



**Global  
Platform®**

The standard for  
secure digital services  
and devices

GlobalPlatform Technology

# TEE API Call Validation

Version 0.0.0.9

Public Review

November 2024

Document Reference: GPP\_TEN\_012

**Copyright © 2024 GlobalPlatform, Inc. All Rights Reserved.**

*Recipients of this document are invited to submit, with their comments, notification of any relevant patents or other intellectual property rights of which they may be aware which might be necessarily infringed by the implementation of the specification or other work product set forth in this document, and to provide supporting documentation. This document (and the information herein) is subject to updates, revisions, and extensions by GlobalPlatform, and may be disseminated without restriction. Use of the information herein (whether or not obtained directly from GlobalPlatform) is subject to the terms of the corresponding GlobalPlatform license agreement on the GlobalPlatform website (the "License"). Any use (including but not limited to sublicensing) inconsistent with the License is strictly prohibited.*

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Introduction .....</b>                          | <b>4</b> |
| 1.1      | Audience .....                                     | 4        |
| 1.2      | IPR Disclaimer .....                               | 4        |
| 1.3      | References .....                                   | 4        |
| 1.4      | Terminology and Definitions .....                  | 5        |
| 1.5      | Abbreviations and Notations .....                  | 5        |
| 1.6      | Revision History .....                             | 5        |
| <b>2</b> | <b>Overview .....</b>                              | <b>6</b> |
| <b>3</b> | <b>Common Pitfalls Addressed by Proposal .....</b> | <b>7</b> |
| <b>4</b> | <b>API Extension .....</b>                         | <b>8</b> |
| 4.1      | New Types .....                                    | 8        |
| 4.1.1    | TEE_ParameterPolicy .....                          | 8        |
| 4.1.2    | TEE_InvokeTACommandPolicy .....                    | 9        |
| 4.1.3    | TEE_OpenTASessionPolicy .....                      | 9        |
| 4.2      | New Commands .....                                 | 10       |
| 4.2.1    | TEE_RegisterCommandPolicy .....                    | 10       |
| 4.2.2    | TEE_RegisterOpenSessionEntryPointPolicy .....      | 11       |
| 4.2.3    | TEE_RegisterPermittedCaller .....                  | 12       |
| 4.3      | Impact on Existing APIs .....                      | 13       |
| 4.3.1    | TA_OpenSessionEntryPoint .....                     | 13       |
| 4.3.2    | TA_InvokeCommandEntryPoint .....                   | 13       |
| 4.4      | Simplification of User TA Code .....               | 14       |
| 4.5      | Example .....                                      | 15       |

# Tables

|            |                                   |   |
|------------|-----------------------------------|---|
| Table 1-1: | Normative References .....        | 4 |
| Table 1-2: | Abbreviations and Notations ..... | 5 |
| Table 1-3: | Revision History .....            | 5 |

# 1 INTRODUCTION

The aim of this document is to provide a modification to the GlobalPlatform TEE Internal Core API ([TEE Core]). It is applicable to all versions of [TEE Core], and provides an OPTIONAL extension that is recommended to TEE implementors.

**If you are implementing this specification and you think it is not clear on something:**

1. Check with a colleague.

**And if that fails:**

2. Contact GlobalPlatform at [TEE-issues-GPP\\_TEN\\_012@globalplatform.org](mailto:TEE-issues-GPP_TEN_012@globalplatform.org)

6

## 1.1 Audience

This document is intended primarily for the use of TEE implementors and Trusted Application developers.

## 1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of intellectual property rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit <https://globalplatform.org/specifications/ip-disclaimers/>. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

## 1.3 References

The table below lists references applicable to this specification. The latest version of each reference applies unless a publication date or version is explicitly stated.

