Tizen Content Screening

Plug-in API Specification

Document version 1.0.1

Tizen Content Screening Plug-in API Specification

Document Information

Document Details

Revision	1.0.1
Author	MMS Development Team

Revision Information

Revision	Revision Date	Author	Details
1.0.0	11/13/2012	MMS Development Team	Created
1.0.1	01/26/2013	MMS Development Team	Add license

Contents

Terms, Abbreviations, Definitions, Conventions

Items	Description
SDK	Software Development Kit
API	Application Programming Interface
Content Screening	Screening content for security consideration
Module	Program, service or any execution entity in the Tizen platform
Application	Executable provided by either system or third-party

Overview

This document defines the Content Screening API for Tizen platform. The API enables caller modules and applications to scan the content inside their logic data. The Content Screening API is defined in native C language. A computer language bundle might be required if calling from any other language. For example, if we want to call Content Screening API from Java, we need to add JNI code (language bundle) to enable Java code to call Content Screening API from virtual machine.
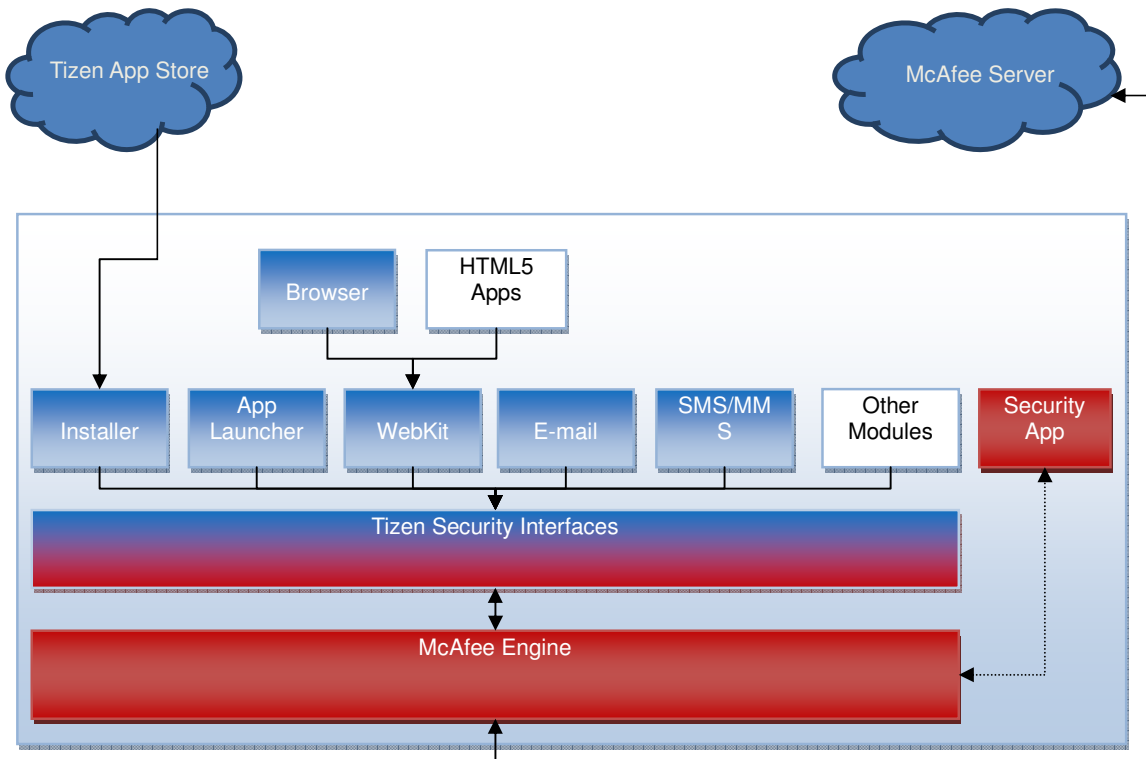
Figure 1   Overview of Content Screening

This document is to define the plug-in API specification in Tizen Security Interfaces for security vendor to implement the plug-in into Tizen OS.

