Tizen Web Protection

Test Specification

Document version 1.1.2

1 Contents

1.1 Document History

Version	Date	Reason
1.0.0	11/28/2012	First draft from McAfee
1.0.1	01/26/2013	Add license
1.1.0	13/03/2014	Add testcases for API TWPCheckURL
1.1.1	27/3/2014	Add testcases for API TWPGetVersion
1.1.2	06/24/2014	Add testcases for API TWPGetInfo

1.2 References

Ref	Document	Issue	Title
[1]	Tizen Web Protection API Specification	1.0.2	Tizen Web Protection API Specification

1.3 Glossary and definitions

API Application Programming Interface

TWP Tizen Web Protection

2 Purpose and Scope

The overall purpose of this document is to describe the conformance test cases for the Tizen Web Protection framework.

This document shall include:

1. Tizen Web Protection Test Configuration
2. Test Case procedures

The scope of this document is the Tizen Web Protection Foundation API functions that are common to all Web Protection implementations. Specific functions of the Web Protection plug-in are not tested. All TWP implementations must include and meet the test cases defined in this document.

TWP validation plug-in

- A security plug-in for Tizen Web Protection Framework validation. Includes the functionalities required for the validation, including scanning, and conforms to the TWP framework API specification.

3 Component Description

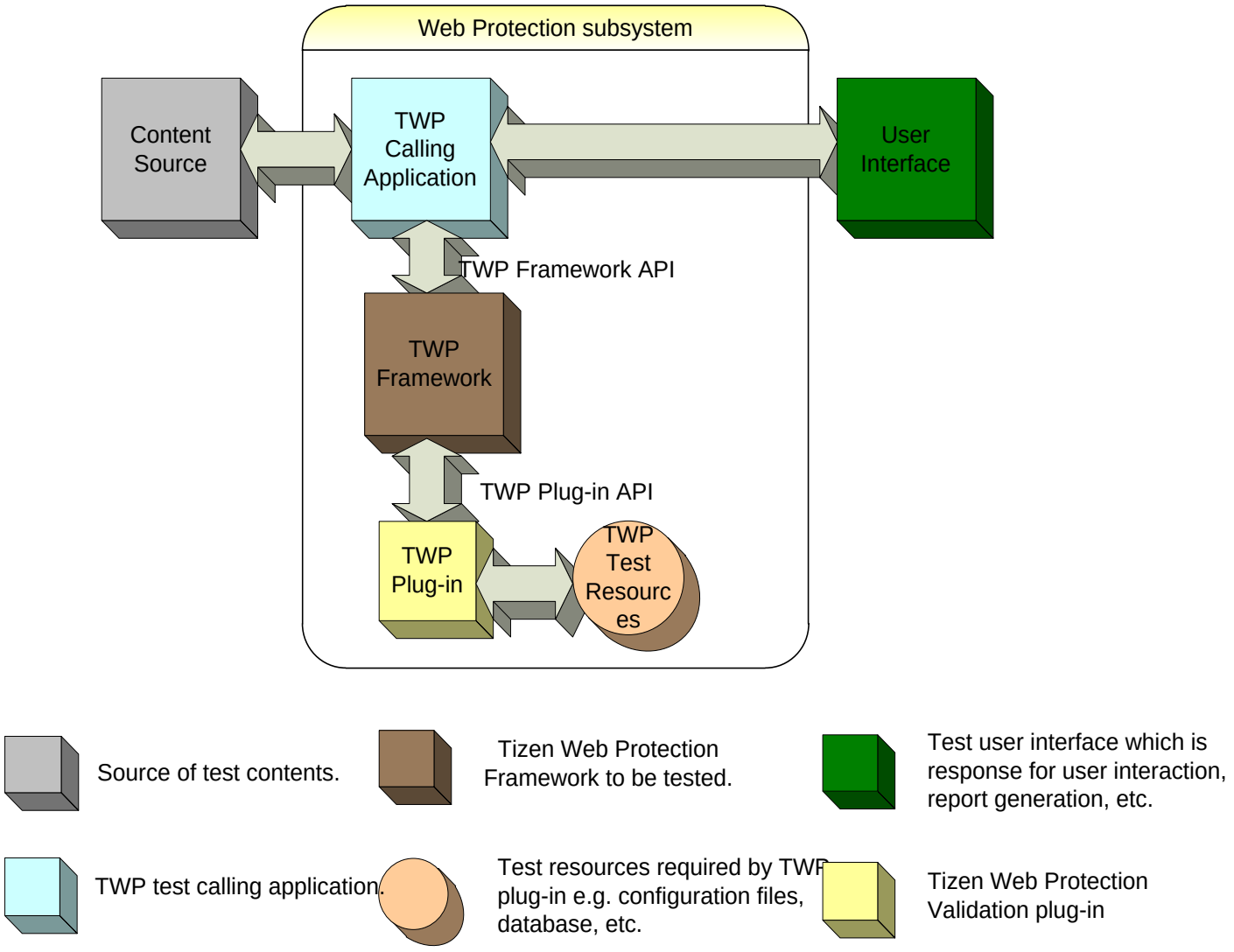

Figure 1: Tizen Web Protection Architecture

The TWP framework (here on will be referenced as “tizen web protectiion library”, “TWP library”) works (interacts) with the calling application through an interface identified as one of the main elements to be tested in this test specification.

TWP plug-in is the web protection function implementation interfacing the TWP framework via Tizen web protection Framework API functions.

“TWP Test Resources” is the resource data used by the TWP plug-in for test purposes (e.g. configurations, test URL database, etc.).

For testing purposes, the TWP library can be interchanged with a test tool. Rather than using software to analyze the content from the calling application and return the result of the scanning, a test tool is used to return the desired result matching the input content and the test case under execution. The test tool should also analyze the request from the calling application implementation to check that the process and the implementation is successful in both of the following ways:

1. The input content received from the calling application triggers the lookup process.
2. The result of the lookup APIs must be understood by the calling application which should take an action with the received content:
   a) Do nothing if the content is correct, or
   b) Request more information from the TWP library (by the test tool).

This test tool can generate a log file with the result of the performed tests for checking purposes.

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 Web Protection 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_WP_TWPInitLibrary_0001

TC_SEC_WP_TWPInitLibrary_0001	**TWP library interface initialization test.**

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPAPIInit *pApiInit);`
`int TWPUninitLibrary(TWPLIB_HANDLE hLib);`	

Test Objectives: This test case verifies that the calling application can correctly initialize the TWP library handle.	
Test pre-conditions: validation plug-in	
Test Procedure: 1. Call `TWPInitLibrary ()`. 2. Verify the API return `value`.	
Test PASS Condition: Step 2 should return valid TWPLIB_HANDLE instead of INVALID_TWPLIB_HANDLE.	
Test Clean-up procedure: Call TWPUninitLibrary() with the TWP library handle returned by TWPInitLibrary ().	

