



Samsung Client API reference

Date	2019.09.18	Version	1.0
Organizator	Mobile Security Technologies Group, IT & Mobile Communications		

Contents

1 Overview	3
1.1 Objective	3
1.2 Scope	3
1.3 Abbreviations and Acronyms	3
1.4 API type	3
1.4.1 Samsung Client API	3
2 Module Documentation	4
2.1 Samsung Client API	4
2.1.1 Detailed Description	5
2.1.2 Enumeration Type Documentation	5
2.1.2.1 anonymous enum	5
2.1.2.2 Cluster	5
2.1.2.3 ta_debug_service_type_t	5
2.1.3 Function Documentation	5
2.1.3.1 get_error_name(TEEC_Result result)	5
2.1.3.2 get_error_text(TEEC_Result result)	6
2.1.3.3 get_return_origin(uint32_t returnOrigin)	6
2.1.3.4 TEECS_Boost(TEEC_Session *session)	6
2.1.3.5 TEECS_GetSessionId(const TEEC_Session *session, TEEC_UUID *session_id)	7
2.1.3.6 TEECS_OpenSession(TEEC_Context *context, TEEC_Session *session, const TEEC_UUID *destination, const void *ta_image, const size_t ta_image_size, uint32_t connectionMethod, const void *connectionData, TEEC_Operation *operation, uint32_t *returnOrigin)	7
2.1.3.7 TEECS_OpenSession1(TEEC_Context *context, TEEC_Session *session, const TEEC_UUID *destination, const char *destination_path, uint32_t connectionMethod, const void *connectionData, TEEC_Operation *operation, uint32_t *returnOrigin)	8
2.1.3.8 TEECS_SetCluster(TEEC_Session *session, Cluster cluster)	9
2.1.3.9 TEECS_SetCryptoClk(TEEC_Context *context, uint32_t state)	10
2.1.3.10 TEECS_TADebugServiceCtl(ta_debug_service_type_t type, const TEEC_UUID *monitoredTA, const TEEC_UUID *monitoredSession)	10
2.1.3.11 TEECS_Unboost(TEEC_Session *session)	11
2.1.3.12 TEECS_WaitTADeath(TEEC_Session *session)	12
3 File Documentation	13
3.1 tees_client_api.h File Reference	13
3.1.1 Detailed Description	14
Bibliography	15
Index	15

Version	Date:	Change Contents	Author	Approver
1.0		Initial version	D.Mazaeva	
2.0	2017.03.27	Autobuild documentation	D.Mazaeva	

1 Overview

1.1 Objective

The goal of this document is to describe Secure OS and Client Library API. The intended audience is Client Application (CA) and Trusted Application developers with permission to access Samsung Client API.

1.2 Scope

The scope of this document is to describe Samsung Client API to Client Application developers. Global Platform (GP) API is out of scope of this document, and (GP) API is explained in TEE Client API document at [Global Platform](#) portal.

1.3 Abbreviations and Acronyms

Terminology / Abbreviation	Description
CA	Client application
TA	Trusted application

1.4 API type

1.4.1 Samsung Client API

[Samsung Client API](#) is Samsung's own API to support any privileged features to internal CA developers. Samsung Client API extends TEE Client API and it provides functionality for setting crypto clocks, modified version of open session function, TEE TA Debug functionality and getting description of errors code.

2 Module Documentation

2.1 Samsung Client API

Enumerations

- enum [Cluster](#) { [CLUSTER_BIG](#), [CLUSTER_LITTLE](#), [CLUSTER_DEFAULT](#), [CLUSTER_MAX_VALUE](#) }
- enum { [DISABLE_CRYPT_CLK](#), [ENABLE_CRYPT_CLK](#) }
- enum [ta_debug_service_type_t](#) { [TA_DEBUG_SERVICE_DLM](#), [TA_DEBUG_SERVICE_PMR](#) }

