Tizen Plugin Control Service Daemon

Test Specification

Document version 1.0.0

1. Contents

1.1 Document History

Version	Date	Reason
1.0.0	7/1/2014	First draft from McAfee

1.2 References

Ref	Document	Issue	Title
[1]	Tizen Security Framework Plugin IPC	0.0.1	Tizen Security Framework Plugin IPC

1.3 Glossary and definitions

API Application Programming Interface

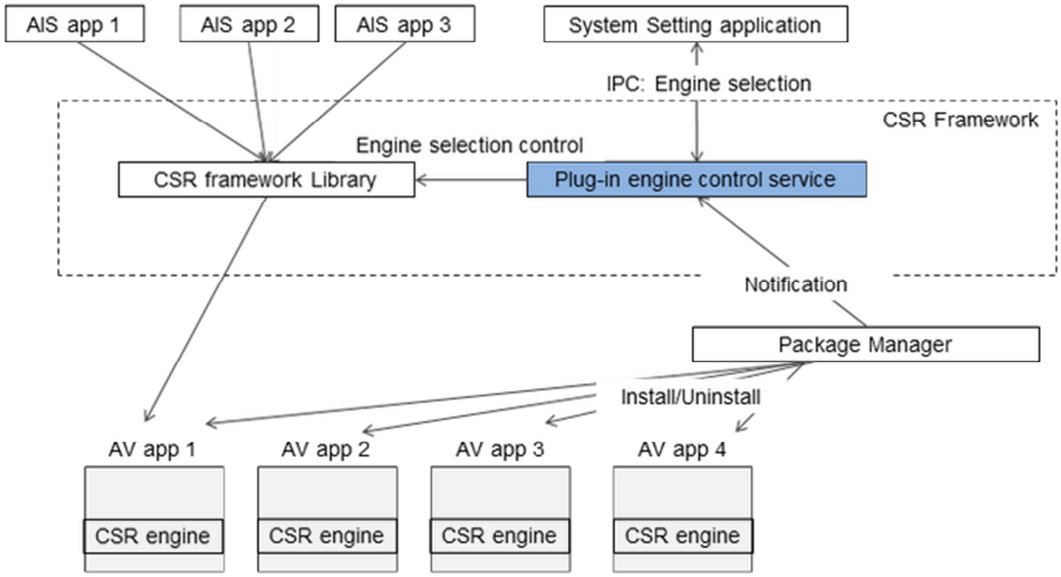
TPCS Tizen Plugin engine Control Service

2. Purpose and Scope

The overall purpose of this document is to describe the test cases for the Tizen Plugin Control Service Daemon. This document includes Test case procedures.

3. Component Description

Plugin engine Control Service



4. Test Environment Description

The test environment used is on Tizen platform.

The following requirements apply to all test cases defined in this document:

1. Any resources required by Tizen Plugin engine Control Service subsystem in runtime should be installed in the test environment.
2. Test samples required by test suite should be installed in the test environment.

5. Test Cases Specifications

5.1 Test Case TC_SEC_TPCS_GetPluginInfo_001

TC_SEC_TPCS_GetPluginInfo_001	**TPCSSerDaemon interface test.**

API Function(s) covered:
 	
`int GetInfoPlugin(void *pData, int req_argc, char **req_argv, char ***res_argv, int *res_argc,`
`CALLBACKFUNC callback,`	

TC_SEC_TPCS_GetPluginInfo_001	TPCSSerDaemon interface test.
TSC_METHOD_HANDLE *handle)	
Test Objectives: This test case verifies that the calling application gets correct result if TPCSSerDaemon is running.	
Test pre-conditions: Daemon running.	
Test Procedure: 1. Start the Daemon. 2. Call TSCSendMessageN() with TPCSGetInfoPlugin 3. Verify the API return value is equal to and number of reply arguments is 2. 4. Verify the status code is RETURN_SUCCESS 5. Verify the content is same as predefined in CONFIG_DEFAULT_STRING	
Test PASS Condition: Step 3, 4 and 5 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop the Daemon.	

