Contents

1. SCOPE .........................................................................................................................................................5

2. REFERENCES ................................................................................................................................................6
   2.1 NORMATIVE REFERENCES ......................................................................................................................6
   2.2 INFORMATIVE REFERENCES ..................................................................................................................6

3. TERMINOLOGY AND CONVENTIONS ...........................................................................................................7
   3.1 CONVENTIONS ........................................................................................................................................7
   3.2 DEFINITIONS ..........................................................................................................................................7
   3.3 ABBREVIATIONS ..................................................................................................................................7

4. INTRODUCTION ...........................................................................................................................................8

5. FRAMEWORK INTERFACES (NORMATIVE) ..................................................................................................9
   5.1 CONTENT SCANNING INTERFACE ...........................................................................................................11
      5.1.1 CSF-1 ..............................................................................................................................................11
   5.2 ERROR RETRIEVAL INTERFACE ............................................................................................................12
      5.2.1 CSF-7 ..............................................................................................................................................12

6. SUPPORTING INTERFACES (INFORMATIVE) ................................................................................................14
   6.1 SCAN ENGINE INITIALIZATION INTERFACE ..........................................................................................14
      6.1.1 CSF-2 ..............................................................................................................................................14
   6.2 VIRUS DATABASE UPDATE INTERFACE ..................................................................................................14
      6.2.1 CSF-3 ..............................................................................................................................................14
      6.2.2 CSF-4 ..............................................................................................................................................15
   6.3 SCAN ENGINE CONFIGURATION INTERFACE .......................................................................................15
      6.3.1 CSF-5 ..............................................................................................................................................15
      6.3.2 CSF-6 ..............................................................................................................................................15

7. SCAN INTERFACE INVOCATION TIME (INFORMATIVE) ..................................................................................17

APPENDIX A. IMPLEMENTER’S NOTE (INFORMATIVE) .........................................................................................19
   A.1 FRAMEWORK INTERFACES IN C .............................................................................................................19
      A.1.1 Content Scanning Interface ..............................................................................................................19
      A.1.2 Error Retrieval Interface ...............................................................................................................19
      A.1.3 Scan Engine Initialization Interface ..............................................................................................20
      A.1.4 Virus Database Update Interface .................................................................................................21
      A.1.5 Scan Engine Configuration Interface ............................................................................................22

APPENDIX B. CHANGE HISTORY (INFORMATIVE) ..............................................................................................24
   B.1 APPROVED VERSION HISTORY ..............................................................................................................24
   B.2 DRAFT/CANDIDATE VERSION 1.0 HISTORY ..........................................................................................24

APPENDIX C. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) ..........................................................25
   C.1 SCR FOR CLIENT SIDE CONTENT SCREENING FRAMEWORK CLIENT .............................................25
      C.1.1 SCR for Calling Enabler ...................................................................................................................25
      C.1.2 SCR for Scan Engine .......................................................................................................................25
   C.2 SCR FOR CLIENT SIDE CONTENT SCREENING FRAMEWORK SERVER .............................................25

Figures

Figure 1: Interfaces of Client Side Content Screening Framework .................................................................10

Figure 2: 32-bit error code format ...................................................................................................................20

Tables

Table 1: Interfaces of Client Side Content Screening Framework .....................................................................10
Table 2: Document types .........................................................................................................................................................11
Table 3: Severity Level ............................................................................................................................................................12
Table 4: Error codes ...............................................................................................................................................................13
Table 5: CSFScanData (CSF-1) Invocation Priority ............................................................................................................18
Table 6: Error Codes ...............................................................................................................................................................20
Table 7: SVerInfo structure ....................................................................................................................................................22
1. Scope

This specification defines technical details of interfaces and interaction mechanism necessary for implementing the OMA Client Side Content Screening Framework to screen malicious content at the mobile terminal. The specification addresses specific requirements enumerated in [CSCSF-RD-v1] and adheres to the architecture described in [CSCSF-AD-v1]. Internal mechanism of the content scanning entity (such as the scan engine, scanning rules, and updating of such engine and rules) are out of the scope of the work.
2. References

