a function of the host environment. Parameters are passed to a specific signaturred function, as well as a pointer where to put the return value, all in `TThoriumValue` (6) format. For more info see `TThoriumSimpleMethod` (5).

### 1.22.2 Method overview

Page	Property	Description
<a href="#">31</a>	CallFromVirtualMachine	
<a href="#">31</a>	Create	

### 1.22.3 Property overview

Page	Property	Access	Description
<a href="#">31</a>	Method	rw	Method pointer to be called.

### 1.22.4 TThoriumHostFunctionSimpleMethod.Create

Declaration: constructor Create; Override

Visibility: default

### 1.22.5 TThoriumHostFunctionSimpleMethod.CallFromVirtualMachine

Declaration: procedure CallFromVirtualMachine  
(AVirtualMachine: TThoriumVirtualMachine)  
; Override

Visibility: protected

### 1.22.6 TThoriumHostFunctionSimpleMethod.Method

Synopsis: Method pointer to be called.

Declaration: Property Method : TThoriumSimpleMethod

Visibility: public

Access: Read,Write

Description: The pointer to the method which will be called.

## 1.23 TThoriumHostMethodAsFunctionNativeCall

### 1.23.1 Description

Works like TThoriumHostFunctionNativeCall ([24](#)), except that a constant is passed as the first parameter, which is assumed to be a Pointer and which must not be specified in the parameter array. If the function is a method, the constant parameter will come out as Self in the method and does not need to be declared in the function signature.

### 1.23.2 Method overview

Page	Property	Description
<a href="#">32</a>	CallFromVirtualMachine	
<a href="#">32</a>	Create	
<a href="#">32</a>	Precompile	

### 1.23.3 Property overview

Page	Property	Access	Description
<a href="#">32</a>	DataPointer	rw	

### 1.23.4 TThoriumHostMethodAsFunctionNativeCall.Create

**Declaration:** constructor Create; Override

Visibility: public

### 1.23.5 TThoriumHostMethodAsFunctionNativeCall.CallFromVirtualMachine

```
Declaration: procedure CallFromVirtualMachine
                                     (AVirtualMachine: TThoriumVirtualMachine)
                                     ; Override
```

Visibility: protected

### 1.23.6 TThoriumHostMethodAsFunctionNativeCall.Precompile

```
Declaration: procedure Precompile; Override
```

Visibility: public

### 1.23.7 TThoriumHostMethodAsFunctionNativeCall.DataPointer

**Declaration:** Property DataPointer : Pointer

Visibility: public

Access: Read, Write

## 1.24 TThoriumHostMethodBase

### 1.24.1 Description

Abstract base class to call methods of the host environment.

### 1.24.2 Method overview

Page	Property	Description
<a href="#">33</a>	CallFromVirtualMachine	
<a href="#">32</a>	Create	

### 1.24.3 TThoriumHostMethodBase.Create

**Declaration:** constructor Create; Override

Visibility: public



### 1.24.4 TThoriumHostMethodBase.CallFromVirtualMachine

Declaration: `procedure CallFromVirtualMachine(OfObject: TObject;  
AVirtualMachine: TThoriumVirtualMachine)  
; Virtual; Abstract`

Visibility: protected

## 1.25 TThoriumHostMethodNativeCall

### 1.25.1 Description

Similar to TThoriumHostFunctionNativeCall (24), except that the Self pointer is adjusted according to the calling context like in TThoriumHostMethodSimple (34).

### 1.25.2 Method overview

Page	Property	Description
<a href="#">33</a>	CallFromVirtualMachine	
<a href="#">33</a>	Create	
<a href="#">33</a>	Destroy	
<a href="#">33</a>	Precompile	Precompile NativeCall subscript.

### 1.25.3 Property overview

Page	Property	Access	Description
<a href="#">34</a>	CallingConvention	rw	Calling convention
<a href="#">34</a>	CodePointer	rw	Pointer to the method.

### 1.25.4 TThoriumHostMethodNativeCall.Create

Declaration: `constructor Create; Override`

Visibility: public

### 1.25.5 TThoriumHostMethodNativeCall.Destroy

Declaration: `destructor Destroy; Override`

Visibility: public

### 1.25.6 TThoriumHostMethodNativeCall.CallFromVirtualMachine

Declaration: `procedure CallFromVirtualMachine(OfObject: TObject;  
AVirtualMachine: TThoriumVirtualMachine)  
; Override`

Visibility: protected

### 1.25.7 TThoriumHostMethodNativeCall.Precompile

Synopsis: Precompile NativeCall subscript.

Declaration: `procedure Precompile`

Visibility: public

Description: This method performs precompilation of the NativeCall subscript which is needed to perform the call. You must call this before the first attempt of a native call but after you have configured the parameters.

### 1.25.8 TThoriumHostMethodNativeCall.CallingConvention

Synopsis: Calling convention

Declaration: `Property CallingConvention : TThoriumNativeCallingConvention`

Visibility: public

Access: Read,Write

Description: Describes the calling convention of the method. See `TThoriumHostFunctionNativeCall.CallingConvention` (25) for more information.

### 1.25.9 TThoriumHostMethodNativeCall.CodePointer

Synopsis: Pointer to the method.

Declaration: `Property CodePointer : Pointer`

Visibility: public

Access: Read,Write

Description: This must be the pointer to the method to be called.

## 1.26 TThoriumHostMethodSimple

### 1.26.1 Description

Similar to `TThoriumHostFunctionSimpleMethod` (30), except that the Self pointer of the function is modified to whatever matches the context (i.e. which host object type variable the method belongs to - gets dereferenced up to the pointer so that you can use it like a normal method).

### 1.26.2 Method overview

Page	Property	Description
<a href="#">35</a>	<code>CallFromVirtualMachine</code>	
<a href="#">34</a>	<code>Create</code>	

### 1.26.3 Property overview

Page	Property	Access	Description
<a href="#">35</a>	<code>ClassMethod</code>	rw	

### 1.26.4 TThoriumHostMethodSimple.Create

Declaration: `constructor Create; Override`

Visibility: public

### 1.26.5 TThoriumHostMethodSimple.CallFromVirtualMachine

Declaration: `procedure CallFromVirtualMachine(OfObject: TObject;  
AVirtualMachine: TThoriumVirtualMachine)  
; Override`

Visibility: protected

### 1.26.6 TThoriumHostMethodSimple.ClassMethod

Declaration: `Property ClassMethod : TThoriumClassMethod`

Visibility: public

Access: Read,Write

## 1.27 TThoriumHostObjectType

### 1.27.1 Description

This class is the base class to publish any class-alike type to Thorium. You will need to override the virtual methods to make it represent the type you want.

### 1.27.2 Method overview

Page	Property	Description
<a href="#">36</a>	AssignValue	Perform assignment - deprecated?
<a href="#">36</a>	Create	
<a href="#">36</a>	Destroy	
<a href="#">37</a>	DisposeValue	Release an instance
<a href="#">36</a>	DuplicateValue	Duplicate the given value
<a href="#">39</a>	FieldID	Get the ID of a field.
<a href="#">40</a>	FieldType	Get the type of a field.
<a href="#">39</a>	FindMethod	
<a href="#">40</a>	GetField	Perform read-access to a field.
<a href="#">40</a>	GetIndex	Perform read access using an index.
<a href="#">36</a>	GetNeededMemoryAmount	Get the memory amount needed for one instance.
<a href="#">41</a>	GetPropertyStoring	Determine whether a field is storing.
<a href="#">41</a>	GetStaticField	Perform read access to a static field.
<a href="#">38</a>	HasFields	Determine whether a type has any fields.
<a href="#">38</a>	HasIndicies	Determine whether your type can be accessed via indicies.
<a href="#">38</a>	HasStaticFields	Determine whether your type has any static fields.
<a href="#">39</a>	IndexType	Get the type of value returned at indexed access.
<a href="#">38</a>	IsTypeCompatible	Determine whether a two-operand operation is possible.
<a href="#">38</a>	IsTypeOperationAvailable	Determine whether an unary operation is possible.
<a href="#">37</a>	PerformEvaluation	Evaluate as integer.
<a href="#">37</a>	PerformNegation	Negate the value
<a href="#">37</a>	PerformNot	Invert the value.
<a href="#">36</a>	PerformOperation	Perform the given operation on a value.
<a href="#">41</a>	SetField	Perform write access to a field.
<a href="#">40</a>	SetIndex	Perform write access using an index.
<a href="#">41</a>	SetStaticField	Perform write access to a static field.
<a href="#">39</a>	StaticFieldID	Get a static field ID.
<a href="#">39</a>	StaticFieldType	Get the type of a static field.

### 1.27.3 Property overview

Page	Property	Access	Description
<a href="#">42</a>	Name	r	Published name

### 1.27.4 TThoriumHostObjectType.Create

Declaration: `constructor Create(ALibrary: TThoriumLibrary); Virtual`

Visibility: `default`

### 1.27.5 TThoriumHostObjectType.Destroy

Declaration: `destructor Destroy; Override`

Visibility: `default`

### 1.27.6 TThoriumHostObjectType.GetNeededMemoryAmount

Synopsis: Get the memory amount needed for one instance.

Declaration: `function GetNeededMemoryAmount : TThoriumSizeInt; Virtual; Abstract`

Visibility: `protected`

Description: This function must return the amount of memory needed for one instance of this type. If you do not need to allocate any memory but you want to assign the value yourself, you should return 0. This will leave the pointer field uninitialized so you can assign anything you want.

### 1.27.7 TThoriumHostObjectType.DuplicateValue

Synopsis: Duplicate the given value

Declaration: `function DuplicateValue(const AValue: TThoriumHostObjectTypeValue)  
: TThoriumValue; Virtual; Abstract`

Visibility: `protected`

Description: This function is supposed to duplicate the Value which is passed as a parameter and return it. This means, that this value should be able to be independently freed by Thorium without destroying the other instance. If you use reference counting, it will be enough to copy the value given and increase the reference by one. It is guaranteed that the input value is of the type this class reflects.

### 1.27.8 TThoriumHostObjectType.AssignValue

Synopsis: Perform assignment - deprecated?

Declaration: `procedure AssignValue(const ASource: TThoriumValue;  
var ADest: TThoriumValue); Virtual; Abstract`

Visibility: `protected`

Description: `Deprecated?`

### 1.27.9 TThoriumHostObjectType.PerformOperation

Synopsis: Perform the given operation on a value.

**Declaration:** `function PerformOperation(const AValue1: TThoriumValue;  
const AValue2: TThoriumValue;  
const Op: TThoriumOperator) : TThoriumValue  
; Virtual; Abstract`

**Visibility:** protected

**Description:** This should perform the given operation on the given values and return the result (if any). There is no need to override this method if your type does not support this.

### 1.27.10 TThoriumHostObjectType.PerformEvaluation

**Synopsis:** Evaluate as integer.

**Declaration:** `function PerformEvaluation(const AValue: TThoriumHostObjectTypeValue)  
: Integer; Virtual; Abstract`

**Visibility:** protected

**Description:** Evaluate the value given as integer (i.e. for boolean evaluations, 0 for false, anything else for true). It is guaranteed that the input value is of the type this class represents.

### 1.27.11 TThoriumHostObjectType.PerformNegation

**Synopsis:** Negate the value

**Declaration:** `function PerformNegation(const AValue: TThoriumHostObjectTypeValue)  
: TThoriumValue; Virtual; Abstract`

**Visibility:** protected

**Description:** Negate the value given (if possible at all) and return the result. There is no need to override this method if your type does not support this.

