Package
fi.aalto.ssg.opentee

Java API Version: V 1.0 beta

This is the main entrance of public APIs. In order to help explaining the APIs, there are several essential key words defined in the following:
1. CA: Client Application that the developer is creating;
2. TA: Trusted Application which is already deployed in TEE;
3. TEE: Trusted Execution Environment in target Android device within which TAs are running;
4. TEE (Proxy) Service Manager: Android service layer abstraction for TEE, which is responsible for handling incoming connections from CAs and for communicating with the TEE with the help of Native Libtee;
5. Native Libtee: A library which enables the communication between TEE and TEE Service Manager;
6. Underlying library: A library which resides in the CA and communicates with the remote TEE service on the behalf of CAs.

Introduction
This public API documentation defines the Java APIs corresponding to the C APIs defined in the GlobalPlatform Device Technology TEE Client API specification V 1.0. It describes how the CA should communicate with a remote TEE service manager.

Target audience
This document suits for software developers implementing:
1. Android version of CAs running within the rich operating environment, which needs to utilizing the functions of TAs;
2. TAs running inside the TEE which need to expose its internal functions to CAs.

Background information
1. what is TEE?
TEE stands for Trusted Execution Environment. There is another notation called Rich Execution Environment (REE). These two are often brought together to help explain both of them by comparisons. Before taking a look at TEE, it is better to start explaining from REE since it is more closer to our daily sense. REE represents the common operating system along with its hardware, such as devices running Windows, Mac OSX, Linux, Android or iOS. It abstracts the underlying hardware and provides resources for the applications to run with. As such, it has rich features for applications to utilize. However, the REE frequently suffer from different kinds of attacks, such as malware, worm, trojan and ransomware. In order to protect very sensitive and private information such as encryption private keys against these attacks, it is good to keep these private information safely in a separate container in case the REE is compromised. It is the similar notion as the safe deposit box. For instance, if bad guys broke into a home, it is still impossible for them to get all your money in the safe deposit box without the right password to open it. So, with such a thought, TEE showed up to meet such needs. Currently, the TEE shipped within devices is physically separated with REE by hardware boundaries. CAs run in the REE and TAs runs in the TEE. Compared with the rich features of REE, TEE mostly comes with very limited hardware capabilities.

2. GP Specification for TEE Client API Specification.
GP is short for GlobalPlatform. It is a non-profit organization that publishes specifications to promote security and interoperability of secure devices. One of the specifications it published, named "GlobalPlatform Device Technology TEE Client API Specification" (GP Client API), standardizes the ways how CAs communicate with TAs. GlobalPlatform also have other specifications for TEE but we only focus on this one specifically. The specification defines the C data types and functions for CAs to communicate with TAs.

3. Open-TEE
Open-TEE is an open virtual Trusted Execution Environment which conforms to the GP TEE Specifications. For devices which are not equipped with real hardware-based TEE, it can provide a virtual TEE for developers to debug and deploy TAs before shipping applications to a real TEE.

API Design
1. What are these Java APIs and what their relationships with GP TEE Client Specification?
In general, these APIs are the Java version of C APIs in GP Client API specification with a reformed design to fit Java development conventions, which mainly target on the Android devices. It can be used to develop Android CAs which want to utilize the functionality which TAs offer. It provides all the necessary functions for CAs to communicate with remote TAs just like the C APIs defined in GP Specification.

2. Why they are needed?

(continued on next page)

(continued from last page)

In GP TEE Client Specification, it only specify the C data types and APIs which limit or complicate the development of CAs which aim for Android devices. Since Java is the mainstream programming language to develop Android applications, for Android developers who wants to utilize the GP C API to enable the communications between CAs and TAs, it would be troublesome to deal with native code development, especially for those who are not familiar with it, which can result in more potential bugs and unexpected behaviours if not handled correctly. Under such circumstances, every developers have to re-write these codes with the similar functionality, which can be a waste of efforts and error-prone. To avoid such awkward situations, an open-sourced design, which can enable the CAs communicate with TAs while provide nice and clean public interfaces for Android developers, is urgent to conquer this issue. With such a thought, the coming public Java APIs are available for Android developers, which can release them from the burdens of dealing native development in Android. It might be not that efficient as directly dealing with C APIs but the performance should be in a tolerant level. In addition, all the implementations of public APIS are open-source for everyone. By taking feedback from developers, these shared codes can be more bug-free and efficient.

3. How to use it and what to expect from the APIs?
a. Prerequisites
- The TA is already deployed in Open-TEE.
- The Android application which provides a remote TEE Proxy service should be running.
b. Check the descriptions for each API.

Bug report to:
rui.yang at aalto.fi

Organization:
Security System Group, Aalto University School of Science, Espoo, Finland.

Appendix: Example code chapter
The following example codes demonstrate how to utilize the Java API to communicate with the TAs residing in the TEE.

Firstly, we assume that we get an `ITEEClient` interface by calling a factory method. The way to obtain an `ITEEClient` interface depends on real implementation. The following code is just an example.

```
ITEEClient client = FactoryMethodWrappers.newTEEClient();
```

Right now, we want to establish a connection to a remote TEE Proxy service so that we can interact with the TAs running inside of TEE. By using the `ITEEClient` interface we obtained from last step, we can establish a connection to a remote TEE by calling `initializeContext` method in `ITEEClient` interface. For the two input parameters, please refer to the API definition in `ITEEClient.IContext` interface. If no exception is caught, a valid `IContext` interface will be returned and program flow continues to next block of code. Otherwise, the returned `IContext` interface will be null and an exception will be threw. For different kinds of exceptions can be threw by this API, please also refer to this API definition in `ITEEClient.IContext` interface. In the handling exception code block, it is recommended to re-initializeContext again and the program flow should not continue until it get a valid `IContext` interface.

```
ITEEClient.IContext ctx = null;

final String param_TEE_NAME = null; // connect to the default TEE.
final android.context.Context param_app_context = getApplicationContext();

try {
    ctx = client.initializeContext(param_TEE_NAME,
                                   param_app_context);
} catch (TEEClientException e) {
    // handle TEEClientException here.
}
```

After we successfully connected to the remote TEE Proxy service, in order to interact with one TA, we must open a session to the TA by providing correct authentication data. To open a session, the function `openSession` within `IContext` interface must be called. For the input parameters for the API and possible exceptions threw by it, please refer to the API definition in `ITEEClient.IContext` interface. For the creation of param_operation parameter, please refer to the example code which creates

(continued on next page)

(continued from last page)

an `IOperation` interface using the factory method `newOperation` in the coming sections.

```
ITEEClient.ISession ses = null;

final UUID param_uuid = new UUID(0x1234567887654321L, 0x0102030405060708L);
final ConnectionMethod param_conn_method =
        ITEEClient.IContext.ConnectionMethod.LoginPublic;
final Integer param_conn_data = null;

try {
    ses = ctx.openSession(param_uuid,
                param_conn_method,
                param_conn_data,
                param_operation);
} catch (TEEClientException e) {
    // handle TEEClientException here.
}
```

After successfully opened a session to a specific TA, a valid `ITEEClient.ISession` interface will be returned. So we can interact with TA by using `invokeCommand` API within the `ITEEClient.ISession` interface. The creation of param_operation please also refer to the same example code which creates an `IOperation` interface.

```
final int param_comm_id = 0x12345678;

try{
    ses.invokeCommand(param_comm_id,
                      param_operation);
}catch (TEEClientException e) {
    // handle TEEClientException here.
}
```

In some cases, data is needed to be transferred between CAs and TAs. So the API provides two different kinds of data encapsulation mechanisms. After that, they can be encapsulated again within `ITEEClient.IOperation` which can be sent to TA during `openSession` or `invokeCommand` calls.

