ETSI TS 102 835 V8.1.0 (2012-03)

Smart Cards;
Test Specification for SCWS
Application Invocation API for Java Card™;
Test Environment and Annexes
(Release 8)

Reference
RTS/SCP-00SCWSv810

Keywords
API, SCWS, testing

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights 8
Foreword 8
1 Scope 9
2 References 9
2.1 Normative references 9
2.2 Informative references 10
3 Definitions and abbreviations 10
3.1 Definitions 10
3.2 Abbreviations 11
4 Applicability 11
4.1 Applicability of the present document 11
4.2 Applicability of the individual test cases 11
4.3 Applicability to different Releases 11
4.4 Definitions 12
4.4.1 Format of table of the table of optional features 12
4.4.2 Format of the applicability table 12
4.4.3 Status and Notations 12
4.5 Table of optional features 13
4.6 Applicability table 14
5 Test environment 15
5.1 Test environment description 15
5.2 Tests format 16
5.2.1 Test area reference 16
5.2.1.1 Conformance requirements 16
5.2.1.2 Test area files 17
5.2.1.3 Test procedure 17
5.2.1.4 Test coverage 17
5.3 Initial conditions 17
5.4 Package name 17
5.5 AID coding 18
5.6 Test equipment 19
5.6.1 Test tool 19
5.6.2 Interfaces and classes use 19
5.6.3 Util package 19
5.6.4 Java Software Development Kit 19
6 Test Cases 19
6.1 class uicc.scws 19
6.1.1 Class ScwsExtensionRegistry 19
6.1.1.1 Method register 19
6.1.1.1.1 Conformance Requirement 20
6.1.1.1.2 Test area files 20
6.1.1.1.3 Test coverage 20
6.1.1.1.4 Test procedure 21
6.1.1.2 Method deregister 21
6.1.1.2.1 Conformance Requirement 22
6.1.1.2.2 Test area files 22
6.1.1.2.3 Test coverage 22
6.1.1.2.4 Test procedure 22
6.1.2 Interface ScwsExtension 23
6.1.3 Class ScwsExtensionService 23
6.1.3.1 Method doDelete 23
6.1.3.1.1 Conformance Requirement 23
6.1.3.1.2 Test area files 23

6.1.3.1.3 Test coverage 23
6.1.3.1.4 Test procedure 24
6.1.3.2 Method doGet 24
6.1.3.2.1 Conformance Requirement 24
6.1.3.2.2 Test area files 24
6.1.3.2.3 Test coverage 24
6.1.3.2.4 Test procedure 25
6.1.3.3 Method doHead 25
6.1.3.3.1 Conformance Requirement 25
6.1.3.3.2 Test area files 25
6.1.3.3.3 Test coverage 25
6.1.3.3.4 Test procedure 26
6.1.3.4 Method doOptions 26
6.1.3.4.1 Conformance Requirement 26
6.1.3.4.2 Test area files 26
6.1.3.4.3 Test coverage 26
6.1.3.4.4 Test procedure 27
6.1.3.5 Method doPost 27
6.1.3.5.1 Conformance Requirement 27
6.1.3.5.2 Test area files 27
6.1.3.5.3 Test coverage 27
6.1.3.5.4 Test procedure 28
6.1.3.6 Method doPut 28
6.1.3.6.1 Conformance Requirement 28
6.1.3.6.2 Test area files 28
6.1.3.6.3 Test coverage 28
6.1.3.6.4 Test procedure 29
6.1.3.7 Method doTrace 29
6.1.3.7.1 Conformance Requirement 29
6.1.3.7.2 Test area files 29
6.1.3.7.3 Test coverage 29
6.1.3.7.4 Test procedure 30
6.1.4 Interface HttpRequest 30
6.1.4.1 Method findAndCopyKeywordValue 30
6.1.4.1.1 Conformance Requirement 30
6.1.4.1.2 Test area files 30
6.1.4.1.3 Test coverage 31
6.1.4.1.4 Test procedure 31
6.1.4.2 Method findAndCopyKeywordValue 31
6.1.4.2.1 Conformance Requirement 31
6.1.4.2.2 Test area files 32
6.1.4.2.3 Test coverage 32
6.1.4.2.4 Test procedure 33
6.1.4.3 Method getContentLength 34
6.1.4.3.1 Conformance Requirement 34
6.1.4.3.2 Test area files 34
6.1.4.3.3 Test coverage 35
6.1.4.3.4 Test procedure 35
6.1.4.4 Method getContentType 35
6.1.4.4.1 Conformance Requirement 35
6.1.4.4.2 Test area files 35
6.1.4.4.3 Test coverage 35
6.1.4.4.4 Test procedure 36
6.1.4.5 Method getRequestHttpVersion 36
6.1.4.5.1 Conformance Requirement 36
6.1.4.5.2 Test area files 36
6.1.4.5.3 Test coverage 36
6.1.4.5.4 Test procedure 37
6.1.4.6 Method readContent 37
6.1.4.6.1 Conformance Requirement 37
6.1.4.6.1.1 Normal execution 37
6.1.4.6.3 Test coverage 38

6.1.4.6.4 Test procedure 38
6.1.4.7 Method readContentType 39
6.1.4.7.1 Conformance Requirement 39
6.1.4.7.2 Test area files 39
6.1.4.7.3 Test coverage 40
6.1.4.7.4 Test procedure 40
6.1.5 Interface HttpResponse 41
6.1.5.1 Method appendContent 41
6.1.5.1.1 Conformance Requirement 41
6.1.5.1.2 Test area files 41
6.1.5.1.3 Test coverage 42
6.1.5.1.4 Test procedure 42
6.1.5.2 Method appendHeaderVariable(byte[] data, short offset, short length) 44
6.1.5.2.1 Conformance Requirement 44
6.1.5.2.2 Test area files 45
6.1.5.2.3 Test coverage 45
6.1.5.2.4 Test procedure 45
6.1.5.3 Method appendHeaderVariable (byte[] name, short nameOffset, short nameLength, byte[] value, short valueOffset, short valueLength) 48
6.1.5.3.1 Conformance Requirement 48
6.1.5.3.2 Test area files 49
6.1.5.3.3 Test coverage 49
6.1.5.3.4 Test procedure 49
6.1.5.4 Method appendHeaderVariable (short headerKeywordNameId, byte[] value, short valueOffset, short valueLength) 53
6.1.5.4.1 Conformance Requirement 53
6.1.5.4.2 Test area files 53
6.1.5.5 Method enableChunkMode 54
6.1.5.5.1 Conformance Requirement 54
6.1.5.5.2 Test area files 54
6.1.5.5.3 Test coverage 54
6.1.5.5.4 Test procedure 55
6.1.5.6 Method finalizeHeader 57
6.1.5.6.1 Conformance Requirement 57
6.1.5.6.2 Test area files 58
6.1.5.6.3 Test coverage 58
6.1.5.6.4 Test procedure 58
6.1.5.7 Method flush 59
6.1.5.7.1 Conformance Requirement 59
6.1.5.7.2 Test area files 60
6.1.5.7.3 Test coverage 60
6.1.5.7.4 Test procedure 60
6.1.5.8 Method getRemainingResponseBufferSize 61
6.1.5.8.1 Conformance Requirement 61
6.1.5.8.2 Test area files 62
6.1.5.8.3 Test coverage 62
6.1.5.8.4 Test procedure 62
6.1.5.9 Method reset 63
6.1.5.9.1 Conformance Requirement 63
6.1.5.9.2 Test area files 64
6.1.5.9.3 Test coverage 64
6.1.5.9.4 Test procedure 64
6.1.5.10 Method sendError 65
6.1.5.10.1 Conformance Requirement 65
6.1.5.10.2 Test area files 66
6.1.5.10.3 Test coverage 66
6.1.5.10.4 Test procedure 67
6.1.5.11 Method setContentType 70
6.1.5.11.1 Conformance Requirement 70
6.1.5.11.2 Test area files 70
6.1.5.11.3 Test coverage 70
6.1.5.11.4 Test procedure 71

6.1.5.12 Method writeStatusCode 73
6.1.5.12.1 Conformance Requirement 73
6.1.5.12.2 Test area files 74
6.1.5.12.3 Test coverage 74
6.1.5.12.4 Test procedure 74
6.2 SCWS Runtime Environment 78
6.2.1 Applet state 78
6.2.1.1 Invokation of applets not in state selectable 78
6.2.1.1.1 Conformance Requirement 78
6.2.1.1.2 Test area files 79
6.2.1.1.3 Test coverage 79
6.2.1.1.4 Test procedure 79
6.2.1.2 Registration remains valid if applet is not in selectable state 81
6.2.1.2.1 Conformance Requirement 81
6.2.1.2.2 Test area files 82
6.2.1.2.3 Test coverage 82
6.2.1.2.4 Test procedure 82
6.2.2 Response sending 82
6.2.3 Exception handling 83
6.2.3.1 No exception shall be propagated as HTTP error to the terminal 83
6.2.3.1.1 Conformance Requirement 83
6.2.3.1.2 Test area files 83
6.2.3.1.3 Test coverage 83
6.2.3.1.4 Test procedure 84
6.2.4 Response Header Management 85
6.2.4.1 Send status code indicating success 85
6.2.4.1.1 Conformance Requirement 85
6.2.4.1.2 Test area files 85
6.2.4.1.3 Test coverage 85
6.2.4.1.4 Test procedure 85
6.2.5 Availability of ProactiveHandler / ProactiveResponseHandler 86
6.2.5.1 Incoming Http request 86
6.2.5.1.1 Conformance Requirement 86
6.2.5.1.2 Test area files 86
6.2.5.1.3 Test coverage 86
6.2.5.1.4 Test procedure 87
6.2.5.2 Available for complete Http response 87
6.2.5.2.1 Conformance Requirement 87
6.2.5.2.2 Test area files 88
6.2.5.2.3 Test coverage 88
6.2.5.2.4 Test procedure 88
6.2.5.3 Availability of ProactiveResponseHandler 88
6.2.5.3.1 Conformance Requirement 88
6.2.5.3.2 Test area files 89
6.2.5.3.3 Test coverage 89
6.2.5.3.4 Test procedure 89
6.2.5.4 Triggering through the ToolkitInterface 89
6.2.5.4.1 Conformance Requirement 89
6.2.5.4.2 Test area files 89
6.2.5.4.3 Test coverage 89
6.2.5.4.4 Test procedure 90
6.2.5.5 Presence of CAT_TP multiplexing application 90
6.2.5.5.1 Conformance Requirement 90
6.2.5.5.2 Test area files 90
6.2.5.5.3 Test coverage 90
6.2.5.5.4 Test procedure 90

Annex A (normative): Class, methods and SCWSFramework tests acronyms 91

A.1 Smart Card Web Server part 91
A.1.1 Class ScwsExtensionRegistry 91
A.1.2 Class ScwsExtensionService 91

A.1.3 HttpRequest interface 91
A.1.4 HttpResponse interface 92
A.2 Acronyms for SCWS Framework tests 92
A.2.1 Applet state 92
A.2.3 Exception handling 92
A.2.4 Response Header Management 92
A.2.5 Availability of ProactiveHandler/ProactiveResponseHandler 92
Annex B (normative): Test file description 93
Annex C (normative): uicc.scws.test.util package and interfaces 94
Annex D (normative): Test Area files 95
Annex E (informative): HTTP-Request and HTTP-Response handling 96
Annex F (informative): Bibliography 97
Annex G (informative): Change history 98
History 99

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Smart Card Platform (SCP).

The contents of the present document are subject to continuing work within TC SCP and may change following formal TC SCP approval. If TC SCP decides to modify the contents of the present document, it will be re-released by TC SCP with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TC SCP for information;
  - 2 presented to TC SCP for approval;
  - 3 or greater indicates TC SCP approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document covers the minimum characteristics considered necessary in order to provide compliance to TS 102 588 [2].

It describes the technical characteristics and methods for testing the SCWS API for Java Card™ (TS 102 588 [2]) implemented in a UICC platform. It specifies the following parts:

- test applicability;
- test environment description;
- tests format;
- test area reference;
- conformance requirements;
- test suite files;
- test procedure;
- test coverage; and
- a description of the associated testing tools that shall be used.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

- In the case of a reference to a TC SCP document, a non specific reference implicitly refers to the latest version of that document in the same Release as the present document.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] ISO/IEC 9646-7 (1995): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 7: Implementation Conformance Statements".

[2] ETSI TS 102 588: "Smart Cards; Application invocation Application Programming Interface (API) by a UICC webserver for Java Card™ platform".

[3] ETSI TS 102 483: "Smart cards; UICC-Terminal interface; Internet Protocol connectivity between UICC and terminal".

[4] OMA: "Smartcard -Web Server Enable Architecture", OMA-AD-Smartcard-Web-Server-V1-0-20080421-A.

NOTE: Available at http://www.openmobilealliance.org/technical/release_program/SCWS_v1_0.aspx.

[5] OMA: "Smartcard-Web-Server", OMA-TS-Smartcard-Web-Server-V1-0-20080421-A.