### 1.27.12 TThoriumHostObjectType.PerformNot

**Synopsis:** Invert the value.

**Declaration:** `function PerformNot(const AValue: TThoriumHostObjectTypeValue)  
: TThoriumValue; Virtual; Abstract`

**Visibility:** protected

**Description:** This is called to execute the not-operator. There is no need to override this method if your type does not support this.

### 1.27.13 TThoriumHostObjectType.DisposeValue

**Synopsis:** Release an instance

**Declaration:** `procedure DisposeValue(var AValue: TThoriumHostObjectTypeValue)  
; Virtual; Abstract`

**Visibility:** protected

**Description:** Release the given instance of your type. If you use reference counting, it will be sufficient if you just decrease the reference by one (and free if no references are left).

### 1.27.14 TThoriumHostObjectType.IsTypeCompatible

Synopsis: Determine whether a two-operand operation is possible.

Declaration: `function IsTypeCompatible(const Value1: TThoriumType;  
const Value2: TThoriumType;  
const Operation: TThoriumOperation;  
out ResultType: TThoriumType) : Boolean  
; Virtual; Abstract`

Visibility: protected

Description: This method is supposed to return whether the given operation is possible. In explicit, this determines which operations the compiler will translate into code and where it will throw an error. Make sure you set *ResultType* to notify the compiler about any type changes (e.g., multiplication of float and integer produce a float).

### 1.27.15 TThoriumHostObjectType.IsTypeOperationAvailable

Synopsis: Determine whether an unary operation is possible.

Declaration: `function IsTypeOperationAvailable(const Value: TThoriumType;  
const Operation: TThoriumOperation;  
out ResultType: TThoriumType) : Boolean  
; Virtual; Abstract`

Visibility: protected

Description: Return whether the given operation is possible on any value of your type. Make sure to set *ResultType* accordingly to any change in the type.

### 1.27.16 TThoriumHostObjectType.HasFields

Synopsis: Determine whether a type has any fields.

Declaration: `function HasFields : Boolean; Virtual; Abstract`

Visibility: protected

Description: Tell the compiler whether your type has any fields. Fields are what properties are in FreePascal, i.e. they are accessed without any parameter handling just like public variables. However, fields may be of an method- or function-type (and in that case of course read-only) so that they can be called. In fact, that is the way to publish methods to Thorium.

### 1.27.17 TThoriumHostObjectType.HasStaticFields

Synopsis: Determine whether your type has any static fields.

Declaration: `function HasStaticFields : Boolean; Virtual; Abstract`

Visibility: protected

Description: Return true if your type has any static fields. Static fields are, other than normal fields, accessed like static methods are in FreePascal, using the type instead of the instance. So if your type is published as "TTestType" to Thorium, a static field will be accessed via "TTestType.staticFieldName", assuming you have a static field called "staticFieldName".

### 1.27.18 TThoriumHostObjectType.HasIndicies

Synopsis: Determine whether your type can be accessed via indicies.

Declaration: `function HasIndicies : Boolean; Virtual; Abstract`

Visibility: protected

Description: If your type supports array-like access (i.e. "Instance[10]" or alike), you must return true.

### 1.27.19 TThoriumHostObjectType.FindMethod

Declaration: `function FindMethod(const AMethodName: String) : TThoriumHostMethodBase  
; Virtual`

Visibility: protected

### 1.27.20 TThoriumHostObjectType.FieldID

Synopsis: Get the ID of a field.

Declaration: `function FieldID(const FieldIdent: String;out ID: QWord) : Boolean  
; Virtual; Abstract`

Visibility: public

Description: This method is supposed to return an ID which is unique to the field identified by *FieldIdent*. If that field does not exist, you must return false instead of an invalid ID. There is no need to implement this method if your implementation of *HasFields* (38) returns false.

### 1.27.21 TThoriumHostObjectType.StaticFieldID

Synopsis: Get a static field ID.

Declaration: `function StaticFieldID(const FieldIdent: String;out ID: QWord) : Boolean  
; Virtual; Abstract`

Visibility: public

Description: Similar to *TThoriumHostObjectType.FieldID* (39), but for static fields. There is no need to override this method if your implementation of *HasStaticFields* (38) returns false.

### 1.27.22 TThoriumHostObjectType.IndexType

Synopsis: Get the type of value returned at indexed access.

Declaration: `function IndexType(const InputType: TThoriumType;  
out ResultType: TThoriumTableEntry) : Boolean  
; Virtual; Abstract`

Visibility: public

Description: Implement this if your type supports indicies. If your type supports the given *InputType* as index type, you must return true and write the type the result of the index operation will have to *ResultType*. Otherwise return false.

### 1.27.23 TThoriumHostObjectType.StaticFieldType

Synopsis: Get the type of a static field.

Declaration: `function StaticFieldType(const AFieldID: QWord;  
out ResultType: TThoriumTableEntry) : Boolean  
; Virtual; Abstract`

Visibility: public

Description: This method must return the type of the static field identified by *ID* in *ResultType*. It is guaranteed that the ID has been fetched using `StaticFieldID` (39). If the type is for some reason not able to return a valid type (e.g. invalid ID or something), the method must return false. The compiler will throw an error about that.

#### 1.27.24 TThoriumHostObjectType.FieldType

Synopsis: Get the type of a field.

Declaration: 

```
function FieldType(const AFieldID: QWord;
                  out ResultType: TThoriumTableEntry) : Boolean
; Virtual; Abstract
```

Visibility: public

Description: This method must return the type of the field identified by *ID* in *ResultType*. It is guaranteed that the ID has been fetched using `FieldID` (39). If the type is for some reason not able to return a valid type (e.g. invalid ID or something), the method must return false. The compiler will throw an error about that.

#### 1.27.25 TThoriumHostObjectType.GetIndex

Synopsis: Perform read access using an index.

Declaration: 

```
function GetIndex(const AInstance: TThoriumValue;
                  const AIndex: TThoriumValue) : TThoriumValue; Virtual
; Abstract
```

Visibility: public

Description: This method is called by the virtual machine whenever an instance of the type is read-accessed via indices. The instance as well as the used index are passed as parameters and the method must return the value at that location or may throw an exception (although this is discouraged). The returned value must have the type which has been announced to the compiler before using `IndexType` (39).

#### 1.27.26 TThoriumHostObjectType.SetIndex

Synopsis: Perform write access using an index.

Declaration: 

```
procedure SetIndex(const AInstance: TThoriumValue;
                   const AIndex: TThoriumValue;
                   const NewValue: TThoriumValue); Virtual; Abstract
```

Visibility: public

Description: This method is called by the virtual machine whenever an instance of the type is write-accessed via indices. The instance, the index used to access as well as the value assigned are passed as parameters. If anything is wrong, the method may throw an exception.

#### 1.27.27 TThoriumHostObjectType.GetField

Synopsis: Perform read-access to a field.

Declaration: 

```
function GetField(const AInstance: TThoriumValue; const AFieldID: QWord)
                  : TThoriumValue; Virtual; Abstract
```

Visibility: public



**Description:** This gets called by the virtual machine whenever a field of your type is read-accessed. This does not match for methods and functions since these references are considered to be static and thus solved at compile time. The instance which is subject to the access as well as the ID of the field are passed as parameters and the function is expected to return the current value of the field.

### 1.27.28 TThoriumHostObjectType.GetStaticField

**Synopsis:** Perform read access to a static field.

**Declaration:**

```
function GetStaticField(const AInstance: TThoriumValue;
                        const AFieldID: QWord) : TThoriumValue; Virtual
; Abstract
```

**Visibility:** public

**Description:** This is similar to `GetField` (40), but for static fields. Note that static fields are not solved at compile time and thus the value is allowed to change during the runtime of the script.

### 1.27.29 TThoriumHostObjectType.SetField

**Synopsis:** Perform write access to a field.

**Declaration:**

```
procedure SetField(const AInstance: TThoriumValue; const AFieldID: QWord;
                  const NewValue: TThoriumValue); Virtual; Abstract
```

**Visibility:** public

**Description:** The virtual machine calls this method whenever a field of the type is write accessed. This is only called, when the `FieldType` (40) call subject to this field ID has not set the Static bit in the type. The method is expected to change the value of the field identified by the given ID in the given instance to the given value.

### 1.27.30 TThoriumHostObjectType.SetStaticField

**Synopsis:** Perform write access to a static field.

**Declaration:**

```
procedure SetStaticField(const AInstance: TThoriumValue;
                        const AFieldID: QWord;
                        const NewValue: TThoriumValue); Virtual
; Abstract
```

**Visibility:** public

**Description:** The virtual machine calls this method whenever a static field of the type is write accessed. This is only called, when the `FieldType` (40) call subject to this field ID has not set the Static bit in the type. The method is expected to change the value of the static field identified by the given ID to the given value.

### 1.27.31 TThoriumHostObjectType.GetPropertyStoring

**Synopsis:** Determine whether a field is storing.

**Declaration:**

```
function GetPropertyStoring(const AFieldID: QWord) : Boolean; Virtual
; Abstract
```

**Visibility:** public

**Description:** Return true if the property is "storing". Storing has only relevancy when a RTTI based host object value is passed to a property or a parameter. If the storing bit is set, the object will be marked as "host controlled" (i.e., the according method is called) and will not be freed by reference counting unless the countermethod is called afterwards.

### 1.27.32 TThoriumHostObjectType.Name

Synopsis: Published name

Declaration: `Property Name : String`

Visibility: public

Access: Read

Description: This is the name which can be used in Thorium to access this type.

## 1.28 TThoriumIdentifierTable

### 1.28.1 Description

Class for internal use - to be described later.

### 1.28.2 Method overview

Page	Property	Description
<a href="#">42</a>	AddConstantIdentifier	
<a href="#">43</a>	AddFunctionIdentifier	
<a href="#">43</a>	AddRegisterVariableIdentifier	
<a href="#">43</a>	AddVariableIdentifier	
<a href="#">43</a>	ClearTable	
<a href="#">43</a>	ClearTableTo	
<a href="#">42</a>	Create	
<a href="#">42</a>	Destroy	
<a href="#">43</a>	FindIdentifier	

### 1.28.3 Property overview

Page	Property	Access	Description
<a href="#">43</a>	Count	r	

### 1.28.4 TThoriumIdentifierTable.Create

Declaration: `constructor Create`

Visibility: default

### 1.28.5 TThoriumIdentifierTable.Destroy

Declaration: `destructor Destroy; Override`

Visibility: default

### 1.28.6 TThoriumIdentifierTable.AddConstantIdentifier

Declaration: `procedure AddConstantIdentifier(Name: String; Scope: Integer;  
Offset: Integer; TypeSpec: TThoriumType;  
Value: TThoriumValue)`

Visibility: public

**1.28.7 TThoriumIdentifierTable.AddVariableIdentifier**

Declaration: `procedure AddVariableIdentifier(Name: String; Scope: Integer;  
Offset: Integer; TypeSpec: TThoriumType)`

Visibility: public

**1.28.8 TThoriumIdentifierTable.AddRegisterVariableIdentifier**

Declaration: `procedure AddRegisterVariableIdentifier(Name: String;  
RegisterID: TThoriumRegisterID;  
TypeSpec: TThoriumType)`

Visibility: public

**1.28.9 TThoriumIdentifierTable.AddFunctionIdentifier**

Declaration: `procedure AddFunctionIdentifier(Name: String; Func: TThoriumFunction)`

Visibility: public

**1.28.10 TThoriumIdentifierTable.ClearTable**

Declaration: `procedure ClearTable`

Visibility: public

**1.28.11 TThoriumIdentifierTable.ClearTableTo**

Declaration: `function ClearTableTo(NewCount: Integer) : Integer`

Visibility: public

**1.28.12 TThoriumIdentifierTable.FindIdentifier**

Declaration: `function FindIdentifier(Name: String; out Ident: TThoriumTableEntry)  
: Boolean`

Visibility: public

**1.28.13 TThoriumIdentifierTable.Count**

Declaration: `Property Count : Integer`

Visibility: public

Access: Read

**1.29 TThoriumInstructions****1.29.1 Description**

Class for internal use - to be fully described later. This class is a container for Thorium instructions with some extra methods for easier handling, such as keeping track in address lists of inserted instructions (and thus changed indicies) and more.