The first approach is to create an `ITEEClient.IValue` interface by calling `newValue` factory method in `ITEEClient`. So up to 2 integer values can be encapsulated. The two values are given when calling `newValue` function and further interactions with this pair of values are defined in the `ITEEClient.IValue` interface.

```
final ITEEClient.IValue.Flag param_flag = ITEEClient.IValue.Flag.TEEC_VALUE_INOUT;

int param_value_A = 66;
int param_value_B = 88;

ITEEClient.IValue val = client.newValue(param_flag,
                                        param_value_A,
                                        param_value_B);
```

Another approach to transfer the data is using shared memory. The notation shared memory in here works as follows. Firstly, CA create a byte array as the buffer for the shared memory. Then, the CA registers the byte array as a shared memory to the TA so that

(continued on next page)

(continued from last page)

TA can also operate on the buffer. To create a shared memory, the CA must call `registerSharedMemory` method in `ITEEClient.IContext`. An `ITEEClient.ISharedMemory` interface will be returned.

```
ITEEClient.ISharedMemory sm = null;

byte[] param_byte_array = new byte[256];

ITEEClient.ISharedMemory param_flags =
        ITEEClient.ISharedMemory.TEEC_MEM_INPUT |
        ITEEClient.ISharedMemory.TEEC_MEM_OUTPUT;

try{
    sm = ctx.registerSharedMemory(param_byte_array,
                                  param_flags);
} catch (TEEClientException e) {
    // handle TEEClientException here.
}
```

After encapsulating the data within `IValue` interface or `ISharedMemory` interface, in order to share the data with TA, we must encapsulate these interfaces again into an `ITEEClient.IOperation` interface which then can be passed to TA during `openSession` or `invokeCommand` calls. The `IValue` interface can be directly used. However, to use the shared memory, the `ISharedMemory` interface must be encapsulated again into an `ITEEClient.IRegisteredMemoryReference` interface. Then along with the `IValue` interface, it can be used to create an `ITEEClient.IOperation` interface. To create an `IRegisteredMemoryReference` interface, the factory method `newRegisteredMemoryReference` within `ITEEClient` must be called.

```
ITEEClient.IRegisteredMemoryReference.Flag param_flags =
        ITEEClient.IRegisteredMemoryReference.Flag.TEEC_MEMREF_INOUT;

final param_offset = 0;

ITEEClient.IRegisteredMemoryReference rmr =
        client.newRegisteredMemoryReference(sm,
                                            param_flags,
                                            param_offset);
```

To create an `IOperation` interface, the factory method `newOperation` within `ITEEClient` must be called. The input parameters can be up to 4 `IValue` or `IRegisteredMemoryReference` interfaces.

```
ITEEClient.IOperation op = client.newOperation(rmr, val);
```

Resource cleaning up

If shared memory is no longer needed, it must be released by calling `releaseSharedMemory` function within `IContext` interface.

(continued on next page)

(continued from last page)

```
try {
    ctx.releaseSharedMemory(sm);
} catch (TEEClientException e) {
    // handle TEEClientException here.
}
```

The session also must be closed if CA no longer wants to interact with the TA.

```
try {
    ses.closeSession();
} catch (TEEClientException e) {
    // handle TEEClientException here.
}
```

Once CA no longer need to communicate with TEE, the context must be finalized. Be aware to release all the resources, mainly shared memory, and close all sessions before finalizing context.

```
try {
    ctx.finalizeContext();
} catch (TEEClientException e) {
    // handle TEEClientException here.
}
```

fi.aalto.ssg.opentee
Interface ITEEClient

public interface **ITEEClient**
extends

Open-TEE Java API entry point. `ITEEClient` interface embraces all APIs and public interfaces. CA can use it to communicate with a remote TEE/TA. The way how an `ITEEClient` can be obtained is determined by real implementations. It is not specified in this Java API.

Nested Class Summary

class	`ITEEClient.IContext` ITEEClient.IContext
class	`ITEEClient.IOperation` ITEEClient.IOperation
class	`ITEEClient.IParameter` ITEEClient.IParameter
class	`ITEEClient.IRegisteredMemoryReference` ITEEClient.IRegisteredMemoryReference
class	`ITEEClient.ISession` ITEEClient.ISession
class	`ITEEClient.ISharedMemory` ITEEClient.ISharedMemory
class	`ITEEClient.IValue` ITEEClient.IValue
class	`ITEEClient.ReturnOriginCode` ITEEClient.ReturnOriginCode

Field Summary

public static final	`TEEC_SUCCESS` The return value for TEEC_SUCCESS. Value: **0**

Method Summary

abstract `ITEEClient.IContext`	`initializeContext(java.lang.String teeName, Context context)` A method which initializes a context to a TEE.
abstract `ITEEClient.IOperation`	`Operation()` a method to create an operation without parameter.
abstract `ITEEClient.IOperation`	`Operation(ITEEClient.IParameter firstParam)` a method to create an operation with one parameter.

abstract ITEEClient.IOperation	Operation(ITEEClient.IParameter firstParam, ITEEClient.IParameter secondParam) a method to create an operation with two parameters.
abstract ITEEClient.IOperation	Operation(ITEEClient.IParameter firstParam, ITEEClient.IParameter secondParam, ITEEClient.IParameter thirdParam) a method to create an operation with three parameters.
abstract ITEEClient.IOperation	Operation(ITEEClient.IParameter firstParam, ITEEClient.IParameter secondParam, ITEEClient.IParameter thirdParam, ITEEClient.IParameter forthParam) a method to create an operation with four parameters.
abstract ITEEClient.IRegisteredMemoryReference	RegisteredMemoryReference(ITEEClient.ISharedMemory sharedMemory, ITEEClient.IRegisteredMemoryReference.Flag flag, int offset) A method to create a IRegisteredMemoryReference interface with a valid ISharedMemory interface.
abstract ITEEClient.IValue	Value(ITEEClient.IValue.Flag flag, int a, int b) A method to create an interface of a pair of two integer values.

Fields

TEEC_SUCCESS

```
public static final int TEEC_SUCCESS
```

The return value for TEEC_SUCCESS.
Constant value: **0**

Methods

Operation

```
public abstract ITEEClient.IOperation Operation()
```

a method to create an operation without parameter.

Returns:
an IOperation interface for created operation.

Operation

```
public abstract ITEEClient.IOperation Operation(ITEEClient.IParameter firstParam)
```

a method to create an operation with one parameter. It is possible to create multiple IOperation interfaces using the same IParameter. But it is not recommended especially when the I/O direction of IParameter is output for TA since it is possible that such an IParameter is in an inconsistent state. This rule also apply to other newOperation overloaded functions which take IParameter(s) as inputs.

Parameters:
firstParam - the first IParameter.

Returns:
an IOperation interface for created operation.

(continued on next page)

(continued from last page)

Operation

```
public abstract ITEEClient.IOperation Operation(ITEEClient.IParameter firstParam,
          ITEEClient.IParameter secondParam)
```

a method to create an operation with two parameters. The order of input parameters should be aligned with the order of required parameters in TA. This rule also apply to other overloaded `newOperation` functions which takes more than two parameters.

Parameters:

`firstParam` - the first `IParameter`.
`secondParam` - the second `IParameter`.

Returns:

an `IOperation` interface for created operation.

Operation

```
public abstract ITEEClient.IOperation Operation(ITEEClient.IParameter firstParam,
          ITEEClient.IParameter secondParam,
          ITEEClient.IParameter thirdParam)
```

a method to create an operation with three parameters.