NOTE: Available at http://www.openmobilealliance.org/technical/release_program/SCWS_v1_0.aspx.

[6] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2.2 Application Programming Interface".

[7] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2.2 Runtime Environment Specification".

[8] Sun Microsystems Java Card™ Specification: "Java Card™ 2.2.2 Virtual Machine Specification".

NOTE: SUN Java Card Specifications can be downloaded at http://java.sun.com/products/javacard.

[9] ETSI TS 102 268: "Smart Cards; Test specification for the UICC Application Programming Interface (AP)I for Java Card™".

[10] ETSI TS 101 220: "Smart Cards; ETSI numbering system for telecommunication application providers".

[11] ETSI TS 102 223: "Smart Cards; Card Application Toolkit (CAT)".

[12] ETSI TS 102 241: "Smart Cards; UICC Application Programming Interface (UICC API) for Java Card (TM)".

[13] ETSI TS 102 221: "Smart Cards; UICC-Terminal interface; Physical and logical characteristics".

[14] ETSI TS 102 225: "Smart Cards; Secured packet structure for UICC based applications (Release 8)".

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 102 600: "Smart Cards; UICC-Terminal interface; Characteristics of the USB interface".

[i.2] HttpClient project.

NOTE: Available at http://hc.apache.org/httpcomponents-client/index.html.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

applet installation parameters: values for applet installation parameters

Conformance Requirement Reference (CRR): description of the expected SCWS behaviour according to TS 102 588 [2]

expected state: state in which the UICC is supposed to be after the execution of the test procedure applied on the relevant initial conditions

security parameters: minimum security requirements defined for the applet installation process

test area: set of Test Cases applicable to a specific part (class method, SCWS RE behaviour, etc.) of TS 102 588 [2]

test case: elementary test that checks for compliance with one or more Conformance Requirement References

test procedure: sequence of actions/commands to perform all the test cases defined in a test area

test source file: java file containing methods that will load and install test applet in the card, execute and verify the test results, and restore the Default Initial Conditions on the UICC (when possible)

test toolkit applet: applet designed to test a specific functionality of the SCWS API (TS 102 588 [2])

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AID	Application IDentifier
API	Application Programming Interface
BIP	Bearer Independent Protocol according to TS 102 223 [11]
CAT	Card Application Toolkit
CRR	Conformance Requirements Reference
CRRC	Conformance Requirement Reference Context Error
CRRN	Conformance Requirement Reference Normal
CRRP	Conformance Requirement Reference Parameter Error
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
JCRE	Java Card™ Run-time Environment
RE	Runtime Environment
SCWS	Smart Card based Web Server according to OMA specifications [4] and [5]
URI	Uniform Resource Identifier
USB	Universal Serial Bus interface according to TS 102 600 [i.1]

4 Applicability

4.1 Applicability of the present document

This present test specification applies to UICCs containing a SCWS invocation API according to TS 102 588 [2].

4.2 Applicability of the individual test cases

Table A.1 lists the optional features for which the supplier of the implementation states the support.

4.3 Applicability to different Releases

The applicability to different releases specified in table B.1 of the present document shall apply, unless otherwise specified.

4.4 Definitions

4.4.1 Format of table of the table of optional features

The columns in table A.1 have the following meaning.

Column	Meaning
Option:	The optional feature supported or not by the implementation.
Status:	See clause 4.4.3 'Status and Notations'
Support:	The support columns shall be filled in by the supplier of the implementation. The following common notations, defined in ISO/IEC 9646-7 [1], are used for the support column in table A.1. Y or y supported by the implementation N or n not supported by the implementation N/A, n/a or - no answer required (allowed only if the status is N/A, directly or after evaluation of a conditional status)
Mnemonic:	The mnemonic column contains mnemonic identifiers for each item.

4.4.2 Format of the applicability table

The applicability of every test in table B.1 is formally expressed by the use of Boolean expression defined in the following clause.

The columns in table B.1 have the following meaning.

Column	Meaning
Test case:	The "Test case" column gives a reference to the test case number(s) detailed in the present document and required to validate the implementation of the corresponding item in the "Description" column
Release:	The "Release" column gives the Release applicable and onwards, for the item in the "Test case" column
Support:	The "Support" column is blank in the proforma, and shall be completed by the manufacturer in respect of each particular requirement to indicate the choices, which have been made in the implementation.

4.4.3 Status and Notations

The following notations, defined in ISO/IEC 9646-7 [1], are used for the status column:

M mandatory - the capability is required to be supported.

O optional - the capability may be supported or not.

N/A not applicable - in the given context, it is impossible to use the capability.

X prohibited (excluded) - there is a requirement not to use this capability in the given context.

O.i qualified optional - for mutually exclusive or selectable options from a set. "i" is an integer which identifies an unique group of related optional items and the logic of their selection which is defined immediately following the table.

Ci conditional - the requirement on the capability ("M", "O", "X" or "N/A") depends on the support of other optional or conditional items. "i" is an integer identifying an unique conditional status expression which is defined immediately following the table. For nested conditional expressions, the syntax "IF ... THEN (IF ... THEN ... ELSE...) ELSE ..." shall be used to avoid ambiguities.

4.5 Table of optional features

Table A.1: Options

Item	Option	Status	Support	Mnemonic
1	ScwsExtension.doGet()	M		doGet
2	ScwsExtension.doPost()	M		doPost
3	ScwsExtension.doHead()	M		doHead
4	ScwsExtension.doPut()	M		doPut
5	ScwsExtension.doDelete()	M		doDelete
6	ScwsExtension.doOptions()	O		doOptions
7	ScwsExtension.doTrace()	O		doTrace

4.6 Applicability table

Table B.1: Applicability of tests

Test case	Description	Release	Rel-7	Rel-8
	Class ScwsExtensionRegistry			
6.1.1.1	Method register	Rel-7	M	M
6.1.1.2	Method deregister	Rel-7	M	M
	Class ScwsExtensionService			
6.1.3.1	Method doDelete	Rel-7	M	M
6.1.3.2	Method doGet	Rel-7	M	M
6.1.3.3	Method doHead	Rel-7	M	M
6.1.3.4	Method doOptions	Rel-7	C001	C001
6.1.3.5	Method doPost	Rel-7	M	M
6.1.3.6	Method doPut	Rel-7	M	M
6.1.3.7	Method doTrace	Rel-7	C002	C002
	Interface HttpRequest			
6.1.4.1	Method findAndCopyKeywordValue	Rel-7	M	M
6.1.4.2	Method findAndCopyKeywordValue	Rel-7	M	M
6.1.4.3	Method getContentLength	Rel-7	M	M
6.1.4.4	Method getContentType	Rel-7	M	M
6.1.4.5	Method getRequestHttpVersion	Rel-7	M	M
6.1.4.6	Method readContent	Rel-7	M	M
6.1.4.7	Method readContentType	Rel-7	M	M
	Interface HttpResponse			
6.1.5.1	Method appendContent	Rel-7	M	M
6.1.5.2	Method appendHeaderVariable(byte[] data, short offset, short length)	Rel-7	M	M
6.1.5.3	Method appendHeaderVariable (byte[] name, short nameOffset, short nameLength, byte[] value, short valueOffset, short valueLength)	Rel-7	M	M
6.1.5.4	Method appendHeaderVariable (short headerKeywordNameId, byte[] value, short valueOffset, short valueLength)	Rel-7	M	M
6.1.5.5	Method enableChunkMode	Rel-7	M	M
6.1.5.6	Method finalizeHeader	Rel-7	M	M
6.1.5.7	Method flush	Rel-7	M	M
6.1.5.8	Method getRemainingResponseBufferSize	Rel-7	M	M
6.1.5.9	Method reset	Rel-7	M	M
6.1.5.10	Method sendError	Rel-7	M	M
6.1.5.11	Method setContentType	Rel-7	M	M
6.1.5.12	Method writeStatusCode	Rel-7	M	M
	SCWS Runtime Environment: Applet state			
6.2.1.1	Invokation of applets not in state selectable	Rel-7	M	M
6.2.1.2	Registration remains valid if applet is not in selectable state	Rel-7	M	M
	SCWS Runtime Environment: Exception handling			
6.2.3.1	No exception shall be propagated as HTTP error to the terminal	Rel-7	M	M
	SCWS Runtime Environment: Response Header Management			
6.2.4.1	Send status code indicating success	Rel-7	M	M
	SCWS Runtime Environment: Availability of ProactiveHandler / ProactiveResponseHandler			
6.2.5.1	Incoming Http request	Rel-8	N/A	M
6.2.5.2	Available for complete Http response	Rel-8	N/A	M
6.2.5.3	Availability of ProactiveResponseHandler	Rel-8	N/A	M
6.2.5.4	Triggering through the ToolkitInterface	Rel-8	N/A	M
6.2.5.5	Presence of CAT_TP multiplexing application	Rel-8	N/A	C003
C001:	IF (doOptions supported) THEN M ELSE N/A.			
C002:	IF (doTrace supported) THEN M ELSE N/A.			
C003:	IF (CAT Multiplexing application present) THEN M ELSE N/A.			

5 Test environment

This clause specifies requirements that shall be met and the testing rules that shall be followed during the test procedure.

The test cases described in the present document shall be performed independently of the interface used between the UICC and the terminal. In particular, any test case shall have the same result regardless of using BIP in Server Mode as defined in TS 102 223 [11] or an IP connection according to TS 102 483 [3] and TS 102 600 [i.1].

5.1 Test environment description

The general architecture for the test environment is:

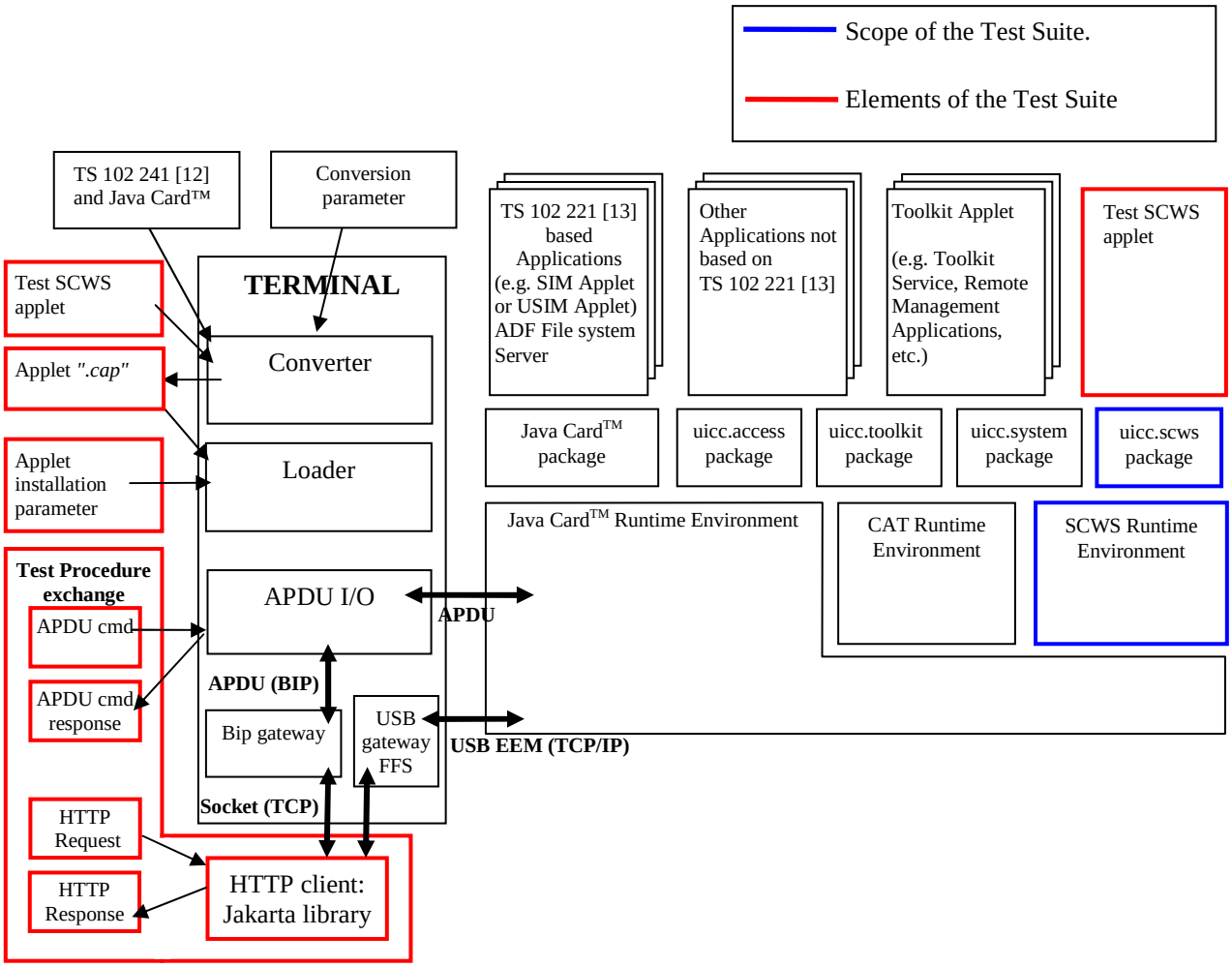


Figure 5.1

5.2 Tests format

5.2.1 Test area reference

Each test area is referenced as follows:

For API Testing: 'API_[package name]_[class name]_[method name]' where

- package name:

  uicc.scws: '1'.