### 1.29.2 Method overview

Page	Property	Description
<a href="#">45</a>	AddInstructionPointer	
<a href="#">44</a>	AppendCode	
<a href="#">45</a>	ClearCode	
<a href="#">44</a>	Create	
<a href="#">44</a>	DeleteInstructions	
<a href="#">44</a>	Destroy	
<a href="#">45</a>	DumpCodeBin	
<a href="#">45</a>	DumpCodeStr	
<a href="#">44</a>	Finish	
<a href="#">45</a>	LoadFromStream	
<a href="#">45</a>	RegisterAddressList	
<a href="#">45</a>	RemoveInstructionPointer	
<a href="#">45</a>	SaveToStream	
<a href="#">45</a>	UnRegisterAddressList	

### 1.29.3 Property overview

Page	Property	Access	Description
<a href="#">46</a>	Capacity	rw	
<a href="#">46</a>	Count	r	
<a href="#">46</a>	Instruction	r	
<a href="#">46</a>	Position	rw	

### 1.29.4 TThoriumInstructions.Create

Declaration: `constructor Create`

Visibility: `default`

### 1.29.5 TThoriumInstructions.Destroy

Declaration: `destructor Destroy; Override`

Visibility: `default`

### 1.29.6 TThoriumInstructions.AppendCode

Declaration: `function AppendCode(AInstruction: TThoriumInstruction) : Integer`  
`function AppendCode(Code: TThoriumInstructionArray) : Integer`

Visibility: `public`

### 1.29.7 TThoriumInstructions.DeleteInstructions

Declaration: `procedure DeleteInstructions(AIndex: Integer; ACount: Integer)`

Visibility: `public`

### 1.29.8 TThoriumInstructions.Finish

Declaration: `procedure Finish`

Visibility: public

### **1.29.9 TThoriumInstructions.ClearCode**

Declaration: procedure ClearCode

Visibility: public

### **1.29.10 TThoriumInstructions.RegisterAddressList**

Declaration: procedure RegisterAddressList (AList: TThoriumIntList)

Visibility: public

### **1.29.11 TThoriumInstructions.UnRegisterAddressList**

Declaration: procedure UnRegisterAddressList (AList: TThoriumIntList)

Visibility: public

### **1.29.12 TThoriumInstructions.AddInstructionPointer**

Declaration: procedure AddInstructionPointer (APointer: PThoriumInstructionAddress)

Visibility: public

### **1.29.13 TThoriumInstructions.RemoveInstructionPointer**

Declaration: procedure RemoveInstructionPointer (APointer: PThoriumInstructionAddress)

Visibility: public

### **1.29.14 TThoriumInstructions.DumpCodeBin**

Declaration: procedure DumpCodeBin (DestStream: TStream)

Visibility: public

### **1.29.15 TThoriumInstructions.DumpCodeStr**

Declaration: function DumpCodeStr : String

Visibility: public

### **1.29.16 TThoriumInstructions.LoadFromStream**

Declaration: procedure LoadFromStream (Stream: TStream)

Visibility: public

### **1.29.17 TThoriumInstructions.SaveToStream**

Declaration: procedure SaveToStream (Stream: TStream)

Visibility: public

### 1.29.18 TThoriumInstructions.Count

Declaration: `Property Count : Integer`

Visibility: `public`

Access: `Read`

### 1.29.19 TThoriumInstructions.Capacity

Declaration: `Property Capacity : Integer`

Visibility: `public`

Access: `Read,Write`

### 1.29.20 TThoriumInstructions.Instruction

Declaration: `Property Instruction[Index: Integer]: PThoriumInstruction; default`

Visibility: `public`

Access: `Read`

### 1.29.21 TThoriumInstructions.Position

Declaration: `Property Position : TThoriumInstructionAddress`

Visibility: `public`

Access: `Read,Write`

## 1.30 TThoriumIntList

### 1.30.1 Description

Class for internal use - to be described later.

### 1.30.2 Method overview

Page	Property	Description
<a href="#">47</a>	AddEntry	
<a href="#">47</a>	Create	
<a href="#">47</a>	DeleteEntry	
<a href="#">47</a>	Destroy	
<a href="#">47</a>	FindValue	

### 1.30.3 Property overview

Page	Property	Access	Description
<a href="#">47</a>	Capacity	rw	
<a href="#">47</a>	Count	rw	
<a href="#">47</a>	Items	rw	

#### **1.30.4 TThoriumIntList.Create**

Declaration: constructor Create

Visibility: default

#### **1.30.5 TThoriumIntList.Destroy**

Declaration: destructor Destroy; Override

Visibility: default

#### **1.30.6 TThoriumIntList.AddEntry**

Declaration: function AddEntry(Value: Integer) : Integer

Visibility: public

#### **1.30.7 TThoriumIntList.FindValue**

Declaration: function FindValue(AValue: Integer) : Integer

Visibility: public

#### **1.30.8 TThoriumIntList.DeleteEntry**

Declaration: procedure DeleteEntry(AIndex: Integer)

Visibility: public

#### **1.30.9 TThoriumIntList.Items**

Declaration: Property Items[Index: Integer]: Integer; default

Visibility: public

Access: Read,Write

#### **1.30.10 TThoriumIntList.Count**

Declaration: Property Count : Integer

Visibility: public

Access: Read,Write

#### **1.30.11 TThoriumIntList.Capacity**

Declaration: Property Capacity : Integer

Visibility: public

Access: Read,Write

## 1.31 TThoriumIntStack

### 1.31.1 Description

Class for internal use - to be described later.

### 1.31.2 Method overview

Page	Property	Description
<a href="#">48</a>	Pop	
<a href="#">48</a>	Push	

---

### 1.31.3 TThoriumIntStack.Push

Declaration: `procedure Push(Value: Integer)`

Visibility: `public`

### 1.31.4 TThoriumIntStack.Pop

Declaration: `function Pop : Integer`

Visibility: `public`

## 1.32 TThoriumJumpList

### 1.32.1 Description

Class for internal use - to be described later.

### 1.32.2 Method overview

Page	Property	Description
<a href="#">48</a>	ChangeAddresses	
<a href="#">48</a>	FillAddresses	

---

### 1.32.3 TThoriumJumpList.FillAddresses

Declaration: `procedure FillAddresses(DownToCount: Integer;  
Address: TThoriumInstructionAddress;  
Instructions: TThoriumInstructions)`

Visibility: `public`

### 1.32.4 TThoriumJumpList.ChangeAddresses

Declaration: `procedure ChangeAddresses(Offset: Integer;  
AfterAddress: TThoriumInstructionAddress;  
Instructions: TThoriumInstructions)`

Visibility: `public`



## **1.33 TThoriumLibrary**

### **1.33.1 Description**

The base class for any library the host environment may want to publish to Thorium. To build a library, you need to override the `GetName` and `InitializeLibrary` methods. For an example see the `thoriumlibpkg` and the `customlib` example.

**1.33.2 Method overview**

Page	Property	Description
<a href="#">52</a>	AddDependency	Add a dependency
<a href="#">52</a>	ClearAll	Clear the whole library
<a href="#">53</a>	ClearFunctions	Clear all functions
<a href="#">53</a>	ClearTypes	Clear all types
<a href="#">51</a>	Create	
<a href="#">55</a>	DeepFindHostType	
<a href="#">56</a>	DeepFindRTTIType	
<a href="#">56</a>	DeepFindRTTITypeByClass	
<a href="#">53</a>	DeleteHostFunction	Delete a function
<a href="#">53</a>	DeleteHostType	Delete a type
<a href="#">51</a>	Destroy	
<a href="#">56</a>	FindConstant	
<a href="#">56</a>	FindHostFunction	
<a href="#">56</a>	FindHostType	
<a href="#">56</a>	FindProperty	
<a href="#">56</a>	FindRTTIType	
<a href="#">56</a>	FindRTTITypeByClass	
<a href="#">51</a>	GetConstant	
<a href="#">51</a>	GetConstantCount	
<a href="#">51</a>	GetHostFunction	
<a href="#">51</a>	GetHostFunctionCount	
<a href="#">51</a>	GetHostType	
<a href="#">52</a>	GetHostTypeCount	
<a href="#">52</a>	GetLibraryProperty	
<a href="#">52</a>	GetLibraryPropertyCount	
<a href="#">53</a>	GetName	Get the library name
<a href="#">52</a>	GetRTTIType	
<a href="#">52</a>	GetRTTITypeCount	
<a href="#">56</a>	IndexOfConstant	
<a href="#">57</a>	IndexOfHostFunction	
<a href="#">57</a>	IndexOfHostType	
<a href="#">57</a>	IndexOfProperty	
<a href="#">57</a>	IndexOfRTTIType	
<a href="#">53</a>	InitializeLibrary	Initialize the library
<a href="#">52</a>	PrecompileFunctions	Precompile all known NativeCall functions.
<a href="#">54</a>	RegisterConstant	
<a href="#">54</a>	RegisterNativeCallFunction	
<a href="#">54</a>	RegisterNativeCallMethodAsFunction	
<a href="#">54</a>	RegisterObjectType	
<a href="#">54</a>	RegisterPropertyCallback	
<a href="#">55</a>	RegisterPropertyCustom	
<a href="#">55</a>	RegisterPropertyDirect	
<a href="#">55</a>	RegisterPropertyDirectCallback	
<a href="#">55</a>	RegisterRTTIType	
<a href="#">55</a>	RegisterSimpleMethod	

### 1.33.3 Property overview

Page	Property	Access	Description
<a href="#">57</a>	Constant	r	
<a href="#">57</a>	ConstantCount	r	
<a href="#">57</a>	HostFunction	r	
<a href="#">57</a>	HostFunctionCount	r	
<a href="#">58</a>	HostType	r	
<a href="#">58</a>	HostTypeCount	r	
<a href="#">58</a>	LibraryProperty	r	
<a href="#">58</a>	LibraryPropertyCount	r	
<a href="#">58</a>	RTTIType	r	
<a href="#">58</a>	RTTITypeCount	r	