Parameters:

`firstParam` - the first `IParameter`.
`secondParam` - the second `IParameter`.
`thirdParam` - the third `IParameter`.

Returns:

an `IOperation` interface for created operation.

Operation

```
public abstract ITEEClient.IOperation Operation(ITEEClient.IParameter firstParam,
          ITEEClient.IParameter secondParam,
          ITEEClient.IParameter thirdParam,
          ITEEClient.IParameter forthParam)
```

a method to create an operation with four parameters.

Parameters:

`firstParam` - the first `IParameter`.
`secondParam` - the second `IParameter`.
`thirdParam` - the third `IParameter`.
`forthParam` - the forth `IParameter`.

Returns:

an `IOperation` interface for created Operation.

RegisteredMemoryReference

```
public abstract ITEEClient.IRegisteredMemoryReference
RegisteredMemoryReference(ITEEClient.ISharedMemory sharedMemory,
          ITEEClient.IRegisteredMemoryReference.Flag flag,
          int offset)
  throws BadParametersException
```

A method to create a `IRegisteredMemoryReference` interface with a valid `ISharedMemory` interface. The flag parameter is only taken into considerations when the I/O direction(s) it implies are a subset of I/O directions of the referenced shared memory. It will not override the flags which the shared memory already have.

(continued on next page)

(continued from last page)

Parameters:

`sharedMemory` - the shared memory to refer.
`flag` - the flag for referenced shared memory.
`offset` - the offset from the beginning of the buffer of shared memory.

Value

```
public abstract ITEEClient.IValue Value(ITEEClient.IValue.Flag flag,
         int a,
         int b)
```

A method to create an interface of a pair of two integer values.

Parameters:

`flag` - The I/O directory of `IValue` for TAs.
`a` - The first integer value.
`b` - The second integer value.

Returns:

an `IValue` interface.

initializeContext

```
public abstract ITEEClient.IContext initializeContext(java.lang.String teeName,
         Context context)
  throws TEEClientException
```

A method which initializes a context to a TEE.

Parameters:

`teeName` - the name of remote TEE. If teeName is null, a context will be initialized within a default TEE.
`context` - Android application context.

Returns:

`IContext` interface.

Throws:

`exception.AccessDeniedException:` - Unable to initialize a context with the remote TEE due to insufficient privileges of the CA.
`exception.BadStateException:` - TEE is not ready to initialize a context for the CA.
`exception.BadParametersException:` - providing an invalid Android context.
`exception.BusyException:` - TEE is busy.
`exception.CommunicationErrorException:` - Communication with remote TEE service failed.
`exception.GenericErrorException:` - Non-specific cause exception.
`exception.TargetDeadException:` - TEE crashed.

fi.aalto.ssg.opentee
Interface ITEEClient.IOperation

public interface **ITEEClient.IOperation**
extends

This interface defines the way to interact with an `Operation` which is a wrapper interface for 0 to 4 `IParameter`(s). It can be created only by calling the function `newOperation`. After a valid `IOperation` interface is returned, developers can refer to the corresponding `Operation` in either `openSession` or `invokeCommand` function calls. When dealing with multiple threads, one `IOperation` interface can be shared between different threads. So it is possible that multiple threads try to access the same `IOperation` interface at the same time. If one or more `IParameter` interfaces wrapped inside the `IOperation` is output for the TA, it is possible that the `IParameter`(s) might be in an inconsistent state which may result in an incorrect read of the corresponding wrapped resources within `IParameter`(s), such as `IValue` and `SharedMemory`. Furthermore, if one thread attempts to apply one `IOperation` interface in its `openSession` or `InvokeCommand` function call while this `IOperation` interface is being used by another thread, a `BusyException` will be thrown. In addition, if one `IOperation` interface is modified by another thread, it is the responsibilities of developers to be aware of the changes. In order to avoid misuse of the `IOperation` interface, developers should not access any wrapped resources in an `IOperation` interface which is in use. The state of the `IOperation` can be obtained by calling its `isStarted` function. So it is recommended that developers should check the state of the `IOperation` interface before accessing it.

Method Summary

`abstract boolean`	`isStarted()` If one `IOperation` interface is being used in an ongoing operation (either openSession or invokeCommand) in a separate thread, this function will return true.

Methods

isStarted

```
public abstract boolean isStarted()
```

If one `IOperation` interface is being used in an ongoing operation (either openSession or invokeCommand) in a separate thread, this function will return true. Developers can utilize this function to test the availability of the `IOperation` interface.

Returns:

true if `IOperation` is under usage. Otherwise false if not being used.

fi.aalto.ssg.opentee
Interface ITEEClient.IParameter

All Subinterfaces:
IValue, IRegisteredMemoryReference

public interface **ITEEClient.IParameter**
extends

`IParameter` interface is the super interface of `IRegisteredMemoryReference` and `IValue` interfaces, It can passed into the `newOperation` to create an `IOperation` interface. It is possible to share the `IParameter` interface between different threads. Developers should be ware of the race condition when accessing the same `IParameter`. It is also their responsibilities to handle such a scenario.

Nested Class Summary

class	`ITEEClient.IParameter.Type` ITEEClient.IParameter.Type

Method Summary

abstract `ITEEClient.IParameter.Type`	`getType()` Get the type of the `IParameter` interface.

Methods

getType

`public abstract ITEEClient.IParameter.Type getType()`

Get the type of the `IParameter` interface.

Returns:
an enum value Type which can be either TEEC_PTYPE_VAL or TEEC_PTYPE_RMR.

fi.aalto.ssg.opentee
Class ITEEClient.IParameter.Type

```
java.lang.Object
   |-java.lang.Enum
      |-fi.aalto.ssg.opentee.ITEEClient.IParameter.Type
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public static final class **ITEEClient.IParameter.Type**
extends java.lang.Enum

The enum to indicates the type of the parameter.

Field Summary

public static final	TEEC_PTYPE_RMR This Parameter is a RegisteredMemoryReference.
public static final	TEEC_PTYPE_VAL This Parameter is a Value.

Method Summary

static ITEEClient.IParameter.Type	valueOf(java.lang.String name)
static ITEEClient.IParameter.Type[]	values()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields

(continued from last page)

TEEC_PTYPE_VAL

`public static final fi.aalto.ssg.opentee.ITEEClient.IParameter.Type` **`TEEC_PTYPE_VAL`**

This Parameter is a Value.

TEEC_PTYPE_RMR

`public static final fi.aalto.ssg.opentee.ITEEClient.IParameter.Type` **`TEEC_PTYPE_RMR`**

This Parameter is a RegisteredMemoryReference.

Methods

values

`public static ITEEClient.IParameter.Type[]` **`values`**`()`

valueOf

`public static ITEEClient.IParameter.Type` **`valueOf`**`(java.lang.String name)`

fi.aalto.ssg.opentee
Interface ITEEClient.IRegisteredMemoryReference

All Superinterfaces:
IParameter

public interface **ITEEClient.IRegisteredMemoryReference**
extends ITEEClient.IParameter

Interface for registered memory reference. When a shared memory needs to be passed to a remote TEE/TA, it must be wrapped within the `IRegisteredMemoryReference`. It can be only obtained by calling the `newRegisteredMemoryReference` function. It is possible that multiple `IRegisteredMemoryReference` interfaces are referencing the same `ISharedMemory` interface. So developers should be aware of such a situation

Nested Class Summary

class	`ITEEClient.IRegisteredMemoryReference.Flag` ITEEClient.IRegisteredMemoryReference.Flag

Method Summary

abstract int	`getOffset()` Get the offset set previously.
abstract int	`getReturnSize()` Get the size of returned buffer from TEE/TA.
abstract ITEEClient.ISharedMemory	`getSharedMemory()` Get the referenced registered shared memory.

Methods inherited from interface `fi.aalto.ssg.opentee.ITEEClient.IParameter`

`getType`

Methods

getSharedMemory

`public abstract ITEEClient.ISharedMemory getSharedMemory()`

Get the referenced registered shared memory.

Returns:
`ISharedMemory` interface for the referenced shared memory.

getOffset

`public abstract int getOffset()`

Get the offset set previously.

Returns:

(continued from last page)

an integer with a value ranging from 0 to the size of referenced shared memory.

getReturnSize