Table 1-1: Normative References

| Standard / Specification | Description   | Ref        |
|--------------------------|---|------------|
| GPD_SPE_010              | GlobalPlatform Technology<br>TEE Internal Core API Specification (all versions) | [TEE Core] |

20

## 21 1.4 Terminology and Definitions

22 Selected terms used in this document are defined in [TEE Core].

## 23 1.5 Abbreviations and Notations

24 **Table 1-2: Abbreviations and Notations**

| Abbreviation / Notation | Meaning                                  |
|-------------------------|--|
| EPFL                    | École Polytechnique Fédérale de Lausanne |
| REE                     | Regular Execution Environment            |
| TA                      | Trusted Application                      |
| TEE                     | Trusted Execution Environment            |

25

## 26 1.6 Revision History

27 GlobalPlatform technical documents numbered  $n.0$  are major releases. Those numbered  $n.1$ ,  $n.2$ , etc., are  
 28 minor releases where changes typically introduce supplementary items that do not impact backward  
 29 compatibility or interoperability of the specifications. Those numbered  $n.n.1$ ,  $n.n.2$ , etc., are maintenance  
 30 releases that incorporate errata and clarifications; all non-trivial changes are indicated, often with revision  
 31 marks.

32 **Table 1-3: Revision History**

| Date     | Version | Description      |
|----------|---------|------------------|
| Aug 2024 | 0.0.0.4 | Committee Review |
| Aug 2024 | 0.0.0.7 | Member Review    |
| Nov 2024 | 0.0.0.9 | Public Review    |
| TBD      | TBD     | Public Release   |

33

## 34 2 OVERVIEW

---

35 The GlobalPlatform TEE Internal Core API ([TEE Core]) describes how calls to the Trusted Application (TA)  
36 are made. In the current [TEE Core], the TA developer is responsible for validating that the caller is as expected  
37 and that parameters passed by the caller are of the appropriate types and values.

38 Security researchers at EPFL in Switzerland noted that many published vulnerabilities in Trusted Applications  
39 are related to TA developers failing to validate parameter types. Additional discussion at GlobalPlatform  
40 identified other areas of common concern.

41 This document proposes additions to [TEE Core] to allow TA developers to declare the intended usage of their  
42 APIs, and have the TEE perform validation for them. It is believed this will reduce the likelihood of future  
43 vulnerabilities. An additional advantage is that the proposed changes will simplify the code the developer has  
44 to write, aiding readability.

45 The proposed changes can be applied to any version of [TEE Core]. While GlobalPlatform intends to add the  
46 new capabilities into the next iteration of the API as an optional extension, the change are published here to  
47 encourage TEE vendors to adopt them early, even if their TEE provides an earlier version of the API.

48

### 49 3 COMMON PITFALLS ADDRESSED BY PROPOSAL

---

- 50
- 51
- 52
- 53
- 54
- 55
- Trusted Application developers are usually responsible for validating that each parameter passed to TA\_OpenSessionEntryPoint or TA\_InvokeCommandEntryPoint is of the appropriate type. Failure to check is a security error that can be and has been exploited. This proposal allows a developer to declare the expected types and have the TEE validate them, avoiding risks of errors/omissions in developer validation code, and additionally avoiding the need for the developer to write this code.
- 56
- 57
- 58
- 59
- 60
- Trusted Application developers are responsible for copying parameters to (private) secure world memory in order to prevent an attack where a shared buffer is modified during operation. Similarly a TA that assembles a result to pass to a client needs to be wary of an attacker viewing or modifying it prior to completion of the call. This proposal allows a developer to indicate when the TEE should copy/isolate buffer parameters.
- 61
- 62
- Trusted Application developers may fail to validate that a buffer parameter is within a given size range. This proposal allows minimum and maximum sizes for each parameter to be specified.
- 63
- 64
- 65
- Trusted Application developers are responsible for validating who the caller is and restricting access for unexpected callers. Although in the general case this logic can be complex, this proposal provides a simple means for listing good callers and having the TEE prevent access by others.
- 66

## 67 4 API EXTENSION

68 The proposed API extension defines a few new calls that may be made by a TA developer, in  
69 TA\_CreateEntryPoint, to register policy for callers and/or commands. Calling these new APIs enables the  
70 additional validation logic.