5.2 Test Case TC_SEC_WP_TWPInitLibrary_0002

TC_SEC_WP_TWPInitLibrary_0002	**TWP library interface initialization test.**
API Function(s) covered: `TWPLIB_HANDLE TWPInitLibrary(TWPAPIInit *pApiInit);`	
Test Objectives: This test case verifies that the calling application can get proper error when there is no TWP plugin found in system.	
Test pre-conditions: Stub functions	
Test Procedure: 1. Call `TWPInitLibrary()`. 2. Verify it returns INVALID_TWPLIB_HANDLE.	

TC_SEC_WP_TWPInitLibrary_0002	TWP library interface initialization test.
Test PASS Condition: Step 2 should return valid INVALID_TWPLIB_HANDLE.	
Test Clean-up procedure: None.	

5.3 Test Case TC_SEC_WP_TWPInitLibrary_0003

TC_SEC_WP_TWPInitLibrary_0003	TWP library replacement test.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary(TWPInitApi *pApiInit);`
`void TWPUninitLibrary(TWPLIB_HANDLE hLib);`	

Test Objectives: This test case verifies that the calling application can get always get the latest TWP library API call after close/open.	
Test pre-conditions: Stub functions	
Test Procedure: 1. Call `TWPInitLibrary ().` 2. Verify it returns INVALID_TWPLIB_HANDLE. 3. Copy validation plug-in to "/opt/usr/share/sec_plugin" 4. Call `TWPInitLibrary ().` 5. Verify it returns valid TWP library handle. 6. Call `TWPUninitLibrary ().`	
Test PASS Condition: Step 2 should pass. Step 5 should pass.	
Test Clean-up procedure: None.	

5.4 Test Case TC_SEC_WP_TWPInitLibrary_0004

TC_SEC_WP_TWPInitLibrary_0004	TWP library replacement test.

TC_SEC_WP_TWPInitLibrary_0004	TWP library replacement test.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);

void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get always get the latest TWP library API call after close/open.

Test pre-conditions:

validation plug-in

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify it returns valid TWP library handle.
3. Delete validation plug-in from "/opt/usr/share/sec_plugin"
4. Call `TWPUninitLibrary ()`.
5. Call `TWPInitLibrary ()`.
6. Verify it returns `INVALID_TWPLIB_HANDLE`.

Test PASS Condition:

Step 2 should pass.

Step 6 should pass.

Test Clean-up procedure:

None.

5.5 Test Case TC_SEC_WP_TWPConfigurationCreate_0001

TC_SEC_WP_TWPConfigurationCreate_0001	**TWP configuration interface initialization.**

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can create the TWP configuration handle.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns valid configuration handle rather than `NULL`.
5. Call `TWPConfigurationDestroy ()`.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.6 Test Case TC_SEC_WP_TWPConfigurationCreate_0002

TC_SEC_WP_TWPConfigurationCreate_0002	TWP configuration interface initialization.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can create the TWP configuration handle.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()` with `pConfigure=NULL`.
4. Verify that the API returns error.
5. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.7 Test Case TC_SEC_WP_TWPConfigurationCreate_0003

TC_SEC_WP_TWPConfigurationCreate_0003	TWP configuration interface initialization.

API Function(s) covered:

```
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
```

TC_SEC_WP_TWPConfigurationCreate_0003	TWP configuration interface initialization.
Test Objectives: This test case verifies that the calling application can create the TWP configuration handle.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPConfigurationCreate ()`. 2. Verify that the API returns error.	
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.8 Test Case TC_SEC_WP_TWPPolicyCreate_0001

TC_SEC_WP_TWPPolicyCreate_0001	TWP policy interface initialization.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPPolicyCreate (TWPLIB_HANDLE hLib,
                            TWPCategories *pCategories,
                            unsigned int uCount,
                            TWPPolicyHandle *phPolicy);
TWP_RESULT TWPPolicyDestroy (TWPLIB_HANDLE hLib,
                             TWPPolicyHandle *phPolicy);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can create the TWP policy handle.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.

TC_SEC_WP_TWPPolicyCreate_0001	TWP policy interface initialization.

2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPPolicyCreate ()` with categories.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Verify `phPolicy` has the valid policy handle.
6. Call `TWPPolicyDestroy ()` to release phPolicy resource.
7. Verify that the API returns `TWP_SUCCESS` rather than error.
8. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

Step 7 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.9 Test Case TC_SEC_WP_TWPPolicyCreate_0002

TC_SEC_WP_TWPPolicyCreate_0002	TWP policy interface initialization.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPPolicyCreate (TWPLIB_HANDLE hLib,
                            TWPCategories *pCategories,
                            unsigned int uCount,
                            TWPPolicyHandle *phPolicy);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can create the TWP policy handle.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.

TC_SEC_WP_TWPPolicyCreate_0002	TWP policy interface initialization.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`. 3. Call `TWPPolicyCreate ()` with `pCategories=NULL`. 4. Verify that the API returns error rather than `TWP_SUCCESS`. 5. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.	
Test PASS Condition: Step 2 should pass verification. Step 4 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.10 Test Case TC_SEC_WP_TWPPolicyCreate_0003

TC_SEC_WP_TWPPolicyCreate_0003	TWP policy interface initialization.

API Function(s) covered:
`TWP_RESULT TWPPolicyCreate (TWPLIB_HANDLE hLib,`
`TWPCategories *pCategories,`
`unsigned int uCount,`
`TWPPolicyHandle *phPolicy);`	

Test Objectives: This test case verifies that the calling application can create the TWP policy handle.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPPolicyCreate ()`. 2. Verify that the API returns error rather than `TWP_SUCCESS`.	
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.11 Test Case TC_SEC_WP_TWPLookupUrls_0001

TC_SEC_WP_TWPLookupUrls_0001	TWP URL lookup interface.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can look up URLs.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()`.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Verify `phResponse` has the valid response handle rather than `NULL`.
8. Call `TWPConfigurationDestroy ()`.
9. Verify that the API returns `TWP_SUCCESS` rather than error.
10. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

TC_SEC_WP_TWPLookupUrls_0001	TWP URL lookup interface.
Test PASS Condition: Step 2 should pass verification. Step 4 should pass verification. Step 6 should pass verification. Step 7 should pass verification. Step 9 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.12 Test Case TC_SEC_WP_TWPLookupUrls_0002

TC_SEC_WP_TWPLookupUrls_0002	TWP URL lookup interface.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,`
`TWPConfiguration *pConfigure,`
`TWPConfigurationHandle *phConfigure);`
`TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,`
`TWPConfigurationHandle hConfigure,`
`TWPRequest *pRequest,`
`int iRedirUrlFlag,`
`const char **ppUrls,`
`unsigned int uCount,`
`TWPResponseHandle *phResponse);`
`TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,`
`TWPConfigurationHandle *phConfigure);`
`void TWPUninitLibrary (TWPLIB_HANDLE hLib);`	

Test Objectives: This test case verifies that the calling application can look up URLs.	
Test pre-conditions: validation plug-in.	

TC_SEC_WP_TWPLookupUrls_0002	TWP URL lookup interface.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with `ppUrls=NULL`.
6. Verify that the API returns error rather than `TWP_SUCCESS`.
7. Call `TWPConfigurationDestroy ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.13 Test Case TC_SEC_WP_TWPLookupUrls_0003

TC_SEC_WP_TWPLookupUrls_0003	TWP URL lookup a-synchronization interface.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
```

TC_SEC_WP_TWPLookupUrls_0003	TWP URL lookup a-synchronization interface.