- class name:

  yyy: 3 letters for each class.

  See annex A for full classes acronyms list.

- method name:

  zzzz[input parameters]:

  See annex A for full methods name acronyms list.

For SCWS Framework testing: 'API_2_[clause name]_[subclause name]' where

- Clause name:

  xxx: 3 letters for each clause.

  See annex A for full clause acronyms list.

- Subclause name:

  yyyy: 4 letters for each subclause.

  See annex A for full subclause acronyms list.

5.2.1.1 Conformance requirements

The conformance requirements are expressed in the following way:

- Method prototype as listed in TS 102 588 [2].
- Normal execution:
  - Contains normal execution and correct parameters limit values, each referenced as a Conformance Requirement Reference Normal (CRRN).
- Parameters error:
  - Contains parameter errors and incorrect parameter limit values, each referenced as a Conformance Requirement Reference Parameter Error (CRRP).
- Context error:
  - Contains errors due to the context the method is used in, each referenced as a Conformance Requirement Reference Context Error (CRRC).

5.2.1.2 Test area files

The files included in the Test Area use the following naming convention:

- Test Source: Test_[Test Area Reference].java
- Test Applet: [Test Area Reference]_[Test applet number].java
- Cap File: [Test Area Reference].cap

The applet numbers start from '1'.

The test source shall use the common interfaces defined in annex C.

The Cap File format is described in Java Card™ Virtual Machine Specification [8].

Test files can be run in any order.

All files from the same test area are located in the same subfolder.

5.2.1.3 Test procedure

Each test procedure contains a table to indicate the expected responses form the API and/or the HTTP level as follows.

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
	Test Case detailed description	*API and/or SCWS Framework expected behaviour.*	*Expected response at HTTP level.*

5.2.1.4 Test coverage

The table above each test procedure indicates the correspondence between the Conformance Requirements Reference (CRR) and the different test cases.

5.3 Initial conditions

The Initial Conditions are a set of general prerequisites for the (U)SIM prior to the execution of testing. For each test procedure described in the present document, the following rules apply to the Initial Conditions:

- unless otherwise stated, before installing the applet(s) relevant to the current test procedure, all packages specific to other test procedures shall not be present.

Prior to any test the test environment shall have the (U)SIM powered on and performed the PROFILE DOWNLOAD procedure.

5.4 Package name

Java packages integrating this Test Suite shall follow this naming convention:

uicc.scws.test.[Test Area Reference]: Java Card packages containing Test Area References for the TS 102 588 [2] uicc.scws package.

uicc.scws.test.scwsframework.[Test Area Reference]: Java Card packages containing Test Area References for the TS 102 588 [2] SCWS Framework.

uicc.scws.test.util: for the Test util package defined in this Test Suite. Requires package uicc.test.util from TS 102 268 [9].

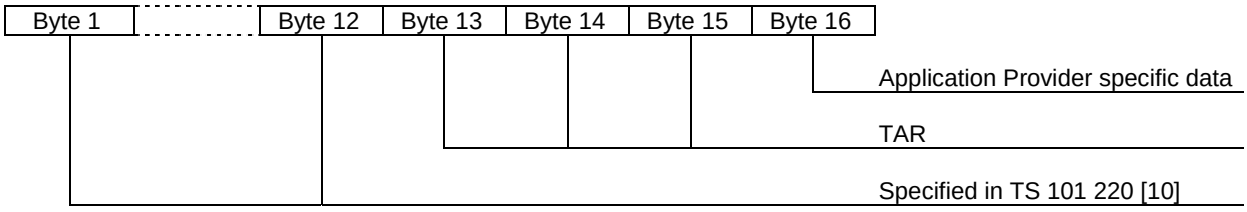
where the Test Area Reference is written in lower case.

EXAMPLE: The package *../uicc.scws.test.[Test Area Reference]* creates the following directory structure *../uicc/scws/test/[Test Area Reference]/Api_1_..._[1..n].**, where '*Api_1_..._[1..n].**' are the different test applets Java source files used in *[Test Area Reference]*.

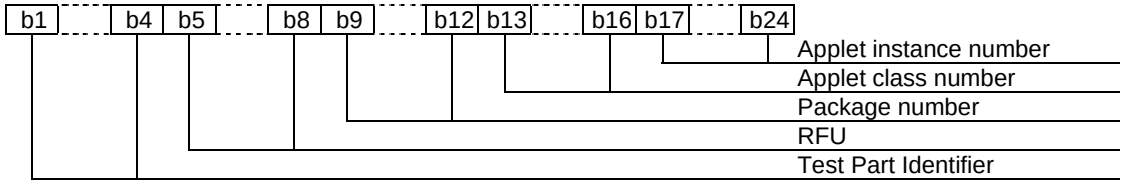
5.5 AID coding

The AID coding for the Test Packages, Applet classes and Applets shall be as specified in TS 101 220 [10]. In addition, the following TAR and Application Provider specific data values are defined for use within the present document:

AID coding



TAR coding (3 bytes/ 24 bits):



Applet instance number, Applet Class number, Package number:

- For package AID, package number shall start from 0 and class and instance numbers shall be 0.
- For class AID, package number is the number of the class package, class number shall start from 1 and instance shall be 0.
- For instance AID, package and class number are the number of class and package of which instance belongs, and instance number shall start from 1.

Test part Identifier (bits b1-b4):

- 0000 reserved (as TAR= '00.00.00' is reserved for Issuer Security Domain).
- 0010 API uicc.scws.
- 0101 SCWS Framework.
- 1111 package uicc.scws.test.util
- other values are RFU.

Application Provider specific data (1 byte):

- '00' for Package.
- '01' for Applet class.
- '02' for Applet Instance.

EXAMPLE: The AID of Package uicc.scws is 'A0 00 00 00 87 10 05 FF FF FF FF 89 20 00 00 00'.

5.6 Test equipment

These clauses recommend a minimum specification for each of the items of test equipment referenced in the tests.

5.6.1 Test tool

This test tool shall meet the following requirements:

- be able to send and receive HTTP messages to the SCWS in the UICC;
- the result of I/O commands must be presented at the application layer;
- be able to provide results of the tests;
- shall send and/or compare all HTTP messages specified in the test files.

5.6.2 Interfaces and classes use

The SCWS test tool extends the UICC test tool defined in TS 102 268 [9]. Therefore the SCWS test tool cannot be run without having implemented the UICC test tool.

The SCWS test tool shall use the interfaces and classes as defined in Annex C. They define the only allowed methods to write the test sources.

Interfaces and classes are defined as follow:

- ScwsApplicationManagementService defines methods to manage applications;
- ScwsCardManagementService defines methods to manage the UICC;
- ScwsToolkitService defines methods to manage toolkit commands;
- ScwsHttpManagement defines methods to send and receive HTTP messages;
- ScwsAPITestCardService defines the static method to get a reference of the class implementing all interfaces.

5.6.3 Util package

Annex C includes java source code of TestScwsApplet abstract class of the uicc.scws package. Each test applet shall extend this abstract class in order to retrieve test results when selecting it.

5.6.4 Java Software Development Kit

Java Card™ software development kit (SDK) version supported by Java Card 2.2.2 specifications ([6], [7], [8]) is 1.4.1.

6 Test Cases

6.1 class uicc.scws

6.1.1 Class ScwsExtensionRegistry

6.1.1.1 Method register

Test Area Reference: API_1_Ser_Regi.

6.1.1.1.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
static void register(ScwsExtension scwsExtension,
                     byte[] appId,
                     short offset,
                     short length)
              throws java.lang.NullPointerException,
                     java.lang.ArrayIndexOutOfBoundsException,
                     ScwsException
```

6.1.1.1.1.1 Normal execution

- CRRN1: registers the scwsExtension object to the SCWS.

6.1.1.1.1.2 Parameter Errors

- CRRP1: if scwsExtension is null, a NullPointerException is thrown.
- CRRP2: if appId is null, a NullPointerException is thrown.
- CRRP3: if offset or length or both would cause access outside of the array bounds, or if length is negative an ArrayIndexOutOfBoundsException is thrown.

6.1.1.1.1.3 Context errors

- CRRC1: if the appId is already registered, an ScwsException with reason code SCWSEXTENSION_REGISTRY_ERROR is thrown.

6.1.1.1.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Ser_Regi.java.

Test Applet: Api_1_Ser_Regi_1.java.

Cap File: Api_1_Ser_Regi.cap.

6.1.1.1.3 Test coverage

CRR number	Test case number
N1	8
P1	1
P2	2
P3	3, 4, 5, 6, 7
C1	9, 10

6.1.1.1.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
0	Instanciate an object implementing the scwsExtension interface		
1	**Null scwsExtension** register()	NullPointerException is thrown	
2	**Null appId** register()	NullPointerException is thrown	
3	**offset > appId.length** register() appId.length = 5 offset = 6 length = 0	ArrayIndexOutOfBoundsException is thrown	
4	**offset < 0** register() appId.length = 5 offset = -1 length = 1	ArrayIndexOutOfBoundsException is thrown	
5	**length > appId.length** register() appId.length = 5 offset = 0 length = 6	ArrayIndexOutOfBoundsException is thrown	
6	**offset + length > appId.length** register() appId.length = 5 offset = 3 length = 3	ArrayIndexOutOfBoundsException is thrown	
7	**length < 0** register() appId.length = 5 offset = 0 length = -1	ArrayIndexOutOfBoundsException is thrown	
	register the object implementing the scwsExtension interface with the name "Name1" to the SCWS.		
8	**successful registration** register() appId = (byte) 'N', (byte) 'a', (byte) 'm', (byte) 'e', (byte) '1' offset = 0 length = 5		
	register the same object a second time with the same name		
9	**unsuccessful registration** register() appId = (byte) 'N', (byte) 'a', (byte) 'm', (byte) 'e', (byte) '1' offset = 0 length = 5	ScwsException with reason code SCWSEXTENSION_REGISTRY_ERROR is thrown	
	retrieve a second object implementing the scwsExtension interface		
10	**register different object with the same appId** register() appId = (byte) 'N', (byte) 'a', (byte) 'm', (byte) 'e', (byte) '1' offset = 0 length = 5	ScwsException with reason code SCWSEXTENSION_REGISTRY_ERROR is thrown	

6.1.1.2 Method deregister

Test Area Reference: API_1_Ser_Dreg, Api_1_Ser_Dreg_Helper

6.1.1.2.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
static void deregister(ScwsExtension scwsExtension)
                throws java.lang.NullPointerException,
                       ScwsException,
                       java.lang.SecurityException
```

6.1.1.2.1.1 Normal execution

- CRRN1: deregisters the scwsExtension object from the SCWS.

6.1.1.2.1.2 Parameter Errors

- CRRP1: if scwsExtension is null, a NullPointerException is thrown.

6.1.1.2.1.3 Context errors

- CRRC1: if the scwsExtension was not registered before, an ScwsException with reason code SCWSEXTENSION_REGISTRY_ERROR is thrown.
- CRRC2: if the scwsExtension was previously registered by a different JCRE context than it is used for the deregistration, a java.lang.SecurityException shall be thrown.
- CRRC3: if the scwsExtension was already deregistered, an ScwsException with reason code SCWSEXTENSION_REGISTRY_ERROR is thrown.

6.1.1.2.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_ Api_1_Ser_Dreg.java.

Test Applet: Api_1_Ser_Dreg_1.java.

Api_1_Ser_Dreg_Helper.java.

Cap File: Api_1_Ser_Dreg.cap.

Cap File: Api_1_Ser_Dreg_Helper.cap.

6.1.1.2.3 Test coverage

CRR number	Test case number
N1	3
P1	2
C1	1
C2	5
C3	4

6.1.1.2.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
0	`Instanciate an object implementing the scwsExtension interface.`		
1	**No previous registration** `deregister()`	ScwsException with reason code SCWSEXTENSION_REGISTRY_ERROR is thrown	
	`Register the object to the SCWS`		

2	**Null scwsExtension**
	`deregister()`
	`scwsExtension = null`
		NullPointerException is thrown	

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
3	**Succesful deregistration** `deregister()`		
4	**Second deregistration** `deregister()`	ScwsException with reason code SCWSEXTENSION_REGISTRY_ERROR is thrown	
5	**Different context** `Request the SCWSExtension object from a different context and deregister it.`	java.lang.SecurityException is thrown	

6.1.2 Interface ScwsExtension

This interface is implicitely tested by the tests covering the SCWSExtensionService.

6.1.3 Class ScwsExtensionService

6.1.3.1 Method doDelete

Test Area Reference: Api_1_Ses_Ddel.