Functions

- const char * [get_error_name](#) (TEEC_Result result)
The function returns string representation of [TEEC_Result](#) type.
- const char * [get_error_text](#) (TEEC_Result result)
The function returns description of [TEEC_Result](#) type.
- const char * [get_return_origin](#) (uint32_t returnOrigin)
*The function description of *origin* type.*
- TEEC_Result [TEECS_SetCluster](#) (TEEC_Session *session, [Cluster](#) cluster)
The function sets TA (main thread and its descendants, created after call) to desired cpu cluster.
- TEEC_Result [TEECS_Boost](#) (TEEC_Session *session)
The function enables performance boosting for current TA session.
- TEEC_Result [TEECS_Unboost](#) (TEEC_Session *session)
The function disables performance boosting for current TA session.
- TEEC_Result [TEECS_SetCryptoClk](#) (TEEC_Context *context, uint32_t state)
The function switches crypto clocks ON/OFF Initially designed to reduce power consumption on Qualcomm chipsets A reference counter is maintained in [TEEC_ContextImp](#).
- TEEC_Result [TEECS_GetSessionId](#) (const TEEC_Session *session, TEEC_UUID *session_id)
Function gets session ID.
- TEEC_Result [TEECS_OpenSession](#) (TEEC_Context *context, TEEC_Session *session, const TEEC_UUID *destination, const void *ta_image, const size_t ta_image_size, uint32_t connectionMethod, const void *connectionData, TEEC_Operation *operation, uint32_t *returnOrigin)
*The function opens TA which is located not exactly in /vendor/tee directory. Function behaves the same way as [TEEC_OpenSession\(\)](#), but provides two additional parameters. *ta_image* and *ta_image_size*.*
- TEEC_Result [TEECS_OpenSession1](#) (TEEC_Context *context, TEEC_Session *session, const TEEC_UUID *destination, const char *destination_path, uint32_t connectionMethod, const void *connectionData, TEEC_Operation *operation, uint32_t *returnOrigin)
*The function opens TA which is located not exactly in /vendor/tee directory. Function behaves the same way as [TEEC_OpenSession\(\)](#), but provides one additional parameter. *destination_path* (e.g. /system/tee, /data/tee) wherever CA has appropriate permission.*
- TEEC_Result [TEECS_TADebugServiceCtl](#) ([ta_debug_service_type_t](#) type, const TEEC_UUID *monitoredTA, const TEEC_UUID *monitoredSession)
Create a request into underlying TEE implementation to provide TEE TA Debug functionality for current Client Application.
- TEEC_Result [TEECS_WaitTADeath](#) (TEEC_Session *session)
Wait for unexpected death of TA.

2.1.1 Detailed Description

2.1.2 Enumeration Type Documentation

2.1.2.1 anonymous enum

```
#include <tees_client_api.h>
```

Crypto clocks enum

Enumerator

DISABLE_CRYPTO_CLK Disable crypto clocks

ENABLE_CRYPTO_CLK Enable crypto clocks

2.1.2.2 enum Cluster

```
#include <tees_client_api.h>
```

This is an cluster enum

2.1.2.3 enum ta_debug_service_type_t

```
#include <tees_client_api.h>
```

Debug service type enum for TEE TA Debug spec

Enumerator

TA_DEBUG_SERVICE_DLM Debug log message type

TA_DEBUG_SERVICE_PMR Post mortem report type

2.1.3 Function Documentation

2.1.3.1 const char* get_error_name (TEEC_Result *result*)

```
#include <tees_client_api.h>
```

The function returns string representation of TEEC_Result type.

Parameters

in	<i>result</i>	the value to get representation.
----	---------------	----------------------------------

Return values

<i>string</i>	representation of TEEC_Result.
---------------	--------------------------------

2.1.3.2 `const char* get_error_text (TEEC_Result result)`

```
#include <tees_client_api.h>
```

The function returns description of TEEC_Result type.

Parameters

in	<i>result</i>	the value to get description.
----	---------------	-------------------------------

Return values

<i>string</i>	description of TEEC_Result.
---------------	-----------------------------

2.1.3.3 `const char* get_return_origin (uint32_t returnOrigin)`

```
#include <tees_client_api.h>
```

The function description of `origin` type.

Parameters

in	<i>returnOrigin</i>	the value to get description.
----	---------------------	-------------------------------

Return values

<i>string</i>	description of <code>origin</code> .
---------------	--------------------------------------

2.1.3.4 `TEEC_Result TEECS_Boost (TEEC_Session * session)`

```
#include <tees_client_api.h>
```

The function enables performance boosting for current TA session.

Parameters

in	<i>session</i>	the session of a TA.
----	----------------	----------------------

Return values

<i>TEEC_SUCCESS</i>	in case of success.
<i>TEEC_ERROR_BAD_PARAMETERS</i>	invalid parameters.

Example of usage:

```

res = TEECS_Boost(&session);
if (res) {
    // Error handling
}

// Invoking commands with boosted performance

res = TEECS_Unboost(&session);
if (res) {
    // Error handling
}

```

2.1.3.5 TEEC_Result TEECS_GetSessionId (const TEEC_Session * session, TEEC_UUID * session_id)

```
#include <tees_client_api.h>
```

Function gets session ID.

Parameters

in	<i>session</i>	pointer to a Session structure to open.
in	<i>session_id</i>	identificator of Session structure.