```
public abstract int getReturnSize()
```

Get the size of returned buffer from TEE/TA. This function will return a valid value (>= 0) only when the following two requirements are met at the same time:

- either TEEC_MEMREF_OUTPUT or TEEC_MEMREF_INOUT is marked as the flag of referenced shared memory;
- the referenced shared memory also can be used as output for TAs.

Otherwise, 0 will be returned. This function is normally called after the TA or TEE writes some data back to the referenced shared memory so that CA can know how big is the size of the returned data.

Returns:

an integer value as the returned size.

fi.aalto.ssg.opentee
Class ITEEClient.IRegisteredMemoryReference.Flag

```
java.lang.Object
   |
   +-java.lang.Enum
      |
      +-fi.aalto.ssg.opentee.ITEEClient.IRegisteredMemoryReference.Flag
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public static final class **ITEEClient.IRegisteredMemoryReference.Flag**
extends java.lang.Enum

Flag enum indicates the I/O direction of the referenced registered shared memory for TAs.

Field Summary

public static final	TEEC_MEMREF_INOUT The I/O directions of the referenced registered shared memory are both input and output for TAs.
public static final	TEEC_MEMREF_INPUT The I/O direction of the referenced registered shared memory is input for TAs.
public static final	TEEC_MEMREF_OUTPUT The I/O direction of the referenced registered shared memory is output for TAs.

Method Summary

static ITEEClient.IRegisteredMemoryReference.Flag	valueOf(java.lang.String name)
static ITEEClient.IRegisteredMemoryReference.Flag[]	values()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

(continued from last page)

Fields

TEEC_MEMREF_INPUT

```
public static final fi.aalto.ssg.opentee.ITEEClient.IRegisteredMemoryReference.Flag
TEEC_MEMREF_INPUT
```

The I/O direction of the referenced registered shared memory is input for TAs.

TEEC_MEMREF_OUTPUT

```
public static final fi.aalto.ssg.opentee.ITEEClient.IRegisteredMemoryReference.Flag
TEEC_MEMREF_OUTPUT
```

The I/O direction of the referenced registered shared memory is output for TAs.

TEEC_MEMREF_INOUT

```
public static final fi.aalto.ssg.opentee.ITEEClient.IRegisteredMemoryReference.Flag
TEEC_MEMREF_INOUT
```

The I/O directions of the referenced registered shared memory are both input and output for TAs.

Methods

values

```
public static ITEEClient.IRegisteredMemoryReference.Flag[] values()
```

valueOf

```
public static ITEEClient.IRegisteredMemoryReference.Flag valueOf(java.lang.String
name)
```

fi.aalto.ssg.opentee
Interface ITEEClient.IValue

All Superinterfaces:
IParameter

public interface **ITEEClient.IValue**
extends ITEEClient.IParameter

Interface to access a pair of two integer values. It can be only obtained by calling the `newValue` method.

Nested Class Summary

class	`ITEEClient.IValue.Flag` ITEEClient.IValue.Flag

Method Summary

abstract int	`getA()` Get the first value.
abstract int	`getB()` Get the second value.

Methods inherited from interface `fi.aalto.ssg.opentee.ITEEClient.IParameter`

`getType`

Methods

getA

`public abstract int getA()`

Get the first value.

Returns:
an integer.

getB

`public abstract int getB()`

Get the second value.

Returns:
an integer.

fi.aalto.ssg.opentee
Class ITEEClient.IValue.Flag

```
java.lang.Object
   |
   +-java.lang.Enum
      |
      +-fi.aalto.ssg.opentee.ITEEClient.IValue.Flag
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public static final class **ITEEClient.IValue.Flag**
extends java.lang.Enum

Flag enum indicates the I/O direction of Values for TAs.

Field Summary

public static final	TEEC_VALUE_INOUT The I/O directions for Value are both input and output for TAs.
public static final	TEEC_VALUE_INPUT The I/O direction for Value is input for TAs.
public static final	TEEC_VALUE_OUTPUT The I/O direction for Value is output for TAs.

Method Summary

static ITEEClient.IValue.Flag	valueOf(java.lang.String name)
static ITEEClient.IValue.Flag[]	values()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields

(continued from last page)

TEEC_VALUE_INPUT

```
public static final fi.aalto.ssg.opentee.ITEEClient.IValue.Flag TEEC_VALUE_INPUT
```

The I/O direction for Value is input for TAs.

TEEC_VALUE_OUTPUT

```
public static final fi.aalto.ssg.opentee.ITEEClient.IValue.Flag TEEC_VALUE_OUTPUT
```

The I/O direction for Value is output for TAs.

TEEC_VALUE_INOUT

```
public static final fi.aalto.ssg.opentee.ITEEClient.IValue.Flag TEEC_VALUE_INOUT
```

The I/O directions for Value are both input and output for TAs.

Methods

values

```
public static ITEEClient.IValue.Flag[] values()
```

valueOf

```
public static ITEEClient.IValue.Flag valueOf(java.lang.String name)
```

fi.aalto.ssg.opentee
Interface ITEEClient.ISession

public interface **ITEEClient.ISession**
extends

For a CA to communicate with a TA within a TEE, a session must be opened between the CA and TA. To open a session, the CA must call `openSession` within a valid context. When a session is opened, an `ISession` interface will be returned. It contains all functions for the CA to communicate with the TA. Within this session, developers can call the `invokeCommand` function to invoke a function within the TA. When the session is no longer needed, the developers should close the session by calling `closeSession` function.

Method Summary

`abstract void`	`closeSession()` Close the connection to the remote TA.
`abstract void`	`invokeCommand(int commandId, ITEEClient.IOperation operation)` Sending a request to the connected TA with agreed commandId and parameters.

Methods

invokeCommand

```
public abstract void invokeCommand(int commandId,
         ITEEClient.IOperation operation)
  throws TEEClientException
```

Sending a request to the connected TA with agreed commandId and parameters. The parameters are encapsulated in the operation.

Parameters:

`commandId` - command identifier that is previously agreed with the TA. Based on the command id, CA can tell TA to perform a certain action. TA will know what to perform.
`operation` - a wrapper of parameters for the action to take.

Throws:

`exception.AccessConflictException:` - using shared resources which are occupied by another thread;
`exception.BadFormatException:` - providing incorrect format of parameters in operation;
`exception.BadParametersException:` - providing parameters with invalid content;
`exception.BusyException:` - 1. the TEE is busy working on something else and does not have the computation power to execute requested operation;
2. the referenced `IOperation` interface is being used by another thread.
`exception.CancelErrorException:` - the provided operation is invalid due to the cancellation from another thread;
`exception.CommunicationErrorException:` - 1. fatal communication error occurred in the remote TEE and TA side;
2. Communication with the TEE proxy service failed.
`exception.ExcessDataException:` - providing too much parameters in the operation.
`exception.ExternalCancelException:` - current operation cancelled by external signal in the CA, remote TEE or TA side.
`exception.GenericErrorException:` - non-specific error.
`exception.ItemNotFoundException:` - providing invalid reference to a registered shared memory.
`exception.NoDataException:` - required data are missing in the operation.
`exception.NotImplementedException:` - action mapped with this command id is not implemented in TA yet.
`exception.NotSupportedException:` - action mapped with this command id is not supported in TA.

(continued from last page)

`exception.OutOfMemoryException:` - the remote system runs out of memory.
`exception.OverflowException:` - an buffer overflow happened in the remote TEE or TA.
`exception.SecurityErrorException:` - incorrect usage of shared memory.
`exception.ShortBufferException:` - the provided output buffer is too short to hold the output.
`exception.TargetDeadException:` - the remote TEE or TA crashed.

closeSession