2.1 Normative References


[PRIVACY] “OMA Privacy Requirements for Mobile Services”, Version 1.0., Open Mobile Alliance™, OMA-RD_Privacy-V1_0, URL:http://www.openmobilealliance.org/

[CSCSF-RD-v1] “OMA Client Side Content Screening Framework Requirements”, Version 1.0, Open Mobile Alliance™, OMA-RD-Client_Side_CS_FW-V1_0, URL:http://www.openmobilealliance.org/


2.2 Informative References

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

For the purposes of this document, the terms and definitions given in [OMA-DICT] apply and the following also apply:

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Side Content Screening</td>
<td>Content screening performed at the mobile terminal.</td>
</tr>
<tr>
<td>Client Side Content Screening Framework</td>
<td>An abstract conceptual structure used as the basis for constructing interaction model between OMA/non-OMA enablers and content scanning functionality through a set of interfaces with the ultimate goal of bringing forth content screening capability to the mobile terminal.</td>
</tr>
<tr>
<td>Content</td>
<td>Data or code delivered to an end-user and/or end-user’s terminal.</td>
</tr>
<tr>
<td>Content Scanning</td>
<td>The actual operation of looking at the data to determine whether it is a potential candidate for screening and level of severity if found to be as such. What this operation consist of would vary according to how content scanning functionality is implemented and falls outside the scope of this document.</td>
</tr>
<tr>
<td>Content Scanning Functionality</td>
<td>Content scanning performed for OMA/non-OMA enabler wishing to determine whether a content under consideration is undesirable or not. This performance is accessed by a set of interfaces specified by the content screening framework.</td>
</tr>
<tr>
<td>Content Screening</td>
<td>The act of protecting an end-user and/or end-user’s terminal from undesirable content by blocking access to the said content. This act may be in the form of warning message, confirmation of deletion, notification of deletion, silent deletion without notification, etc. Exact detail would vary according to severity level reported, I/O capability of mobile terminal, user preferences, etc.</td>
</tr>
<tr>
<td>Mobile Terminal</td>
<td>A device that receives content as part of its normal running operation.</td>
</tr>
<tr>
<td>Scan Engine</td>
<td>Component of client side content screening framework that performs content scanning service to OMA/non-OMA enablers related to end-user content delivery and/or processing.</td>
</tr>
<tr>
<td>Screening Action</td>
<td>The act of blocking an undesirable content (see ‘Content Screening’).</td>
</tr>
<tr>
<td>Server Side Content Screening</td>
<td>Content screening performed at the network by servers with content screening functionality. E.g. Proxy server, mail server, firewall, etc.</td>
</tr>
</tbody>
</table>

3.3 Abbreviations

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMA</td>
<td>Open Mobile Alliance</td>
</tr>
</tbody>
</table>
4. Introduction

There is an urgent market demand for an effective countermeasure to the growing amount of malicious content delivered to mobile terminals before more lethal variants, such as self-spreading viruses and worms create havoc on networks and users as richer content become available. The goal of the OMA Client Side Content Screening Framework is to facilitate adaptation of existing, or the development of, client-side content screening technologies to the mobile environment, providing a timely solution for an effective countermeasure to these threats.

This specification defines the framework interfaces for use by OMA and non-OMA enablers residing in mobile terminals for utilizing content scanning functionality to detect and screen malicious content. Details on how such enablers interact with the content scanning functionality through these interfaces are also defined. Recommended interface invocation time, screening actions, and use of interfaces in particular execution environments are described as informative purpose.
5. Framework Interfaces (Normative)

This section specifies technical details of framework interfaces of the client side content screening framework as identified in [CSCSF-AD-v1]. In order for OMA and non-OMA enablers to use the scan interface CSFScanData (CSF-1), a set of supporting interfaces is needed in order to provide the CSF-1 interface in mobile terminals. These supporting interfaces can be grouped into four types according to their functions, namely:

1. Scan Engine Initialization Interface
   - CSF-2: CSFSystemInit

2. Virus Database Update Interface
   - CSF-3: CSFScanUpdate
   - CSF-4: CSFScanVersion

3. Scan Engine Configuration Interface
   - CSF-5: CSFConfigSet
   - CSF-6: CSFConfigGet

4. Error Retrieval Interface
   - CSF-7: CSFGetLastError

Figure 1 shows the supporting interfaces in the context of architectural model of the client side content screening framework. The complete list of framework interfaces is shown in Table 1.