71

### 72 4.1 New Types

#### 73 4.1.1 TEE\_ParameterPolicy

74 A set of one to four TEE\_ParameterPolicy defines the policy for a single parameter.

- 75 • type One of the TEE\_PARAM\_TYPE\_\* values listed in [TEE Core] section 4.2.1.
- 76 • mayBeShared If false, then the TEE must isolate the buffer from the external caller.  
77 If true, then the buffer may or may not be shared, depending on the TEE  
78 implementation.  
79 Note that isolation must be applied for both REE and TEE callers. A simple approach  
80 would be to copy the parameter to/from TA private memory, although implementations  
81 are free to use other approaches such as manipulation of memory firewalls.
- 82 • minSize Minimum buffer size for any TEE\_PARAM\_TYPE\_MEMREF\_\* type. Ignored for other  
83 types.
- 84 • maxSize Maximum buffer size for any TEE\_PARAM\_TYPE\_MEMREF\_\* type, or 0 indicating  
85 unbounded size. Ignored for other types.

86

```
87 typedef struct {  
88     uint32_t    type;  
89     bool        mayBeShared;  
90     size_t      minSize;  
91     size_t      maxSize; // 0=don't check  
92 } TEE_ParameterPolicy;
```

93

94 *Note that TEE\_ParameterPolicy is constructed to allow default values to be omitted in static initializers, for*  
95 *brevity and readability.*

96



97 **4.1.2 TEE\_InvokeTACommandPolicy**

98 TEE\_InvokeTACommandPolicy indicates a valid policy for a call to TEE\_InvokeTACommand.

- 99
- commandId The identifier of the command this policy relates to.
  - params[4] A valid set of parameter policies to use with this commandId.
- 100

101

```

102 typedef struct {
103     uint32_t      commandId;
104     TEE_ParameterPolicy params[4];
105 } TEE_InvokeTACommandPolicy;
    
```

106

107 **4.1.3 TEE\_OpenTASessionPolicy**

108 TEE\_OpenTASessionPolicy indicates a valid policy for a call to TEE\_OpenTASession.

- 109
- params[4] A valid set of parameter policies to use in a call to TEE\_OpenTASession.
- 110

110

```

111 typedef struct {
112     TEE_ParameterPolicy params[4];
113 } TEE_OpenTASessionPolicy;
    
```

114

## 115 4.2 New Commands

### 116 4.2.1 TEE\_RegisterCommandPolicy

```

117 TEE_Result TEE_RegisterCommandPolicy(
118     const TEE_InvokeTACommandPolicy *policy,
119     size_t numPolicy);
    
```

#### 120 Description

121 The `TEE_RegisterCommandPolicy` function may only be called in `TA_CreateEntryPoint` or a  
 122 subroutine called from it. The function registers a set of permitted command policies and modifies the behavior  
 123 of any subsequent call to `TA_InvokeCommandEntryPoint`.

#### 124 Note

- 125 • If the function is not called, then no checking/isolating of parameters to  
 126 `TA_OpenSessionEntryPoint` takes place.
- 127 • The function may be called more than once, and all policies are applied.
- 128 • It is an error to register more than one policy for a (commandId, parameter type combination).
- 129 • In general, is advisable to only have one policy per `commandId`, in order to keep code simple and  
 130 readable; however, multiple policies for the same `commandId` with different parameter types are  
 131 permitted in order to allow legacy code to be migrated to use the new capabilities.
- 132 • If `TEE_RegisterCommandPolicy` is called one or more times, then the TEE implementation must  
 133 validate all calls to `TEE_InvokeCommandEntryPoint` to ensure that the policy is met.

#### 134 Parameters

- 135 • `policy` A pointer to an array of policies. The array must not be freed until  
 136 `TA_DestroyEntryPoint` is called. (Typically, the array will be statically defined.)
- 137 • `numPolicies` The size of the policy array.

#### 138 Return Value

139 `TEE_SUCCESS` if the policy is acceptable.

140 In other cases, the effect of the call is a 'no op' with a returned error code indicating the issue:

- 141 • `TEE_ERROR_BAD_PARAMETERS` if the policy conflicts with one previously registered, or if two or more  
 142 entries within the policy conflict with each other.
- 143 • `TEE_ERROR_OUT_OF_MEMORY` if the policy cannot be stored.

144

## 145 4.2.2 TEE\_RegisterOpenSessionEntryPointPolicy

```

146 TEE_Result TEE_RegisterOpenSessionEntryPointPolicy(
147     const TEE_OpenTASessionPolicy *policy,
148     size_t numPolicy);
    
```

### 149 Description

150 The `TEE_RegisterOpenSessionEntryPointPolicy` function may only be called in  
 151 `TA_CreateEntryPoint` or a subroutine called from it. The function registers a set of permitted policies for  
 152 the open session operation and modifies the behavior of any subsequent call to  
 153 `TA_OpenSessionEntryPoint`.