```
                              unsigned int uCount,
                              TWPResponseHandle *phResponse);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseWrite (TWPLIB_HANDLE hLib,
                               TWPResponseHandle hResponse,
                               const void *pData,
                               unsigned uLength);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can a-synchronized look up URLs.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with `TWPRequest::receivefunc=NULL` and known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseWrite ()` with `uLength=0`.
8. Verify response data is as expected.
9. Call `TWPConfigurationDestroy ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Test Clean-up procedure:

TC_SEC_WP_TWPLookupUrls_0003	TWP URL lookup a-synchronization interface.
No specific cleanup required.	

5.14 Test Case TC_SEC_WP_TWPLookupUrls_0004

TC_SEC_WP_TWPLookupUrls_0004	TWP URL lookup synchronization interface.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can synchronized look up URLs.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with `TWPRequest::receivefunc!=NULL` and known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Verify response data is as expected.
8. Call `TWPResponseDestroy ()`.

TC_SEC_WP_TWPLookupUrls_0004	TWP URL lookup synchronization interface.

9. Verify that the API returns `TWP_SUCCESS` rather than error.
10. Call `TWPConfigurationDestroy ()`.
11. Verify that the API returns `TWP_SUCCESS` rather than error.
12. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 7 should pass verification.

Step 9 should pass verification.

Step 11 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.15 Test Case TC_SEC_WP_TWPLookupUrls_0005

TC_SEC_WP_TWPLookupUrls_0005	TWP URL lookup interface.

API Function(s) covered:

```
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
```

Test Objectives:

This test case verifies that the calling application can get proper error with stub library.

Test pre-conditions:

Stub library.

Test Procedure:

1. Call `TWPLookupUrls ()`.
2. Verify that the API returns error rather than `TWP_SUCCESS`.

TC_SEC_WP_TWPLookupUrls_0005	TWP URL lookup interface.
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.16 Test Case TC_SEC_WP_TWPGetUrlRating_0001

TC_SEC_WP_TWPGetUrlRating_0001	TWP get URL rating interface.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);

TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByIndex (TWPLIB_HANDLE hLib,
                                           TWPResponseHandle hResponse,
                                           unsigned int uIndex,
                                           TWPUrlRatingHandle *phRating);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

TC_SEC_WP_TWPGetUrlRating_0001	TWP get URL rating interface.

Test Objectives:

This test case verifies that the calling application can get rating from response.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ().`
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ().`
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByIndex ().`
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPResponseDestroy ().`
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPConfigurationDestroy ().`
12. Verify that the API returns `TWP_SUCCESS` rather than error.
13. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ().`

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 12 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.17 Test Case TC_SEC_WP_TWPGetUrlRating_0002

TC_SEC_WP_TWPGetUrlRating_0002	TWP get URL rating interface.

TC_SEC_WP_TWPGetUrlRating_0002	TWP get URL rating interface.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get rating from response.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.

TC_SEC_WP_TWPGetUrlRating_0002	TWP get URL rating interface.

6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPResponseDestroy ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPConfigurationDestroy ()`.
12. Verify that the API returns `TWP_SUCCESS` rather than error.
13. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 12 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.18 Test Case TC_SEC_WP_TWPGetUrlRating_0003

TC_SEC_WP_TWPGetUrlRating_0003	TWP get URL rating interface.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
```

TC_SEC_WP_TWPGetUrlRating_0003	TWP get URL rating interface.

```
                              TWPResponseHandle *phResponse);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                                  TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get rating from response.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()` with `pUrl=NULL`.
8. Verify that the API returns error rather than `TWP_SUCCESS`.
9. Call `TWPResponseDestroy ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPConfigurationDestroy ()`.
12. Verify that the API returns `TWP_SUCCESS` rather than error.
13. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

TC_SEC_WP_TWPGetUrlRating_0003	TWP get URL rating interface.
Step 6 should pass verification. Step 8 should pass verification. Step 10 should pass verification. Step 12 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.19 Test Case TC_SEC_WP_TWPGetUrlRating_0004

TC_SEC_WP_TWPGetUrlRating_0004	TWP get URL rating interface.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByIndex (TWPLIB_HANDLE hLib,
                                           TWPResponseHandle hResponse,
                                           unsigned int uIndex,
                                           TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

TC_SEC_WP_TWPGetUrlRating_0004	TWP get URL rating interface.

Test Objectives:

This test case verifies that the calling application can get rating from response.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ().`
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ().`
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByIndex ()` with outbound index.
8. Verify that the API returns error rather than `TWP_SUCCESS`.
9. Call `TWPResponseDestroy ().`
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPConfigurationDestroy ().`
12. Verify that the API returns `TWP_SUCCESS` rather than error.
13. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ().`

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 12 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.20 Test Case TC_SEC_WP_TWPGetUrlRating_0005

TC_SEC_WP_TWPGetUrlRating_0005	TWP get URL rating interface.

TC_SEC_WP_TWPGetUrlRating_0005	TWP get URL rating interface.
API Function(s) covered: `TWP_RESULT TWPResponseGetUrlRatingByIndex (TWPLIB_HANDLE hLib, TWPResponseHandle hResponse, unsigned int uIndex, TWPUrlRatingHandle *phRating);`	
Test Objectives: This test case verifies that the calling application can get rating from response.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPResponseGetUrlRatingByIndex ()`. 2. Verify that the API returns error rather than `TWP_SUCCESS`.	
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.21 Test Case TC_SEC_WP_TWPGetUrlRating_0006

TC_SEC_WP_TWPGetUrlRating_0006	TWP get URL rating interface.
API Function(s) covered: `TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib, TWPResponseHandle hResponse, const char *pUrl, unsigned int uUrlLength, TWPUrlRatingHandle *phRating);`	
Test Objectives: This test case verifies that the calling application can get rating from response.	
Test pre-conditions: Stub library.	
Test Procedure:	

TC_SEC_WP_TWPGetUrlRating_0006	TWP get URL rating interface.

1. Call `TWPResponseGetUrlRatingByUrl ()`.
2. Verify that the API returns error rather than `TWP_SUCCESS`.

Test PASS Condition:

Step 2 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.22 Test Case TC_SEC_WP_TWPGetUrlRatingsCount_0001

TC_SEC_WP_TWPGetUrlRatingsCount_0001	TWP get URL rating count.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByIndex (TWPLIB_HANDLE hLib,
                                           TWPResponseHandle hResponse,
                                           unsigned int uIndex,
                                           TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseGetUrlRatingsCount (TWPLIB_HANDLE hLib,
                                          TWPResponseHandle hResponse,
                                          unsigned int *puCount);
```

TC_SEC_WP_TWPGetUrlRatingsCount_0001	TWP get URL rating count.

```
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get rating count from rating handle.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByIndex ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPResponseGetUrlRatingsCount ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Verify that the `puCount` contain right rating number.
12. Call `TWPResponseDestroy ()`.
13. Verify that the API returns `TWP_SUCCESS` rather than error.
14. Call `TWPConfigurationDestroy ()`.
15. Verify that the API returns `TWP_SUCCESS` rather than error.
16. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 11 should pass verification.