CSFScanData (CSF-1) SHALL be supported by the framework as it is this interface which returns the result of the scan to the calling enabler. CSFGetLastError (CSF-7) SHOULD be supported by the framework for the CSFScanData (CSF-1) interface to be meaningful in case of error.

Other interfaces, CSFSystemInit (CSF-2) through CSFConfigGet (CSF-6), are described in section 6 as informative interfaces.
Figure 1: Interfaces of Client Side Content Screening Framework

<table>
<thead>
<tr>
<th>Interface ID</th>
<th>Interface Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSF-1</td>
<td>CSFScanData</td>
<td>Requests scanning of content.</td>
</tr>
<tr>
<td>CSF-2</td>
<td>CSFSystemInit</td>
<td>Initializes scan engine.</td>
</tr>
<tr>
<td>CSF-3</td>
<td>CSFScanUpdate</td>
<td>Triggers update of virus database.</td>
</tr>
<tr>
<td>CSF-4</td>
<td>CSFScanVersion</td>
<td>Returns virus database version information.</td>
</tr>
<tr>
<td>CSF-5</td>
<td>CSFConfigSet</td>
<td>Sets scan engine configuration variable.</td>
</tr>
<tr>
<td>CSF-6</td>
<td>CSFConfigGet</td>
<td>Retrieves scan engine configuration variable.</td>
</tr>
<tr>
<td>CSF-7</td>
<td>CSFGetLastError</td>
<td>Retrieves last error set by the scan engine.</td>
</tr>
</tbody>
</table>

Table 1: Interfaces of Client Side Content Screening Framework
5.1 Content Scanning Interface

5.1.1 CSF-1

5.1.1.1 Name

CSFScanData

5.1.1.2 Description

This interface is used by enabler related to end-user content delivery and/or processing for content scanning. The input comprises of data and type of the content that needs to be scanned. The output comprises of result of the scan, status code, name of the threat, and severity level.

Upon invocation, scan engine performs scanning of the forwarded content and returns the result to the calling enabler. The caller screens the content if it was determined to be malicious based on the result from the scan engine. When the content is found to be malicious, it SHOULD be screened by the caller in order to protect the end-user and/or end-user’s terminal. This screening action MAY be in the form of simple warning message, confirmation of deletion, notification of deletion, or even silent deletion without any notification whatsoever. Exact detail is implementation dependent and would vary according to severity level reported, I/O capability of mobile terminal, user preferences, etc.

5.1.1.3 Input

Data of the content SHALL be provided by the invoking enabler as input to the interface.

Type of the content MAY be provided by the invoking enabler as input to the interface (see Table 2). Scan engine MAY use the type of content to assist the scanning process if the information is provided as input.

<table>
<thead>
<tr>
<th>Document type</th>
<th>Meaning</th>
<th>Expected action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unknown</td>
<td>Scan data for malicious content with no assumption on document type.</td>
</tr>
<tr>
<td>1</td>
<td>HTML</td>
<td>Scan for malicious content in HTML.</td>
</tr>
<tr>
<td>2</td>
<td>URL address</td>
<td>Scan for URL with malicious content.</td>
</tr>
<tr>
<td>3</td>
<td>Email address</td>
<td>Scan for email-address with malicious intent.</td>
</tr>
<tr>
<td>4</td>
<td>Phone number</td>
<td>Scan for phone number with malicious intent.</td>
</tr>
<tr>
<td>5</td>
<td>Text data</td>
<td>Scan text data for malicious content.</td>
</tr>
</tbody>
</table>

Table 2: Document types

5.1.1.4 Output

Result of the scan SHALL be returned by the scan engine as output of the interface as one of follows:

0 if benign.
1 if malicious.

Status code SHALL be returned by the scan engine as output of the interface.

0 if success.
-1 if failure.
Name of threat detected MAY be returned by the scan engine as output of the interface.