### 1.33.4 TThoriumLibrary.Create

Declaration: `constructor Create(AThorium: TThorium)`

Visibility: `public`

### 1.33.5 TThoriumLibrary.Destroy

Declaration: `destructor Destroy; Override`

Visibility: `public`

### 1.33.6 TThoriumLibrary.GetConstant

Declaration: `function GetConstant(AIndex: Integer) : TThoriumLibraryConstant`

Visibility: `protected`

### 1.33.7 TThoriumLibrary.GetConstantCount

Declaration: `function GetConstantCount : Integer`

Visibility: `protected`

### 1.33.8 TThoriumLibrary.GetHostFunction

Declaration: `function GetHostFunction(AIndex: Integer) : TThoriumHostFunctionBase`

Visibility: `protected`

### 1.33.9 TThoriumLibrary.GetHostFunctionCount

Declaration: `function GetHostFunctionCount : Integer`

Visibility: `protected`

### 1.33.10 TThoriumLibrary.GetHostType

Declaration: `function GetHostType(AIndex: Integer) : TThoriumHostObjectType`

Visibility: `protected`

### 1.33.11 TThoriumLibrary.GetHostTypeCount

Declaration: `function GetHostTypeCount : Integer`

Visibility: `protected`

### 1.33.12 TThoriumLibrary.GetLibraryProperty

Declaration: `function GetLibraryProperty(AIndex: Integer) : TThoriumLibraryProperty`

Visibility: `protected`

### 1.33.13 TThoriumLibrary.GetLibraryPropertyCount

Declaration: `function GetLibraryPropertyCount : Integer`

Visibility: `protected`

### 1.33.14 TThoriumLibrary.GetRTTIType

Declaration: `function GetRTTIType(AIndex: Integer) : TThoriumRTTIObjectType`

Visibility: `protected`

### 1.33.15 TThoriumLibrary.GetRTTITypeCount

Declaration: `function GetRTTITypeCount : Integer`

Visibility: `protected`

### 1.33.16 TThoriumLibrary.PrecompileFunctions

Synopsis: Precompile all known NativeCall functions.

Declaration: `procedure PrecompileFunctions`

Visibility: `protected`

Description: This will precompile any function which derives from `TThoriumFunctionNativeCall` ([1](#)) or `TThoriumMethodNativeCall` ([1](#)) and which is in the host function list of the library. This method is automatically called by the constructor after calling `InitializeLibrary` ([53](#)).

### 1.33.17 TThoriumLibrary.AddDependency

Synopsis: Add a dependency

Declaration: `procedure AddDependency(const ALibName: String)`  
`procedure AddDependency(const ALib: TThoriumLibrary)`

Visibility: `protected`

Description: This method adds a dependency to the library and throws an exception if the dependency cannot be fulfilled.

### 1.33.18 TThoriumLibrary.ClearAll

Synopsis: Clear the whole library

Declaration: `procedure ClearAll`

Visibility: protected

Description: Deletes anything related to this library. Properties, functions, types, anything I said, did you hear me? Anything!!

### 1.33.19 TThoriumLibrary.ClearFunctions

Synopsis: Clear all functions

Declaration: `procedure ClearFunctions`

Visibility: protected

Description: Delete any functions registered with this library.

### 1.33.20 TThoriumLibrary.ClearTypes

Synopsis: Clear all types

Declaration: `procedure ClearTypes`

Visibility: protected

Description: Deletes all types associated with this library.

### 1.33.21 TThoriumLibrary.DeleteHostFunction

Synopsis: Delete a function

Declaration: `procedure DeleteHostFunction (AIndex: Integer)`

Visibility: protected

Description: This removes the function at index *AIndex* from the library.

### 1.33.22 TThoriumLibrary.DeleteHostType

Synopsis: Delete a type

Declaration: `procedure DeleteHostType (AIndex: Integer)`

Visibility: protected

Description: This removes the type at index *AIndex* from the library.

### 1.33.23 TThoriumLibrary.GetName

Synopsis: Get the library name

Declaration: `function GetName : String; Virtual; Abstract`

Visibility: protected

Description: This class method must return the name of the library, that is the one under which the library should be able to be loaded in Thorium.

### 1.33.24 TThoriumLibrary.InitializeLibrary

Synopsis: Initialize the library

Declaration: `procedure InitializeLibrary; Virtual`

Visibility: protected

Description: This function should probably be overridden by any descendant class. It is called by the constructor to let the library initialize itself. That is adding host functions and types as well as library properties.

#### 1.33.25 TThoriumLibrary.RegisterConstant

Declaration: `function RegisterConstant(const AName: String;  
const AValue: TThoriumValue) : PThoriumValue`

Visibility: protected

#### 1.33.26 TThoriumLibrary.RegisterNativeCallFunction

Declaration: `function RegisterNativeCallFunction(const AName: String;  
const ACodePointer: Pointer;  
const AParameters: Array of TThoriumHostType;  
const AReturnType: TThoriumHostType;  
const ACallingConvention: TThoriumNativeCall  
: TThoriumHostFunctionNativeCall`

Visibility: protected

#### 1.33.27 TThoriumLibrary.RegisterNativeCallMethodAsFunction

Declaration: `function RegisterNativeCallMethodAsFunction(const AName: String;  
const ACodePointer: Pointer;  
const ADataPointer: Pointer;  
const AParameters: Array of TThoriumHostType;  
const AReturnType: TThoriumHostType;  
const ACallingConvention: TThoriumNativeCall  
: TThoriumHostMethodAsFunctionNativeCall`

Visibility: protected

#### 1.33.28 TThoriumLibrary.RegisterObjectType

Declaration: `function RegisterObjectType(const AName: String;  
const ATypeClass: TThoriumHostObjectTypeClass)  
: TThoriumHostObjectType`

Visibility: protected

#### 1.33.29 TThoriumLibrary.RegisterPropertyCallback

Declaration: `function RegisterPropertyCallback(const AName: String;  
const ATypeSpec: TThoriumType;  
Static: Boolean;  
const AGetCallback: TThoriumOnPropertyGet;  
const ASetCallback: TThoriumOnPropertySet)  
: TThoriumLibraryPropertyCallback`

Visibility: protected

**1.33.30 TThoriumLibrary.RegisterPropertyCustom**

Declaration: `function RegisterPropertyCustom(const AName: String;  
const AClass: TThoriumLibraryPropertyClass)  
: TThoriumLibraryProperty`

Visibility: protected

**1.33.31 TThoriumLibrary.RegisterPropertyDirect**

Declaration: `function RegisterPropertyDirect(const AName: String;  
const ATypeSpec: TThoriumType;  
Static: Boolean)  
: TThoriumLibraryPropertyDirect`

Visibility: protected

**1.33.32 TThoriumLibrary.RegisterPropertyDirectCallback**

Declaration: `function RegisterPropertyDirectCallback(const AName: String;  
const ATypeSpec: TThoriumType;  
Static: Boolean;  
Callback: TThoriumOnPropertySetCallback)  
: TThoriumLibraryPropertyDirectSetCallback`

Visibility: protected

**1.33.33 TThoriumLibrary.RegisterRTTIType**

Declaration: `function RegisterRTTIType(const AClass: TThoriumPersistentClass;  
AbstractClass: Boolean)  
: TThoriumRTTIObjectType  
function RegisterRTTIType(const AClass: TClass;  
AMethodsCallback: TThoriumRTTIMethodsCallback;  
AStaticMethodsCallback: TThoriumRTTIStaticMethodsCallback;  
AbstractClass: Boolean)  
: TThoriumRTTIObjectType`

Visibility: protected

**1.33.34 TThoriumLibrary.RegisterSimpleMethod**

Declaration: `function RegisterSimpleMethod(const AName: String;  
const AFunction: TThoriumSimpleMethod;  
const AParameters: Array of TThoriumHostType;  
const AReturnType: TThoriumHostType)  
: TThoriumHostFunctionSimpleMethod`

Visibility: protected

**1.33.35 TThoriumLibrary.DeepFindHostType**

Declaration: `function DeepFindHostType(const AName: String) : TThoriumHostObjectType`

Visibility: public

### 1.33.36 TThoriumLibrary.DeepFindRTTIType

Declaration: `function DeepFindRTTIType(const AName: String) : TThoriumRTTIObjectType`

Visibility: public

### 1.33.37 TThoriumLibrary.DeepFindRTTITypeByClass

Declaration: `function DeepFindRTTITypeByClass(const AClass: TClass)  
: TThoriumRTTIObjectType`

Visibility: public

### 1.33.38 TThoriumLibrary.FindConstant

Declaration: `function FindConstant(const AName: String) : TThoriumLibraryConstant`

Visibility: public

### 1.33.39 TThoriumLibrary.FindHostFunction

Declaration: `function FindHostFunction(const AName: String)  
: TThoriumHostFunctionBase`

Visibility: public

### 1.33.40 TThoriumLibrary.FindHostType

Declaration: `function FindHostType(const AName: String) : TThoriumHostObjectType`

Visibility: public

### 1.33.41 TThoriumLibrary.FindProperty

Declaration: `function FindProperty(const AName: String) : TThoriumLibraryProperty`

Visibility: public

### 1.33.42 TThoriumLibrary.FindRTTIType

Declaration: `function FindRTTIType(const AName: String) : TThoriumRTTIObjectType`

Visibility: public

### 1.33.43 TThoriumLibrary.FindRTTITypeByClass

Declaration: `function FindRTTITypeByClass(const AClass: TClass)  
: TThoriumRTTIObjectType`

Visibility: public

### 1.33.44 TThoriumLibrary.IndexOfConstant

Declaration: `function IndexOfConstant(const AName: String) : Integer`

Visibility: public



#### **1.33.45 TThoriumLibrary.IndexOfHostFunction**

Declaration: `function IndexOfHostFunction(const AName: String) : Integer`

Visibility: public

#### **1.33.46 TThoriumLibrary.IndexOfHostType**

Declaration: `function IndexOfHostType(const AName: String) : Integer`

Visibility: public

#### **1.33.47 TThoriumLibrary.IndexOfProperty**

Declaration: `function IndexOfProperty(const AName: String) : Integer`

Visibility: public

#### **1.33.48 TThoriumLibrary.IndexOfRTTIType**

Declaration: `function IndexOfRTTIType(const AName: String) : Integer`

Visibility: public

#### **1.33.49 TThoriumLibrary.Constant**

Declaration: `Property Constant[AIndex: Integer]: TThoriumLibraryConstant`

Visibility: public

Access: Read

#### **1.33.50 TThoriumLibrary.ConstantCount**

Declaration: `Property ConstantCount : Integer`

Visibility: public

Access: Read

#### **1.33.51 TThoriumLibrary.HostFunction**

Declaration: `Property HostFunction[AIndex: Integer]: TThoriumHostFunctionBase`

Visibility: public

Access: Read

#### **1.33.52 TThoriumLibrary.HostFunctionCount**

Declaration: `Property HostFunctionCount : Integer`

Visibility: public

Access: Read

### 1.33.53 TThoriumLibrary.HostType

Declaration: `Property HostType[AIndex: Integer]: TThoriumHostObjectType`

Visibility: public

Access: Read

### 1.33.54 TThoriumLibrary.HostTypeCount

Declaration: `Property HostTypeCount : Integer`

Visibility: public

Access: Read

### 1.33.55 TThoriumLibrary.LibraryProperty

Declaration: `Property LibraryProperty[AIndex: Integer]: TThoriumLibraryProperty`

Visibility: public

Access: Read

### 1.33.56 TThoriumLibrary.LibraryPropertyCount

Declaration: `Property LibraryPropertyCount : Integer`

Visibility: public

Access: Read

### 1.33.57 TThoriumLibrary.RTTIType

Declaration: `Property RTTIType[AIndex: Integer]: TThoriumRTTIObjectType`

Visibility: public

Access: Read

### 1.33.58 TThoriumLibrary.RTTITypeCount

Declaration: `Property RTTITypeCount : Integer`

Visibility: public

Access: Read

## 1.34 TThoriumLibraryConstant

### 1.34.1 Description

This class implements a constant to be exported by a library.