Step 13 should pass verification.

TC_SEC_WP_TWPGetUrlRatingsCount_0001	TWP get URL rating count.
Step 15 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.23 Test Case TC_SEC_WP_TWPGetUrlRatingsCount_0002

TC_SEC_WP_TWPGetUrlRatingsCount_0002	TWP get URL rating count.

API Function(s) covered:
`TWP_RESULT TWPResponseGetUrlRatingsCount (TWPLIB_HANDLE hLib,`
`TWPResponseHandle hResponse,`
`unsigned int *puCount);`	

Test Objectives: This test case verifies that the calling application can get error from stub library.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPResponseGetUrlRatingsCount ()`. 2. Verify that the API returns error rather than `TWP_SUCCESS`.	
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.24 Test Case TC_SEC_WP_TWPGetRedirUrlFor_0001

TC_SEC_WP_TWPGetRedirUrlFor_0001	TWP get redirection URL.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,`
`TWPConfiguration *pConfigure,`	

TC_SEC_WP_TWPGetRedirUrlFor_0001	TWP get redirection URL.

```
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByIndex (TWPLIB_HANDLE hLib,
                                           TWPResponseHandle hResponse,
                                           unsigned int uIndex,
                                           TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseGetRedirUrlFor (TWPLIB_HANDLE hLib,
                                      TWPResponseHandle hResponse,
                                      TWPUrlRatingHandle hRating,
                                      TWPPolicyHandle hPolicy,
                                      char **ppUrl,
                                      unsigned int *piLength);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get redirection URL.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.

TC_SEC_WP_TWPGetRedirUrlFor_0001	TWP get redirection URL.

5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByIndex ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPPolicyCreate ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPResponseGetRedirUrlFor ()`.
12. Verify that the API returns `TWP_SUCCESS` rather than error.
13. Verify that the `ppUrl` contain right URL.
14. Call `TWPPolicyDestroy ()`.
15. Verify that the API returns `TWP_SUCCESS` rather than error.
16. Call `TWPResponseDestroy ()`.
17. Verify that the API returns `TWP_SUCCESS` rather than error.
18. Call `TWPConfigurationDestroy ()`.
19. Verify that the API returns `TWP_SUCCESS` rather than error.
20. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 11 should pass verification.

Step 13 should pass verification.

Step 15 should pass verification.

Step 17 should pass verification.

Step 19 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.25 Test Case TC_SEC_WP_TWPGetRedirUrlFor_0002

TC_SEC_WP_TWPGetRedirUrlFor_0002	TWP get redirection URL.

API Function(s) covered:

```
TWP_RESULT TWPResponseGetRedirUrlFor (TWPLIB_HANDLE hLib,
                                      TWPResponseHandle hResponse,
                                      TWPUrlRatingHandle hRating,
                                      TWPPolicyHandle hPolicy,
                                      char **ppUrl,
                                      unsigned int *piLength);
```

Test Objectives:

This test case verifies that the calling application can get error from stub library.

Test pre-conditions:

Stub library.

Test Procedure:

1. Call `TWPResponseGetRedirUrlFor ()`.
2. Verify that the API returns error rather than `TWP_SUCCESS`.

Test PASS Condition:

Step 2 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.26 Test Case TC_SEC_WP_TWPPolicyValidate_0001

TC_SEC_WP_TWPPolicyValidate_0001	TWP policy validation.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPPolicyCreate (TWPLIB_HANDLE hLib,
                            TWPCategories *pCategories,
                            unsigned int uCount,
                            TWPPolicyHandle *phPolicy);
```

TC_SEC_WP_TWPPolicyValidate_0001	TWP policy validation.

```
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPPolicyValidate (TWPLIB_HANDLE hLib,
                              TWPPolicyHandle hPolicy,
                              TWPRULRatingHandle hRating,
                              Int *piViolated);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
TWP_RESULT TWPPolicyDestroy (TWPLIB_HANDLE hLib,
                             TWPPolicyHandle *phPolicy);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can validate the response and rating against policy.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.

TC_SEC_WP_TWPPolicyValidate_0001	TWP policy validation.

5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPPolicyCreate ()` with proper test categories.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPPolicyValidate ()`.
12. Verify that the API returns `TWP_SUCCESS` rather than error.
13. Verify that the `piViolted` contains 1.
14. Call `TWPPolicyDestroy ()` with proper test categories.
15. Verify that the API returns `TWP_SUCCESS` rather than error.
16. Call `TWPResponseDestroy ()`.
17. Verify that the API returns `TWP_SUCCESS` rather than error.
18. Call `TWPConfigurationDestroy ()`.
19. Verify that the API returns `TWP_SUCCESS` rather than error.
20. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 12 should pass verification.

Step 13 should pass verification.

Step 15 should pass verification.

Step 17 should pass verification.

Step 19 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.27 Test Case TC_SEC_WP_TWPPolicyValidate_0002

TC_SEC_WP_TWPPolicyValidate_0002	TWP policy validation.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPPolicyCreate (TWPLIB_HANDLE hLib,
                            TWPCategories *pCategories,
                            unsigned int uCount,
                            TWPPolicyHandle *phPolicy);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPPolicyValidate (TWPLIB_HANDLE hLib,
                              TWPPolicyHandle hPolicy,
                              TWPRULRatingHandle hRating,
                              Int *piViolated);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
TWP_RESULT TWPPolicyDestroy (TWPLIB_HANDLE hLib,
                             TWPPolicyHandle *phPolicy);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

TC_SEC_WP_TWPPolicyValidate_0002	TWP policy validation.

Test Objectives:

This test case verifies that the calling application can validate the response and rating against policy.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPPolicyCreate ()` with proper test categories.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPPolicyValidate ()`.
12. Verify that the API returns `TWP_SUCCESS` rather than error.
13. Verify that the `piViolted` contains 0.
14. Call `TWPPolicyDestroy ()` with proper test categories.
15. Verify that the API returns `TWP_SUCCESS` rather than error.
16. Call `TWPResponseDestroy ()`.
17. Verify that the API returns `TWP_SUCCESS` rather than error.
18. Call `TWPConfigurationDestroy ()`.
19. Verify that the API returns `TWP_SUCCESS` rather than error.
20. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 12 should pass verification.

TC_SEC_WP_TWPPolicyValidate_0002	TWP policy validation.
Step 13 should pass verification. Step 15 should pass verification. Step 17 should pass verification. Step 19 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.28 Test Case TC_SEC_WP_TWPPolicyValidate_0003

TC_SEC_WP_TWPPolicyValidate_0003	TWP policy validation.

API Function(s) covered:
`TWP_RESULT TWPPolicyValidate (TWPLIB_HANDLE hLib,`
`TWPPolicyHandle hPolicy,`
`TWPRULRatingHandle hRating,`
`Int *piViolated);`	

Test Objectives: This test case verifies that the calling application can error from stub library.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPPolicyValidate ()`. 2. Verify that the API returns `TWP_SUCCESS` rather than error.	
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.29 Test Case TC_SEC_WP_TWPPolicyGetViolations_0001

TC_SEC_WP_TWPPolicyGetViolations_0001	TWP policy validation.

TC_SEC_WP_TWPPolicyGetViolations_0001	TWP policy validation.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPPolicyCreate (TWPLIB_HANDLE hLib,
                            TWPCategories *pCategories,
                            unsigned int uCount,
                            TWPPolicyHandle *phPolicy);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPPolicyGetViolations (TWPLIB_HANDLE hLib,
                                   TWPPolicyHandle hPolicy,
                                   TWPURLRatingHandle hRating,
                                   TWPCategories **ppViolated,
                                   unsigned int *puLength);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
TWP_RESULT TWPPolicyDestroy (TWPLIB_HANDLE hLib,
                             TWPPolicyHandle *phPolicy);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

TC_SEC_WP_TWPPolicyGetViolations_0001	TWP policy validation.

Test Objectives:

This test case verifies that the calling application can validate the response and rating against policy.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPPolicyCreate ()` with proper test categories.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPPolicyGetViolations ()`.
12. Verify that the API returns `TWP_SUCCESS` rather than error.
13. Verify that the `ppViolted` contains correct categories.
14. Verify that the `puLength` contains correct number.
15. Call `TWPPolicyDestroy ()` with proper test categories.
16. Verify that the API returns `TWP_SUCCESS` rather than error.
17. Call `TWPResponseDestroy ()`.
18. Verify that the API returns `TWP_SUCCESS` rather than error.
19. Call `TWPConfigurationDestroy ()`.
20. Verify that the API returns `TWP_SUCCESS` rather than error.
21. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

TC_SEC_WP_TWPPolicyGetViolations_0001	TWP policy validation.
Step 12 should pass verification. Step 13 should pass verification. Step 14 should pass verification. Step 16 should pass verification. Step 18 should pass verification. Step 20 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.30 Test Case TC_SEC_WP_TWPPolicyGetViolations_0002

TC_SEC_WP_TWPPolicyGetViolations_0002	TWP policy validation.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPPolicyCreate (TWPLIB_HANDLE hLib,
                            TWPCategories *pCategories,
                            unsigned int uCount,
                            TWPPolicyHandle *phPolicy);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPPolicyGetViolations (TWPLIB_HANDLE hLib,
                                   TWPPolicyHandle hPolicy,
                                   TWPURLRatingHandle hRating,
                                   TWPCategories **ppViolated,
                                   unsigned int *puLength);
```