Severity level MAY be returned by the scan engine as output of the interface when the result of the scan is found to be malicious. Severity level specifies the level of threat posed by a content that was found to be malicious and recommended screening action for the calling enabler. See table below for exact details.

<table>
<thead>
<tr>
<th>Severity level</th>
<th>Meaning</th>
<th>Recommended screening behavior</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Low</td>
<td>Process with a warning</td>
<td>This severity level may be assigned to content previously considered malicious.</td>
</tr>
<tr>
<td>1</td>
<td>Medium Low</td>
<td>Prompt the user before processing</td>
<td>Ask the user if he/she wants the enabler to process the content.</td>
</tr>
<tr>
<td>2</td>
<td>Medium</td>
<td>Do not process the content</td>
<td>e.g. Content may not need to be deleted as it may just be inappropriate for the particular device.</td>
</tr>
<tr>
<td>3</td>
<td>Medium High</td>
<td>Do not process the content and prompt user for removal if stored.</td>
<td>Ask the user if he/she wants the enabler to remove the content.</td>
</tr>
<tr>
<td>4</td>
<td>High</td>
<td>Do not process the content and automatically remove if stored.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Severity Level

5.2 Error Retrieval Interface

5.2.1 CSF-7

5.2.1.1 Name

CSFGetLastError

5.2.1.2 Description

This interface returns the last-error code set by the scan engine.

5.2.1.3 Input

None.

5.2.1.4 Output

The last error code set by the scan engine SHALL be returned as output of the interface. See table 4 for list of platform independent error codes.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success, not an error.</td>
</tr>
<tr>
<td>1</td>
<td>Operation cancelled.</td>
</tr>
<tr>
<td>2</td>
<td>Failed to access data (e.g. read failed).</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>Invalid input parameter.</td>
</tr>
<tr>
<td>4</td>
<td>Insufficient resource (e.g. out of memory)</td>
</tr>
<tr>
<td>5</td>
<td>Internal error.</td>
</tr>
</tbody>
</table>

**Table 4: Error codes**
6. Supporting Interfaces (Informative)

This section specifies informative interfaces of the client side content screening framework that are usually required in client terminals for supporting normative interfaces defined in the previous section. They are provided for guidance only.

6.1 Scan Engine Initialization Interface

6.1.1 CSF-2

6.1.1.1 Name

CSFSystemInit

6.1.1.2 Description

This interface performs initialization of the scan engine and is invoked (e.g. during boot-time) before other CSF interfaces. Specifically, validation and environment initialization of data kept at persistent storage locations of a mobile terminal are performed. These data are commonly virus database, configuration settings, and synchronization objects used by the scan engine. The exact data are implementation dependent.

6.1.1.3 Input

None.

6.1.1.4 Output

Status code:

0 if success.

-1 if failure.

6.2 Virus Database Update Interface

6.2.1 CSF-3

6.2.1.1 Name

CSFSysScanUpdate

6.2.1.2 Description

This interface triggers the scan engine to perform update of its virus database.

6.2.1.3 Input

None.

6.2.1.4 Output

Status code:

0 if success.

-1 if failure.
6.2.2 CSF-4

6.2.2.1 Name
CSFScanVersion

6.2.2.2 Description
This interface obtains version information of the virus database.

6.2.2.3 Input
None.

6.2.2.4 Output
Virus database version information.
Scan engine version information.
Status code:
  0 if success.
  -1 if failure.

6.3 Scan Engine Configuration Interface

6.3.1 CSF-5

6.3.1.1 Name
CSFConfigSet

6.3.1.2 Description
This interface sets a value for the specified scan engine configuration variable (e.g. to turn on or off the scanner). Only one request can be made at one time. Each variable can only take one value.

6.3.1.3 Input
Name of the scan engine configuration variable.

6.3.1.4 Output
New configuration setting/value for the variable specified.
Status code:
  0 if success.
  -1 if failure.

6.3.2 CSF-6

6.3.2.1 Name
CSFConfigGet
6.3.2.2 Description

This interface retrieves a value for the specified scan engine configuration variable. Only one request can be made at one time. Each variable can only take one value.