### 154 Note

- 155 • If the function is not called, then no checking/isolating of parameters to  
 156 `TA_OpenSessionEntryPoint` takes place.
- 157 • The function may be called only once. (An exception being if all previous calls returned an error.)
- 158 • It is an error to register more than one policy for each set of parameter types.
- 159 • If `TEE_RegisterOpenSessionEntryPointPolicy` is called, then the TEE implementation must  
 160 validate all calls to `TA_OpenSessionEntryPoint` to ensure that the policy is met.

### 161 Parameters

- 162 • `policy` A pointer to an array of policies. The array must not be freed until  
 163 `TA_DestroyEntryPoint` is called. (Typically, the array will be statically defined.)
- 164 • `numPolicies` The size of the policy array.

### 165 Return Value

166 `TEE_SUCCESS` if the policy is acceptable.

167 In other cases, the effect of the call is a 'no op' with a returned error code indicating the issue:

- 168 • `TEE_ERROR_BAD_PARAMETERS` if the function has already been called, or if two or more entries within  
 169 the policy conflict with each other.
- 170 • `TEE_ERROR_OUT_OF_MEMORY` if the policy cannot be stored.

171

172 **4.2.3 TEE\_RegisterPermittedCaller**173 

```
TEE_Result TEE_RegisterPermittedCaller(const TEE_Identity *caller);
```

174 **Description**

175 The `TEE_RegisterPermittedCaller` function may only be called in `TA_CreateEntryPoint` or a  
176 subroutine called from it. The function registers the identity of a caller permitted to create a session to the TA,  
177 and modifies the behavior of any subsequent call to `TA_OpenSessionEntryPoint`.

178 **Note**

- 179 • If the function is not called, then no checking of the caller is made. In this circumstance a TEE  
180 implementation may choose to log a warning each time a client is about to call  
181 `TA_OpenSessionEntryPoint`, logging the identity of the caller and indicating that no access policy is  
182 in place.
- 183 • The function may be called multiple times to register multiple permitted callers.
- 184 • If `TEE_RegisterPermittedCaller` is called, then the TEE implementation must validate all calls to  
185 `TA_OpenSessionEntryPoint` to ensure that the caller has been registered.
- 186 • Additional registration methods may be added in future to identify clients by other means.

187 **Parameters**

- 188 • `caller` A pointer to a permitted client identity. This identity structure must not be freed until  
189 `TA_DestroyEntryPoint` is called. (Typically, the identity structure will be statically  
190 defined.)

191 **Return Value**

192 `TEE_SUCCESS` if the caller is successfully added to the list of permitted callers.

193 `TEE_ERROR_OUT_OF_MEMORY` if the caller identity cannot be stored.

194

195 **4.3 Impact on Existing APIs**

196 **4.3.1 TA\_OpenSessionEntryPoint**

197 **Impact**

198 If a call to `TEE_RegisterPermittedCaller` was made and the caller is NOT one of the permitted callers,  
 199 then the TEE will return `TEE_ERROR_ACCESS_DENIED` with the call origin `TEE_ORIGIN_TEE`.

200 If a call to `TEE_RegisterOpenSessionEntryPointPolicy` was made, then:

- 201 • If the caller-supplied parameters match a policy included in the list of registered policies, then:
  - 202 ○ If the TEE is able to enforce that policy (e.g. allocate and copy memory), then the policy is enforced
  - 203 and the call proceeds.
  - 204 ○ If the TEE is unable able to enforce that policy (e.g. allocate and copy memory), then an
  - 205 appropriate error (for example `TEE_ERROR_OUT_OF_MEMORY`) is returned with call origin
  - 206 `TEE_ORIGIN_TEE`.
- 207 • If the caller-supplied parameters do not match any of the registered policies, then the TEE will return
- 208 `TEE_ERROR_BAD_PARAMETERS` with the call origin `TEE_ORIGIN_TEE`.