TC_SEC_WP_TWPPolicyGetViolations_0002	TWP policy validation.

```
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
TWP_RESULT TWPPolicyDestroy (TWPLIB_HANDLE hLib,
                             TWPPolicyHandle *phPolicy);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can validate the response and rating against policy.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPPolicyCreate ()` with proper test categories.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Call `TWPPolicyGetViolations ()`.
12. Verify that the API returns `TWP_SUCCESS` rather than error.
13. Verify that the `ppViolted` does not contain any category.
14. Call `TWPPolicyDestroy ()` with proper test categories.
15. Verify that the API returns `TWP_SUCCESS` rather than error.

TC_SEC_WP_TWPPolicyGetViolations_0002	TWP policy validation.

16. Call `TWPResponseDestroy ()`.
17. Verify that the API returns `TWP_SUCCESS` rather than error.
18. Call `TWPConfigurationDestroy ()`.
19. Verify that the API returns `TWP_SUCCESS` rather than error.
20. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 12 should pass verification.

Step 13 should pass verification.

Step 15 should pass verification.

Step 17 should pass verification.

Step 19 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.31 Test Case TC_SEC_WP_TWPPolicyGetViolations_0003

TC_SEC_WP_TWPPolicyGetViolations_0003	TWP policy validation.

API Function(s) covered:

```
TWP_RESULT TWPPolicyGetViolations (TWPLIB_HANDLE hLib,
                                   TWPPolicyHandle hPolicy,
                                   TWPURLRatingHandle hRating,
                                   TWPCategories **ppViolated,
                                   unsigned int *puLength);
```

Test Objectives:

This test case verifies that the calling application can get error from stub library.

Test pre-conditions:

Stub library.

TC_SEC_WP_TWPPolicyGetViolations_0003	TWP policy validation.

Test Procedure:

1. Call `TWPPolicyGetViolations ()`.
2. Verify that the API returns error rather than `TWP_SUCCESS`.

Test PASS Condition:

Step 2 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.32 Test Case TC_SEC_WP_TWPRatingGetScore_0001

TC_SEC_WP_TWPRatingGetScore_0001	TWP get URL score.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPUrlRatingGetScore (TWPLIB_HANDLE hLib,
                                 TWPUrlRatingHandle hRating,
                                 int *piScore);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
```

TC_SEC_WP_TWPRatingGetScore_0001	TWP get URL score.

```
                                        unsigned int uUrlLength,
                                        TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get URL score from their rating.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPUrlRatingGetScore ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Verify that the `piScore` contains correct score.
12. Call `TWPResponseDestroy ()`.
13. Verify that the API returns `TWP_SUCCESS` rather than error.
14. Call `TWPConfigurationDestroy ()`.
15. Verify that the API returns `TWP_SUCCESS` rather than error.
16. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

TC_SEC_WP_TWPRatingGetScore_0001	TWP get URL score.
Step 11 should pass verification. Step 13 should pass verification. Step 15 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.33 Test Case TC_SEC_WP_TWPRatingGetScore_0002

TC_SEC_WP_TWPRatingGetScore_0002	TWP get URL score.

API Function(s) covered:
`TWP_RESULT TWPUrlRatingGetScore (TWPLIB_HANDLE hLib,`
`TWPUrlRatingHandle hRating,`
`int *piScore);`	

Test Objectives: This test case verifies that the calling application can get error from stub library.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPUrlRatingGetScore ()`. 2. Verify that the API returns error rather than `TWP_SUCCESS`.	
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.34 Test Case TC_SEC_WP_TWPRatingGetUrl_0001

TC_SEC_WP_TWPRatingGetUrl_0001	TWP get URL from rating.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,`	

TC_SEC_WP_TWPRatingGetUrl_0001	TWP get URL from rating.

```
                                    TWPConfiguration *pConfigure,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPUrlRatingGetUrl (TWPLIB_HANDLE hLib,
                               TWPUrlRatingHandle hRating,
                               const char **ppUrl,
                               unsigned int *puLength);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get URL from their rating.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.

TC_SEC_WP_TWPRatingGetUrl_0001	TWP get URL from rating.

5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPUrlRatingGetUrl ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Verify that the `ppUrl` contains correct URL.
12. Verify that the puLength contains correct URL length.
13. Call `TWPResponseDestroy ()`.
14. Verify that the API returns `TWP_SUCCESS` rather than error.
15. Call `TWPConfigurationDestroy ()`.
16. Verify that the API returns `TWP_SUCCESS` rather than error.
17. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 11 should pass verification.

Step 12 should pass verification.

Step 14 should pass verification.