```
public abstract void closeSession()
  throws TEEClientException
```

Close the connection to the remote TA. When dealing with multi-threads, this function is recommended to be called with the same thread which opens this session.

Throws:

`exception.CommunicationErrorException:` - Communication with remote TEE service failed.
`exception.TargetDeadException:` - the remote TEE or TA crashed.

fi.aalto.ssg.opentee
Interface ITEEClient.ISharedMemory

public interface **ITEEClient.ISharedMemory**
extends

In order to enable data sharing between a CA and TEE/TA, the notation called shared memory has been introduced to avoid expensive memory copies. A shared memory is a block of memory resides in the CA and a TEE/TA can operate on it directly. But how effective the shared memory is depends on the real implementation on specific systems. To create a shared memory, the CA firstly allocate a buffer which can be used as a shared memory. Then, the CA calls the `registerSharedMemory` to register the buffer as a shared memory to the remote TEE so that the TA can also operate on it. When the CA tries to register a shared memory, the I/O direction of this shared memory must be provided along with the buffer of the shared memory. The I/O direction is a bit mask of TEEC_MEM_INPUT and TEEC_MEM_OUTPUT. Note that the I/O direction of this shared memory is for the remote TEE/TA. See the detailed explanation of these two flags in the field description. The size of the shared memory is the same as the buffer that it holds. When the CA successfully register this buffer as a shared memory with a flag of TEEC_MEM_INPUT, any modification on this buffer will be synced to the TEE/TA during each function call from the CA to the TEE. Similarly, if the shared memory is flagged with TEEC_MEM_OUTPUT, any modification of the shared memory from the TEE side will be synced back to the CA after each remote function call from the CA to the TEE.

`ISharedMemory` interface provides operations on the shared memory. It is only valid in a `IContext` interface. This interface can be only obtained by calling `registerSharedMemory` function. If the registered shared memory is not longer needed, developers should release it by calling `releaseSharedMemory` function. After the shared memory is released, the buffer it holds will not longer used as a shared memory. So, any modification on it will no longer be synced to the remote the TEE/TA.

Field Summary

`public static final`	`TEEC_MEM_INPUT` This value indicates the I/O direction of the shared memory is input for both TEE and TA. Value: **`1`**
`public static final`	`TEEC_MEM_OUTPUT` This value indicates the I/O direction of the shared memory is output for both TEE and TA. Value: **`2`**

Method Summary

`abstract byte[]`	`asByteArray()` Get the content of the shared memory.
`abstract int`	`getFlags()` Get the I/O direction of the shared memory.

Fields

TEEC_MEM_INPUT

`public static final int` **`TEEC_MEM_INPUT`**

This value indicates the I/O direction of the shared memory is input for both TEE and TA.
Constant value: **`1`**

(continued from last page)

TEEC_MEM_OUTPUT

```
public static final int TEEC_MEM_OUTPUT
```

This value indicates the I/O direction of the shared memory is output for both TEE and TA.
Constant value: **2**

Methods

getFlags

```
public abstract int getFlags()
```

Get the I/O direction of the shared memory.

Returns:
the flags of `ISharedMemory`.

---

asByteArray

```
public abstract byte[] asByteArray()
```

Get the content of the shared memory. This function returns a reference to the buffer that the shared memory holds.

Returns:
an byte array reference.

fi.aalto.ssg.opentee
Class ITEEClient.ReturnOriginCode

```
java.lang.Object
   |
   +-java.lang.Enum
      |
      +-fi.aalto.ssg.opentee.ITEEClient.ReturnOriginCode
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public static final class **ITEEClient.ReturnOriginCode**
extends java.lang.Enum

A enum indicates the origin when an exception is threw. It can be obtained by calling `getReturnOrigin` of a caught exception. Developers can get a valid return origin only when the exceptions are threw by these two functions: `openSession` and `invokeCommand`. Otherwise, the return origin will be null.

Field Summary

public static final	TEEC_ORIGIN_API The exception is originated within the TEE Client API implementation.
public static final	TEEC_ORIGIN_COMMS The exception is originated within the underlying communications stack linking: 1.
public static final	TEEC_ORIGIN_TA The exception is originated within the TA.
public static final	TEEC_ORIGIN_TEE The exception is originated within the common TEE code.

Method Summary

static ITEEClient.ReturnOriginCode	valueOf(java.lang.String name)
static ITEEClient.ReturnOriginCode[]	values()

Methods inherited from class `java.lang.Enum`

`clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`

`compareTo`

Fields

TEEC_ORIGIN_API

```
public static final fi.aalto.ssg.opentee.ITEEClient.ReturnOriginCode TEEC_ORIGIN_API
```

The exception is originated within the TEE Client API implementation.

TEEC_ORIGIN_COMMS

```
public static final fi.aalto.ssg.opentee.ITEEClient.ReturnOriginCode TEEC_ORIGIN_COMMS
```

The exception is originated within the underlying communications stack linking:
1. the CA with remote TEE Proxy service;
2. the TEE Proxy service with the TEE.

TEEC_ORIGIN_TEE

```
public static final fi.aalto.ssg.opentee.ITEEClient.ReturnOriginCode TEEC_ORIGIN_TEE
```

The exception is originated within the common TEE code.

TEEC_ORIGIN_TA

```
public static final fi.aalto.ssg.opentee.ITEEClient.ReturnOriginCode TEEC_ORIGIN_TA
```

The exception is originated within the TA.

Methods

values

```
public static ITEEClient.ReturnOriginCode[] values()
```

valueOf

```
public static ITEEClient.ReturnOriginCode valueOf(java.lang.String name)
```

fi.aalto.ssg.opentee
Interface ITEEClient.IContext

public interface **ITEEClient.IContext**
extends

`IContext` interface provides all the functions to interact with an initialized context in remote TEE. This interface is returned by the `initializeContext` function call. When a context is no longer needed, it should be closed by calling `finalizeContext`. When the `IContext` interface is passed into different threads, developers are responsible for providing thread-safe mechanism to avoid the conflict between different threads.

Nested Class Summary

class	ITEEClient.IContext.ConnectionMethod ITEEClient.IContext.ConnectionMethod

Method Summary

abstract void	finalizeContext() Finalizing the context and close the connection to the TEE after all sessions have been terminated and all shared memories have been released.
abstract ITEEClient.ISession	openSession(java.util.UUID uuid, ITEEClient.IContext.ConnectionMethod connectionMethod, java.lang.Integer connectionData, ITEEClient.IOperation operation) Open a session with a TA within the current context.
abstract ITEEClient.ISharedMemory	registerSharedMemory(byte[] buffer, int flags) Register a block of existing CA memory as a shared memory within.
abstract void	releaseSharedMemory(ITEEClient.ISharedMemory sharedMemory) Releases the Shared Memory which was previously obtained using `registerSharedMemory`.
abstract void	requestCancellation(ITEEClient.IOperation operation) Requests the cancellation of a pending open session or a command invocation operation.