The API is composed by following categories:
- Initialization and clean-up functions
- Scan functions
- Support functions

Tizen Content Screening Plug-in API Specification
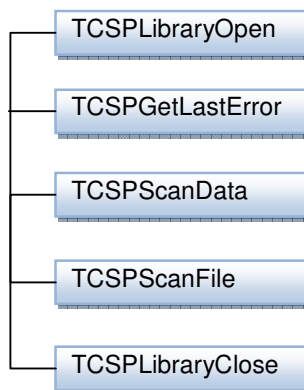
Figure 2   API Call Diagram

TCSPLibraryOpen() and TCSPLibraryClose() are used to initialize Tizen Content Screening library or clean up resource on application exit. TCSPGetLastError() is a support function to return caller an error code to indicate the latest error during this library function call.

TCSPScanData() is used to scan content in the memory while TCSPScanFile() is used to scan the content on permanent storages, like SD card.

Library

The delivery of Tizen Content Screening plug-in API should be a .so library: `libengine.so`

Initialize Functions

Summary

Methods	
`TCSLIB_HANDLE`	`TCSPLibraryOpen()`
	Initialize Tizen Content Screening library.
`void`	`TCSPLibraryClose(TCSLIB_HANDLE hLib)`
	Close Tizen Content Screening library by releasing all resources it occupied.

Methods

`TCSLIB_HANDLE TCSPLibraryOpen()`

Initialize Tizen Content Screening library. For example, allocating memory for internal data use, loading signature database, etc.

Parameters
    None.

Returns
    An instance of Tizen Content Screening library context.

`void TCSPLibraryClose(TCSLIB_HANDLE hLib)`

Destroy Tizen Content Screening library instance. Release all resources it occupies.

Parameters
    hLib   Tizen Content Screening library instance returned
                                  by `TCSPLibraryOpen()`.
Returns
    None.

Scan Functions

Summary

Methods		

int	`TCSPScanData(TCSLIB_HANDLE hLib,`
	`            TCSScanParam *pParam,`
	`            TCSScanResult *pResult)`	

	Scan content in memory.	

int	`TCSPScanFile(TCSLIB_HANDLE hLib,`
	`            char const *pszFileName,`
	`            int iDataType,`
	`            int iAction,`
	`            int iCompressFlag,`
	`            TCSScanResult *pResult)`	

	Scan content in file system.	

Methods

```
int TCSPScanData (TCSLIB_HANDLE hLib,
                  TCSScanParam *pParam,
                  TCSScanResult *pResult)
```

Scan content in memory. Caller need to pass callback functions in `pParam` so that scanner can read or write data back and forth. Scan result will be returned in a data structure `pResult`. The integer return value of this function call is just to indicate if this call is success or not. For any failure of this function call please use `TCSPGetLastError()` to get detail information.

Parameters

`hLib`	Tizen Content Screening library instance returned by `TCSPLibraryOpen()`.
`pParam`	Memory address of data structure instance `TCSScanParam`, see detail at [TCSScanParam](#).
`pResult`	Memory address of data structure instance `TCSScanResult`, see detail at [TCSScanResult](#).

Returns

0 – on success.

−1 – on failure

```
int TCSPScanFile (TCSLIB_HANDLE hLib,
                  char const *pszFileName,
                  int iDataType,
                  int iAction,
                  int iCompressFlag,
                  TCSScanResult *pResult)
```

Scan content on file system. It requires scanner instance, file path, data type the file could be, and type of action for malware. It will return detail scan result in data structure instance `pResult`. The integer return value of this function call indicates if the call is success or not. For any failure of this function call please use `TCSPGetLastError()` to get detail

information.

Parameters

hLib	Tizen Content Screening library instance returned by `TCSPLibraryOpen()`.
pszFileName	Path to the file.
iDataType	Data type of the file. It could be set to unknown type, which leaves the scanner to determine. But by specifying the data type, potentially can accelerate the scanning progress. For detail information please see <u>Data Type</u>.
iAction	Action type if malware detected. Please find detail at <u>Action Type</u>.
iCompressFlag	0 – decompression disabled, 1 – decompress enabled
pResult	Memory address of data structure instance `TCSScanResult`, see detail at <u>TCSScanResult</u>.

Returns

0 – on success.

–1 – on failure

Support Function

Summary

Methods	
int	`TCSPGetLastError(TCSLIB_HANDLE hLib)`
	Return last error code.