Return values

<i>TEEC_SUCCESS</i>	in case of success.
<i>TEEC_ERROR_BAD_PARAMETERS</i>	
<i>TEEC_ERROR_GENERIC</i>	

Example of usage:

TODO: provide workable example

2.1.3.6 TEEC_Result TEECS_OpenSession (TEEC_Context * context, TEEC_Session * session, const TEEC_UUID * destination, const void * ta_image, const size_t ta_image_size, uint32_t connectionMethod, const void * connectionData, TEEC_Operation * operation, uint32_t * returnOrigin)

```
#include <tees_client_api.h>
```

The function opens TA which is located not exactly in /vendor/tee directory. Function behaves the same way as TEEC_OpenSession(), but provides two additional parameters. *ta_image* and *ta_image_size*.

Parameters

in	<i>context</i>	pointer to a session context.
in	<i>session</i>	pointer to a session structure to open.
in	<i>destination</i>	pointer to a structure containing the TA uuid.
in	<i>ta_image</i>	pointer to TA image.
in	<i>ta_image_size</i>	size of TA image.
in	<i>connectionMethod</i>	the method of connection to use.
in	<i>connectionData</i>	any necessary data required to support the connection method chosen.
in	<i>operation</i>	pointer to an operation containing parameters to exchange with the TA.
in	<i>returnOrigin</i>	pointer to a variable which will contain the return origin.

Return values

<i>TEEC_SUCCESS</i>	in case of success.
<i>TEEC_ERROR_BAD_PARAMETERS</i>	wrong parameters were passed to the function.

Example of usage:

```
int fd = open(TA_NAME, O_RDONLY);

struct stat st;
fstat(fd, &st);
void *ta_image = malloc(st.st_size);

size_t ta_image_size = read(fd, (char *)ta_image, st.st_size);
close(fd);

res = TEECS_OpenSession(context,
                        &session,
                        &ta_uuid,
                        ta_image,
                        ta_image_size,
                        0,
                        NULL,
                        NULL,
                        &returnOrigin);
```

2.1.3.7 TEEC_Result TEECS_OpenSession1 (TEEC_Context * context, TEEC_Session * session, const TEEC_UUID * destination, const char * destination_path, uint32_t connectionMethod, const void * connectionData, TEEC_Operation * operation, uint32_t * returnOrigin)

```
#include <tees_client_api.h>
```

The function opens TA which is located not exactly in /vendor/tee directory. Function behaves the same way as TEEC_OpenSession(), but provides one additional parameter. *destination_path* (e.g. /system/tee, /data/tee) wherever CA has appropriate permission.

Parameters

in	<i>context</i>	pointer to a session context.
in	<i>session</i>	pointer to a session structure to open.
in	<i>destination</i>	pointer to a structure containing the TA uuid.
in	<i>destination_path</i>	pointer to directory location
in	<i>connectionMethod</i>	the method of connection to use.
in	<i>connectionData</i>	any necessary data required to support the connection method chosen.
in	<i>operation</i>	pointer to an operation containing parameters to exchange with the TA.
in	<i>returnOrigin</i>	pointer to a variable which will contain the return origin.

Return values

<i>TEEC_SUCCESS</i>	in case of success.
<i>TEEC_ERROR_BAD_PARAMETERS</i>	wrong parameters were passed to the function.

Example of usage:

```
const char *path = "/system/tee/";

res = TEECS_OpenSession1(context,
                        &session,
                        &ta_uuid,
                        path,
                        0,
                        NULL,
                        NULL,
                        &returnOrigin);
```

2.1.3.8 TEEC_Result TEECS_SetCluster (TEEC_Session * session, Cluster cluster)

```
#include <tees_client_api.h>
```

The function sets TA (main thread and its descendants, created after call) to desired cpu cluster.

Parameters

in	<i>session</i>	the session of a TA.
in	<i>cluster</i>	one of the value CLUSTER_LITTLE, CLUSTER_BIG, or CLUSTER_DEFAULT.

Return values

<i>TEEC_SUCCESS</i>	in case of success.
<i>TEEC_ERROR_BAD_PARAMETERS</i>	invalid parameters.

Example of usage: Use this function carefully. It will affect main thread (and its descendants) for any subsequent TEEC_* call till session termination. Use 'TEECS_SetCluster(session, CLUSTER_DEFAULT)' when you don't need any specific cluster anymore.

TODO: provide workable example

2.1.3.9 TEEC_Result TEECS_SetCryptoClk (TEEC_Context * context, uint32_t state)

```
#include <tees_client_api.h>
```

The function switches crypto clocks ON/OFF Initially designed to reduce power consumption on Qualcomm chipsets A reference counter is maintained in TEEC_ContextImp.