Step 16 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.35 Test Case TC_SEC_WP_TWPRatingGetUrl_0002

TC_SEC_WP_TWPRatingGetUrl_0002	TWP get URL from rating.

API Function(s) covered:

```
TWP_RESULT TWPUrlRatingGetUrl (TWPLIB_HANDLE hLib,
                               TWPUrlRatingHandle hRating,
```

TC_SEC_WP_TWPRatingGetUrl_0002	TWP get URL from rating.
const char **ppUrl, unsigned int *puLength);	
Test Objectives: This test case verifies that the calling application can get error from stub library.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPUrlRatingGetUrl ()`. 2. Verify that the API returns error rather than `TWP_SUCCESS`.	
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.36 Test Case TC_SEC_WP_TWPRatingGetDLAUrl_0001

TC_SEC_WP_TWPRatingGetDLAUrl_0001	TWP get DLA URL from rating.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,`
`TWPConfiguration *pConfigure,`
`TWPConfigurationHandle *phConfigure);`
`TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,`
`TWPConfigurationHandle hConfigure,`
`TWPRequest *pRequest,`
`int iRedirUrlFlag,`
`const char **ppUrls,`
`unsigned int uCount,`
`TWPResponseHandle *phResponse);`
`TWP_RESULT TWPUrlRatingGetDLAUrl (TWPLIB_HANDLE hLib,`
`TWPUrlRatingHandle hRating,`	

TC_SEC_WP_TWPRatingGetDLAUrl_0001	TWP get DLA URL from rating.

```
                                    const char **ppDlaUrl,
                                    unsigned int *puLength);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                     TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                                TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get DLA URL from their rating.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPUrlRatingGetDLAUrl ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Verify that the `ppDlaUrl` contains correct URL.
12. Verify that the puLength contains correct URL length.
13. Call `TWPResponseDestroy ()`.
14. Verify that the API returns `TWP_SUCCESS` rather than error.
15. Call `TWPConfigurationDestroy ()`.

TC_SEC_WP_TWPRatingGetDLAUrl_0001	TWP get DLA URL from rating.
16. Verify that the API returns `TWP_SUCCESS` rather than error. 17. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.	
Test PASS Condition: Step 2 should pass verification. Step 4 should pass verification. Step 6 should pass verification. Step 8 should pass verification. Step 10 should pass verification. Step 11 should pass verification. Step 12 should pass verification. Step 14 should pass verification. Step 16 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.37 Test Case TC_SEC_WP_TWPRatingGetDLAUrl_0002

TC_SEC_WP_TWPRatingGetDLAUrl_0002	TWP get DLA URL from rating.

API Function(s) covered:
`TWP_RESULT TWPUrlRatingGetDLAUrl (TWPLIB_HANDLE hLib,`
`TWPUrlRatingHandle hRating,`
`const char **ppDlaUrl,`
`unsigned int *puLength);`	

Test Objectives: This test case verifies that the calling application can get error from stub library.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPUrlRatingGetDLAUrl ()`. 2. Verify that the API returns error rather than `TWP_SUCCESS`.	
Test PASS Condition: Step 2 should pass verification.	

TC_SEC_WP_TWPRatingGetDLAUrl_0002	TWP get DLA URL from rating.

Test Clean-up procedure:

No specific cleanup required.

5.38 Test Case TC_SEC_WP_TWPRatingHasCategory_0001

TC_SEC_WP_TWPRatingHasCategory_0001	TWP check category.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPUrlRatingHasCategory (TWPLIB_HANDLE hLib,
                                    TWPUrlRatingHandle hRating,
                                    TWPCategories Category,
                                    int *piPresent);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

TC_SEC_WP_TWPRatingHasCategory_0001	TWP check category.

Test Objectives:

This test case verifies that the calling application can check if rating contains specified category.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPUrlRatingHasCategory ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Verify that the `piPresent` contains 1.
12. Call `TWPResponseDestroy ()`.
13. Verify that the API returns `TWP_SUCCESS` rather than error.
14. Call `TWPConfigurationDestroy ()`.
15. Verify that the API returns `TWP_SUCCESS` rather than error.
16. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 11 should pass verification.

Step 13 should pass verification.

Step 15 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.39 Test Case TC_SEC_WP_TWPRatingHasCategory_0002

TC_SEC_WP_TWPRatingHasCategory_0002	TWP check category.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
                          TWPResponseHandle *phResponse);
TWP_RESULT TWPUrlRatingHasCategory (TWPLIB_HANDLE hLib,
                                    TWPUrlRatingHandle hRating,
                                    TWPCategories Category,
                                    int *piPresent);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                               TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can check if rating contains specified category.

Test pre-conditions:

validation plug-in.

TC_SEC_WP_TWPRatingHasCategory_0002	TWP check category.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPUrlRatingHasCategory ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Verify that the `piPresent` contains 0.
12. Call `TWPResponseDestroy ()`.
13. Verify that the API returns `TWP_SUCCESS` rather than error.
14. Call `TWPConfigurationDestroy ()`.
15. Verify that the API returns `TWP_SUCCESS` rather than error.
16. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 11 should pass verification.

Step 13 should pass verification.

Step 15 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.40 Test Case TC_SEC_WP_TWPRatingHasCategory_0003

TC_SEC_WP_TWPRatingHasCategory_0003	TWP check category.

TC_SEC_WP_TWPRatingHasCategory_0003	TWP check category.

API Function(s) covered:
`TWP_RESULT TWPUrlRatingHasCategory (TWPLIB_HANDLE hLib,`
`TWPUrlRatingHandle hRating,`
`TWPCategories Category,`
`int *piPresent);`	

Test Objectives: This test case verifies that the calling application can error from stub library.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPUrlRatingHasCategory ()`. 2. Verify that the API returns error rather than `TWP_SUCCESS`.	
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.41 Test Case TC_SEC_WP_TWPRatingGetCategories_0001

TC_SEC_WP_TWPRatingGetCategories_0001	TWP get all categories in rating.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,`
`TWPConfiguration *pConfigure,`
`TWPConfigurationHandle *phConfigure);`
`TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,`
`TWPConfigurationHandle hConfigure,`
`TWPRequest *pRequest,`
`int iRedirUrlFlag,`
`const char **ppUrls,`
`unsigned int uCount,`	

TC_SEC_WP_TWPRatingGetCategories_0001	TWP get all categories in rating.

```
                              TWPResponseHandle *phResponse);
TWP_RESULT TWPUrlRatingGetCategories (TWPLIB_HANDLE hLib,
                                      TWPUrlRatingHandle hRating,
                                      TWPCategories **ppCategory,
                                      unsigned int *puLength);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                                  TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get all categories in rating.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPUrlRatingGetCategories ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Verify that the `ppCategories` contains all categories that the rating has.
12. Verify that the `puLength` contains correct length.

TC_SEC_WP_TWPRatingGetCategories_0001	TWP get all categories in rating.

13. Call `TWPResponseDestroy ()`.
14. Verify that the API returns `TWP_SUCCESS` rather than error.
15. Call `TWPConfigurationDestroy ()`.
16. Verify that the API returns `TWP_SUCCESS` rather than error.
17. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 11 should pass verification.