## 1.35 TThoriumLibraryProperty

### 1.35.1 Description

Like a public variable in a Thorium module, a library can export properties which may even be changed by modules.

### 1.35.2 Method overview

Page	Property	Description
<a href="#">59</a>	Create	
<a href="#">59</a>	GetStatic	Determine whether the property is readonly.
<a href="#">59</a>	GetType	Get the type of the value.
<a href="#">59</a>	GetValue	Get the value of the property.
<a href="#">59</a>	SetValue	Set the value of the property.

### 1.35.3 TThoriumLibraryProperty.Create

Declaration: `constructor Create; Virtual`

Visibility: `public`

### 1.35.4 TThoriumLibraryProperty.GetValue

Synopsis: Get the value of the property.

Declaration: `procedure GetValue(const AThoriumValue: PThoriumValue); Virtual  
; Abstract`

Visibility: `protected`

Description: This function is supposed to write the value of the property to value pointed to by *AThoriumValue*.

### 1.35.5 TThoriumLibraryProperty.GetStatic

Synopsis: Determine whether the property is readonly.

Declaration: `function GetStatic : Boolean; Virtual; Abstract`

Visibility: `protected`

Description: Must return true when the value is read only.

### 1.35.6 TThoriumLibraryProperty.GetType

Synopsis: Get the type of the value.

Declaration: `function GetType : TThoriumType; Virtual; Abstract`

Visibility: `protected`

Description: Return the type of the value. This must not change during the whole program runtime.

### 1.35.7 TThoriumLibraryProperty.SetValue

Synopsis: Set the value of the property.

Declaration: `procedure SetValue(const AThoriumValue: PThoriumValue); Virtual  
; Abstract`

Visibility: protected

Description: This method should read the given value and assign it to the property.

## 1.36 TThoriumLibraryPropertyCallback

### 1.36.1 Description

This implementation is a virtual property. Any read or write from or to the property is redirected to callbacks allowing the owner to get the value from elsewhere.

### 1.36.2 Method overview

Page	Property	Description
<a href="#">60</a>	CalcHash	
<a href="#">60</a>	Create	
<a href="#">60</a>	GetStatic	
<a href="#">60</a>	GetType	
<a href="#">60</a>	GetValue	
<a href="#">61</a>	SetValue	

### 1.36.3 TThoriumLibraryPropertyCallback.Create

Declaration: `constructor Create; Override`

Visibility: public

### 1.36.4 TThoriumLibraryPropertyCallback.CalcHash

Declaration: `procedure CalcHash; Override`

Visibility: protected

### 1.36.5 TThoriumLibraryPropertyCallback.GetValue

Declaration: `procedure GetValue(const AThoriumValue: PThoriumValue); Override`

Visibility: protected

### 1.36.6 TThoriumLibraryPropertyCallback.GetStatic

Declaration: `function GetStatic : Boolean; Override`

Visibility: protected

### 1.36.7 TThoriumLibraryPropertyCallback.GetType

Declaration: `function GetType : TThoriumType; Override`

Visibility: protected

### 1.36.8 TThoriumLibraryPropertyCallback.SetValue

Declaration: `procedure SetValue(const AThoriumValue: PThoriumValue);`   Override

Visibility: `protected`

## 1.37 TThoriumLibraryPropertyDirect

### 1.37.1 Description

This class implements a library property using a private variable as storage without any control over the values assigned to it.

### 1.37.2 Method overview

Page	Property	Description
<a href="#">61</a>	CalcHash	
<a href="#">61</a>	Create	
<a href="#">61</a>	Destroy	
<a href="#">61</a>	GetStatic	
<a href="#">62</a>	GetType	
<a href="#">61</a>	GetValue	
<a href="#">62</a>	SetValue	

### 1.37.3 TThoriumLibraryPropertyDirect.Create

Declaration: `constructor Create;`   Override

Visibility: `public`

### 1.37.4 TThoriumLibraryPropertyDirect.Destroy

Declaration: `destructor Destroy;`   Override

Visibility: `public`

### 1.37.5 TThoriumLibraryPropertyDirect.CalcHash

Declaration: `procedure CalcHash;`   Override

Visibility: `protected`

### 1.37.6 TThoriumLibraryPropertyDirect.GetValue

Declaration: `procedure GetValue(const AThoriumValue: PThoriumValue);`   Override

Visibility: `protected`

### 1.37.7 TThoriumLibraryPropertyDirect.GetStatic

Declaration: `function GetStatic : Boolean;`   Override

Visibility: `protected`

### 1.37.8 TThoriumLibraryPropertyDirect.GetType

Declaration: `function GetType : TThoriumType; Override`

Visibility: `protected`

### 1.37.9 TThoriumLibraryPropertyDirect.SetValue

Declaration: `procedure SetValue(const AThoriumValue: PThoriumValue); Override`

Visibility: `protected`

## 1.38 TThoriumLibraryPropertyDirectSetCallback

### 1.38.1 Description

This class implements a library property using a private variable but providing also a callback which is called when a value is assigned to the property with the possibility to abort the assignment.

### 1.38.2 Method overview

Page	Property	Description
<a href="#">62</a>	Create	
<a href="#">62</a>	SetValue	

### 1.38.3 Property overview

Page	Property	Access	Description
<a href="#">62</a>	OnPropertySet	rw	

### 1.38.4 TThoriumLibraryPropertyDirectSetCallback.Create

Declaration: `constructor Create; Override`

Visibility: `public`

### 1.38.5 TThoriumLibraryPropertyDirectSetCallback.SetValue

Declaration: `procedure SetValue(const AThoriumValue: PThoriumValue); Override`

Visibility: `protected`

### 1.38.6 TThoriumLibraryPropertyDirectSetCallback.OnPropertySet

Declaration: `Property OnPropertySet : TThoriumOnPropertySetCallback`

Visibility: `public`

Access: `Read,Write`

## 1.39 TThoriumModule

### 1.39.1 Description

TThoriumModule represents one Thorium module. This class is capable of compiling a module from source and loading/saving it from/to a binary stream. It manages dependencies on other modules and libraries as well as the functions and variables published by the module.

### 1.39.2 Method overview

Page	Property	Description
<a href="#">64</a>	CalcHash	
<a href="#">65</a>	CompileFromStream	Compile Thorium script source code.
<a href="#">64</a>	Create	
<a href="#">64</a>	Destroy	
<a href="#">65</a>	Dump	Dump information to console
<a href="#">65</a>	DumpCodeStr	Format instructions and return string.
<a href="#">66</a>	DumpLibStr	Format library and return string.
<a href="#">66</a>	ExecuteMain	Deprecated?
<a href="#">64</a>	FillHeader	
<a href="#">64</a>	FindHostFunction	
<a href="#">64</a>	FindHostObjectType	
<a href="#">64</a>	FindHostRTTIType	
<a href="#">64</a>	FindLibraryConstant	
<a href="#">65</a>	FindLibraryProperty	
<a href="#">66</a>	FindPublicFunction	Return public function.
<a href="#">66</a>	IndexOfPublicFunction	Return index of public function.
<a href="#">65</a>	InternalLoadFromStream	
<a href="#">65</a>	InternalSaveToStream	
<a href="#">66</a>	IsTypeCompatible	Move to private?
<a href="#">66</a>	IsTypeOperationAvailable	Move to private?
<a href="#">67</a>	LoadFromStream	Load module from stream
<a href="#">67</a>	SaveToStream	Save module in a binary format.

### 1.39.3 Property overview

Page	Property	Access	Description
<a href="#">67</a>	Compiled	r	Whether the module is ready for use.
<a href="#">67</a>	Compress	rw	Whether to compress the data.
<a href="#">67</a>	InstructionCount	r	Amount of instructions.
<a href="#">68</a>	LastCompilerError	r	Last compiler error.
<a href="#">68</a>	LibraryString	r	Access to library strings.
<a href="#">68</a>	LibraryStringCount	r	Amount of library'd strings
<a href="#">68</a>	Name	r	Name of the module.
<a href="#">68</a>	OptimizedInstructions	r	Amount of removed instructions.
<a href="#">69</a>	PublicFunction	r	Access to public functions.
<a href="#">69</a>	PublicFunctionCount	r	Amount of public functions.
<a href="#">69</a>	PublicVariable	r	Access to public variables
<a href="#">69</a>	PublicVariableCount	r	Amount of public variables.
<a href="#">69</a>	Thorium	r	Owning thorium engine.

#### 1.39.4 TThoriumModule.Create

Declaration: `constructor Create(AThorium: TThorium); Virtual`  
`constructor Create(AThorium: TThorium; AName: String)`

Visibility: default

#### 1.39.5 TThoriumModule.Destroy

Declaration: `destructor Destroy; Override`

Visibility: default

#### 1.39.6 TThoriumModule.CalcHash

Declaration: `procedure CalcHash; Override`

Visibility: protected

#### 1.39.7 TThoriumModule.FillHeader

Declaration: `procedure FillHeader(out Header: TThoriumModuleHeader); Virtual`

Visibility: protected

#### 1.39.8 TThoriumModule.FindHostFunction

Declaration: `function FindHostFunction(const AName: String)`  
`: TThoriumHostFunctionBase`

Visibility: protected

#### 1.39.9 TThoriumModule.FindHostObjectType

Declaration: `function FindHostObjectType(const AName: String)`  
`: TThoriumHostObjectType`

Visibility: protected

#### 1.39.10 TThoriumModule.FindHostRTTIType

Declaration: `function FindHostRTTIType(const AName: String) : TThoriumRTTIObjectType`

Visibility: protected

#### 1.39.11 TThoriumModule.FindLibraryConstant

Declaration: `function FindLibraryConstant(const AName: String)`  
`: TThoriumLibraryConstant`

Visibility: protected



### 1.39.12 TThoriumModule.FindLibraryProperty

Declaration: `function FindLibraryProperty(const AName: String)  
: TThoriumLibraryProperty`

Visibility: protected

### 1.39.13 TThoriumModule.InternalLoadFromStream

Declaration: `procedure InternalLoadFromStream(Stream: TStream;  
const Header: TThoriumModuleHeader)  
; Virtual`

Visibility: protected

### 1.39.14 TThoriumModule.InternalSaveToStream

Declaration: `procedure InternalSaveToStream(Stream: TStream;  
const Header: TThoriumModuleHeader)  
; Virtual`

Visibility: protected

### 1.39.15 TThoriumModule.CompileFromStream

Synopsis: Compile Thorium script source code.

Declaration: `function CompileFromStream(SourceStream: TStream;  
Flags: TThoriumCompilerFlags) : Boolean`

Visibility: public

Description: This method clears the whole module and tries to compile the code delivered with SourceStream using the Flags passed as second parameter. It returns whether the compilation was successful or not. In the latter case, the module is cleared again to bring it in a stable state.

### 1.39.16 TThoriumModule.Dump

Synopsis: Dump information to console

Declaration: `procedure Dump`

Visibility: public

Description: This dumps a lot of information about the module to stdout, like the whole instruction array, exports and dependencies.