6.1.3.1.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
public void doDelete(HttpRequest request,
                     HttpResponse response)
            throws ScwsException
```

6.1.3.1.1.1 Normal execution

- n/a.

6.1.3.1.1.2 Parameter Errors

- n/a.

6.1.3.1.1.3 Context errors

- CRRC1: if a ScwsException with reason code METHOD_NOT_SUPPORTED was thrown by the application then an appropriate HTML status page is returned by the SCWS (status code 5xx, 4xx).

6.1.3.1.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Ses_Ddel.java.

Test Applet: Api_1_Ses_Ddel_1.java.

Cap File: Api_1_Ses_Ddel.cap.

6.1.3.1.3 Test coverage

CRR number	Test case number
C1	1

6.1.3.1.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
0	`Instantiate an doDeleteScwsExtensionServiceObject object and register to the SCWS. Map the object to the path "/Test_Api_1_Ses_Ddel" in the SCWS and send a HTTP Delete request to the SCWS triggering the application.`		
1	**Throw ScwsException with reason code METHOD_NOT_SUPPORTED**		Appropriate status code (405) is returned by the SCWS

6.1.3.2 Method doGet

Test Area Reference: Api_1_Ses_Dget.

6.1.3.2.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
public void doGet(HttpRequest request,
                  HttpResponse response)
            throws ScwsException
```

6.1.3.2.1.1 Normal execution

- n/a.

6.1.3.2.1.2 Parameter Errors

- n/a.

6.1.3.2.1.3 Context errors

- CRRC1: if an ScwsException with reason code METHOD_NOT_SUPPORTED was thrown by the application then an appropriate HTML status page is returned by the SCWS (status code 4xx, 5xx).

6.1.3.2.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Ses_Dget.java.

Test Applet: Api_1_Ses_Dget_1.java.

Cap File: Api_1_Ses_Dget.cap.

6.1.3.2.3 Test coverage

CRR number	Test case number
C1	1

6.1.3.2.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
0	`Instantiate an doGetScwsExtensionServiceObject object and register to the SCWS. Map the object to the path "/Test_Api_1_Ses_Dget" in the SCWS and send a HTTP Get request to the SCWS triggering the application.`		
1	**Throw ScwsException with reason code METHOD_NOT_SUPPORTED**		Appropriate status code (405) is returned by the SCWS

6.1.3.3 Method doHead

Test Area Reference: Api_1_Ses_Dhea.

6.1.3.3.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
public void doHead(HttpRequest request,
                   HttpResponse response)
            throws ScwsException
```

6.1.3.3.1.1 Normal execution

- n/a.

6.1.3.3.1.2 Parameter Errors

- n/a.

6.1.3.3.1.3 Context errors

- CRRC1: if an ScwsException with reason code METHOD_NOT_SUPPORTED was thrown by the application then an appropriate HTML status page is returned by the SCWS (status code 4xx, 5xx).

6.1.3.3.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Ses_Dhea.java.

Test Applet: Api_1_Ses_Dhea_1.java.

Cap File: Api_1_Ses_Dhea.cap.

6.1.3.3.3 Test coverage

CRR number	Test case number
C1	1

6.1.3.3.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
0	`Instantiate an doHeadScwsExtensionServiceObject object and register to the SCWS. Map the object to the path "/Test_Api_1_Ses_Dhea" in the SCWS and send a HTTP Head request to the SCWS triggering the application.`		
1	**Throw ScwsException with reason code METHOD_NOT_SUPPORTED**		Appropriate status code (405) is returned by the SCWS

6.1.3.4 Method doOptions

Test Area Reference: Api_1_Ses_Dopt.

6.1.3.4.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
public void doOptions(HttpRequest request,
                      HttpResponse response)
            throws ScwsException
```

6.1.3.4.1.1 Normal execution

- n/a.

6.1.3.4.1.2 Parameter Errors

- n/a.

6.1.3.4.1.3 Context errors

- CRRC1: if a ScwsException with reason code METHOD_NOT_SUPPORTED was thrown by the application (which is Intercept or Non Intercept Application) then an appropriate HTML status page is returned by the SCWS (status code 4xx, 5xx).

6.1.3.4.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Ses_Dopt.java.

Test Applet: Api_1_Ses_Dopt_1.java.

Cap File: Api_1_Ses_Dopt.cap.

6.1.3.4.3 Test coverage

CRR number	Test case number
C1	1

6.1.3.4.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
0	`Instantiate an doOptionsScwsExtensionServiceObject object and register to the SCWS. Map the object to the path "/Test_Api_1_Ses_Dopt" in the SCWS and send a HTTP Options request to the SCWS triggering the application.`		
1	**Throw ScwsException with reason code METHOD_NOT_SUPPORTED**		Appropriate status code (405) is returned by the SCWS

6.1.3.5 Method doPost

Test Area Reference: Api_1_Ses_Dpos.

6.1.3.5.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
public void doPost(HttpRequest request,
                   HttpResponse response)
            throws ScwsException
```

6.1.3.5.1.1 Normal execution

- n/a.

6.1.3.5.1.2 Parameter Errors

- n/a.

6.1.3.5.1.3 Context errors

- CRRC1: if an ScwsException with reason code METHOD_NOT_SUPPORTED was thrown by the application then an appropriate HTML status page is returned by the SCWS (status code 4xx, 5xx).

6.1.3.5.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Ses_Dpos.java.

Test Applet: Api_1_Ses_Dpos_1.java.

Cap File: Api_1_Ses_Dpos.cap.

6.1.3.5.3 Test coverage

CRR number	Test case number
C1	1

6.1.3.5.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
0	`Instantiate an doPostScwsExtensionServiceObject object and register to the SCWS. Map the object to the path "/Test_Api_1_Ses_Dpos" in the SCWS and send a HTTP Post request to the SCWS triggering the application.`		
1	**Throw ScwsException with reason code METHOD_NOT_SUPPORTED**		Appropriate status code (405) is returned by the SCWS

6.1.3.6 Method doPut

Test Area Reference: Api_1_Ses_Dput.

6.1.3.6.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
public void doPut(HttpRequest request,
                  HttpResponse response)
            throws ScwsException
```

6.1.3.6.1.1 Normal execution

- n/a.

6.1.3.6.1.2 Parameter Errors

- n/a.

6.1.3.6.1.3 Context errors

- CRRC1: if a ScwsException with reason code METHOD_NOT_SUPPORTED was thrown by the application (which is Intercept or Non Intercept Application) then an appropriate HTML status page is returned by the SCWS (status code 4xx, 5xx).

6.1.3.6.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Ses_Dput.java.

Test Applet: Api_1_Ses_Dput_1.java.

Cap File: Api_1_Ses_Dput.cap.

6.1.3.6.3 Test coverage

CRR number	Test case number
C1	1

6.1.3.6.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
0	`Instantiate an doPutScwsExtensionServiceObject object and register to the SCWS. Map the object to the path "/Test_Api_1_Ses_Dput" in the SCWS and send a HTTP Put request to the SCWS triggering the application.`		
1	**Throw ScwsException with reason code METHOD_NOT_SUPPORTED**		Appropriate status code (405) is returned by the SCWS

6.1.3.7 Method doTrace

Test Area Reference: Api_1_Ses_Dtra.

6.1.3.7.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
public void doTrace(HttpRequest request,
                    HttpResponse response)
             throws ScwsException
```

6.1.3.7.1.1 Normal execution

- n/a.

6.1.3.7.1.2 Parameter Errors

- n/a.

6.1.3.7.1.3 Context errors

- CRRC1: if a ScwsException with reason code METHOD_NOT_SUPPORTED was thrown by the application (which is Intercept or Non Intercept Application) then an appropriate HTML status page is returned by the SCWS (status code 4xx, 5xx).

6.1.3.7.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Ses_Dtra.java.

Test Applet: Api_1_Ses_Dtra_1.java.

Cap File: Api_1_Ses_Dtra.cap.

6.1.3.7.3 Test coverage

CRR number	Test case number
C1	1

6.1.3.7.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
0	Instantiate an doTraceScwsExtensionServiceObject object and register to the SCWS. Map the object to the path "/Test_Api_1_Ses_Dtra" in the SCWS and send a HTTP Trace request to the SCWS triggering the application.		
1	**Throw ScwsException with reason code METHOD_NOT_SUPPORTED**		Appropriate status code (405) is returned by the SCWS.

6.1.4 Interface HttpRequest

6.1.4.1 Method findAndCopyKeywordValue

Test Area Reference: Api_1_Hrq_Fckw.

6.1.4.1.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
short findAndCopyKeywordValue(byte[] headerKeywordName,
                              short nameOffs,
                              short nameLength,
                              byte[] buffer,
                              short bufferOffs,
                              short maxLength)
                       throws: java.lang.NullPointerException,
                              java.lang.ArrayIndexOutOfBoundsException.
```

6.1.4.1.1.1 Normal execution

- CRRN1: Find and copy the value of a header.
- CRRN2: bufferOffs+length of the copied value or KEYWORD_NOT_FOUND in case the keyword is not found.

6.1.4.1.1.2 Parameter Errors

- CRRP1: if headerKeywordName is equal to null a java.lang.NullPointerException is thrown.
- CRRP2: if buffer is equal to null a NullPointerException is thrown.
- CRRP3: if the copy operation would cause access out of bounds an ArrayIndexOutOfBoundsException is thrown.

6.1.4.1.1.3 Context errors

- n/a.

6.1.4.1.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Hrq_Fckw_Bss_Bss.java.

Test Applet: Api_1_Hrq_Fckw_Bss_Bss_1.java.

Cap File: Api_1_Hrq_Fckw_Bss_Bss.cap.

6.1.4.1.3 Test coverage

CRR number	Test case number
P1	1
P2	2
P3	3, 4, 5, 6, 7
N1	9
N2	8

6.1.4.1.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**findAndCopyKeywordValue () with a null headerKeywordName**	NullPointerException is thrown	
2	**findAndCopyKeywordValue () with a null buffer**	NullPointerException is thrown	

3	**bufferOffs ≥ buffer.length**
	`findAndCopyKeywordValue ()`
	`nameOffs = 0`
	`buffer.length = 2`
	`bufferOffs = 2`
	`maxLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

4	**bufferOffs < 0**
	`findAndCopyKeywordValue()`
	`buffer.length = 2`
	`bufferOffs = -1`
	`maxLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

5	**maxLength > buffer.length**
	`findAndCopyKeywordValue()`
	`buffer.length = 2`
	`bufferOffs = 0`
	`maxLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

6	**bufferOffs + maxLength > buffer.length**
	`findAndCopyKeywordValue()`
	`buffer.length = 2`
	`bufferOffs = 1`
	`maxLength = 2`
		ArrayIndexOutOfBoundsException is thrown	

7	**maxLength < 0**
	`findAndCopyKeywordValue()`
	`buffer.length = 2`
	`bufferOffs = 0`
	`maxLength = -1`
		ArrayIndexOutOfBoundsException is thrown	

8	`findAndCopyKeywordValue()`
	`headerKeywordName= HEADER_AB [ This header is` **`not`** `present in the request]`
	`bufferOffs = 0`
		ScwsConstants.KEYWORD_NOT_FOUND	

9	`findAndCopyKeywordValue()`
	`headerKeywordName= HEADER_AB [ This header is present in the request]`
	`bufferOffs = 1`
		lenght of Header AB (1) + bufferOffs (1) = 2	

6.1.4.2 Method findAndCopyKeywordValue

Test Area Reference: Api_1_Hrq_Fckw.