Methods

```
int TCSPGetLastError (TCSLIB_HANDLE hLib)
```

This function is used to retrieve the error code previous function call occurs. All scan functions return zero to indicate success, and –1 for failure. The error code gives the detail of the failure reason for trouble shooting.

Parameters

hLib	Tizen Content Screening library instance returned by `TCSPLibraryOpen ()`.

Returns

Error code, please find detail at <u>Error Code</u>.

TCSSScanParam

Description

Data structure for caller to pass input data for scanning.

Summary

Fields	
iAction	The type of action that caller want to take when malware detected. Please find detail at Action Type.
iDataType	The type of content data. For example, archived file. Please find detail at Data Type.
iCompressFlag	0 – decompression disabled, 1 – decompression enabled.
pPrivate	Caller context data. Instead performing direct access to this field, scanner will pass this context data back to caller via below callbacks, so that caller can track the access status inside their own context data without creating global variables.
Unsigned int	`(*pfGetSize)(void *pPrivate)`
	It is used by scanner to obtain content data size in bytes from caller via this callback function.
int	`(*pfSetSize)(void *pPrivate,` `            unsigned int uSize)`
	It is used by scanner to change content data size in bytes via this callback function. (For example, repair infected data)

Unsigned int	`(*pfRead)(void *pPrivate,`
	`        unsigned int uOffset,`
	`        void *pBuffer,`
	`        unsigned int uCount)`

	It is used by scanner to read content data in bytes from caller via this callback function.

Unsigned int	`(*pfGetWrite)(void *pPrivate,`
	`            unsigned int uOffset,`
	`            void const *pBuffer,`
	`            unsigned int uCount)`

	It is used by scanner to change content data in bytes via this callback function.

int	`(*pfGetCallBack)(void *pPrivate,`
	`            int iReason,`
	`            void *pParam)`

	It is used by scanner to notify caller for specific events via this callback function.

CallBack methods

```
unsigned int (*pfGetSize) (void *pPrivate)
```

To scan content data in memory, scanner needs to know the size of the data to be scanned. This callback function is supposed to return the content data size in bytes to scanner.

Parameters

pPrivate	Caller context data.

Returns

Size in bytes.

```
int (*pfSetSize) (void *pPrivate,
                  unsigned int uSize)
```

When scanner try to repair destroyed content data by malware, it usually needs to change the size of content data size, so that caller can do coordinate work for this change.

Parameters

pPrivate	Caller context data.
uSize	New size in bytes

Returns

Size in bytes, not equal to expected size indicating failure of this call.

```
unsigned int (*pfRead) (void *pPrivate,
                        unsigned int uOffset,
                        void const *pBuffer,
                        unsigned int uCount)
```

When scanner scan the content data in memory it needs to read data from caller instead of directly access the memory, this enables flexibility for scanner to handle variable format of content data and stream scanning.

Parameters

pPrivate	Caller context data.
uOffset	Offset where to start reading
pBuffer	The memory address of buffer which is to be filled with read data as result of this read call.
uCount	Bytes to be read

Returns

Read bytes count, not equal to expected size indicating failure of this call.

```
unsigned int (*pfWrite) (void *pPrivate,
                         unsigned int uOffset,
                         void const *pBuffer,
                         unsigned int uCount)
```

When scanner repair the broken content data in memory it needs to write data through this callback.

Parameters

pPrivate	Caller context data.
uOffset	Offset where to start writing
pBuffer	The memory address of buffer which is to be copied to the specified offset.
uCount	Bytes to be written

Returns

Written bytes count, not equal to expected size indicating failure of this call.

```
int (*pfCallBack) (void *pPrivate,
                   int iReason,
```

```
                    void *pParam)
```

When scanner repair the broken content data in memory it needs to write data through this callback.

Parameters

pPrivate	Caller context data.
iReason	Reason for scanner to call caller. Please find detail at [CallBack Reason]().
pParam	Coordinate parameter for specific reason. Please find detail at [CallBack Reason]().

Returns

0 – indicating success
Negative value – indicating stop scanning

TCSScanResult

Description

Data structure for scanner to pass detail scan result back to caller.