### 1.39.17 TThoriumModule.DumpCodeStr

Synopsis: Format instructions and return string.

Declaration: `function DumpCodeStr : String`

Visibility: public

Description: This puts the instructions of the module in a more or less human readable form and returns them as a string predestined to be printed to a console.

### 1.39.18 TThoriumModule.DumpLibStr

Synopsis: Format library and return string.

Declaration: `function DumpLibStr : String`

Visibility: public

Description: This formats the string library in a human readable format and returns it as a string.

### 1.39.19 TThoriumModule.ExecuteMain

Synopsis: Deprecated?

Declaration: `procedure ExecuteMain`

Visibility: public

### 1.39.20 TThoriumModule.FindPublicFunction

Synopsis: Return public function.

Declaration: `function FindPublicFunction(const AName: String) : TThoriumFunction`

Visibility: public

Description: Searches for a public function with the given name in the module and returns it if found or nil if not.

### 1.39.21 TThoriumModule.IndexOfPublicFunction

Synopsis: Return index of public function.

Declaration: `function IndexOfPublicFunction(const AName: String) : Integer`

Visibility: public

Description: Searches for a public function with the given name in the module and returns its index if found or -1 if not.

### 1.39.22 TThoriumModule.IsTypeCompatible

Synopsis: Move to private?

Declaration: `function IsTypeCompatible(Value1: TThoriumType; Value2: TThoriumType;  
Operation: TThoriumOperation;  
out ResultType: TThoriumType) : Boolean`

Visibility: public

### 1.39.23 TThoriumModule.IsTypeOperationAvailable

Synopsis: Move to private?

Declaration: `function IsTypeOperationAvailable(Value: TThoriumType;  
Operation: TThoriumOperation;  
out ResultType: TThoriumType) : Boolean`

Visibility: public

### 1.39.24 TThoriumModule.LoadFromStream

Synopsis: Load module from stream

Declaration: `procedure LoadFromStream(Stream: TStream)`

Visibility: public

Description: This method loads a whole module from a stream. It is assumed that the module is in the binary format `SaveToStream` generates. References to other modules, libraries, types, methods or functions have been encoded and are decoded by this method and verified. If any verification fails, this method throws an exception and leaves the module in an empty state.

### 1.39.25 TThoriumModule.SaveToStream

Synopsis: Save module in a binary format.

Declaration: `procedure SaveToStream(Stream: TStream)`

Visibility: public

Description: This method saves the complete module in a binary format. References to other modules, libraries, functions and types are encoded so that they can be verified when loading the module again. If compression is enabled and supported, the module instructions will be compressed using the zlib library.

### 1.39.26 TThoriumModule.Compiled

Synopsis: Whether the module is ready for use.

Declaration: `Property Compiled : Boolean`

Visibility: public

Access: Read

Description: This property shows whether the module is ready for use - i.e. has been compiled or loaded from a binary.

### 1.39.27 TThoriumModule.Compress

Synopsis: Whether to compress the data.

Declaration: `Property Compress : Boolean`

Visibility: public

Access: Read,Write

Description: This switch defines whether the module will be compressed when it gets saved to a stream. Since the Thorium instructions are optimized for speed rather than size they contain a lot of unused space and zeros, which can be very good compressed by the zlib algorithm.

### 1.39.28 TThoriumModule.InstructionCount

Synopsis: Amount of instructions.

Declaration: `Property InstructionCount : Integer`

Visibility: public

Access: Read

Description: The amount of instructions in this module.

### 1.39.29 TThoriumModule.LastCompilerError

Synopsis: Last compiler error.

Declaration: `Property LastCompilerError : String`

Visibility: public

Access: Read

Description: This is the last error thrown by the compiler. Only set to anything else than an empty string if compilation failed at least once.

### 1.39.30 TThoriumModule.LibraryString

Synopsis: Access to library strings.

Declaration: `Property LibraryString[Index: Integer]: String`

Visibility: public

Access: Read

Description: Constant strings which occur in the source code of Thorium scripts are saved in a so called library. Access to those from the instructions is then only handled by indices to speed things up. You can access the strings stored in the library using this property.

### 1.39.31 TThoriumModule.LibraryStringCount

Synopsis: Amount of library'd strings

Declaration: `Property LibraryStringCount : Integer`

Visibility: public

Access: Read

Description: This is the amount of strings contained in the module library. See [TThoriumModule.LibraryString \(68\)](#)LibraryString

### 1.39.32 TThoriumModule.Name

Synopsis: Name of the module.

Declaration: `Property Name : String`

Visibility: public

Access: Read

Description: The name of the module under which it can also be referenced in other modules.

### 1.39.33 TThoriumModule.OptimizedInstructions

Synopsis: Amount of removed instructions.

Declaration: `Property OptimizedInstructions : LongInt`

Visibility: public

Access: Read

Description: This is the amount of instructions which have been removed by the internal optimizer of Thorium. The optimizer makes patterns which are known to be created by the compiler or certain code phrases more performant (and sometimes, it even breaks the whole code :)).

### 1.39.34 TThoriumModule.PublicFunction

Synopsis: Access to public functions.

Declaration: `Property PublicFunction[Index: Integer]: TThoriumFunction`

Visibility: public

Access: Read

Description: Using this property one can access the public functions declared in the module. The amount of published functions can be queried using `TThoriumModule.PublicFunctionCount` (69)`PublicFunctionCount`

### 1.39.35 TThoriumModule.PublicFunctionCount

Synopsis: Amount of public functions.

Declaration: `Property PublicFunctionCount : Integer`

Visibility: public

Access: Read

Description: This is the amount of public functions declared in this module.

### 1.39.36 TThoriumModule.PublicVariable

Synopsis: Access to public variables

Declaration: `Property PublicVariable[Index: Integer]: TThoriumVariable`

Visibility: public

Access: Read

Description: This property provides access to variables made public by the module.

### 1.39.37 TThoriumModule.PublicVariableCount

Synopsis: Amount of public variables.

Declaration: `Property PublicVariableCount : Integer`

Visibility: public

Access: Read

Description: This property reflects the amount of variables which have been made public by the module.

### 1.39.38 TThoriumModule.Thorium

Synopsis: Owning thorium engine.

Declaration: `Property Thorium : TThorium`

Visibility: public

Access: Read

Description: The Thorium engine which owns the module.

## 1.40 TThoriumParameters

### 1.40.1 Description

This class is a container class to hold a list of Thorium types. It is used as parameter list in function specifications of Thorium functions. For host environment functions there is a separate class.

### 1.40.2 Method overview

Page	Property	Description
<a href="#">70</a>	AddParameter	
<a href="#">70</a>	Clear	
<a href="#">70</a>	Create	
<a href="#">70</a>	Destroy	
<a href="#">71</a>	Duplicate	
<a href="#">71</a>	GetParameterSpec	
<a href="#">71</a>	LoadFromStream	
<a href="#">70</a>	RemoveParameter	
<a href="#">71</a>	SaveToStream	

### 1.40.3 Property overview

Page	Property	Access	Description
<a href="#">71</a>	Count	r	

### 1.40.4 TThoriumParameters.Create

Declaration: `constructor Create`

Visibility: `default`

### 1.40.5 TThoriumParameters.Destroy

Declaration: `destructor Destroy; Override`

Visibility: `default`

### 1.40.6 TThoriumParameters.AddParameter

Declaration: `function AddParameter : PThoriumType`

Visibility: `protected`

### 1.40.7 TThoriumParameters.Clear

Declaration: `procedure Clear`

Visibility: `protected`

### 1.40.8 TThoriumParameters.RemoveParameter

Declaration: `procedure RemoveParameter(const Index: Integer)`

Visibility: `protected`

### 1.40.9 TThoriumParameters.Duplicate

Declaration: `function Duplicate : TThoriumParameters`

Visibility: `public`

### 1.40.10 TThoriumParameters.GetParameterSpec

Declaration: `procedure GetParameterSpec(const Index: Integer;  
out ParamSpec: TThoriumType)`

Visibility: `public`

### 1.40.11 TThoriumParameters.LoadFromStream

Declaration: `procedure LoadFromStream(Stream: TStream)`

Visibility: `public`

### 1.40.12 TThoriumParameters.SaveToStream

Declaration: `procedure SaveToStream(Stream: TStream)`

Visibility: `public`

### 1.40.13 TThoriumParameters.Count

Declaration: `Property Count : Integer`

Visibility: `public`

Access: `Read`

## 1.41 TThoriumPersistent

### 1.41.1 Description

The easiest way to publish a class to Thorium is deriving it from `TThoriumPersistent`. This keeps you from building your own implementation of the methods specified in the `IThoriumPersistent` (9) interface.

### 1.41.2 Method overview

Page	Property	Description
<a href="#">72</a>	<code>\_AddRef</code>	Increase reference count.
<a href="#">72</a>	<code>\_Release</code>	Decrease reference count.
<a href="#">72</a>	<code>Create</code>	
<a href="#">73</a>	<code>DisableHostControl</code>	Unset the host controlled flag.
<a href="#">73</a>	<code>EnableHostControl</code>	Set the host controlled flag
<a href="#">72</a>	<code>FreeReference</code>	Release a reference.
<a href="#">73</a>	<code>GetMethodList</code>	Determine method list.
<a href="#">73</a>	<code>GetReference</code>	Increase the reference counter and return the instance.
<a href="#">73</a>	<code>GetReferenceCount</code>	Get the amount of references.
<a href="#">72</a>	<code>GetStaticMethodList</code>	Determine the list of static methods.
<a href="#">72</a>	<code>QueryInterface</code>	Query for an interface.

### 1.41.3 TThoriumPersistent.Create

Declaration: `constructor Create`

Visibility: `public`

### 1.41.4 TThoriumPersistent.FreeReference

Synopsis: Release a reference.

Declaration: `procedure FreeReference`

Visibility: `public`

Description: Decreases the reference counter by one and, if applicable, frees the instance.

See also: [TThoriumPersistent.GetReference \(73\)](#), [TThoriumPersistent.\\_Release \(72\)](#)

### 1.41.5 TThoriumPersistent.\_AddRef

Synopsis: Increase reference count.

Declaration: `function _AddRef : LongInt`

Visibility: `protected`

Description: IUnknown implementation to increase the reference counter of the instance.

See also: [TThoriumPersistent.GetReference \(73\)](#), [TThoriumPersistent.\\_Release \(72\)](#)

### 1.41.6 TThoriumPersistent.\_Release

Synopsis: Decrease reference count.

Declaration: `function _Release : LongInt`

Visibility: `protected`

Description: IUnknown implementation to decrease the reference counter.

See also: [TThoriumPersistent.FreeReference \(72\)](#), [TThoriumPersistent.\\_AddRef \(72\)](#)

### 1.41.7 TThoriumPersistent.QueryInterface

Synopsis: Query for an interface.

Declaration: `function QueryInterface(const IID: TGuid;out Obj) : LongInt`

Visibility: `protected`

Description: Default implementation of QueryInterface from IUnknown.

### 1.41.8 TThoriumPersistent.GetStaticMethodList

Synopsis: Determine the list of static methods.

Declaration: 

```
procedure GetStaticMethodList(Sender: TThoriumRTTIObjectType;
                             var Methods: TThoriumRTTIStaticMethods)
; Virtual
```

Visibility: `protected`

Description: This is called by the constructor of an [TThoriumRTTIObjectType \(75\)](#) instance to fetch the list of static methods a class type supports. You should pass the entries in the array given.