5.2 Test Case TC_SEC_TPCS_GetPluginInfo_002

TC_SEC_TPCS_GetPluginInfo_002	TPCSSerDaemon interface test.
API Function(s) covered: int GetInfoPlugin(void *pData, int req_argc, char **req_argv, char ***res_argv, int *res_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle)	
Test Objectives: This test case verifies that if the tpcs_config.xml is invalid, but tpcs_config_new.xml exists, the calling application gets correct result tpcs_config_new.xml if TPCSSerDaemon is running.	
Test pre-conditions:	

TC_SEC_TPCS_GetPluginInfo_002	TPCSSerDaemon interface test.
Daemon running. tpcs_config.xml is invalid and tpcs_config_new.xml exists and is valid.	
Test Procedure: 1. Start the Daemon. 2. Call `TSCSendMessageN()` with `TPCSGetInfoPlugin` 3. Verify the API return value is equal to and number of reply arguments is 2. 4. Verify the status code is RETURN_SUCCESS 5. Verify the content is same as predefined in CONFIG_FILE_NEW which is same as tpcs_config_new.xml 6. Verify the content is same as file stored in /usr/bin/tpcs_config_new.xml	
Test PASS Condition: Step 3, 4, 5 and 6 hould pass.	
Test Clean-up procedure: Cleanup Reply. Stop the Daemon.	

5.3 Test Case TC_SEC_TPCS_GetPluginInfo_003

TC_SEC_TPCS_GetPluginInfo_003	TPCSSerDaemon interface test.

API Function(s) covered:
`int GetInfoPlugin(void *pData, int req_argc, char **req_argv, char ***res_argv, int *res_argc,`
`CALLBACKFUNC callback,`
`TSC_METHOD_HANDLE *handle)`	

Test Objectives: This test case verifies that the calling application gets the default config file when tpcs_config.xml is invalide when TPCSSerDaemon is running.	
Test pre-conditions: Daemon running tpcs_config.xml is invalid	
Test Procedure: 1. Start the Daemon. 2. Call `TSCSendMessageN()` with `TPCSGetInfoPlugin`	

TC_SEC_TPCS_GetPluginInfo_003	TPCSSerDaemon interface test.

3. Verify the API return value is equal to and number of reply arguments is 2.
4. Verify the status code is RETURN_SUCCESS
5. Verify the content is same as predefined in CONFIG_DEFAULT_STRING.

Test PASS Condition:

Step 2, 3, 4 and 5 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop Daemon.

5.4 Test Case TC_SEC_TPCS_GetPluginInfo_004

TC_SEC_TPCS_GetPluginInfo_004	TPCSSerDaemon interface test.

API Function(s) covered:

```
int GetInfoPlugin(void *pData, int req_argc, char **req_argv, char
***res_argv, int *res_argc,
                  CALLBACKFUNC callback,
                  TSC_METHOD_HANDLE *handle)
```

Test Objectives:

This test case verifies that the calling application gets the default config file when tpcs_config.xml is corrupted when TPCSSerDaemon is running.

Test pre-conditions:

Daemon running and tpcs_config.xml is corrupted.

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with `TPCSGetInfoPlugin`
3. Verify the API return value is equal to and number of reply arguments is 2.
4. Verify the status code is RETURN_SUCCESS
5. Verify the content is same as predefined in CONFIG_DEFAULT_STRING.

Test PASS Condition:

Step 2, 3, 4 and 5 should pass.

TC_SEC_TPCS_GetPluginInfo_004	TPCSSerDaemon interface test.
Test Clean-up procedure: Cleanup Reply. Stop the Daemon.	

5.5 Test Case TC_SEC_TPCS_GetPluginInfo_005

TC_SEC_TPCS_GetPluginInfo_005	TPCSSerDaemon interface test.

API Function(s) covered:
 	
`int GetInfoPlugin(void *pData, int req_argc, char **req_argv, char ***res_argv, int *res_argc,`
`CALLBACKFUNC callback,`
`TSC_METHOD_HANDLE *handle)`	

Test Objectives: This test case verifies that the calling application gets default tpcs_config.xml when there is no tpcs_config.xml or tpcs_config_new.xml and TPCSSerDaemon is running.	
Test pre-conditions: Daemon running. tpcs_config.xml and tpcs_config_new.xml not exists.	
Test Procedure: 1. Start the Daemon. 2. Call `TSCSendMessageN()` with `TPCSGetInfoPlugin` 3. Verify the API return value is equal to and number of reply arguments is 2. 4. Verify the status code is RETURN_SUCCESS 5. Verify the content is same as predefined in CONFIG_DEFAULT_STRING.	
Test PASS Condition: Step 2, 3, 4 and 5 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop Daemon.	

5.6 Test Case TC_SEC_TPCS_GetPluginInfo_006

TC_ SEC_TPCS_GetPluginInfo_006	TPCSSerDaemon interface test.