6.1.4.2.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
short findAndCopyKeywordValue(short keywordId,
                              byte[] buffer,
                              short bufferOffs,
                              short maxLength)
                       throws ScwsException,
                              NullPointerException,
                              ArrayIndexOutOfBoundsException
```

6.1.4.2.1.1 Normal execution

- CRRN1 Find and copy the value of a header or the value of a keyword/value part of the URI indicated by a `keywordId`. The supported keywords are defined in `ScwsConstants` Interface.

6.1.4.2.1.2 Parameter Errors

- CRRP1 `KEYWORD_NOT_FOUND` in case the keyword is not found.
- CRRP2 if the `keywordId` is not recognized a `ScwsException` - with reason `UNKNOWN_KEYWORD_ID`.
- CRRP3: if buffer is equal to null a NullPointerException is thrown.
- CRRP4: if the copy operation would cause access out of bounds an ArrayIndexOutOfBoundsException is thrown.

6.1.4.2.1.3 Context errors

- n/a.

6.1.4.2.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Hrq_Fckws_Bss.java.

Test Applet: Api_1_Hrq_Fckws_Bss_1.java.

Cap File: Api_1_Hrq_Fckws_Bss.cap.

6.1.4.2.3 Test coverage

CRR number	Test case number
P1	1
P2	2
P3	19
P4	20, 21, 22, 23, 24
N1	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

6.1.4.2.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_ACCEPT_ENCODING [not present in request] bufferOffs = 0	ScwsConstants.KEYWORD_NOT_FOUND	
2	findAndCopyKeywordValue() keywordId = 99 (Unknown) bufferOffs = 0	ScwsException - with reason UNKNOWN_KEYWORD_ID	
3	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_ACCEPT [present in request] bufferOffs = 0		
4	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_ACCEPT_CHARSET [present in request] bufferOffs = 0		
5	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_ACCEPT_ENCODING [present in request] bufferOffs = 0		
6	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_ACCEPT_LANGUAGE [present in request] bufferOffs = 0		
7	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_HOST [present in request] bufferOffs = 0		
8	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_USER_AGENT [present in request] bufferOffs = 0		
9	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_CONTENT_ENCODING [present in request] bufferOffs = 0		
10	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_CONTENT_LANGUAGE [present in request] bufferOffs = 0		
11	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_CONTENT_LENGTH [present in request] bufferOffs = 0		
12	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_CONTENT_TYPE [present in request] bufferOffs = 0		
13	findAndCopyKeywordValue() keywordId = ScwsConstants.HEADER_SERVER [present in request] bufferOffs = 0		
14	findAndCopyKeywordValue() keywordId = ScwsConstants.URI_TAG [present in request] bufferOffs = 0		
15	findAndCopyKeywordValue() keywordId = ScwsConstants.URI_SCHEMA_TAG [present in request] bufferOffs = 0		
16	findAndCopyKeywordValue() keywordId = ScwsConstants.URI_AUTHORITY_TAG [present in request] bufferOffs = 0		
17	findAndCopyKeywordValue() keywordId = ScwsConstants.URI_PATH_TAG [present in request] bufferOffs = 0		
18	findAndCopyKeywordValue() keywordId = ScwsConstants.URI_QUERY_TAG [present in request] bufferOffs = 0		

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
19	`findAndCopyKeywordValue() with a null buffer`	NullPointerException is thrown	

20	**bufferOffs ≥ buffer.length**
	`findAndCopyKeywordValue ()`
	`buffer.length = 2`
	`bufferOffs = 3`
	`maxLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

21	**bufferOffs < 0**
	`findAndCopyKeywordValue()`
	`buffer.length = 2`
	`bufferOffs = -1`
	`maxLength = 1`
		ArrayIndexOutOfBoundsException is thrown	

22	**maxLength > buffer.length**
	`findAndCopyKeywordValue()`
	`buffer.length = 2`
	`bufferOffs = 0`
	`maxLength = 3`
		ArrayIndexOutOfBoundsException is thrown	

23	**bufferOffs + maxLength > buffer.length**
	`findAndCopyKeywordValue()`
	`buffer.length = 2`
	`bufferOffs = 1`
	`maxLength = 2`
		ArrayIndexOutOfBoundsException is thrown	

24	**maxLength < 0**
	`findAndCopyKeywordValue()`
	`buffer.length = 2`
	`bufferOffs = 0`
	`maxLength = -1`
		ArrayIndexOutOfBoundsException is thrown	

6.1.4.3 Method getContentLength

Test Area Reference: Api_1_Hrq_Gcle.

6.1.4.3.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
int getContentLength()
```

6.1.4.3.1.1 Normal execution

- CRRN1: Returns the length of the HTTP request content.

6.1.4.3.1.2 Parameter Errors

- None.

6.1.4.3.1.3 Context errors

- None.

6.1.4.3.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Hrq_Gcle.java.

Test Applet: Api_1_Hrq_Gcle_1.java.

Cap File: Api_1_Hrq_Gcle.cap.

6.1.4.3.3 Test coverage

CRR number	Test case number
N1	1, 2

6.1.4.3.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**Normal execution** `Send a HTTP request of 0xFF bytes.`	getContentLenght() returns `0xFF`	
2	**Normal execution** `Send a HTTP request of 0x00 bytes.`	getContentLenght() returns `0x00`	
NOTE: Tests regarding the upper boundaries of the HTTPRequest buffer can not be performed since a minimum size of the request buffer is not specified.			

6.1.4.4 Method getContentType

Test Area Reference: Api_1_Hrq_Gcty.

6.1.4.4.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
short getContentType()
```

6.1.4.4.1.1 Normal execution

- CRRN1: Returns the content-type keyword of the HTTP request or an appropriate error code.

6.1.4.4.1.2 Parameter Errors

- None.

6.1.4.4.1.3 Context errors

- None.

6.1.4.4.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Hrq_Gcty.java.

Test Applet: Api_1_Hrq_Gcty_1.java.

Cap File: Api_1_Hrq_Gcty.cap.

6.1.4.4.3 Test coverage

CRR number	Test case number
N1	1,2,3,4

6.1.4.4.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**Normal execution** Send a HTTP request with content type set to: text/html.	Returns ScwsConstants.CONTENT_TYPE_TEXT_HTML	
2	**Normal execution** Send a HTTP request with content type set to: text/plain.	Returns ScwsConstants.CONTENT_TYPE_TEXT_PLAIN	
3	**Keyword not found** Send a HTTP request with content type header field missing	Returns ScwsConstants.KEYWORD_NOT_FOUND	
4	**Content type unknown** Send a HTTP request with content type which doesn't match one of the possible contents	Returns ScwsConstants.CONTENT_TYPE_UNKNOWN	

6.1.4.5 Method getRequestHttpVersion

Test Area Reference: Api_1_Hrq_Grhv.

6.1.4.5.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
short getRequestHttpVersion()
```

6.1.4.5.1.1 Normal execution

- CRRN1: Returns the version of the HTTP protocol of the current request.

6.1.4.5.1.2 Parameter Errors

- None.

6.1.4.5.1.3 Context errors

- None.

6.1.4.5.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Hrq_Grhv.java.

Test Applet: Api_1_Hrq_Grhv_1.java.

Cap File: Api_1_Hrq_Grhv.cap.

6.1.4.5.3 Test coverage

CRR number	Test case number
N1	1, 2

6.1.4.5.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**Normal execution** Send a HTTP request with protocol version set to: HTTP/1.0.	Returns ScwsConstants.HTTP_PROTOCOL_VERSION_10	
2	**Normal execution** Send a HTTP request with protocol version set to: HTTP/1.1.	Returns ScwsConstants.HTTP_PROTOCOL_VERSION_11	

6.1.4.6 Method readContent

Test Area Reference: Api_1_Hrq_Rcon.

6.1.4.6.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
short readContent(int requestOffset,
                  byte[] data,
                  short offset,
                  short len)
          throws: ScwsException
                  java.lang.NulPointerException
                  java.lang.ArrayIndexOutOfBoundsException
```

6.1.4.6.1.1 Normal execution

- CRRN1: Read a part of the request content and copy it in the data buffer.

6.1.4.6.1.2 Parameter Errors

- CRRP1: if the data buffer is equal to null, a java.lang.Null PointerException is thrown.
- CRRP2: if the copy operation would cause access out of bounds, a java.lang.ArrayIndexOutOfBoundsException will be thrown.
- CRRP3: if the requested offset is out of the request bounds, a ScwsException.UNREACHABLE_OFFSET will be thrown.

6.1.4.6.1.3 Context errors

- None.

6.1.4.6.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Hrq_Rcon.java.

Test Applet: Api_1_Hrq_Rcon_1.java.

Cap File: Api_1_Hrq_Rcon.cap.

6.1.4.6.3 Test coverage

CRR number	Test case number
N1	1, 2
P1	3
P2	4, 5, 6, 7, 8
P3	9, 10

6.1.4.6.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**Normal execution** Send a request with Content-length set to 0x20 readContent() requestOffset = 0 offset = 0 len = 0x10	Data buffers contains the first bytes of the request content	
2	**Normal execution** Send a request with Content-length set to 0x20 readContent() requestOffset = 0 offset = 0 len = 0x20	Data buffers contains the entire request content	
3	**Null pointer exception** Send a request with Content-length set to 0x20 readContent() data[] is set to null	NullPointerException is thrown	
4	**offset > data.length** Send a request with Content-length set to 0x20 readContent() data.length = 0x20 offset = 0x20 length = 0x10	ArrayIndexOutOfBoundException is thrown	
5	**length > data.length** Send a request with Content-length set to 0x20 readContent() data.length = 0x1F offset = 0x00 length = 0x20	ArrayIndexOutOfBoundException is thrown	
6	**offset + length > data.length** Send a request with Content-length set to 0x20 readContent() data.length = 0x20 offset = 0x10 length = 0x11	ArrayIndexOutOfBoundException is thrown	
7	**length < 0** Send a request with Content-length set to 0x20 readContent() data.length = 0x20 offset = 0x00 length = -1	ArrayIndexOutOfBoundException is thrown	
8	**offset < 0** Send a request with Content-length set to 0x20 readContent() data.length = 0x20 offset = -1 length = 0x10	ArrayIndexOutOfBoundException is thrown	

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
9	**requestOffset > request length** Send a request with Content-length set to 0x20 readContent() Request length is 0x20 requestOffset = 0x20 data.length = 0x20 offset = 0 length = 0x10	ScwsException.UNREACHABLE_OFFSET is thrown	
10	**requestOffset < 0** Send a request with Content-length set to 0x20 readContent() Request length is 0x20 requestOffset = -1 data.length = 0x20 offset = 0 length = 0x10	ScwsException.UNREACHABLE_OFFSET is thrown	

6.1.4.7 Method readContentType

Test Area Reference: Api_1_Hrq_Rcty.

6.1.4.7.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
short readContentType(byte[] data,
                      short offs,
                      short len)
              throws: java.lang.NulPointerException
                      java.lang.ArrayIndexOutOfBoundsException
```

6.1.4.7.1.1 Normal execution

- CRRN1: Reads the content type given in the HTTP request and copy it in the data[ ] buffer. Returns the offset of the last data written in the buffer.

6.1.4.7.1.2 Parameter Errors

- CRRP1: if the data buffer is null, a java.lang.NullPointerException will be thrown.
- CRRP2: if the copy operation would cause access out of bounds, a java.lang.ArrayIndexOutOfBoundsException will be thrown.

6.1.4.7.1.3 Context errors

- None.

6.1.4.7.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_1_Hrq_Rcty.java.

Test Applet: Api_1_Hrq_Rcty_1.java.

Cap File: Api_1_Hrq_Rcty.cap.

6.1.4.7.3 Test coverage

CRR number	Test case number
N1	1, 2
P1	3
P2	4, 5, 6, 7, 8

6.1.4.7.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**Normal execution** Send a HTTP request with content type set to: text/html. readContentType() data.length = 0x20 offs = 0x00 len = 0x0A	text/html is written in the data buffer and the method returns 0x08	
2	**Normal execution** Send a HTTP request with content type set to: text/html. readContentType() data.length = 0x20 offs = 0x00 len = 0x05	text/ is written in the data buffer and the method returns 0x04	
3	**Null pointer** Send a HTTP request with content type set to: text/html readContentType() data = null offs = 0x00 len = 0x0A	NullPointerException is thrown	
4	**offs > data.length** Send a HTTP request with content type set to: text/html. readContentType() data.length = 0x20 offs = 0x20 len = 0x10	ArrayIndexOutOfBoundException is thrown	
5	**len > data.length** Send a HTTP request with content type set to: text/html. readContentType() data.length = 0x05 offs = 0x00 len = 0x09	ArrayIndexOutOfBoundException is thrown	
6	**offs + len > data.length** Send a HTTP request with content type set to: text/html. readContentType() data.length = 0x05 offs = 0x01 len = 0x05	ArrayIndexOutOfBoundException is thrown	
7	**len < 0** Send a HTTP request with content type set to: text/html. readContent() data.length = 0x20 offs = 0x00 Len = -1	ArrayIndexOutOfBoundException is thrown	
8	**offs < 0** Send a HTTP request with content type set to: text/html. readContent() data.length = 0x20 offs = -1 Len = 0x10	ArrayIndexOutOfBoundException is thrown	

6.1.5 Interface HttpResponse

6.1.5.1 Method appendContent

Test Area Reference: Api_1_Hrs_Acon.