See also: [TThoriumPersistent.GetMethodList \(73\)](#), [TThoriumRTTIObjectType \(75\)](#)



### 1.41.9 TThoriumPersistent.GetMethodList

Synopsis: Determine method list.

Declaration: `procedure GetMethodList (Sender: TThoriumRTTIObjectType;  
var Methods: TThoriumRTTIMethods); Virtual`

Visibility: protected

Description: This method is called by the constructor of a TThoriumRTTIObjectType (75) instance to determine which methods are to be published by the class. The methods are to be written to the dynamic array passed.

See also: TThoriumPersistent.GetStaticMethodList (72), TThoriumRTTIObjectType (75)

### 1.41.10 TThoriumPersistent.EnableHostControl

Synopsis: Set the host controlled flag

Declaration: `procedure EnableHostControl`

Visibility: public

Description: Sets the instance to be host controlled. This wants to say that it will never be freed when the reference counter reaches zero. This is for example called when the instance is assigned to a parameter or property which has been flagged as storing.

See also: TThoriumPersistent.DisableHostControl (73), TThoriumHostObjectType.GetPropertyStoring (41)

### 1.41.11 TThoriumPersistent.DisableHostControl

Synopsis: Unset the host controlled flag.

Declaration: `procedure DisableHostControl`

Visibility: public

Description: Removes the host controlled flag from the instance and thus let it free if the reference counter reaches zero.

See also: TThoriumPersistent.EnableHostControl (73)

### 1.41.12 TThoriumPersistent.GetReference

Synopsis: Increase the reference counter and return the instance.

Declaration: `function GetReference : TObject`

Visibility: public

Description: The method increases the reference counter for this instance and returns the instance too.

See also: TThoriumPersistent.\_Add (71)

### 1.41.13 TThoriumPersistent.GetReferenceCount

Synopsis: Get the amount of references.

Declaration: `function GetReferenceCount : LongInt`

Visibility: public

Description: Return the amount of known references to this instance. The virtual reference created by the host controlled flag is not counted.

## 1.42 TThoriumPublicValue

### 1.42.1 Description

This is an abstract baseclass to describe identifiers (mostly public) like variables and functions which are declared in a Thorium script.

### 1.42.2 Method overview

Page	Property	Description
<a href="#">74</a>	Create	
<a href="#">74</a>	LoadFromStream	Load specification from stream.
<a href="#">74</a>	SaveToStream	Save specification to stream.

### 1.42.3 Property overview

Page	Property	Access	Description
<a href="#">74</a>	Module	r	Owning module.
<a href="#">75</a>	Name	r	Identifier name.

### 1.42.4 TThoriumPublicValue.Create

**Declaration:** `constructor Create(AModule: TThoriumModule); Virtual`

**Visibility:** default

### 1.42.5 TThoriumPublicValue.LoadFromStream

**Synopsis:** Load specification from stream.

**Declaration:** `procedure LoadFromStream(Stream: TStream); Virtual`

**Visibility:** public

**Description:** Loads the specification of this identifier from a given stream.

**See also:** [TThoriumPublicValue.SaveToStream \(74\)](#)

### 1.42.6 TThoriumPublicValue.SaveToStream

**Synopsis:** Save specification to stream.

**Declaration:** `procedure SaveToStream(Stream: TStream); Virtual`

**Visibility:** public

**Description:** Saves the specification of this identifier to a stream. References to other identifiers are declared by using their name and their hash to identify them uniquely.

**See also:** [TThoriumPublicValue.LoadFromStream \(74\)](#)

### 1.42.7 TThoriumPublicValue.Module

**Synopsis:** Owning module.

**Declaration:** `Property Module : TThoriumModule`

**Visibility:** public

Access: Read

Description: The module which owns this identifier.

### 1.42.8 TThoriumPublicValue.Name

Synopsis: Identifier name.

Declaration: `Property Name : String`

Visibility: public

Access: Read

Description: The name of this identifier.

## 1.43 TThoriumRTTIObjectType

### 1.43.1 Description

This uses `TThoriumHostObjectType` (35) as ancestor class and implements a generic host object type which is used to represent Pascal classes using all available RTTI information. This speciality of the class is hard coded and there are many places in Thorium where special code is used for type instances of this class to assure you do not need to derive a class from this one for each type you want to publish to Thorium.

### 1.43.2 Method overview

Page	Property	Description
<a href="#">77</a>	<code>CalcHash</code>	Create an instance.
<a href="#">76</a>	<code>Create</code>	
<a href="#">76</a>	<code>Destroy</code>	
<a href="#">76</a>	<code>DisposeValue</code>	
<a href="#">76</a>	<code>DuplicateValue</code>	
<a href="#">77</a>	<code>FieldID</code>	
<a href="#">78</a>	<code>FieldType</code>	
<a href="#">77</a>	<code>FindMethod</code>	
<a href="#">78</a>	<code>GetField</code>	
<a href="#">76</a>	<code>GetNeededMemoryAmount</code>	
<a href="#">78</a>	<code>GetPropertyStoring</code>	
<a href="#">77</a>	<code>HasFields</code>	
<a href="#">77</a>	<code>HasIndicies</code>	
<a href="#">77</a>	<code>HasStaticFields</code>	
<a href="#">77</a>	<code>IsTypeCompatible</code>	
<a href="#">77</a>	<code>IsTypeOperationAvailable</code>	
<a href="#">79</a>	<code>NewNativeCallMethod</code>	Helper function.
<a href="#">79</a>	<code>NewNativeCallStaticFunction</code>	Helper function.
<a href="#">79</a>	<code>NewNativeCallStaticMethod</code>	Helper function.
<a href="#">78</a>	<code>SetField</code>	
<a href="#">78</a>	<code>SetPropertyStoring</code>	Set whether a property is storing.
<a href="#">78</a>	<code>StaticFieldID</code>	
<a href="#">78</a>	<code>StaticFieldType</code>	

### 1.43.3 Property overview

Page	Property	Access	Description
<a href="#">80</a>	BaseClass	r	Class represented by this type.

### 1.43.4 TThoriumRTTIObjectType.Create

Synopsis: Create an instance.

**Declaration:** `constructor Create(ALibrary: TThoriumLibrary); Override`  
`constructor Create(ALibrary: TThoriumLibrary;`  
`ABaseClass: TThoriumPersistentClass;`  
`AbstractClass: Boolean)`  
`constructor Create(ALibrary: TThoriumLibrary; ABaseClass: TClass;`  
`MethodCallback: TThoriumRTTIMethodsCallback;`  
`StaticMethodCallback: TThoriumRTTIStaticMethodsCallback;`  
`AbstractClass: Boolean)`

Visibility: default

**Description:** You must not use the constructor inherited from the parent class type to create an instance of this class. To publish a class type to Thorium, you may either derive it from TThoriumPersistent (71) or implement IThoriumPersistent (9) interface in it. Depending on which method you choose, you have to choose a different constructor. If you use TThoriumPersistent directly, you should use the first variant of the constructor which expects the class type as its second parameter. Otherwise you must use the second variant and specify the wanted methods accordingly. The first constructor internally calls the second one. To specify a base class which is unable to be ever instantiated (e.g. when you publish the TStream-class tree, you cannot alter TStream to implement IThoriumPersistent. You would derive a class from those streams you want to have published to Thorium implementing IThoriumPersistent and specify TStream as an abstract class).

### 1.43.5 TThoriumRTTIObjectType.Destroy

**Declaration:** `destructor Destroy; Override`

Visibility: default

### 1.43.6 TThoriumRTTIObjectType.GetNeededMemoryAmount

**Declaration:** `function GetNeededMemoryAmount : TThoriumSizeInt; Override`

Visibility: protected

### 1.43.7 TThoriumRTTIObjectType.DuplicateValue

**Declaration:** `function DuplicateValue(const AValue: TThoriumHostObjectTypeValue)`  
`: TThoriumValue; Override`

Visibility: protected

### 1.43.8 TThoriumRTTIObjectType.DisposeValue

**Declaration:** `procedure DisposeValue(var AValue: TThoriumHostObjectTypeValue)`  
`; Override`

Visibility: protected

**1.43.9 TThoriumRTTIObjectType.IsTypeCompatible**

Declaration: function IsTypeCompatible(const Value1: TThoriumType;  
                                       const Value2: TThoriumType;  
                                       const Operation: TThoriumOperation;  
                                       out ResultType: TThoriumType) : Boolean  
                                       ; Override

Visibility: protected

**1.43.10 TThoriumRTTIObjectType.IsTypeOperationAvailable**

Declaration: function IsTypeOperationAvailable(const Value: TThoriumType;  
   const Operation: TThoriumOperation;  
   out ResultType: TThoriumType) : Boolean  
   ; Override

Visibility: protected

**1.43.11 TThoriumRTTIObjectType.HasFields**

Declaration: function HasFields : Boolean; Override

Visibility: protected

**1.43.12 TThoriumRTTIObjectType.HasStaticFields**

Declaration: function HasStaticFields : Boolean; Override

Visibility: protected

**1.43.13 TThoriumRTTIObjectType.HasIndices**

Declaration: function HasIndices : Boolean; Override

Visibility: protected

**1.43.14 TThoriumRTTIObjectType.FindMethod**

Declaration: function FindMethod(const AMethodName: String) : TThoriumHostMethodBase  
                                       ; Override

Visibility: protected

**1.43.15 TThoriumRTTIObjectType.CalcHash**

Declaration: procedure CalcHash; Override

Visibility: protected

**1.43.16 TThoriumRTTIObjectType.FieldID**

Declaration: function FieldID(const FieldIdent: String; out ID: QWord) : Boolean  
                                       ; Override

Visibility: public

### 1.43.17 TThoriumRTTIObjectType.StaticFieldID

Declaration: `function StaticFieldID(const FieldIdent: String; out ID: QWord) : Boolean  
; Override`

Visibility: public

### 1.43.18 TThoriumRTTIObjectType.FieldType

Declaration: `function FieldType(const AFieldID: QWord;  
out ResultType: TThoriumTableEntry) : Boolean  
; Override`

Visibility: public

### 1.43.19 TThoriumRTTIObjectType.StaticFieldType

Declaration: `function StaticFieldType(const AFieldID: QWord;  
out ResultType: TThoriumTableEntry) : Boolean  
; Override`

Visibility: public

### 1.43.20 TThoriumRTTIObjectType.GetField

Declaration: `function GetField(const AInstance: TThoriumValue; const AFieldID: QWord)  
: TThoriumValue; Override`

Visibility: public

### 1.43.21 TThoriumRTTIObjectType.SetField

Declaration: `procedure SetField(const AInstance: TThoriumValue; const AFieldID: QWord;  
const NewValue: TThoriumValue); Override`

Visibility: public

### 1.43.22 TThoriumRTTIObjectType.GetPropertyStoring

Declaration: `function GetPropertyStoring(const PropertyName: String) : Boolean  
function GetPropertyStoring(const PropInfo: PPropInfo) : Boolean  
function GetPropertyStoring(const AFieldID: QWord) : Boolean; Override`

Visibility: public

### 1.43.23 TThoriumRTTIObjectType.SetPropertyStoring

Synopsis: Set whether a property is storing.