API Function(s) covered:

```
int GetInfoPlugin(void *pData, int req_argc, char **req_argv, char
***res_argv, int *res_argc,
                  CALLBACKFUNC callback,
                  TSC_METHOD_HANDLE *handle)
```

Test Objectives:

This test case verifies that the calling application gets success response when there is no tpcs_config.xml or tpcs_config_new.xml and after there is a plugin installed and TPCSSerDaemon is running.

Test pre-conditions:

Daemon running.

tpcs_config.xml and tpcs_config_new.xml not exists

There is a plugin application.

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with `TPCSGetInfoPlugin`
3. Verify the API return value is equal to and number of reply arguments is 2.
4. Verify the status code is RETURN_SUCCESS

Test PASS Condition:

Step 2, 3 and 4 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop Daemon.

5.7 Test Case TC_SEC_TPCS_InstallPlugin_001

TC_SEC_TPCS_InstallPlugin_001	TPCSSerDaemon interface test.

API Function(s) covered:

```
int InstallPlugin(void *pData, int req_argc, char **req_argv, char
***res_argv, int *res_argc,
```

TC_SEC_TPCS_InstallPlugin_001	TPCSSerDaemon interface test.

```
                    CALLBACKFUNC callback,
                    TSC_METHOD_HANDLE *handle)
```

Test Objectives:

This test case verifies the installed plugin successfully installed while TPCSSerDaemon running

Test pre-conditions:

Daemon running.

After successfully installed a plugin application

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with `TPCSInstallPlugin`
3. Verify the API return value is equal to and number of reply arguments is 2.
4. Verify the status code is RETURN_SUCCESS
5. Verify the tpcs_config.xml has the plugin node
6. Verify the tpcs_config.xml active plugin node is the one installed.

Test PASS Condition:

Step 2, 3, 4, 5 and 6 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop Daemon.

5.8 Test Case TC_SEC_TPCS_InstallPlugin_002

TC_SEC_TPCS_InstallPlugin_002	TPCSSerDaemon interface test.

API Function(s) covered:

```
int InstallPlugin(void *pData, int req_argc, char **req_argv, char
***res_argv, int *res_argc,
                    CALLBACKFUNC callback,
                    TSC_METHOD_HANDLE *handle)
```

Test Objectives:

This test case verifies the two plugins info get updated successfully while TPCSSerDaemon running

Test pre-conditions:

TC_SEC_TPCS_InstallPlugin_002	TPCSSerDaemon interface test.
Daemon running. Two installed plugin application.	
Test Procedure: 1. Start the Daemon. 2. Call `TSCSendMessageN()` with `TPCSInstallPlugin` 3. Verify the API return value is equal to and number of reply arguments is 2. 4. Verify the status code is RETURN_SUCCESS 5. Verify the tpcs_config.xml has the plugin node 6. Call `TSCSendMessageN()` with `TPCSInstallPlugin` 7. Verify the API return value is equal to and number of reply arguments is 2. 8. Verify the status code is RETURN_SUCCESS 9. Verify the tpcs_config.xml has the plugin node	
Test PASS Condition: Step 2, 3, 4, 5, 6, 7, 8 and 9 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop the Daemon.	

5.9 Test Case TC_SEC_TPCS_InstallPlugin_003

TC_SEC_TPCS_InstallPlugin_003	TPCSSerDaemon interface test.

API Function(s) covered:
 	
`int InstallPlugin(void *pData, int req_argc, char **req_argv, char ***res_argv, int *res_argc,`
`CALLBACKFUNC callback,`
`TSC_METHOD_HANDLE *handle)`	

Test Objectives: This test case verifies the plugins get upgraded successfully while TPCSSerDaemon running	
Test pre-conditions: Daemon running.	

TC_SEC_TPCS_InstallPlugin_003	TPCSSerDaemon interface test.
Upgrade a plugin application	
Test Procedure: 1. Start the Daemon. 2. Call `TSCSendMessageN()` with `TPCSInstallPlugin` 3. Verify the API return value is equal to and number of reply arguments is 2. 4. Verify the status code is RETURN_SUCCESS 5. Verify the tpcs_config.xml has the plugin node	
Test PASS Condition: Step 2, 3, 4, and 5 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop the Daemon.	

5.10 Test Case TC_SEC_TPCS_InstallPlug_004

TC_SEC_TPCS_InstallPlugin_006	TPCSSerDaemon interface test.