6.3.2.3 Input

Name of the scan engine configuration variable.

6.3.2.4 Output

Value of the specified scan engine configuration variable.

Status code:

0 if success.

-1 if failure.
7. Scan Interface Invocation Time (Informative)

Client side content screening framework provides identification of malicious content before a client enabler processes or renders a given content. The scanning of the content is transparent to the user until the scan engine detects a malicious content. Table 2 specifies recommended scanning time according to type of content and client enablers. Note that multiple invocations of the scan interface are not necessary. What is important is that scanning takes place early in the priority list.

<table>
<thead>
<tr>
<th>Recommended CSFScanData (CSF-1) Invocation Time</th>
<th>Content Type</th>
<th>Enablers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. After receiving</td>
<td>XHTML, Email message, SMS message, MMS message, Instruction code, Multimedia data</td>
<td>Browser, Message Handler, File installer, HTTP protocol handler, Data exchange</td>
</tr>
<tr>
<td>2. Before storing</td>
<td>XHTML, Email message, SMS message, MMS message, Instruction code, Multimedia data</td>
<td>Browser, Message Handler, Phone, File installer, HTTP protocol handler, Data exchange</td>
</tr>
<tr>
<td>3. Before rendering (or execution)</td>
<td>XHTML, Email message, SMS message, MMS message, Instruction code, Multimedia data</td>
<td>Browser, Message Handler</td>
</tr>
<tr>
<td>4. Before forwarding to other enablers</td>
<td>Instruction code, Multimedia data, URL address</td>
<td>Browser, Message Handler, File installer, HTTP protocol handler, Data exchange</td>
</tr>
</tbody>
</table>
5. Before requesting

<table>
<thead>
<tr>
<th>URL address</th>
<th>Email address</th>
<th>Phone number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>Message Handler</td>
<td>File installer</td>
</tr>
<tr>
<td>HTTP protocol handler</td>
<td>Data exchange</td>
<td>Phone</td>
</tr>
</tbody>
</table>

Table 5: CSFScanData (CSF-1) Invocation Priority
Appendix A. Implementer’s Note (Informative)

This section is provided as informative purpose to assist implementors of client side content screening framework by describing the framework interfaces in particular execution environments. Note that CSFScanOpen() and other calls are considered as platform dependent functions.

A.1 Framework Interfaces in C

A.1.1 Content Scanning Interface

A.1.1.1 CSFScanData

Description

This interface is used by enabler related to end-user content delivery and/or processing for content scanning. Upon invocation, scan engine performs scanning of the forwarded content and returns the result to the calling enabler. The caller screens the content if it was determined to be malicious based on the result from the scan engine. The caller specifies scanner action, scan target data type(s), a set I/O functions to access the data, and an optional callback function for information retrieval. The result of the scan is returned in a caller provided data structure.

Prototype

```c
int CSFScanData( CSFSCAN_HANDLE hScan,
                 SScanParam*   pParam,
                 SScanResult*  pResult );
```

Parameters

- **hScan**: [in] Scan engine handle obtained from a call to the CSFScanOpen() function.
- **pParam**: [in] Pointer to a structure containing data scan parameters.
- **pResult**: [out] Pointer to a structure containing data scan results.

Return Value (status code)

0 if successful, -1 otherwise.

A.1.2 Error Retrieval Interface

A.1.2.1 CSFGetLastError

Description

This interface is used for retrieving error information when a CSF interface fails.

Prototype

```c
CSFErrorCode CSFGetLastError(CSFLIB_HANDLE hLib);
```

Parameter
hLib

[in] CSF library handled returned by CSFLibraryOpen.

**Return Value**

[out] 32-bit error code value.

Table 6 lists a set of error codes to be reported using the CSFGetLastError interface which returns a 32-bit value formed by combining a component code with an error code (see Figure 2). The last error set by the scan engine when an interface fails is retrieved using CSFGetLastError interface, and an appropriate action is to be taken by the invoking enablers.