Declaration: `procedure SetPropertyStoring(const PropertyName: String;  
IsStoring: Boolean)  
procedure SetPropertyStoring(const PropInfo: PPropInfo;  
IsStoring: Boolean)`

Visibility: public

Description: Using these methods you can set the storing bit of any property of the class represented by this type implementation. For more infos about the storing bit see `TThoriumHostObjectType.GetPropertyStoring` ([41](#)).

#### 1.43.24 TThoriumRTTIObjectType.NewNativeCallMethod

Synopsis: Helper function.

Declaration: 

```
function NewNativeCallMethod(const AName: String;
                             const ACodePointer: Pointer;
                             const AParameters: Array of TThoriumHostType;
                             const AReturnType: TThoriumHostType;
                             const ACallingConvention: TThoriumNativeCallingConvention
                             : TThoriumHostMethodNativeCall
```

Visibility: public

Description: This is an helper function which creates a new instance of a native call method. The instance is returned, but not registered with the type. This is to be used in the callbacks given to the constructor or in the methods which determine the methods published by a type in `TThoriumPersistent` ([71](#)).

#### 1.43.25 TThoriumRTTIObjectType.NewNativeCallStaticMethod

Synopsis: Helper function.

Declaration: 

```
function NewNativeCallStaticMethod(const AName: String;
                                   const ACodePointer: Pointer;
                                   const ADataPointer: Pointer;
                                   const AParameters: Array of TThoriumHostType;
                                   const AReturnType: TThoriumHostType;
                                   const ACallingConvention: TThoriumNativeCallingConvention
                                   : TThoriumHostMethodAsFunctionNativeCall
```

Visibility: public

Description: This is an helper function which creates a new instance of a native call static (= class) method. The instance is returned, but not registered with the type. This is to be used in the callbacks given to the constructor or in the methods which determine the methods published by a type in `TThoriumPersistent` ([71](#)).

#### 1.43.26 TThoriumRTTIObjectType.NewNativeCallStaticFunction

Synopsis: Helper function.

Declaration: 

```
function NewNativeCallStaticFunction(const AName: String;
                                     const ACodePointer: Pointer;
                                     const AParameters: Array of TThoriumHostType;
                                     const AReturnType: TThoriumHostType;
                                     const ACallingConvention: TThoriumNativeCallingConvention
                                     : TThoriumHostFunctionNativeCall
```

Visibility: public

Description: This is an helper function which creates a new instance of a native call function. The instance is returned, but not registered with the type. This is to be used in the callbacks given to the constructor or in the methods which determine the methods published by a type in `TThoriumPersistent` ([71](#)).

### 1.43.27 TThoriumRTTIObjectType.BaseClass

Synopsis: Class represented by this type.

Declaration: `Property BaseClass : TClass`

Visibility: public

Access: Read

Description: This is the class which is represented by this type.

## 1.44 TThoriumScanner

### 1.44.1 Description

Class for internal use - to be described later.

### 1.44.2 Method overview

Page	Property	Description
<a href="#">80</a>	Create	
<a href="#">80</a>	Destroy	
<a href="#">80</a>	ScanForSymbol	

### 1.44.3 Property overview

Page	Property	Access	Description
<a href="#">81</a>	CurrentLine	r	
<a href="#">81</a>	CurrentStr	r	
<a href="#">80</a>	CurrentSym	r	

### 1.44.4 TThoriumScanner.Create

Declaration: `constructor Create(AInputString: String)`  
`constructor Create(AInputStream: TStream)`

Visibility: default

### 1.44.5 TThoriumScanner.Destroy

Declaration: `destructor Destroy; Override`

Visibility: default

### 1.44.6 TThoriumScanner.ScanForSymbol

Declaration: `procedure ScanForSymbol(var Sym: TThoriumSymbol; var Str: String)`

Visibility: protected

### 1.44.7 TThoriumScanner.CurrentSym

Declaration: `Property CurrentSym : TThoriumSymbol`

Visibility: public



Access: Read

### 1.44.8 TThoriumScanner.CurrentStr

Declaration: `Property CurrentStr : String`

Visibility: public

Access: Read

### 1.44.9 TThoriumScanner.CurrentLine

Declaration: `Property CurrentLine : Integer`

Visibility: public

Access: Read

## 1.45 TThoriumStack

### 1.45.1 Description

Class for internal use - to be described later.

### 1.45.2 Method overview

Page	Property	Description
<a href="#">82</a>	ClearStack	
<a href="#">81</a>	Create	
<a href="#">82</a>	Destroy	
<a href="#">82</a>	FastGetStackEntry	
<a href="#">82</a>	GetTop	
<a href="#">82</a>	GetTopStackEntry	
<a href="#">82</a>	Pop	
<a href="#">82</a>	PopTop	
<a href="#">82</a>	Prealloc	
<a href="#">82</a>	Push	

### 1.45.3 Property overview

Page	Property	Access	Description
<a href="#">83</a>	Capacity	rw	
<a href="#">83</a>	EntryCount	r	
<a href="#">83</a>	StackEntry	r	

### 1.45.4 TThoriumStack.Create

Declaration: `constructor Create`

Visibility: default

### 1.45.5 TThoriumStack.Destroy

Declaration: `destructor Destroy; Override`

Visibility: `default`

### 1.45.6 TThoriumStack.FastGetStackEntry

Declaration: `function FastGetStackEntry (ScopeRoot: Integer; Index: Integer)  
: PThoriumStackEntry`

Visibility: `public`

### 1.45.7 TThoriumStack.GetTopStackEntry

Declaration: `function GetTopStackEntry : PThoriumStackEntry`

Visibility: `public`

### 1.45.8 TThoriumStack.GetTop

Declaration: `function GetTop (Offset: Integer) : PThoriumStackEntry`

Visibility: `public`

### 1.45.9 TThoriumStack.Prealloc

Declaration: `function Prealloc : PThoriumStackEntry`

Visibility: `public`

### 1.45.10 TThoriumStack.Push

Declaration: `function Push : PThoriumStackEntry  
procedure Push (AEntry: PThoriumStackEntry)`

Visibility: `public`

### 1.45.11 TThoriumStack.Pop

Declaration: `procedure Pop (Amount: Integer; FreeValues: Boolean)`

Visibility: `public`

### 1.45.12 TThoriumStack.PopTop

Declaration: `function PopTop : PThoriumStackEntry`

Visibility: `public`

### 1.45.13 TThoriumStack.ClearStack

Declaration: `procedure ClearStack`

Visibility: `public`

### 1.45.14 TThoriumStack.StackEntry

Declaration: `Property StackEntry[ScopeRoot: Integer; Index: Integer]: PThoriumStackEntry`

Visibility: public

Access: Read

### 1.45.15 TThoriumStack.EntryCount

Declaration: `Property EntryCount : Integer`

Visibility: public

Access: Read

### 1.45.16 TThoriumStack.Capacity

Declaration: `Property Capacity : Integer`

Visibility: public

Access: Read, Write

## 1.46 TThoriumVariable

### 1.46.1 Description

This class represents a (probably public) variable declared in a Thorium script.

### 1.46.2 Method overview

Page	Property	Description
<a href="#">83</a>	Create	Creates an instance.
<a href="#">83</a>	LoadFromStream	Loads specification from stream.
<a href="#">84</a>	SaveToStream	Saves specification to stream.

### 1.46.3 Property overview

Page	Property	Access	Description
<a href="#">84</a>	IsStatic	r	Whether the value is static.
<a href="#">84</a>	StackPosition	r	Position of the variable on the stack.
<a href="#">84</a>	TypeSpec	r	Type of the variable.

### 1.46.4 TThoriumVariable.Create

Synopsis: Creates an instance.

Declaration: `constructor Create(AModule: TThoriumModule);`   Override

Visibility: default

### 1.46.5 TThoriumVariable.LoadFromStream

Synopsis: Loads specification from stream.

**Declaration:** `procedure LoadFromStream(Stream: TStream); Override`

**Visibility:** public

### 1.46.6 TThoriumVariable.SaveToStream

**Synopsis:** Saves specification to stream.

**Declaration:** `procedure SaveToStream(Stream: TStream); Override`

**Visibility:** public

### 1.46.7 TThoriumVariable.IsStatic

**Synopsis:** Whether the value is static.

**Declaration:** `Property IsStatic : Boolean`

**Visibility:** public

**Access:** Read

**Description:** If this is true, no changes can be made to the value of this variable. It has been declared as static and most references have been replaced by the compiler with the actual value to optimize the code.

### 1.46.8 TThoriumVariable.StackPosition

**Synopsis:** Position of the variable on the stack.

**Declaration:** `Property StackPosition : Integer`

**Visibility:** public

**Access:** Read

**Description:** This property tells where on the module local part of the stack the variable can be found.

### 1.46.9 TThoriumVariable.TypeSpec

**Synopsis:** Type of the variable.

**Declaration:** `Property TypeSpec : TThoriumType`

**Visibility:** public

**Access:** Read

**Description:** The type specification of the variable.

## 1.47 TThoriumVirtualMachine

### 1.47.1 Description

This class is responsible to execute the bytecode generated by the Thorium compiler. It also keeps track of the stack. While a virtual machine is attached to a Thorium engine, no changes should be made to ensure the consistency of the virtual machine state.

### 1.47.2 Method overview

Page	Property	Description
<a href="#">85</a>	Create	
<a href="#">85</a>	Destroy	
<a href="#">85</a>	DumpStack	Dump stack to stdout.
<a href="#">85</a>	Execute	Execute instructions.
<a href="#">85</a>	GetStack	Return the stack of the virtual machine.

### 1.47.3 TThoriumVirtualMachine.Create

Declaration: constructor `Create(AThorium: TThorium)`

Visibility: default

### 1.47.4 TThoriumVirtualMachine.Destroy

Declaration: destructor `Destroy; Override`

Visibility: default

### 1.47.5 TThoriumVirtualMachine.DumpStack

Synopsis: Dump stack to stdout.

Declaration: procedure `DumpStack`

Visibility: public

Description: Dumps the whole stack to stdout. Be aware that it also tries to read values and thus may crash if the stack is in an inconsistent state (which should of course not occur normally).

### 1.47.6 TThoriumVirtualMachine.GetStack

Synopsis: Return the stack of the virtual machine.

Declaration: function `GetStack : TThoriumStack`

Visibility: public

Description: Returns the stack which is being used by this virtual machine. Handle with care. You should not attempt to make any changes to the stack by yourself.

### 1.47.7 TThoriumVirtualMachine.Execute

Synopsis: Execute instructions.

Declaration: procedure `Execute(StartModuleIndex: Integer; StartInstruction: Integer;  
CreateDefaultStackFrame: Boolean); Virtual`

Visibility: public

Description: This function starts the execution of Thorium bytecode instructions. The execution begins at the instruction index supplied via *StartInstruction* in the module which can be found at the index given with *StartModuleIndex*. The execution stops only when a jump to THORIUM\_JMP\_EXIT ([1](#)) occurs. If *CreateDefaultStackframe* is true, a stack frame is generated whose return value points to THORIUM\_JMP\_EXIT so that a *ret* instruction will finish the execution. You would normally only set *CreateDefaultStackframe* to False if you would want to initialize a module since that is the only

situation where jmp-instructions to THORIUM\_JMP\_EXIT are placed. If you want to call a function, you set CreateDefaultStackframe to True, although you should call functions always using their Call (18) or even SafeCall (19) method since these take care of the stack for you.