Step 12 should pass verification.

Step 14 should pass verification.

Step 16 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.42 Test Case TC_SEC_WP_TWPRatingGetCategories_0002

TC_SEC_WP_TWPRatingGetCategories_0002	TWP get all categories in rating.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);
TWP_RESULT TWPConfigurationCreate (TWPLIB_HANDLE hLib,
                                   TWPConfiguration *pConfigure,
                                   TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPLookupUrls (TWPLIB_HANDLE hLib,
                          TWPConfigurationHandle hConfigure,
                          TWPRequest *pRequest,
                          int iRedirUrlFlag,
                          const char **ppUrls,
                          unsigned int uCount,
```

TC_SEC_WP_TWPRatingGetCategories_0002	TWP get all categories in rating.

```
                              TWPResponseHandle *phResponse);
TWP_RESULT TWPUrlRatingGetCategories (TWPLIB_HANDLE hLib,
                                      TWPUrlRatingHandle hRating,
                                      TWPCategories **ppCategory,
                                      unsigned int *puLength);
TWP_RESULT TWPConfigurationDestroy (TWPLIB_HANDLE hLib,
                                    TWPConfigurationHandle *phConfigure);
TWP_RESULT TWPResponseGetUrlRatingByUrl (TWPLIB_HANDLE hLib,
                                         TWPResponseHandle hResponse,
                                         const char *pUrl,
                                         unsigned int uUrlLength,
                                         TWPUrlRatingHandle *phRating);
TWP_RESULT TWPResponseDestroy (TWPLIB_HANDLE hLib,
                                TWPResponseHandle *phRespone);
void TWPUninitLibrary (TWPLIB_HANDLE hLib);
```

Test Objectives:

This test case verifies that the calling application can get all categories in rating.

Test pre-conditions:

validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPConfigurationCreate ()`.
4. Verify that the API returns `TWP_SUCCESS` rather than error.
5. Call `TWPLookupUrls ()` with known test URL.
6. Verify that the API returns `TWP_SUCCESS` rather than error.
7. Call `TWPResponseGetUrlRatingByUrl ()`.
8. Verify that the API returns `TWP_SUCCESS` rather than error.
9. Call `TWPUrlRatingGetCategories ()`.
10. Verify that the API returns `TWP_SUCCESS` rather than error.
11. Verify that the `ppCategories` contains no category.
12. Call `TWPResponseDestroy ()`.

TC_SEC_WP_TWPRatingGetCategories_0002	TWP get all categories in rating.

13. Verify that the API returns `TWP_SUCCESS` rather than error.
14. Call `TWPConfigurationDestroy ()`.
15. Verify that the API returns `TWP_SUCCESS` rather than error.
16. Call `TWPUninitLibrary ()` with the TWP library handle returned by `TWPInitLibrary ()`.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 6 should pass verification.

Step 8 should pass verification.

Step 10 should pass verification.

Step 11 should pass verification.

Step 13 should pass verification.

Step 15 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.43 Test Case TC_SEC_WP_TWPRatingGetCategories_0003

TC_SEC_WP_TWPRatingGetCategories_0003	TWP get all categories in rating.

API Function(s) covered:

```
TWP_RESULT TWPUrlRatingGetCategories (TWPLIB_HANDLE hLib,
                                      TWPUrlRatingHandle hRating,
                                      TWPCategories **ppCategory,
                                      unsigned int *puLength);
```

Test Objectives:

This test case verifies that the calling application can get error from stub library.

Test pre-conditions:

Stub library.

Test Procedure:

1. Call `TWPUrlRatingGetCategories ()`.
2. Verify that the API returns error rather than `TWP_SUCCESS`.

Test PASS Condition:

TC_SEC_WP_TWPRatingGetCategories_0003	TWP get all categories in rating.
Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.44 Test Case TC_SEC_WP_TWPCheckURL_0001

TC_SEC_WP_TWPCheckURL_0001	TWP get the risk level.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPCheckURL(TWPLIB_HANDLE hLib, const char *pUrl, char **ppBlkUrl, unsigned int *puBlkUrlLen, int *pRiskLevel);`	

Test Objectives: This test case verifies that the calling application can get riskLevel and redirect URL.	
Test pre-conditions: Validation plug-in.	
Test Procedure: 1. Call `TWPInitLibrary ()`. 2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`. 3. Call `TWPCheckURL () with High risk URL`. 4. Verify that the API returns TWP_SUCCESS. 5. Verify returned risk level is TWP_High and contains a redirect URL with length greater than zero.	
Test PASS Condition: Step 2 should pass verification. Step 4 should pass verification. Step 5 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.45 Test Case TC_SEC_WP_TWPCheckURL_0002

TC_SEC_WP_TWPCheckURL_0002	TWP get the risk level.

TC_SEC_WP_TWPCheckURL_0002	TWP get the risk level.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);

TWP_RESULT TWPCheckURL(TWPLIB_HANDLE hLib, const char *pUrl, char **ppBlkUrl, unsigned int *puBlkUrlLen, int *pRiskLevel);
```

Test Objectives:

This test case verifies that the calling application can get riskLevel and redirect URL.

Test pre-conditions:

Validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ()`.
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPCheckURL () with Medium risk URL`.
4. Verify that the API returns TWP_SUCCESS.
5. Verify returned risk level is TWP_Medium and contains a redirect URL with length greater than zero.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.46 Test Case TC_SEC_WP_TWPCheckURL_0003

TC_SEC_WP_TWPCheckURL_0003	TWP get the risk level.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);

TWP_RESULT TWPCheckURL(TWPLIB_HANDLE hLib, const char *pUrl, char **ppBlkUrl, unsigned int *puBlkUrlLen, int *pRiskLevel);
```

Test Objectives:

This test case verifies that the calling application can get riskLevel and redirect URL.

Test pre-conditions:

Validation plug-in.

TC_SEC_WP_TWPCheckURL_0003	TWP get the risk level.

Test Procedure:

1. Call `TWPInitLibrary ().`
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPCheckURL () with Unverified URL`.
4. Verify that the API returns TWP_SUCCESS.
5. Verify returned risk level is TWP_Unverified and contains a redirect URL with length greater than zero.

Test PASS Condition:

Step 2 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

Test Clean-up procedure:

No specific cleanup required.

5.47 Test Case TC_SEC_WP_TWPCheckURL_0004

TC_SEC_WP_TWPCheckURL_0004	TWP get the risk level.

API Function(s) covered:

```
TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);