6.1.5.1.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void appendContent(byte[] data,
                   short offset,
                   short length)
                   throws ScwsException
```

6.1.5.1.1.1 Normal execution

- CRRN1: Writes the content of `data` into the `HttpResponse`.
- CRRN2: Invoking this method implicitly finalizes the header of the `HttpResponse`.
- CRRN3: In case of chunked transfer encoding hex length of chunked data is automatically added by the framework.

6.1.5.1.1.2 Parameter Errors

- CRRP1: if data is null, a java.lang.NullPointerException is thrown.
- CRRP2: if parameter offset is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP3: if parameter length is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP4: if parameter offset plus length are greater than the length of array data, a java.lang.ArrayIndexOutOfBoundsException is thrown.

6.1.5.1.1.3 Context errors

- CRRC1: if appending the content would cause an overflow of the response buffer, a uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown.
- CRRC2: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC3: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.1.2 Test area files

Specific triggering: None:

Test Source:	Test_Api_1_Hrs_Acon.java.
Test Applet:	Api_1_Hrs_Acon _1.java.
Cap File:	Api_1_Hrs_Acon.cap.

6.1.5.1.3 Test coverage

CRR number	Test case number
N1	1, 2, 3, 4, 5, 14
N2	4
N3	2, 14
P1	6
P2	7
P3	8
P4	9
C1	12, 13
C2	10
C3	11

6.1.5.1.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**append chunk off** writeStatusCode(200) finalizeHeader() appendContent(data = [Hello World] offset = 0 length = 6) appendContent(data = [Hello World] offset = 6 length = 5) flush()		HTTPResponse (statusCode = "**200**" Header("**Content-Length**") = "**11**" contentBody = "**Hello World**")
2	**append chunk on** writeStatusCode(200) enableChunkMode() finalizeHeader() appendContent(data = [Hello World] offset = 0 length = 6) appendContent(data = [Hello World] offset = 6 length = 5) flush()		HTTPResponse (statusCode = "**200**" Header("**Transfer-Encoding**") = "**chunked**" contentBody = "**Hello World**")

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
3	**append up to remaining buffer length** writeStatusCode(200) finalizeHeader() appendContent(data = 'aaa...' offset = 0 length = remainingBufferLength) flush()		HTTPResponse (statusCode = **"200"** Header(**"Content-Length"**) = *bufferLength* contentBody = **"aaa..."**)
4	**append implicitly finalizes the header** writeStatusCode(200) appendContent(data= [Hello World] offset= 0 length= 11) flush()		HTTPResponse (statusCode = **"200"** Header(**"Content-Length"**) = **"11"** contentBody = **"Hello World"**)
5	**length = 0** writeStatusCode(200) finalizeHeader() appendContent(data = [Hello World] offset= 0 length= 0) flush()		HTTPResponse (statusCode = **"200"** Header(**"Content-Length"**) = **"0"** contentBody = **""**)
6	**data[] is null** appendContent(data= null offset = 0 length = 10)	java.lang.NullPointerException is thrown : Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = **"204"**)
7	**offset<0** appendContent(data = [Hello World] offset = -1 length = 10)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = **"204"**)
8	**length<0** appendContent(data = [Hello World] offset = 0 length = -1)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = **"204"**)
9	**offset + length > data.length** appendContent(data = [Hello World] offset = 10 length = 11)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = **"204"**)

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
10	**flush called previously** writeStatusCode(200) finalizeHeader() appendContent(data = 'Hello' offset = 0 length = 5) flush() appendContent(data = 'World' offset = 0 length = 5)	uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown. Catch the exception and continue the procedure.	HTTPResponse (statusCode = **"200"** Header(**"Content-Length"**) = **"5"** contentBody = **"Hello"**)
11	**sendError called previously** sendError(500) appendContent(data = 'World' offset = 0 length = 5)	uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown. Catch the exception and continue the procedure.	HTTPResponse (statusCode = **"500"**)
12	**response buffer overflow** writeStatusCode(200) finalizeHeader() appendContent(data = 'aaa...' offset = 0 length = responseBuffer+1)	uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown. Catch the exception and continue the procedure.	HTTPResponse (statusCode = **"200"** Header(**"Content-Length"**) = **"0"** contentBody = **""**)
13	**Void**		
14	**Void**		
15	**Not present** writeStatusCode(200) finalizeHeader() flush()		HTTPResponse (statusCode = **"200"** Header(**"Content-Length"**) = **"0"** ContentBody = **""**)

6.1.5.2 Method appendHeaderVariable(byte[] data, short offset, short length)

Test Area Reference: Api_1_Hrs_Ahva_Bss.

6.1.5.2.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void appendHeaderVariable(byte[] data,
                          short offset,
                          short length)
                          throws ScwsException
```

6.1.5.2.1.1 Normal execution

- CRRN1: Writes a header variable to outgoing buffer, "CRLF" will be appended automatically.
- CRRN2: The header variables content-length or transfer-encoding are inserted by the framework depending on the transfer mode.

6.1.5.2.1.2 Parameter Errors

- CRRP1: if data is null, a java.lang.NullPointerException is thrown.
- CRRP2: if parameter offset is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP3: if parameter length is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP4: if parameter offset plus length are greater than the length of array data, a java.lang.ArrayIndexOutOfBoundsException is thrown.

6.1.5.2.1.3 Context errors

- CRRC1: if the HTTP header was already finalized, a uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
- CRRC2: if appending the header variable would cause an overflow of the response buffer, a uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown.
- CRRC3: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC4: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.2.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Ahva_Bss.java.

Test Applet: Api_1_Hrs_Ahva_Bss_1.java.

Cap File: Api_1_Hrs_Ahva_Bss.cap.

6.1.5.2.3 Test coverage

CRR number	Test case number
N1	1, 2, 3, 4
N2	1, 2, 3, 4
P1	5
P2	6
P3	7
P4	8
C1	9, 10
C2	11
C3	12
C4	13

6.1.5.2.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

1	**Basic**
	
	`writeStatusCode(200)`
	
	`appendHeaderVariable(`
	`  data= [A: B],`
	`  offset= 0,`
	`  length= 4)`
	
	`finalizeHeader();`
	`appendContent([OK],0,2);`
	`flush();`
			`HTTPResponse (`
			` statusCode = "200"`
			` Header("A") = "B"`
			` Header("Content-Length") = "2"`
			` contentBody = "OK"`
			`)`

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

2	**offset>0**
	
	writeStatusCode(200)
	
	appendHeaderVariable(
	data= [A: BC: Dd],
	offset= 4,
	length= 5)
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "200"
			Header("C")= "Dd"
			Header("Content-Length") = "2"
			contentBody= "OK"
			)

3	**No SP char**
	
	writeStatusCode(200)
	
	appendHeaderVariable(
	data= [A:B],
	offset= 0,
	length= 3)
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "200"
			Header("A")= "B"
			Header("Content-Length")= "2"
			contentBody= "OK"
			)

4	**length = 0**
	
	writeStatusCode(200)
	
	appendHeaderVariable(
	data= [A: B],
	offset = 0,
	length = 0)
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "200"
			Header("Content-Length")= "2"
			contentBody= "OK"
			)

5	**data[] is null** appendHeaderVariable(data = null, offset = 0, length = 2)	java.lang.NullPointerException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")
6	**offset<0** appendHeaderVariable(data= [A: B], offset= -1, length= 4)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")
7	**Length<0** appendHeaderVariable(data = [A: B], offset = 0, length = -1)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")
8	**valueOffset + valueLength > value.length** appendHeaderVariable(data = [A: B], valueOffset= 1, length = 4)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
9	**finalizeHeader called previously** writeStatusCode(200) appendHeaderVariable(data = [A: B], valueOffset= 0, length = 4) finalizeHeader();		HTTPResponse (statusCode= "**200**" Header("**A**")= "**B**" Header("**Content-Length**")= "**2**" HeaderNotPresent("C") contentBody= "**OK**")

	appendHeaderVariable(
	data = [C: D],
	valueOffset= 0,
	length = 4)
	
	appendContent([OK],0,2);
	flush();
		Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
		Catch the exception and continue the procedure.	

10	**appendContent called previously** writeStatusCode(200) appendHeaderVariable(data = [A: B], valueOffset= 0, length = 4) appendContent([OK],0,2);		HTTPResponse (statusCode= "**200**" Header("**A**")= "**B**" Header("**Content-Length**")= "**2**" HeaderNotPresent("C") contentBody= "**OK**")
	appendHeaderVariable(data = [C: D], valueOffset= 0, length = 4) flush();	Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown. Catch the exception and continue the procedure.	
11	**BUFFER_OVERFLOW** writeStatusCode(200) appendHeaderVariable(data = [A: BbB...], valueOffset= 0, length = remainingBufferLength+1)	uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown. Catch the exception and continue the procedure.	HTTPResponse (statusCode = "200" Header("**Content-Length**")= "**0**" HeaderNotPresent("A")).

12	**flush called previously**
	
	writeStatusCode(200)
	
	appendHeaderVariable(
	data = [A: B],
	valueOffset= 0,
	length = 4)
	
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "**200**"
			Header("**A**")= "**B**"
			Header("**Content-Length**") = "**2**"
			HeaderNotPresent("C")
			contentBody= "**OK**"
			)

	appendHeaderVariable(data = [C: D], valueOffset= 0, length = 4)	uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown. Catch the exception and continue the procedure.	
13	**sendError called previously** sendError(500)		HTTPResponse (statusCode= "500" HeaderNotPresent("A"))
	appendHeaderVariable(data = [A: B], valueOffset= 0, length = 4)	uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown. Catch the exception and continue the procedure.	

6.1.5.3 Method appendHeaderVariable (byte[] name, short nameOffset, short nameLength, byte[] value, short valueOffset, short valueLength)

Test Area Reference: Api_1_Hrs_Ahva_Bss_Bss.

6.1.5.3.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void appendHeaderVariable(byte[] name,
                          short nameOffset,
                          short nameLength,
                          byte[] value,
                          short valueOffset,
                          short valueLength)
                          throws ScwsException
```

6.1.5.3.1.1 Normal execution

- CRRN1: Appends a header variable to outgoing. The colon and space (":") as well as the "CRLF" will be appended automatically.
- CRRN2: The header variables content-length or transfer-encoding are inserted by the framework depending on the transfer mode.

6.1.5.3.1.2 Parameter Errors

- CRRP1: if name is null, a java.lang.NullPointerException is thrown.
- CRRP2: if value is null, a java.lang.NullPointerException is thrown.
- CRRP3: if parameter nameOffset is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP4: if parameter nameLength is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP5: if parameter nameOffset plus nameLength are greater than the length of array name, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP6: if parameter valueOffset is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP7: if parameter valueLength is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP8: if parameter valueOffset plus valueLength are greater than the length of array value, a java.lang.ArrayIndexOutOfBoundsException is thrown.

6.1.5.3.1.3 Context errors

- CRRC1: if the HTTP header was already finalized, a uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
- CRRC2: if appending the header variable would cause an overflow of the response buffer, a uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown.
- CRRC3: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC4: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.3.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Ahva_Bss_Bss .java.

Test Applet: Api_1_Hrs_Ahva_Bss_Bss_1.java.

Cap File: Api_1_Hrs_Ahva_Bss_Bss.cap.

6.1.5.3.3 Test coverage

CRR number	Test case number
N1	1, 2, 3, 4, 5
N2	1, 2, 3, 4, 5
P1	6
P2	7
P3	8
P4	10
P5	12
P6	9
P7	11
P8	13
C1	14, 15
C2	16
C3	17
C4	18

6.1.5.3.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

1	**Basic**
	
	writeStatusCode(200)
	
	appendHeaderVariable(
	name= [A],
	nameOffset= 0,
	nameLength= 1,
	value=[B],
	value Offset= 0,
	value Length= 1)
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= **"200"**
			Header(**"A"**)= **"B"**
			Header(**"Content-Length"**)= **"2"**
			contentBody= **"OK"**
			)

2	**nameOffset>0**
	
	writeStatusCode(200)
	
	appendHeaderVariable(
	name= [XAa],
	nameOffset= 1,
	nameLength= 2,
	value=[B],
	value Offset= 0,
	value Length= 1)
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= **"200"**
			Header(**"Aa"**)= **"B"**
			Header(**"Content-Length"**)= **"2"**
			contentBody= **"OK"**
			)

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

3	**valueOffset>0**
	
	writeStatusCode(200)
	
	appendHeaderVariable(
	name= [A],
	nameOffset= 0,
	nameLength= 1,
	value=[XBb],
	valueOffset= 1,
	valueLength= 2)
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "**200**"
			Header("**A**")= "**Bb**"
			Header("**Content-Length**")= "**2**"
			contentBody= "**OK**"
			)

4	**nameLength = 0**
	
	writeStatusCode(200)
	
	appendHeaderVariable(
	name= [A],
	nameOffset= 0,
	nameLength= 0,
	value=[B],
	valueOffset= 0,
	valueLength= 1)
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush()
		When header name length is 0 then nothing is append to the responseBuffer.	HTTPResponse (
			statusCode= "**200**"
			Header("**Content-Length**")= "**2**"
			contentBody= "**OK**"
			HeaderNotPresent("A")
			)

5	**valueLength = 0**
	
	writeStatusCode(200)
	
	appendHeaderVariable(
	name= [A],
	nameOffset= 0,
	nameLength= 1,
	value=[B],
	valueOffset= 0,
	valueLength= 0)
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "**200**"
			Header("**Content-Length**")= "**2**"
			Header("**A**")= ""
			contentBody= "**OK**"
			)

6	**name[] is null** appendHeaderVariable(name= null, nameOffset= 0, nameLength= 1, value=[B], value Offset= 0, value Length= 1)	java.lang.NullPointerException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")
7	**value[] is null** appendHeaderVariable(name= [A], nameOffset= 0, nameLength= 1, value= null, value Offset= 0, value Length= 1)	java.lang.NullPointerException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")
8	**nameOffset<0** appendHeaderVariable(name= [A], nameOffset= -1, nameLength= 1, value=[B], valueOffset= 0, valueLength= 1)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
9	**valueOffset<0** appendHeaderVariable(name= [A], nameOffset= 0, nameLength= 1, value=[B], valueOffset= -1, valueLength= 1)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")
10	**nameLength<0** appendHeaderVariable(name= [A], nameOffset= 0, nameLength= -1, value=[B], valueOffset= 0, valueLength= 1)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")
11	**valueLength<0** appendHeaderVariable(name= [A], nameOffset= 0, nameLength= 1, value=[B], valueOffset= 0, valueLength= -1)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")
12	**nameOffset + nameLength > name.length** appendHeaderVariable(name= [A], nameOffset= 0, nameLength= 2, value=[B], valueOffset= 0, valueLength= 1)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")
13	**valueOffset + valueLength > value.length** appendHeaderVariable(name= [A], nameOffset= 0, nameLength= 1, value=[B], valueOffset= 0, valueLength= 2)	java.lang.ArrayIndexOutOfBoundsException is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")

14	**finalizeHeader called previously**
	
	writeStatusCode(200)
	appendHeaderVariable(
	name= [A],
	nameOffset= 0,
	nameLength= 1,
	value=[B],
	valueOffset= 0,
	valueLength= 1)
	finalizeHeader();
	
	appendHeaderVariable(
	name= [C],
	nameOffset= 0,
	nameLength= 1,
	value=[D],
	valueOffset= 0,
	valueLength= 1)
	
	appendContent([OK],0,2);
	
	flush();
		Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= "**200**"
			Header("**A**")= "**B**"
			Header("**Content-Length**")= "**2**"
			HeaderNotPresent("C")
			contentBody= "**OK**"
			)

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
15	**appendContent called previously** writeStatusCode(200) appendHeaderVariable(name= [A], nameOffset= 0, nameLength= 1, value=[B], valueOffset= 0, valueLength= 1) appendContent([OK],0,2);		HTTPResponse (statusCode= "**200**" Header("**A**")= "**B**" Header("**Content-Length**")= "**2**" HeaderNotPresent("C") contentBody= "**OK**")
	appendHeaderVariable(name= [C], nameOffset= 0, nameLength= 1, value=[D], valueOffset= 0, valueLength= 1) flush();	Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown. Catch the exception and continue the procedure.	
16	**BUFFER_OVERFLOW** writeStatusCode(200) appendHeaderVariable(name= [A], nameOffset= 0, nameLength= 1, value=[B], valueOffset= 0, valueLength= (remainingBufferLength+1))	uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown. Catch the exception and continue the procedure.	HTTPResponse (statusCode = "200" Header("**Content-Length**")= "**0**" HeaderNotPresent("A"))

17	**flush called previously**
	
	writeStatusCode(200)
	appendHeaderVariable(
	name= [A],
	nameOffset= 0,
	nameLength= 1,
	value=[B],
	valueOffset= 0,
	valueLength= 1)
	appendContent([OK],0,2);
	flush();
	
	appendHeaderVariable(
	name= [C],
	nameOffset= 0,
	nameLength= 1,
	value=[D],
	valueOffset= 0,
	valueLength= 1)
		uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= "**200**"
			Header("**A**")= "**B**"
			Header("**Content-Length**")= "**2**"
			HeaderNotPresent("C")
			contentBody= "**OK**"
			)

18	**sendError called previously** sendError(500) appendHeaderVariable(name= [A], nameOffset= 0, nameLength= 1, value=[B], valueOffset= 0, valueLength= 1)	uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown. Catch the exception and continue the procedure.	HTTPResponse (statusCode= "**500**" HeaderNotPresent("A"))

6.1.5.4 Method appendHeaderVariable (short headerKeywordNameId, byte[] value, short valueOffset, short valueLength)

Test Area Reference: Api_1_Hrs_Ahvas_Bss.

6.1.5.4.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void appendHeaderVariable(short headerKeywordNameId,
                          byte[] value,
                          short valueOffset,
                          short valueLength)
                          throws ScwsException
```