<table>
<thead>
<tr>
<th>Component</th>
<th>Code</th>
<th>Error</th>
<th>Value²</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default 00h</td>
<td>CSF_ERR_SUCCESS</td>
<td>000000h</td>
<td>Success; not an error</td>
<td></td>
</tr>
<tr>
<td>Default 00h</td>
<td>CSF_ERR_CANCELED</td>
<td>000001h</td>
<td>Operation cancelled.</td>
<td></td>
</tr>
<tr>
<td>Default 00h</td>
<td>CSF_ERR_DATA_ACCESS</td>
<td>000002h</td>
<td>Failed to access data (e.g. read failed).</td>
<td></td>
</tr>
<tr>
<td>Default 00h</td>
<td>CSF_ERR_INVALID_PARAM</td>
<td>000003h</td>
<td>Invalid input parameter.</td>
<td></td>
</tr>
<tr>
<td>Default 00h</td>
<td>CSF_ERR_INSUFFICIENT_RSC</td>
<td>000004h</td>
<td>Insufficient resource (e.g. out of memory)</td>
<td></td>
</tr>
<tr>
<td>Default 00h</td>
<td>CSF_ERR_INTERNAL</td>
<td>000005h</td>
<td>Internal error.</td>
<td></td>
</tr>
</tbody>
</table>

**Table 6: Error Codes**

![Figure 2: 32-bit error code format](image)

**A.1.3 Scan Engine Initialization Interface**

**A.1.3.1 CSFSystemInit**

**Description**

Verifies and initializes system environment information.

**Prototype**

```c
int CSFSystemInit( void );
```

**Parameters**

None.

**Return Value**

0 if successful, -1 otherwise.
A.1.4 Virus Database Update Interface

A.1.4.1 CSFScanUpdate

Description
This interface triggers the scan engine to perform update of the virus database.

Prototype

```c
int CSFScanUpdate( CSFSCAN_HANDLE hScan,
                    SUpdateParam* pParam );
```

Parameters

- **hScan**
  - [in] CSF scan handle obtained using the CSFScanOpen() function.
- **pParam**
  - [in] Pointer to an update parameter structure containing a callback function pointer for update cancellation/abort and progress status update.

Return Value

0 if successful, -1 otherwise.

A.1.4.2 CSFScanVersion

Description
This interface obtains version information of the virus database.

Prototype

```c
int CSFScanVersion( CSFSCAN_HANDLE hScan,
                      SVerInfo* pVer );
```

Parameter

- **hScan**
  - [in] Scan engine handle obtained using the CSFScanOpen() function.
- **pVer**
  - [out] Pointer to SVerInfo structure described below. Null-character ("\0") terminated strings are stored in the fields of this structure.

```c
#define CSF_VERSION_MAX 16

typedef struct
{
    char szVer[CSF_VERSION_MAX];
    char szEngineVer[CSF_VERSION_MAX];
} SVerInfo;
```
Table 7: SVerInfo structure

Return Value
0 if successful, -1 otherwise.

A.1.5 Scan Engine Configuration Interface
A.1.5.1 CSFConfigSet

Description
This interface sets a value for the specified scan engine configuration variable (e.g. to turn on or off the scanner). Only one request can be made at one time. Each variable can only take one value.

Prototype
int CSFConfigSet( CSFCONFIG_HANDLE hConfig,
        char const* pszName,
        char const* pszValue );

Parameters
hConfig
•  [in] Configuration handle returned by the CSFConfigOpen() function.
pszName
•  [in] NULL terminated configuration variable name.
pszValue
•  [int] NULL terminated new configuration setting/value for the variable specified

Return Value
0 if successful, -1 otherwise.

A.1.5.2 CSFConfigGet

Description
This interface retrieves a value for the specified scan engine configuration variable. Only one request can be made at one time. Each variable can only take one value.

Prototype
int CSFConfigGet( CSFCONFIG_HANDLE hConfig
        char const* pszName,
        char* pBuffer,
        unsigned int uSize );

Parameter
hConfig
•  [in] Configuration handle returned by the CSFConfigOpen() function.
pszName
- [in] null terminated configuration variable name.

pBuffer
- [out] null terminated configuration setting/value for the variable specified

uSize
- [in] Length of pBuffer in bytes.