TWP_RESULT TWPCheckURL(TWPLIB_HANDLE hLib, const char *pUrl, char **ppBlkUrl, unsigned int *puBlkUrlLen, int *pRiskLevel);
```

Test Objectives:

This test case verifies that the calling application can get riskLevel and redirect URL.

Test pre-conditions:

Validation plug-in.

Test Procedure:

1. Call `TWPInitLibrary ().`
2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`.
3. Call `TWPCheckURL () with Minimal risk URL.`
4. Verify that the API returns TWP_SUCCESS.
5. Verify returned risk level is TWP_Minimal and contains a redirect URL with length greater than zero.

Test PASS Condition:

Step 2 should pass verification.

TC_SEC_WP_TWPCheckURL_0004	TWP get the risk level.
Step 4 should pass verification. Step 5 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.48 Test Case TC_SEC_WP_TWPCheckURL_0005

TC_SEC_WP_TWPCheckURL_0005	TWP get the risk level.
API Function(s) covered: `TWPRiskLevel TWPCheckURL(TWPLIB_HANDLE hLib, const char *pUrl, char **ppBlkUrl, unsigned int *puBlkUrlLen);`	
Test Objectives: This test case verifies that the calling application can get error from stub library.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPCheckURL ()` with `INVALID_TWPLIB_HANDLE`. 2. Verify the API does not return TWP_SUCCESS.	
Test PASS Condition: Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.49 Test Case TC_SEC_WP_TWPGetVersion_0001

TC_SEC_WP_TWPGetVersion_0001	TWP get the Framework and Plugin version.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPGetVersion(TWPLIB_HANDLE hLib, TWPVerInfo *pVerInfo);`	

Test Objectives: This test case verifies that the calling application can get version number of Framework and Plugin.	
Test pre-conditions:	

TC_SEC_WP_TWPGetVersion_0001	TWP get the Framework and Plugin version.
Validation plug-in.	
Test Procedure: 1. Call `TWPInitLibrary ().` 2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`. 3. Call `TWPGetVersion () with handle.` 4. Verify that the API returns TWP_SUCCESS. 5. Verify string length of plugin version is greater than zero. 6. Verify string length of framework version is greater than zero. 7. Verify the framework version matches in format and value with TWP_FRAMEWORK_VERSION.	
Test PASS Condition: Step 2 should pass verification. Step 4 should pass verification. Step 5 should pass verification. Step 6 should pass verification. Step 7 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.50 Test Case TC_SEC_WP_TWPGetVersion_0002

TC_SEC_WP_TWPGetVersion_0002	TWP get the framework and plugin version.
API Function(s) covered: `TWP_RESULT TWPGetVersion(TWPLIB_HANDLE hLib, TWPVerInfo *pVerInfo);`	
Test Objectives: This test case verifies that the calling application can get error from stub library.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPGetVersion ()` with `INVALID_TWPLIB_HANDLE.` 2. Verify that the API returns TWP_INVALID_PARAMETER.	
Test PASS Condition:	

TC_SEC_WP_TWPGetVersion_0002	TWP get the framework and plugin version.
Step 2 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.51 Test Case TC_SEC_WP_TWPGetVersion_0003

TC_SEC_WP_TWPGetVersion_0003	TWP get the Framework and Plugin version.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPGetVersion(TWPLIB_HANDLE hLib, TWPVerInfo *pVerInfo);`	

Test Objectives: This test case verifies that the calling application gets error from library.	
Test pre-conditions: Validation plug-in.	
Test Procedure: 1. Call `TWPInitLibrary ()`. 2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`. 3. Call `TWPGetVersion () with handle and NULL for TWPVerInfo Parameter.` 4. Verify that the API returns TWP_INVALID_PARAMETER.	
Test PASS Condition: Step 2 should pass verification. Step 4 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.52 Test Case TC_SEC_WP_TWPGetInfo_0001

TC_SEC_WP_TWPGetInfo_0001	TWP get meta information.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPGetInfo(TWPLIB_HANDLE hLib, char *pszInfo);`	

TC_SEC_WP_TWPGetInfo_0001	TWP get meta information.
Test Objectives: This test case verifies that the calling application can get the meta information.	
Test pre-conditions: Validation plug-in.	
Test Procedure: 1. Call `TWPInitLibrary ().` 2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`. 3. Call `TWPGetInfo () with handle.` 4. Verify that the API returns TWP_SUCCESS. 5. Verify string length of meta information is greater than zero.	
Test PASS Condition: Step 2 should pass verification. Step 4 should pass verification. Step 5 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

5.53 Test Case TC_SEC_WP_TWPGetInfo_0002

TC_SEC_WP_TWPGetInfo_0002	Check if API handles invalid input.
API Function(s) covered: `TWP_RESULT TWPGetInfo(TWPLIB_HANDLE hLib, char *pszInfo);`	
Test Objectives: This test case verifies that the calling application can get error from stub library.	
Test pre-conditions: Stub library.	
Test Procedure: 1. Call `TWPGetInfo ()` with `INVALID_TWPLIB_HANDLE.` 2. Verify that the API returns TWP_INVALID_PARAMETER.	
Test PASS Condition: Step 2 should pass verification.	

TC_SEC_WP_TWPGetInfo_0002	Check if API handles invalid input.
Test Clean-up procedure: No specific cleanup required.	

5.54 Test Case TC_SEC_WP_TWPGetInfo_0003

TC_SEC_WP_TWPGetInfo_0003	Check if API handles invalid input.

API Function(s) covered:
`TWPLIB_HANDLE TWPInitLibrary (TWPInitApi *pApiInit);`
`TWP_RESULT TWPGetInfo(TWPLIB_HANDLE hLib, char *pszInfo);`	

Test Objectives: This test case verifies that the calling application gets error from library.	
Test pre-conditions: Validation plug-in.	
Test Procedure: 1. Call `TWPInitLibrary ()`. 2. Verify that the API returns valid `TWPLIB_HANDLE` instead of `INVALID_TWPLIB_HANDLE`. 3. Call `TWPGetInfo () with handle and NULL for pszInfo Parameter.` 4. Verify that the API returns TWP_INVALID_PARAMETER.	
Test PASS Condition: Step 2 should pass verification. Step 4 should pass verification.	
Test Clean-up procedure: No specific cleanup required.	

6 Test Guide

To run test cases, we need to have:

- TWP plug-in for test purpose
- Test contents
- Test cases
- TWP security framework

Test cases need to be compiled with TWP security framework. A TWP plug-in need to be created which can lookup the test contents as expected. All test contents, test cases and test TWP plug-in will be provided as a test suite along with accordinate script file which will automate the test process.

7 Test Contents

URL	Categories	Score
http://twp.test.drugs	Drugs	8
http://twp.test.gambling	Gambling	14
http://twp.test.malicioussites	Malicioussites	29
http://twp.test.pornography	Pornography	47
http://twp.test.phishing	Phishing	67
http://twp.test.spamurls	Spamurls	69
http://twp.test.maliciousdownloads	Maliciousdownloads	28
http://twp.test.potentiallyunwantedprograms	Potentiallyunwantedprograms	105
http://twp.test.games	Games	15
http://twp.test.health	Health	18
http://twp.test.chat.browserexploits	Chat / Browserexploits	100
http://twp.test.remoteaccess.drugs	Remoteaccess / Drugs	8
http://twp.test.sports.games	Sports / Games	15
http://twp.test.searchengines.spamurls	Searchengines / Spamurls	43
http://twp.test.spywareadwarekeyloggers.phishing	Spywareadwarekeyloggers / Phishing	48
http://twp.test.webads.malicioussites	Webads / Malicioussites	29
http://twp.test.travel.porngraphy.tobacco	Travel / Porngraphy / Tobacco	47