Methods

finalizeContext

```
public abstract void finalizeContext()
  throws TEEClientException
```

Finalizing the context and close the connection to the TEE after all sessions have been terminated and all shared memories have been released. This function is recommended to be called at the end of the thread which initialized the context.

Throws:

`exception.CommunicationErrorException:` - Communication with remote TEE service failed.

(continued from last page)

registerSharedMemory

```
public abstract ITEEClient.ISharedMemory registerSharedMemory(byte[] buffer,
          int flags)
  throws TEEClientException
```

Register a block of existing CA memory as a shared memory within. When this function tries to register a buffer as a shared memory which is already used by another shared memory, this function will also return a valid `ISharedMemory` interface. The TEE will regard this buffer as two identical shared memory. This will cause problems such as an `MacInvalidException`. However, when a shared memory is released, the buffer it holds can be registered again as a new shared memory. For the CA, the new shared memory has the same buffer but it is identical for the TEE.

Parameters:

`buffer` - pre-allocated byte array which is to be shared.

`flags` - indicates I/O direction of this shared memory for TAs.

Throws:

`exception.BadParametersException:` - 1. try to register a null/empty buffer as a shared memory; 2. providing incorrect flag value.

`exception.BadStateException:` - TEE is not ready to register a shared memory.

`exception.BusyException:` - TEE is busy.

`exception.CommunicationErrorException:` - Communication with remote TEE service failed.

`exception.ExternalCancelException:` - Current operation is cancelled by external signal in TEE.

`exception.GenericErrorException:` - Non-specific causes error.

`exception.NoStorageSpaceException:` - Insufficient storage in TEE.

`exception.OutOfMemoryException:` - Insufficient memory in TEE.

`exception.OverflowException:` - Buffer overflow in TEE.

`exception.TargetDeadException:` - TEE/TA crashed.

---

releaseSharedMemory

```
public abstract void releaseSharedMemory(ITEEClient.ISharedMemory sharedMemory)
  throws TEEClientException
```

Releases the Shared Memory which was previously obtained using `registerSharedMemory`. As stated in the description of the `ISharedMemory` interface, when the shared memory is released, the TEE/TA will no longer be able to read or write data to the shared memory. However, the buffer that this shared memory holds will still remain valid. When using the same shared memory within multi-threads, it is recommended to release the shared memory in the same thread which registered it.

Parameters:

`sharedMemory` - the reference to an `ISharedMemory` instance.

Throws:

`exception.CommunicationErrorException:` - Communication with the remote TEE service failed.

`exception.BadParametersException:` - Incorrect `ISharedMemory` instance such as passing a null object.

---

openSession

```
public abstract ITEEClient.ISession openSession(java.util.UUID uuid,
          ITEEClient.IContext.ConnectionMethod connectionMethod,
          java.lang.Integer connectionData,
          ITEEClient.IOperation operation)
  throws TEEClientException
```

Open a session with a TA within the current context. A session is a channel through which a CA can communicate with a specific TA (specified by the uuid). In order to open such a channel successfully, the CA must provide precise and correct data to authenticate itself to the TA.

Parameters:

`uuid` - UUID of the TA.

`connectionMethod` - the method of connection to use.

(continued from last page)

`connectionData` - any necessary data for connectionMethod.
`operation` - operation to perform.

Returns:

an `ISession` interface.

Throws:

`exception.AccessDeniedException:` - Insufficient privilege.
`exception.BadFormatException:` - Using incorrect format of parameter(s).
`exception.BadParametersException:` - Unexpected value(s) for parameter(s).
`exception.BadStateException:` - TEE is not ready to open a session or the referenced IOperation interface is occupied by another thread.
`exception.BusyException:` - TEE is busy.
`exception.CancelErrorException:` - Current operation is cancelled by another thread.
`exception.CommunicationErrorException:` - Communication with remote TEE service failed.
`exception.ExternalCancelException:` - Cancelled by external interrupt.
`exception.GenericErrorException:` - Non-specific cause.
`exception.ItemNotFoundException:` - Referred shared memory not found.
`exception.NoDataException:` - Extra data expected.
`exception.NoStorageSpaceException:` - Insufficient data storage in TEE.
`exception.OutOfMemoryException:` - TEE runs out of memory.
`exception.OverflowException:` - Buffer overflow in TEE.
`exception.SecurityErrorException:` - Incorrect usage of shared memory.
`exception.ShortBufferException:` - the provided output buffer is too short to hold the output.
`exception.TargetDeadException:` - TEE/TA crashed.

requestCancellation

```
public abstract void requestCancellation(ITEEClient.IOperation operation)
  throws TEEClientException
```

Requests the cancellation of a pending open session or a command invocation operation. This can be called from a different thread from that which is waiting for the `IOperation` interface. It is not guaranteed that the operation can be cancelled.

Parameters:

`operation` - the started or pending operation instance.

Throws:

`exception.CommunicationErrorException:` - Communication with remote TEE service failed.

fi.aalto.ssg.opentee
Class ITEEClient.IContext.ConnectionMethod

```
java.lang.Object
   ┗-java.lang.Enum
      ┗-fi.aalto.ssg.opentee.ITEEClient.IContext.ConnectionMethod
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public static final class **ITEEClient.IContext.ConnectionMethod**
extends java.lang.Enum

Connection Method enum with fixed value corresponding to GP specification when calling `openSession`.

Field Summary

public static final	`LoginApplication` Login data about the running CA process itself is provided.
public static final	`LoginGroup` Login data about the group running the CA process is provided.
public static final	`LoginGroupApplication` Login data about the group running the CA and about the Client Application and the about the CA itself is provided.
public static final	`LoginPublic` No login data is provided.
public static final	`LoginUser` Login data about the user running the CA process is provided.
public static final	`LoginUserApplication` Login data about the user running the CA and about the Client Application itself is provided.

Method Summary

static ITEEClient.IContext.ConnectionMethod	`valueOf(java.lang.String name)`
static ITEEClient.IContext.ConnectionMethod[]	`values()`

Methods inherited from class `java.lang.Enum`

`clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf`

Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface `java.lang.Comparable`
compareTo

Fields

LoginPublic

```
public static final fi.aalto.ssg.opentee.ITEEClient.IContext.ConnectionMethod
LoginPublic
```

No login data is provided.

LoginUser

```
public static final fi.aalto.ssg.opentee.ITEEClient.IContext.ConnectionMethod
LoginUser
```

Login data about the user running the CA process is provided.

LoginGroup

```
public static final fi.aalto.ssg.opentee.ITEEClient.IContext.ConnectionMethod
LoginGroup
```

Login data about the group running the CA process is provided.

LoginApplication

```
public static final fi.aalto.ssg.opentee.ITEEClient.IContext.ConnectionMethod
LoginApplication
```

Login data about the running CA process itself is provided.

LoginUserApplication

```
public static final fi.aalto.ssg.opentee.ITEEClient.IContext.ConnectionMethod
LoginUserApplication
```

Login data about the user running the CA and about the Client Application itself is provided.

LoginGroupApplication

```
public static final fi.aalto.ssg.opentee.ITEEClient.IContext.ConnectionMethod
LoginGroupApplication
```

Login data about the group running the CA and about the Client Application and the about the CA itself is provided.

Methods

(continued from last page)

values

```
public static ITEEClient.IContext.ConnectionMethod[] values()
```

---

valueOf