API Function(s) covered:
`int InstallPlugin(void *pData, int req_argc, char **req_argv, char ***res_argv, int *res_argc,`
`CALLBACKFUNC callback,`
`TSC_METHOD_HANDLE *handle)`	

Test Objectives: This test case verifies install a no-exist plugins get failure while TPCSSerDaemon running	
Test pre-conditions: Daemon running. Pass a no exist plugin	
Test Procedure: 1. Start the Daemon. 2. Call `TSCSendMessageN()` with `TPCSInstallPlugin` 3. Verify the API return value is equal to and number of reply arguments is 1. 4. Verify the status code is RETURN_FAILURE	

TC_SEC_TPCS_InstallPlugin_006	TPCSSerDaemon interface test.

5. Verify the tpcs_config.xml has not such plugin node

Test PASS Condition:

Step 2, 3, 4 and 5 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

5.11 Test Case TC_SEC_TPCS_UninstallPlugin_001

TC_SEC_TPCS_UninstallPlugin_001	TPCSSerDaemon interface test.

API Function(s) covered:

```
int UninstallPlugin(void *pData, int req_argc, char **req_argv, char
***res_argv, int *res_argc,
                    CALLBACKFUNC callback,
                    TSC_METHOD_HANDLE *handle)
```

Test Objectives:

This test case verifies uninstall a plug successfully while TPCSSerDaemon running

Test pre-conditions:

Daemon running.

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with `TPCSUninstallPlugin`
3. Verify the API return value is equal to and number of reply arguments is 2.
4. Verify the status code is RETURN_SUCCESS

Test PASS Condition:

Step 2, 3 and 4 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

5.12 Test Case TC_SEC_TPCS_SetActivePlugin_001

TC_SEC_TPCS_SetActivePlugin_001	TPCSSerDaemon interface test.

API Function(s) covered:

```
int SetActivePlugin(void *pData, int req_argc, char **req_argv, char
***res_argv, int *res_argc,
                    CALLBACKFUNC callback,
                    TSC_METHOD_HANDLE *handle)
```

Test Objectives:

This test case verifies that plugin was set active while TPCSSerDaemon running

Test pre-conditions:

Daemon running.

Plugin to set active

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with `TPCSSetActivePlugin`
3. Verify the API return value is equal to and number of reply arguments is 2.
4. Verify the status code is RETURN_SUCCESS

Test PASS Condition:

Step 2, 3 and 4 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

5.13 Test Case TC_SEC_TPCS_SetActivePlugin_002

TC_SEC_TPCS_SetActivePlugin_002	TPCSSerDaemon interface test.

TC_SEC_TPCS_SetActivePlugin_002	TPCSSerDaemon interface test.

API Function(s) covered:

```
int SetActivePlugin(void *pData, int req_argc, char **req_argv, char
***res_argv, int *res_argc,
                    CALLBACKFUNC callback,
                    TSC_METHOD_HANDLE *handle)
```

Test Objectives:

This test case verifies that no-exist plugin was set active unsuccessfully while TPCSSerDaemon running

Test pre-conditions:

Daemon running.

No exist plugin

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with `TPCSSetActivePlugin`
3. Verify the API return value is equal to and number of reply arguments is 1.
4. Verify the status code is RETURN_FAILURE
5. Verify the tpcs_config.xml active plugin does not have this plugin node

Test PASS Condition:

Steps 2 to 5 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

1. Test Guide

To run test cases, we need to have:

- TPCSSerDaemon for test purpose
- Test contents
- Test cases

All test contents, test cases and test TPCSSerDaemon will be provided as a test suite along with script file which will automate the test process.

2. Test Contents

Test Library	Its TCSGetInfo Return
n709.so	<Root><Plug><Version>2.2.1</Version><VendorName>NQ</VendorName><ProductName>What'sApp</ProductName><AppId>n7097a278m</AppId></Plug></Root>
u709.so	<?xml version=\"1.0\" encoding=\"UTF-8\"?> \ <Root><Plug> \ <Version>2.2.1</Version> \ <VendorName>Symantec</VendorName> \ <ProductName>WP</ProductName> \ <AppId>u7097a278m</AppId> \ </Plug></Root>
q709.so	<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n\ <Root><Plug>\n\ <Version>3.2.1</Version>\n\ <VendorName>McAfee</VendorName>\n\ <ProductName>TPCS</ProductName>\n\ <AppId>q7097a278m</AppId>\n\ </Plug></Root>\n
u709v4.2.1.so	<Root><Plug><Version>2.2.1</Version><VendorName>NQ</VendorName><ProductName>What'sApp</ProductName><AppId>n7097a278m</AppId></Plug></Root>