Return Value
0 if successful, -1 otherwise.
Appendix B. Change History

### B.1 Approved Version History

<table>
<thead>
<tr>
<th>Reference</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>n/a</td>
<td>No prior version –or- No previous version within OMA</td>
</tr>
</tbody>
</table>

### B.2 Draft/Candidate Version 1.0 History

<table>
<thead>
<tr>
<th>Document Identifier</th>
<th>Date</th>
<th>Sections</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft Versions</td>
<td>13 Apr 2005</td>
<td>All</td>
<td>1st Draft as per OMA-MAE-2005-0058-TS_Client_Side_CS_FW.</td>
</tr>
<tr>
<td></td>
<td>18 Jan 2006</td>
<td>2.1, 2.2, C.1, C.2</td>
<td>Revised according to consistency review report: OMA-CONRR-Client_Side_CS_FW-V1_0-20060117-D</td>
</tr>
<tr>
<td>Candidate Versions</td>
<td>09 Feb 2006</td>
<td>n/a</td>
<td>Status changed to Candidate by TP TP doc.ref# OMA-TP-2006-0059-Client_Side_CS_FW-V1_0_for_Candidate_approval</td>
</tr>
</tbody>
</table>
Appendix C. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [IOPPROC].

C.1 SCR for Client Side Content Screening Framework Client

C.1.1 SCR for Calling Enabler

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Status</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCSF-CE-001</td>
<td>Invoke CSFSData (CSF-1) interface to request scanning of content.</td>
<td>5.1</td>
<td>M</td>
<td>CSCSF-SE-001 AND CSCSF-CE-003</td>
</tr>
<tr>
<td>CSCSF-CE-002</td>
<td>Invoke CSFGetLastError (CSF-7) interface to retrieve last error code set by scan engine.</td>
<td>5.2</td>
<td>O</td>
<td>CSCSF-SE-002</td>
</tr>
<tr>
<td>CSCSF-CE-003</td>
<td>Provide content data as input to CSFSData (CSF-1) interface.</td>
<td>5.1</td>
<td>M</td>
<td>CSCSF-CE-001</td>
</tr>
<tr>
<td>CSCSF-CE-004</td>
<td>Provide content type as input to CSFSData (CSF-7) interface.</td>
<td>5.2</td>
<td>O</td>
<td>CSCSF-CE-001</td>
</tr>
</tbody>
</table>

C.1.2 SCR for Scan Engine

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Status</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCSF-SE-001</td>
<td>Support for CSFSData (CSF-1) interface</td>
<td>5.1</td>
<td>M</td>
<td>CSCSF-SE-003 AND CSCSF-SE-004</td>
</tr>
<tr>
<td>CSCSF-SE-002</td>
<td>Support for CSFGetLastError (CSF-7) interface</td>
<td>5.2</td>
<td>O</td>
<td>CSCSF-SE-007</td>
</tr>
<tr>
<td>CSCSF-SE-003</td>
<td>Provide result of the scan as output of CSFSData (CSF-1) interface.</td>
<td>5.1</td>
<td>M</td>
<td>CSCSF-SE-001</td>
</tr>
<tr>
<td>CSCSF-SE-004</td>
<td>Provide status code as output of CSFSData (CSF-1) interface.</td>
<td>5.1</td>
<td>M</td>
<td>CSCSF-SE-001</td>
</tr>
<tr>
<td>CSCSF-SE-005</td>
<td>Provide name of threat detected as output of CSFSData (CSF-1) interface.</td>
<td>5.1</td>
<td>O</td>
<td>CSCSF-SE-001</td>
</tr>
<tr>
<td>CSCSF-SE-006</td>
<td>Provide severity level as output of CSFSData (CSF-1) interface.</td>
<td>5.1</td>
<td>O</td>
<td>CSCSF-SE-001</td>
</tr>
<tr>
<td>CSCSF-SE-007</td>
<td>Provide last error code as output of CSFGetLastError (CSF-7) interface.</td>
<td>5.2</td>
<td>O</td>
<td>CSCSF-SE-002</td>
</tr>
</tbody>
</table>

C.2 SCR for Client Side Content Screening Framework Server

No server requirements exist for Client Side Content Screening Framework.