```
public static ITEEClient.IContext.ConnectionMethod valueOf(java.lang.String name)
```

fi.aalto.ssg.opentee
Interface OTHelper

public interface **OTHelper**
extends

Open-TEE specific util functions for CA.

Method Summary

abstract boolean	installTA(java.lang.String taFileName)
abstract void	installTA(java.lang.String taName, byte[] taInBytes)

Methods

installTA

```
public abstract void installTA(java.lang.String taName,
         byte[] taInBytes)
  throws CommunicationErrorException
```

installTA

```
public abstract boolean installTA(java.lang.String taFileName)
  throws CommunicationErrorException
```

Package
fi.aalto.ssg.opentee.exception

This package contains the exceptions used by the client library.

fi.aalto.ssg.opentee.exception
Class AccessConflictException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.AccessConflictException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **AccessConflictException**
extends TEEClientException

Concurrent accesses caused conflict. This exception can be threw by underlying library when one thread of the Client Application tries to access shared resources, such as ISharedMemory and ISession, while the same resource is held by another thread.

Constructor Summary

public	AccessConflictException(java.lang.String msg)
public	AccessConflictException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

AccessConflictException

public **AccessConflictException**(java.lang.String msg)

(continued from last page)

AccessConflictException

```
public AccessConflictException(java.lang.String msg,
                               ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class AccessDeniedException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.AccessDeniedException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **AccessDeniedException**
extends TEEClientException

Access privileges are not sufficient. This exception can be threw from underlying library when the Client Application has insufficient and/or incorrect authentication data to prove its identity when try to connect to a remote TEE or a Trusted Application.

Constructor Summary

public	AccessDeniedException(java.lang.String msg)
public	AccessDeniedException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

AccessDeniedException

public **AccessDeniedException**(java.lang.String msg)

(continued from last page)

AccessDeniedException

```
public AccessDeniedException(java.lang.String msg,
                             ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class BadFormatException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.BadFormatException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **BadFormatException**
extends TEEClientException

Input data was of invalid format. This exception can be threw by underlying library when the Client Application gives an input data with a wrong format in a sense that either the remote TEE or TA can not parse it correctly.

Constructor Summary

public	BadFormatException(java.lang.String msg)
public	BadFormatException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

BadFormatException

public **BadFormatException**(java.lang.String msg)

(continued from last page)

BadFormatException

```
public BadFormatException(java.lang.String msg,
                          ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class BadParametersException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.BadParametersException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **BadParametersException**
extends TEEClientException

Input parameters were invalid. This exception can be threw from underlying library when the TEE or TA get a parameter(s) which is not expected as a valid value(s).

Constructor Summary

public	BadParametersException(java.lang.String msg)
public	BadParametersException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

BadParametersException

public **BadParametersException**(java.lang.String msg)

(continued from last page)

BadParametersException

```
public BadParametersException(java.lang.String msg,
                              ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class BadStateException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.BadStateException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **BadStateException**
extends TEEClientException

Operation is not valid in the current state. This exception can be threw by underlying library when the Client Application tries to initialize context when the TEE is not ready to do so.

Constructor Summary

public	BadStateException(java.lang.String msg)
public	BadStateException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

BadStateException

public **BadStateException**(java.lang.String msg)

(continued from last page)

BadStateException

```
public BadStateException(java.lang.String msg,
                         ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class BusyException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.BusyException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **BusyException**
extends TEEClientException

This exception can be threw by underlying library when the system is busy working on something and will not accept any incoming operation requests.

Constructor Summary

public	BusyException(java.lang.String msg)
public	BusyException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

BusyException

public **BusyException**(java.lang.String msg)

(continued from last page)

BusyException

```
public BusyException(java.lang.String msg,
                     ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class CancelErrorException

```
java.lang.Object
   |-java.lang.Throwable
      |-java.lang.Exception
         |-fi.aalto.ssg.opentee.exception.TEEClientException
            |-fi.aalto.ssg.opentee.exception.CancelErrorException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **CancelErrorException**
extends TEEClientException

The operation was cancelled. This exception can be threw by underlying library when the remote TEE or TA tries to access one operation which has already been cancelled by another thread in Client Application. This scenario can only happen when the following two constraints are satisfied: 1. this operation must has a started field set to 0. If developer does not want one operation to be cancelled, he/she can set this field to 1 to tell the TEE that this operation can be cancelled before being invoked by TEE; 2. this operation has not been invoked by the TEE yet before it has been cancelled by another thread in Client Application who managed to call requestCancellation function and the remote TEE actually cancelled it.

Constructor Summary

public	CancelErrorException(java.lang.String msg)
public	CancelErrorException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

CancelErrorException

public **CancelErrorException**(java.lang.String msg)

(continued from last page)

CancelErrorException

```
public CancelErrorException(java.lang.String msg,
                            ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class CommunicationErrorException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.CommunicationErrorException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **CommunicationErrorException**
extends TEEClientException

Communication with a remote party failed. This exception includes the communication errors with Android IPC calls which have already been defined in RemoteException and the developers are also suppose to handle it too. On the basis of the CA able to communicate with remote service, this exception is threw by underlying library when the NativeLibtee fails to communicate with the TEE or TAs. This situation can be caused by the following cases: 1. when there are internal errors with TEE which disable the NativeLibtee taking to TEE. Under such a circumstances, the developers are suggested to check the states and configurations of TEE on target device and adjust TEE to run properly before interacting with in Client Application; 2. when TEE is unable to talk to TA especially when the TA is crashed due to internal errors. Under such a circumstance, the developers are suggested to debug the TA and fix corresponding errors before using CA talks to it.

Constructor Summary

public	CommunicationErrorException(java.lang.String msg)
public	CommunicationErrorException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

(continued from last page)

CommunicationErrorException

```
public CommunicationErrorException(java.lang.String msg)
```

CommunicationErrorException

```
public CommunicationErrorException(java.lang.String msg,
                                   ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class ExcessDataException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.ExcessDataException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **ExcessDataException**
extends TEEClientException

Too much data for the requested operation was passed. This exception can be threw by under lying library when the CA provides unexpected amount of data to TEE.

Constructor Summary

public	ExcessDataException(java.lang.String msg)
public	ExcessDataException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

ExcessDataException

public **ExcessDataException**(java.lang.String msg)

(continued from last page)

ExcessDataException

```
public ExcessDataException(java.lang.String msg,
                           ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class ExternalCancelException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-fi.aalto.ssg.opentee.exception.TEEClientException
            |
            +-fi.aalto.ssg.opentee.exception.ExternalCancelException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **ExternalCancelException**
extends TEEClientException

An external event has caused a User Interface operation to be aborted, which is defined by the Trusted User Interface specification.

Constructor Summary

public	ExternalCancelException(java.lang.String msg)
public	ExternalCancelException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

ExternalCancelException

public **ExternalCancelException**(java.lang.String msg)

(continued from last page)

ExternalCancelException

```
public ExternalCancelException(java.lang.String msg,
                               ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class GenericErrorException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.GenericErrorException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **GenericErrorException**
extends TEEClientException

Non-specific cause exception. This exception can be threw by underlying library when there is an error(s) excluding the errors already defined when CA is interacting with TEE.

Constructor Summary

public	GenericErrorException(java.lang.String msg)
public	GenericErrorException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

GenericErrorException

public **GenericErrorException**(java.lang.String msg)

(continued from last page)

GenericErrorException

```
public GenericErrorException(java.lang.String msg,
                             ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class ItemNotFoundException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.ItemNotFoundException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **ItemNotFoundException**
extends TEEClientException

The requested data item is not found. This exception can be threw by underlying library when CA tries to refer a shared memory which has already been released.

Constructor Summary

public	ItemNotFoundException(java.lang.String msg)
public	ItemNotFoundException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

ItemNotFoundException

public **ItemNotFoundException**(java.lang.String msg)

(continued from last page)

ItemNotFoundException

```
public ItemNotFoundException(java.lang.String msg,
                             ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class NoDataException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-fi.aalto.ssg.opentee.exception.TEEClientException
            |
            +-fi.aalto.ssg.opentee.exception.NoDataException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **NoDataException**
extends TEEClientException

Expected data was missing. This exception can be threw by underlying library when the CA does not provide enough data for the remote TEE/TA. As a result, the corresponding operation will fail.