6.1.5.4.1.1 Normal execution

- CRRN1: Appends a header variable to outgoing. The colon and space (":") as well as the "CRLF" will be appended automatically.
- CRRN2: The header variables content-length or transfer-encoding are inserted by the framework depending on the transfer mode.

6.1.5.4.1.2 Parameter Errors

- CRRP1: if value is null, a java.lang.NullPointerException is thrown.
- CRRP2: if parameter valueOffset is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP3: if parameter valueLength is negative, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP4: if parameter valueOffset plus valueLength are greater than the length of array data, a java.lang.ArrayIndexOutOfBoundsException is thrown.
- CRRP5: if parameter `headerKeywordNameId` is not correct according to `SCWSConstants, a uicc.scws.ScwsException with reason codes UNKNOWN_KEYWORD_ID` is thrown.

6.1.5.4.1.3 Context errors

- CRRC1: if the HTTP header was already finalized, a uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
- CRRC2: if appending the header variable would cause an overflow of the response buffer, a uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown.
- CRRC3: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC4: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.4.2 Test area files

Specific triggering: None

Test Source:	Test_Api_1_Hrs_Ahva_sBss.java.
Test Applet:	Api_1_Hrs_ Ahva_sBss_1.java.
Cap File:	Api_1_Hrs_ Ahva_sBss.cap.

6.1.5.5 Method enableChunkMode

Test Area Reference: Api_1_Hrs_Encm.

6.1.5.5.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void enableChunkMode()
```

6.1.5.5.1.1 Normal execution

- CRRN1: Informs the SCWS to use the chunked mode for sending the response.
- CRRN2: Chunked mode off is the default mode.
- CRRN3: This allows the SCWS to handle large amounts of response data.

6.1.5.5.1.2 Parameter Errors

- n/a.

6.1.5.5.1.3 Context errors

- CRRC1: if the HTTP header was already finalized, a uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
- CRRC2: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC3: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.5.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Encm.java.

Test Applet: Api_1_Hrs_Encm_1.java.

Cap File: Api_1_Hrs_Encm.cap.

6.1.5.5.3 Test coverage

CRR number	Test case number
N1	1, 2, 3, 11
N2	4
N3	5, 6
C1	7, 8
C2	9
C3	10

6.1.5.5.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

1	**call in First**
	
	enableChunkMode()
	
	writeStatusCode(200)
	appendHeaderVariable(
	headerKeywordNameId= 1,
	value= [header1header2],
	valueOffset= 0,
	valueLength= 7)
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "**200**"
			Header("**Accept**")= "**header1**"
			Header("**Transfer-Encoding**")= "**chunked**"
			contentBody= "OK"
			)

2	**write StatusCode call before**
	
	writeStatusCode(200)
	
	enableChunkMode()
	
	appendHeaderVariable(
	headerKeywordNameId= 1,
	value= [header1header2],
	valueOffset= 0,
	valueLength= 7)
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "**200**"
			Header("**Accept**")= "**header1**"
			Header("**Transfer-Encoding**")= "**chunked**"
			contentBody= "**OK**"
			)

3	**appendHeaderVariable call before**
	
	writeStatusCode(200)
	appendHeaderVariable(
	headerKeywordNameId= 1,
	value= [header1header2],
	valueOffset= 0,
	valueLength= 7)
	
	enableChunkMode()
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "**200**"
			Header("**Accept**")= "**header1**"
			Header("**Transfer-Encoding**")= "**chunked**"
			contentBody= "**OK**"
			)

4	**default mode off**
	
	writeStatusCode(200)
	appendHeaderVariable(
	headerKeywordNameId= 1,
	value= [header1header2],
	valueOffset= 0,
	valueLength= 7)
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= "**200**"
			Header("**Accept**")= "**header1**"
			Content-Length: "**2**"
			HeaderNotPresent("**Transfer-Encoding**")
			contentBody= "**OK**"
			)

5	**large data several append**
	
	writeStatusCode(200)
	
	enableChunkMode()
	
	finalizeHeader();
	appendContent([aaa...],0, 2048);
	appendContent([aaa...],0, 2048);
	appendContent([aaa...],0, 2048);
	appendContent([aaa...],0, 2048);
	appendContent([aaa...],0, 2048);
	appendContent([aaa...],0, 2048);
	appendContent([aaa...],0, 2048);
	appendContent([aaa...],0, 2048);
	appendContent([aaa...],0, 2048);
	appendContent([aaa...],0, 2048);
	flush();
			HTTPResponse (
			statusCode= "**200**"
			Header("**Transfer-Encoding**")= "**chunked**"
			contentBody= "**aaa...**"
			)

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

6	**large data one append**
	
	writeStatusCode(200)
	
	enableChunkMode()
	
	finalizeHeader();
	appendContent([aaa...],0, 10240);
	flush();
			HTTPResponse (
			statusCode= "200"
			Header("**Transfer-Encoding**")= "**chunked**"
			contentBody= "aaa..."
			)

7	**header already finalized 01**
	
	writeStatusCode(200)
	appendHeaderVariable(
	headerKeywordNameId= 1,
	value= [header1header2],
	valueOffset= 0,
	valueLength= 7)
	finalizeHeader();
	
	enableChunkMode();
	
	appendContent([OK],0,2);
	flush();
		Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= "**200**"
			Header("**Accept**")= "**header1**"
			Header("**Content-Length**")= "**2**"
			HeaderNotPresent("**Transfer-Encoding**")
			contentBody= "**OK**"
			)

8	**header already finalized 02**
	
	writeStatusCode(200)
	appendHeaderVariable(
	headerKeywordNameId= 1,
	value= [header1header2],
	valueOffset= 0,
	valueLength= 7)
	finalizeHeader();
	appendContent([OK],0,2);
	
	enableChunkMode();
	
	flush();
		Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= "**200**"
			Header("**Accept**")= "**header1**"
			Header("**Content-Length**")= "**2**"
			HeaderNotPresent("**Transfer-Encoding**")
			contentBody= "**OK**"
			)

9	**flush called previously**
	
	writeStatusCode(200)
	appendHeaderVariable(
	headerKeywordNameId= 1,
	value= [header1header2],
	valueOffset= 0,
	valueLength= 7)
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
	
	enableChunkMode();
	
		Uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= "**200**"
			Header("**Accept**")= "**header1**"
			Header("**Content-Length**")= "**2**"
			HeaderNotPresent("**Transfer-Encoding**")
			contentBody= "**OK**"
			)

10	**sendError called previously** sendError(500) enableChunkMode(); 	Uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown. Catch the exception and continue the procedure.	HTTPResponse (statusCode= "**500**")

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

11	**Multiple success call**
	
	`enableChunkMode()`
	
	`writeStatusCode(200)`
	
	`enableChunkMode()`
	
	`appendHeaderVariable(`
	`  headerKeywordNameId= 1,`
	`  value= [header1header2],`
	`  valueOffset= 0,`
	`  valueLength= 7)`
	
	`enableChunkMode()`
	`enableChunkMode()`
	
	`finalizeHeader();`
	`appendContent([OK],0,2);`
	`flush();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Accept")= "header1"`
			` Header("**Transfer-Encoding**")= "**chunked**"`
			` contentBody= "OK"`
			`)`

6.1.5.6 Method finalizeHeader

Test Area Reference: Api_1_Hrs_Finh.

6.1.5.6.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void finalizeHeader()throws ScwsException
```

6.1.5.6.1.1 Normal execution

- CRRN1: Finalizes the header data in the outgoing buffer. It automatically adds missing header data with default values by the SCWS.

6.1.5.6.1.2 Parameter Errors

- n/a.

6.1.5.6.1.3 Context errors

- CRRC1: if the HTTP header was already finalized, a uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
- CRRC2: if the adding of missing header data would cause access outside the Response buffer, a uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown.
- CRRC3: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC4: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.6.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Finh.java.

Test Applet: Api_1_Hrs_Finh_1.java.

Cap File: Api_1_Hrs_Finh.cap.

6.1.5.6.3 Test coverage

CRR number	Test case number
N1	1
C1	5, 6
C2	-
C3	7
C4	8

6.1.5.6.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

1	**Normal call**
	
	`writeStatusCode(200)`
	`appendHeaderVariable(`
	`  headerKeywordNameId= 1,`
	`  value= [header1header2],`
	`  valueOffset= 0,`
	`  valueLength= 7)`
	
	`finalizeHeader();`
	
	`appendContent([OK],0,2);`
	`flush();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Accept")= "header1"`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

2	**header already finalized 01**
	
	`writeStatusCode(200)`
	`appendHeaderVariable(`
	`  headerKeywordNameId= 1,`
	`  value= [header1header2],`
	`  valueOffset= 0,`
	`  valueLength= 7)`
	
	`finalizeHeader();`
	
	`finalizeHeader();`
	
	`appendContent([OK],0,2);`
	`flush();`
		`Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Accept")= "header1"`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

3	**header already finalized 02**
	
	`writeStatusCode(200)`
	`appendHeaderVariable(`
	`  headerKeywordNameId= 1,`
	`  value= [header1header2],`
	`  valueOffset= 0,`
	`  valueLength= 7)`
	
	`finalizeHeader();`
	
	`appendContent([OK],0,2);`
	
	`finalizeHeader();`
	
	`flush();`
		`Uicc.scws.ScwsException` with reason codes `HEADER_ALREADY_FINALIZED` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Accept")= "header1"`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

4	**flush called previously**
	
	`writeStatusCode(200)`
	`appendHeaderVariable(`
	`  headerKeywordNameId= 1,`
	`  value= [header1header2],`
	`  valueOffset= 0,`
	`  valueLength= 7`
	`)`
	`appendContent([OK],0,2);`
	`flush();`
	
	`finalizeHeader();`
		`Uicc.scws.ScwsException` with reason codes `HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Accept")= "header1"`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

5	**sendError called previously**
	
	`sendError(500)`
	
	`finalizeHeader();`
		`Uicc.scws.ScwsException` with reason codes `HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= "500"`
			`)`

6.1.5.7 Method flush

Test Area Reference: Api_1_Hrs_Flus.