209

210 **4.3.2 TA\_InvokeCommandEntryPoint**

211 **Impact**

212 If no call to `TEE_RegisterCommandPolicy` was made, then the operation of this function is not modified.

213 If the caller supplies a combination of command id and parameters that matches a registered policy, then:

- 214 • If the TEE is able to enforce that policy (e.g. allocate and copy memory), then the policy is enforced
- 215 and the call proceeds.
- 216 • If the TEE is unable to meet the policy, then an appropriate error (typically
- 217 `TEE_ERROR_OUT_OF_MEMORY`) is returned with the call origin `TEE_ORIGIN_TEE`.

218 If the caller supplies a combination of command id and parameters that does not match a registered policy,  
 219 then `TEE_ERROR_BAD_PARAMETERS` is returned with the call origin `TEE_ORIGIN_TEE`.

220

## 221 4.4 Simplification of User TA Code

222 The proposed changes are backward compatible. Existing user code will not be invalidated by application of  
 223 these changes; however, there is scope for simplification of user code.

224

225 • When `TEE_RegisterCommandPolicy` has been used

226 ○ `TEE_InvokeCommandEntryPoint` does not need to check the types of parameters for a given  
 227 `commandId`, unless more than one permitted set of parameter types was registered for that  
 228 `commandId` (which is supported but not recommended).

229 ○ `TEE_InvokeCommandEntryPoint` does not need to copy IN/INOUT parameters to TEE memory,  
 230 or copy INOUT/OUT parameters from TEE memory *unless* the parameter was marked as  
 231 `maybeShared` in the policy in force.

232 ○ Note that it remains the responsibility of the developer to validate the content of any buffer.

233

234 • When `TEE_RegisterOpenSessionEntryPointPolicy` has been used

235 ○ `TA_OpenSessionEntryPoint` does not need to check the types of parameters unless more than  
 236 one permitted set of parameter types was registered.

237 ○ `TA_OpenSessionEntryPoint` does not need to copy IN/INOUT parameters to TEE memory, or  
 238 copy INOUT/OUT parameters from TEE memory *unless* the parameter was marked as  
 239 `maybeShared` in the policy in force.

240 ○ Note that it remains the responsibility of the developer to validate the content of any buffer.

241

242 • When `TEE_RegisterPermittedCaller` has been used

243 ○ `TA_OpenSessionEntryPoint` and `TEE_InvokeCommandEntryPoint` do not need to validate  
 244 the caller for simple access control case.

245 ○ If the TA wishes to restrict calls based on data passed, or other contextual information, it remains  
 246 the responsibility of user code to make that determination.

247

248 **4.5 Example**

249 A portion of an example 'Hello World' TA:

```
250 static const TEE_InvokeTACommandPolicy policy[] = {
251     {
252         .commandId = CMD_SAY_HELLO,
253         .params = {
254             {
255                 .type = TEE_PARAM_TYPE_MEMREF_INPUT,
256                 .minSize = 1,
257                 .maxSize = 1024
258                 // Note maybeShared=false is a default value so may be omitted
259             },{
260                 .type = TEE_PARAM_TYPE_MEMREF_OUTPUT,
261                 .maxSize = 1024,
262                 .maybeShared = true
263             }
264         }
265     },{
266         .commandId = CMD_ADD_TWO_NUMBERS
267         .params {
268             {
269                 .type = TEE_PARAM_TYPE_VALUE_INPUT
270             },{
271                 .type = TEE_PARAM_TYPE_VALUE_OUTPUT
272             }
273         }
274     }
275 };

276
277 static const TEE_Identity caller = {
278     .login = TEE_LOGIN_APPLICATION_USER,
279     .uuid = { 0xfd2a7830U, 0xab65U, 0x565bU,
280             { 0xa4U, 0xc8U, 0x80U, 0x32U, 0x43U, 0xf2U, 0xa5U, 0x09U }}
281 };
282
283 TEE_Result TA_CreateEntryPoint(void)
284 {
285     TEE_RegisterCommandPolicy(policy, sizeof(policy) /
286                               sizeof(TEE_InvokeTACommandPolicy));
287     TEE_RegisterPermittedCaller(& caller);
288     return TEE_SUCCESS;
289 }
```

290