Constructor Summary

public	NoDataException(java.lang.String msg)
public	NoDataException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

NoDataException

public **NoDataException**(java.lang.String msg)

(continued from last page)

NoDataException

```
public NoDataException(java.lang.String msg,
                       ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class NotImplementedException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.NotImplementedException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **NotImplementedException**
extends TEEClientException

The requested operation should exist but is not yet implemented. This exception can threw by underlying library when the CA invokes an operation which has not been implemented in TA yet. TA can use this exception to notify the CA that the function it invoked is not ready right now but might be available in the future.

Constructor Summary

public	NotImplementedException(java.lang.String msg)
public	NotImplementedException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

NotImplementedException

public **NotImplementedException**(java.lang.String msg)

(continued from last page)

NotImplementedException

```
public NotImplementedException(java.lang.String msg,
                               ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class NotSupportedException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.NotSupportedException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **NotSupportedException**
extends TEEClientException

The requested operation is valid but is not supported in this implementation. This exception can be threw by underlying library when the CA tries to invoke a valid operation which does not exists in current implementation of TA. This exception can be used to notify the CA that it talks to an older version of TA which does not support such an operation. So, this exception can help the CA to be backward compatible.

Constructor Summary

public	NotSupportedException(java.lang.String msg)
public	NotSupportedException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

NotSupportedException

public **NotSupportedException**(java.lang.String msg)

NotSupportedException

```
public NotSupportedException(java.lang.String msg,
                             ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class OutOfMemoryException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.OutOfMemoryException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **OutOfMemoryException**
extends TEEClientException

System ran out of resources. This exception can be threw by underlying library when the remote service runs out of resources. Under such a circumstance, the developer is suggested to release some unused resources or limit the number of calls to remote service.

Constructor Summary

public	OutOfMemoryException(java.lang.String msg)
public	OutOfMemoryException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

OutOfMemoryException

public **OutOfMemoryException**(java.lang.String msg)

(continued from last page)

OutOfMemoryException

```
public OutOfMemoryException(java.lang.String msg,
                            ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class SecurityErrorException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.SecurityErrorException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **SecurityErrorException**
extends TEEClientException

A security fault was detected. This exception can be threw by underlying library when the CA tries to access a number of resources in a wrong way. For instance, if the shared memory only marked with input for TA, the CA should not require the TA to use the shared memory as an output.

Constructor Summary

public	SecurityErrorException(java.lang.String msg)
public	SecurityErrorException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

SecurityErrorException

```
public SecurityErrorException(java.lang.String msg)
```

(continued from last page)

SecurityErrorException

```
public SecurityErrorException(java.lang.String msg,
                              ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class ShortBufferException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.ShortBufferException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **ShortBufferException**
extends TEEClientException

The supplied buffer is too short for the generated output. This exception can be threw by underlying library when the TA tries to copy the result to a shorter output buffer which is previously given by CA. This scenario can happen when the provided Shared Memory or Value for output is too short. Under such a circumstance, the developers are suggested to allocate a bigger buffer for output. If the Value is not bigger enough for the output, the Shared Memory should be used instead.

Constructor Summary

public	ShortBufferException(java.lang.String msg)
public	ShortBufferException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

ShortBufferException

```
public ShortBufferException(java.lang.String msg)
```

ShortBufferException

```
public ShortBufferException(java.lang.String msg,
                            ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class TargetDeadException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.TargetDeadException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **TargetDeadException**
extends TEEClientException

The Trusted Application has terminated.

Constructor Summary

public	TargetDeadException(java.lang.String msg)
public	TargetDeadException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

TargetDeadException

public **TargetDeadException**(java.lang.String msg)

(continued from last page)

TargetDeadException

```
public TargetDeadException(java.lang.String msg,
                           ITEEClient.ReturnOriginCode retOrigin)
```

fi.aalto.ssg.opentee.exception
Class TEEClientException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
```

All Implemented Interfaces:
java.io.Serializable

Direct Known Subclasses:
TrustedApplicationException, TargetDeadException, ShortBufferException, SecurityErrorException, OutOfMemoryException, NotSupportedException, NotImplementedException, NoDataException, ItemNotFoundException, GenericErrorException, ExternalCancelException, ExcessDataException, CommunicationErrorException, CancelErrorException, BusyException, BadStateException, BadParametersException, BadFormatException, AccessDeniedException, AccessConflictException

---

public abstract class **TEEClientException**
extends java.lang.Exception

TEEClientException extends the java.lang.Exception class. All exceptions in this project should subclass it excluding exceptions defined by Android. The origin which causes this exception can be obtained using getReturnOrigin function. In most cases, all exceptions come without a return origin. Only exceptions come from the openSession and invokeCommand function calls have it. So developer should be able to distinguish these two kinds of exceptions. The getReturnOrigin will return null if one exception does not have a return origin.

Constructor Summary

public	TEEClientException()
public	TEEClientException(ITEEClient.ReturnOriginCode returnOriginCode)
public	TEEClientException(java.lang.String message)
public	TEEClientException(java.lang.String message, ITEEClient.ReturnOriginCode returnOriginCode)
public	TEEClientException(java.lang.String message, java.lang.Throwable cause)
public	TEEClientException(java.lang.Throwable cause)

Method Summary

ITEEClient.ReturnOriginCode	getReturnOrigin() Get the return origin.

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

TEEClientException

public **TEEClientException**()

TEEClientException

public **TEEClientException**(ITEEClient.ReturnOriginCode returnOriginCode)

TEEClientException

public **TEEClientException**(java.lang.String message)

TEEClientException

public **TEEClientException**(java.lang.String message,
ITEEClient.ReturnOriginCode returnOriginCode)

TEEClientException

public **TEEClientException**(java.lang.String message,
java.lang.Throwable cause)

TEEClientException

public **TEEClientException**(java.lang.Throwable cause)

Methods

getReturnOrigin

public ITEEClient.ReturnOriginCode **getReturnOrigin**()

(continued from last page)

Get the return origin.

Returns:

The return origin code.

fi.aalto.ssg.opentee.exception
Class TrustedApplicationException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-fi.aalto.ssg.opentee.exception.TEEClientException
            ┗-fi.aalto.ssg.opentee.exception.TrustedApplicationException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **TrustedApplicationException**
extends TEEClientException

Exception for error code customized by developer. This exception can be threw by underlying library within openSession function call when an customized error code is returned. This error code should differ itself from other return code defined by GP specification.

Constructor Summary

public	TrustedApplicationException(java.lang.String msg, int errorCode)
public	TrustedApplicationException(java.lang.String msg, ITEEClient.ReturnOriginCode retOrigin)

Method Summary

int	getErrorCode()

Methods inherited from class fi.aalto.ssg.opentee.exception.TEEClientException

getReturnOrigin

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

(continued from last page)

TrustedApplicationException

```
public TrustedApplicationException(java.lang.String msg,
                                   int errorCode)
```

TrustedApplicationException

```
public TrustedApplicationException(java.lang.String msg,
                                   ITEEClient.ReturnOriginCode retOrigin)
```

Methods

getErrorCode

```
public int getErrorCode()
```