Summary

Fields	
iNumDetected	The number of detections.
pDList	Detection list, please find detail at [TCSDetected](#).
void	(*pfFreeResult)(struct TCSScanResult_struct *pResult)
	It is used by caller to release detection list resources when needed.

CallBack methods

```
void (*pfFreeResult) (struct TCSScanResult_struct *pResult)
```

Caller has to pass scan result instance back to this callback function to release resources used by detection list.

Parameters

pResult   The scan result instance.

Returns

None.

TCSDetected

Description

Data structure for scanner to pass detection information to caller.

Summary

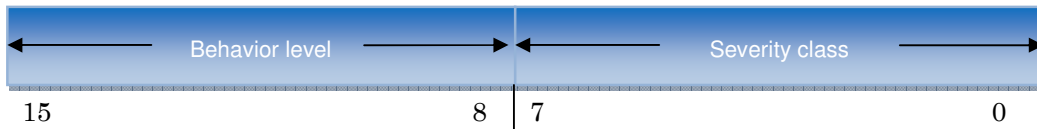| Fields	
pNext	Detection structure is a link list node.
pszName	Malware name.
pszVariant	Malware variant name.
uType	Malware type. Please see below table for detail.
uAction	Bit-field of malware severity, class and behaviour level. Please find detail in below table.
pszFileName	Path of the infected file, it can be ignored if current is a memory data scan.

Malware type

Fields	
TCS_VTYPE_MALWARE	It is a malware.
TCS_VTYPE_PUP	It is a potentially unwanted program.

Malware action

This is a bit-field variable which contains malware severity flags and application behavior levels in bits. Bits $31 - 16$ are reserved.

Behavior level		Severity class	
15	8	7	0

Behavior level

Fields	
TCS_BC_LEVEL0	Process with a warning. The severity is assigned to data previously considered malicious.
TCS_BC_LEVEL1	Prompt user before processing. Ask user if they want the application to process the data.
TCS_BC_LEVEL2	Do not process the data.
TCS_BC_LEVEL3	Do not process the data and prompt user for removal. If the content stored on the terminal, prompt the user for permission before removal.
TCS_BC_LEVEL4	Do not process the data and automatically remove if stored.

Severity class

Fields	
TCS_SC_USER	The malware is harmful to end user.
TCS_SC_TERMINAL	The malware is harmful to the terminal.

Data Type

Description

Data type that Tizen Content Screening library supports.

Summary

Fields	
TCS_DTYPE_UNKNOWN	Data type is unknown, scanner is to determine the data type by itself.
TCS_DTYPE_HTML	HTML content.

TCS_DTYPE_URL	Content data is URL.
TCS_DTYPE_EMAIL	Content data is e-mail.
TCS_DTYPE_PHONE	Content data is phone number.
TCS_DTYPE_JAVA	Content data is Java code.
TCS_DTYPE_JAVAS	Content data is JavaScript.
TCS_DTYPE_TEXT	Content data is plain text.

Action Type

Description

Action type that caller want to take on detected malware.

Summary

Fields	
TCS_SA_SCANONLY	Tell scanner scan content data without changing anything.
TCS_SA_SCANREPAIR	Tell scanner to repair content data if detected.

CallBack Reason

Description

Reason codes for scanner to callback on caller.

Summary

Fields	
TCS_CB_DETECTED	Tell caller that malware was detected.

The coordinate parameter for this callback reason is TCSDetected

Error Code

Description

Error code definition is a bit-field.

Component code	Error code
31 24	23 0

Component code

Fields	
TCS_ERROR_MODULE_GENERIC	Generic error can be occurred in all components.

Error code

Fields	
TCS_ERROR_NOT_IMPLEMENTED	Tizen Content Screening library is not implemented.

Call Sequence

Caller   TCS

TCSPLibraryOpen()

TCS library instance

TCSPScanData()

pfGetSize()

Content data size

<<iterate>>

pfRead()

data

<<iterate>>

[Repair on detection] pfWrite()

Written size

[Repair on detection] pfSetSize()

[detected] pfCallBack(malware information)

0

0

pfFreeResult(detetion list)

TCSPLibraryClose()

Tizen Content Screening Plug-in API Specification