Parameters

in	<i>context</i>	session contex.
in	<i>state</i>	one of the value DISABLE_CRYPTOK, ENABLE_CRYPTOK.

Return values

<i>TEEC_SUCCESS</i>	in case of success.
---------------------	---------------------

Example of usage:

TODO: provide workable example

2.1.3.10 TEEC_Result TEECS_TADebugServiceCtl (ta_debug_service_type_t type, const TEEC_UUID * monitoredTA, const TEEC_UUID * monitoredSession)

```
#include <tees_client_api.h>
```

Create a request into underlying TEE implementation to provide TEE TA Debug functionality for current Client Application.

Parameters

in	<i>type</i>	Specifies what kind of TEE TA Debug Service current Client Application wants to get. type is an enumeration of ta_debug_service_type_t .
in	<i>monitoredTA</i>	Pointer to TEEC_UUID. It's a UUID of the Monitored TA. This value is optional and may be Nil UUID to indicate an empty value. If a Nil UUID is specified, then all debuggable sessions on the monitored TEE shall report DLM output to current Client Application and the <i>monitoredSession</i> value is ignored. Refer to TEE TA Debug Specification, 5.3.4 Client Side DLM_State Structure.
in	<i>monitoredSession</i>	Pointer to TEEC_UUID. It's a session ID of connection established between Client Application and its Trusted Application being debugged (a.k.a Monitored TA). Client Application can obtain session ID by means of call to TEECS_GetSessionId() . Refer to TEE TA Debug Specification, 5.3.4 Client Side DLM_State Structure.

Return values

<i>TEEC_SUCCESS</i>	in case of success.
<i>TEEC_ERROR_NOT_IMPLEMENTED</i>	when part of functionality is not implemented yet.
<i>TEEC_ERROR_BAD_PARAMETERS</i>	wrong parameters were passed to the function.
<i>TEEC_ERROR_COMMUNICATION</i>	can't establish communication with service daemon or communication error.
<i>TEEC_ERROR_BAD_STATE</i>	DLM service can not perform operation, because client's stdout and stderr are closed.

Example of usage:

TODO: provide workable example

2.1.3.11 TEEC_Result TEECS_Unboost (TEEC_Session * session)

```
#include <tees_client_api.h>
```

The function disables performance boosting for current TA session.

Parameters

in	<i>session</i>	the session of a TA.
----	----------------	----------------------

Return values

<i>TEEC_SUCCESS</i>	in case of success.
<i>TEEC_ERROR_BAD_PARAMETERS</i>	invalid parameters.

Example of usage:

```
res = TEECS_Boost(&session);
if (res) {
    // Error handling
}

// Invoking commands with boosted performance

res = TEECS_Unboost(&session);
if (res) {
    // Error handling
}
```

2.1.3.12 TEEC_Result TEECS_WaitTADeath (TEEC_Session * session)

```
#include <tees_client_api.h>
```

Wait for unexpected death of TA.

Parameters

in	session	Specifies successfully opened session with monitored TA.
----	---------	--

Return values

<i>TEEC_ERROR_TARGET_DEAD</i>	monitored TA unexpectedly died.
<i>TEEC_ERROR_BAD_PARAMETERS</i>	wrong parameters were passed to the function.
<i>TEEC_ERROR_COMMUNICATION</i>	communication with tzdaemon failed.
<i>TEEC_ERROR_ITEM_NOT_FOUND</i>	client context haven't been found.

3 File Documentation

3.1 tees_client_api.h File Reference

TEE Samsung Client API.

Enumerations

- enum [Cluster](#) { [CLUSTER_BIG](#), [CLUSTER_LITTLE](#), [CLUSTER_DEFAULT](#), [CLUSTER_MAX_VALUE](#) }
- enum { [DISABLE_CRYPTOK](#), [ENABLE_CRYPTOK](#) }
- enum [ta_debug_service_type_t](#) { [TA_DEBUG_SERVICE_DLM](#), [TA_DEBUG_SERVICE_PMR](#) }

Functions

- const char * [get_error_name](#) (TEEC_Result result)
The function returns string representation of [TEEC_Result](#) type.
- const char * [get_error_text](#) (TEEC_Result result)
The function returns description of [TEEC_Result](#) type.
- const char * [get_return_origin](#) (uint32_t returnOrigin)
The function description of [origin](#) type.
- TEEC_Result [TEEC_SetCluster](#) (TEEC_Session *session, [Cluster](#) cluster)
The function sets TA (main thread and its descendants, created after call) to desired cpu cluster.
- TEEC_Result [TEEC_Boost](#) (TEEC_Session *session)
The function enables performance boosting for current TA session.
- TEEC_Result [TEEC_Unboost](#) (TEEC_Session *session)
The function disables performance boosting for current TA session.
- TEEC_Result [TEEC_SetCryptoClk](#) (TEEC_Context *context, uint32_t state)
The function switches crypto clocks ON/OFF Initially designed to reduce power consumption on Qualcomm chipsets A reference counter is maintained in [TEEC_ContextImp](#).
- TEEC_Result [TEEC_GetSessionId](#) (const TEEC_Session *session, TEEC_UUID *session_id)
Function gets session ID.
- TEEC_Result [TEEC_OpenSession](#) (TEEC_Context *context, TEEC_Session *session, const TEEC_UUID *destination, const void *ta_image, const size_t ta_image_size, uint32_t connectionMethod, const void *connectionData, TEEC_Operation *operation, uint32_t *returnOrigin)
The function opens TA which is located not exactly in /vendor/tee directory. Function behaves the same way as [TEEC_OpenSession\(\)](#), but provides two additional parameters. [ta_image](#) and [ta_image_size](#).
- TEEC_Result [TEEC_OpenSession1](#) (TEEC_Context *context, TEEC_Session *session, const TEEC_UUID *destination, const char *destination_path, uint32_t connectionMethod, const void *connectionData, TEEC_Operation *operation, uint32_t *returnOrigin)
The function opens TA which is located not exactly in /vendor/tee directory. Function behaves the same way as [TEEC_OpenSession\(\)](#), but provides one additional parameter. [destination_path](#) (e.g. /system/tee, /data/tee) wherever CA has appropriate permission.
- TEEC_Result [TEEC_TADebugServiceCtl](#) ([ta_debug_service_type_t](#) type, const TEEC_UUID *monitoredTA, const TEEC_UUID *monitoredSession)
Create a request into underlying TEE implementation to provide TEE TA Debug functionality for current Client Application.
- TEEC_Result [TEEC_WaitTADeath](#) (TEEC_Session *session)
Wait for unexpected death of TA.

3.1.1 Detailed Description

TEE Samsung Client API.

Copyright

(C) 2012-2019, Samsung Electronics Co., Ltd.

3 Index

- Cluster
 - Samsung Client API, [5](#)
- DISABLE_CRYPT0_CLK
 - Samsung Client API, [5](#)
- ENABLE_CRYPT0_CLK
 - Samsung Client API, [5](#)
- get_error_name
 - Samsung Client API, [5](#)
- get_error_text
 - Samsung Client API, [5](#)
- get_return_origin
 - Samsung Client API, [6](#)
- Samsung Client API, [4](#)
 - Cluster, [5](#)
 - DISABLE_CRYPT0_CLK, [5](#)
 - ENABLE_CRYPT0_CLK, [5](#)
 - get_error_name, [5](#)
 - get_error_text, [5](#)
 - get_return_origin, [6](#)
 - TA_DEBUG_SERVICE_DLM, [5](#)
 - TA_DEBUG_SERVICE_PMR, [5](#)
 - TEECS_Boost, [6](#)
 - TEECS_GetSessionId, [7](#)
 - TEECS_OpenSession, [7](#)
 - TEECS_OpenSession1, [8](#)
 - TEECS_SetCluster, [9](#)
 - TEECS_SetCryptoClk, [10](#)
 - TEECS_TADebugServiceCtl, [10](#)
 - TEECS_Unboost, [11](#)
 - TEECS_WaitTADeath, [11](#)
 - ta_debug_service_type_t, [5](#)
- TA_DEBUG_SERVICE_DLM
 - Samsung Client API, [5](#)
- TA_DEBUG_SERVICE_PMR
 - Samsung Client API, [5](#)
- TEECS_Boost
 - Samsung Client API, [6](#)
- TEECS_GetSessionId
 - Samsung Client API, [7](#)
- TEECS_OpenSession
 - Samsung Client API, [7](#)
- TEECS_OpenSession1
 - Samsung Client API, [8](#)
- TEECS_SetCluster
 - Samsung Client API, [9](#)
- TEECS_SetCryptoClk
 - Samsung Client API, [10](#)
- TEECS_TADebugServiceCtl
 - Samsung Client API, [10](#)
- TEECS_Unboost
 - Samsung Client API, [11](#)
- TEECS_WaitTADeath
 - Samsung Client API, [11](#)
- ta_debug_service_type_t
 - Samsung Client API, [5](#)