6.1.5.7.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void flush()
```

6.1.5.7.1.1 Normal execution

- CRRN1: Sends all data contained in the HTTP response.
- CRRN2: If this method is invoked in fixed buffer mode after a previous invocation of the same method the SCWS will do nothing.
- CRRN3: If this method is invoked in fixed buffer mode and the sendError() message was invoked previously the SCWS will do nothing.
- CRRN4: Invoking this method for the first time implicitly finalizes the header of the `HttpResponse`.

6.1.5.7.1.2 Parameter Errors

- n/a.

6.1.5.7.1.3 Context errors

- CRRC1: if the adding of missing header data would cause access outside the Response buffer, a uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown.

6.1.5.7.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Flus.java.

Test Applet: Api_1_Hrs_Flus_1.java.

Cap File: Api _1_Hrs_Flus.cap.

6.1.5.7.3 Test coverage

CRR number	Test case number
N1	1, 2, 3, 5
N2	2
N3	6
N4	4
C1	-

6.1.5.7.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

1	**Normal case + Chunk off**
	
	`writeStatusCode(200)`
	`finalizeHeader();`
	`appendContent([OK],0,2);`
	
	`flush();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

2	**Repeat flush + chunk off**
	
	`writeStatusCode(200)`
	`finalizeHeader();`
	`appendContent([OK],0,2);`
	
	`flush();`
	`flush();`
	`flush();`
	`flush();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

3	**Normal case + chunk on**
	
	`writeStatusCode(200)`
	`enableChunkMode();`
	`finalizeHeader();`
	`appendContent([Hello],0,5);`
	
	`appendContent([ ],0,1);`
	
	`appendContent([World],0,5);`
	
	`flush();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Transfer-Encoding")= "chunk"`
			` contentBody= "Hello World"`
			`)`

4	**Finalized the http header**
	
	`writeStatusCode(200);`
	
	`flush();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "0"`
			` contentBody= ""`
			`)`

5	**No flush**
	
	`writeStatusCode(200)`
	`finalizeHeader();`
	`appendContent([OK],0,2);`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

6	**SendError before**
	
	`sendError(500);`
	
	`flush();`
			`HTTPResponse (`
			` statusCode= "500"`
			`)`

6.1.5.8 Method getRemainingResponseBufferSize

Test Area Reference: Api_1_Hrs_Grrs.

6.1.5.8.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
int getRemainingResponseBufferSize()
```

6.1.5.8.1.1 Normal execution

- CRRN1: Returns the remaining size of the Response Buffer available for this applet.
- CRRN2: The size returned by this method is guaranteed until the next header data is added to the Response Buffer.
- CRRN3: The size returned by this method is guaranteed until the content data is added to the Response Buffer.
- CRRN4: Returns -1 if chunked mode is switched on.

6.1.5.8.1.2 Parameter Errors

- n/a.

6.1.5.8.1.3 Context errors

- CRRC1: if the flush method was called previously and the selected transfer mode is "Fixed transfer mode", a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC2: if the sendError method was called previously and the selected transfer mode is "Fixed transfer mode", a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.8.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Grrs.java.

Test Applet: Api_1_Hrs_Grrs_1.java.

Cap File: Api_1_Hrs_Grrs.cap.

6.1.5.8.3 Test coverage

CRR number	Test case number
N1	1, 2, 3
N2	3
N3	1
N4	2
C1	4
C2	5

6.1.5.8.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**Remaining content data** writeStatusCode(200) finalizeHeader() remaining = getRemainingResponseBufferSize(); appendContent(data = [aaa...] offset = 0 length = remaining) flush()		HTTPResponse (statusCode= "**200**" Header("**Content-Length**")= remaining contentBody = "AaA[...]")

2	**Remaining + Chunk on**
	
	writeStatusCode(200)
	enableChunkMode();
	finalizeHeader();
	
	remaining =
	getRemainingResponseBufferSize();
	
	appendContent([OK],0,2);
	flush();
		remaining = -1	HTTPResponse (
			statusCode= "200"
			Header("**Transfer-Encoding**")=
			"**chunked**"
			contentBody= "**OK**"
			)

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

3	**Remaining header data**
	
	`writeStatusCode(200)`
	
	`remainingA = getRemainingResponseBufferSize();`
	
	`appendHeaderVariable(`
	`  headerKeywordNameId= 1,`
	`  value= [header1header2],`
	`  valueOffset= 7,`
	`  valueLength= 7`
	`)`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Accept")= "header2"`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

	`remainingB = getRemainingResponseBufferSize();`
	
	`remainingC = getRemainingResponseBufferSize();`
	
	`finalizeHeader();`
	`appendContent([OK],0,2);`
	`flush();`
		`remainingB < remainingA`
		
		`remainingB = remainingC`	

4	**flush called previously**
	
	`writeStatusCode(200)`
	`finalizeHeader();`
	`appendContent([OK],0,2);`
	`flush();`
	
	`getRemainingResponseBufferSize();`
		`Uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

5	**sendError called previously**
	
	`sendError(500)`
	
	`getRemainingResponseBufferSize();`
		`Uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= "500"`
			`)`

6.1.5.9 Method reset

Test Area Reference: Api_1_Hrs_Rset.

6.1.5.9.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void reset() throws ScwsException
```

6.1.5.9.1.1 Normal execution

- CRRN1: Clears the Response Buffer content.
- CRRN2: Reset the state of the `HttpResponse` Object to its initial state.
- CRRN3: All the header variables and content that was appended to the `HttpResponse` until the invocation of the `reset` method is lost.

6.1.5.9.1.2 Parameter Errors

- n/a.

6.1.5.9.1.3 Context errors

- CRRC1: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC2: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.9.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Rset.java.

Test Applet: Api_1_Hrs_Rset_1.java.

Cap File: Api_1_Hrs_Rset.cap.

6.1.5.9.3 Test coverage

CRR number	Test case number
N1	1, 2, 3
N2	1, 2, 3
N3	1, 2, 3
C1	6
C2	7

6.1.5.9.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

1	**Reset StatusCode**
	`writeStatusCode(204)`
	`reset();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "2"`
			` contentBody = "OK"`
			`)`

	`writeStatusCode(200);`
	`finalizeHeader();`
	`appendContent([OK],0, 2);`
	`flush();`		

2	**Reset Header**
	`writeStatusCode(500);`
	`appendHeaderVariable(8,[L1], 0, 2);`
	`reset();`
	`appendHeaderVariable(8,[L2], 0, 2);`
	`appendContent([OK],0, 2);`
	`flush();`
			`HTTPResponse (`
			` statusCode= 200`
			` Header("Content-Length")= "2"`
			` Header("Content-Language")= "L2"`
			` contentBody = "OK"`
			`)`

3	**Reset ContentType**
	`setContentType(4);`
	`reset();`
	`writeStatusCode(200);`
	`appendContent([OK],0, 2);`
	`flush();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "2"`
			` Header("Content-Type") =`
			`"text/plain" (default value).`
			` contentBody = "OK"`
			`)`

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

4	**Reset Header Finalized**
	
	`writeStatusCode(200);`
	`appendHeaderVariable(8,[L1], 0, 2);`
	`finalizeHeader();`
	
	`reset();`
	
	`appendHeaderVariable(8,[L2], 0, 2);`
	
	`appendContent([OK],0, 2);`
	`flush();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "2"`
			` Header("Content-Language")= "L2"`
			` contentBody = "OK"`
			`)`

5	**Reset Data**
	
	`writeStatusCode(200);`
	`appendHeaderVariable(8,[L1], 0, 2);`
	`finalizeHeader();`
	`appendContent([NOT OK],0, 6);`
	
	`reset();`
	
	`appendContent([OK],0, 2);`
	`flush();`
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "2"`
			` HeaderNotPresent("Content-Language")`
			` contentBody = "OK"`
			`)`

6	**flush() called previously**
	
	`writeStatusCode(200);`
	`finalizeHeader();`
	`appendContent([OK],0, 2);`
	`flush();`
	
	`reset();`
		`uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= "200"`
			` Header("Content-Length")= "2"`
			` contentBody = "OK"`
			`)`

7	**sendError called previously**
	
	`sendError(500);`
	
	`reset();`
		`uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= "500"`
			`)`

6.1.5.10 Method sendError

Test Area Reference: Api_1_Hrs_Sene.

6.1.5.10.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void sendError(short errorCode)
```

6.1.5.10.1.1 Normal execution

- CRRN1: Sends an error code to the SCWS.
- CRRN2: The HTTP response is sent immediately by the SCWS no other data is send by the SCWS.
- CRRN3: Missing header variables are added by the SCWS with their default values.
- CRRN4: Invoking this method implicitly finalizes the header of the `HttpResponse`.

6.1.5.10.1.2 Parameter Errors

- CRRP1: if parameter errorCode is unknown with respect to the HTTP 1.1 protocol, a `uicc.scws.ScwsException with reason codes` HTTP_CODE_UNKNOWN is thrown.

6.1.5.10.1.3 Context errors

- CRRC1: if the HTTP header was already finalized, a uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
- CRRC2: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC3: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.10.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Sene.java.

Test Applet: Api_1_Hrs_Sene_1.java.

Cap File: Api_1_Hrs_Sene.cap.

6.1.5.10.3 Test coverage

CRR number	Test case number
N1	1, 2, 3, 4, 5, 6
N2	1, 2, 3, 4, 5, 6
N3	1, 2, 3, 4, 5, 6
N4	1, 2, 3, 4, 5, 6
P1	7
C1	8, 9
C2	10
C3	11

6.1.5.10.4 Test procedure

Test Case				
Id		**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**Normal use case** : Return an Informational code (1xx)			

	a	// SC_CONTINUE
		sendError(100)
				Not testable

	b	// SC_SWITCHING_PROTOCOLS
		sendError(101)
				Not testable

2	**Normal use case** : Return a Successful code (2xx)			

	a	// SC_OK
		sendError(200)
				HTTPResponse(
				statusCode= 200
				)

	b	// SC_CREATED
		sendError(201)
				HTTPResponse(
				statusCode= 201
				)

	c	// SC_ACCEPTED
		sendError(202)
				HTTPResponse(
				statusCode= 202
				)

	d	// SC_NON_AUTHORITATIVE_INFORMATION sendError(203)		HTTPResponse(statusCode= 203)

	e	// SC_NO_CONTENT
		sendError(204)
				HTTPResponse(
				statusCode= 204
				)

	f	// SC_RESET_CONTENT
		sendError(205)
				HTTPResponse(
				statusCode= 205
				)

		// SC_PARTIAL_CONTENT
		sendError(206)
				HTTPResponse(
				statusCode= 206
				)

3	**Normal use case** : Return a Redirection code (3xx)			

	a	// SC_MULTIPLE_CHOICES
		sendError(300)
				HTTPResponse(
				statusCode= 300
				)

	b	// SC_MOVED_PERMANENTLY
		sendError(301)
				HTTPResponse(
				statusCode= 301
				)

	c	// SC_FOUND
		sendError(302)
				HTTPResponse(
				statusCode= 302
				)

	d	// SC_SEE_OTHER
		sendError(303)
				HTTPResponse(
				statusCode= 303
				)

	e	// SC_NOT_MODIFIED
		sendError(304)
				HTTPResponse(
				statusCode= 304
				)

	f	// SC_USE_PROXY
		sendError(305)
				HTTPResponse(
				statusCode= 305
				)

	g	// SC_TEMPORARY_REDIRECT
		sendError(307)
				HTTPResponse(
				statusCode= 307
				)

4	**Normal use case** : Return a client error code (4xx)			

	a	// SC_BAD_REQUEST
		sendError(400)
				HTTPResponse(
				statusCode= 400
				)

	b	// SC_UNAUTHORIZED
		sendError(401)
				HTTPResponse(
				statusCode= 401
				)

	c	// SC_PAYMENT_REQUIRED
		sendError(402)
				HTTPResponse(
				statusCode= 402
				)

	d	// SC_FORBIDDEN
		sendError(403)
				HTTPResponse(
				statusCode= 403
				)

	e	// SC_NOT_FOUND
		sendError(404)
				HTTPResponse(
				statusCode= 404
				)

Test Case				
Id		**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

	f	`// SC_METHOD_NOT_ALLOWED`
		`sendError(405)`
				`HTTPResponse(`
				` statusCode= 405`
				`)`

	g	`// SC_NOT_ACCEPTABLE`
		`sendError(406)`
				`HTTPResponse(`
				` statusCode= 406`
				`)`

	h	`//`
		`SC_PROXY_AUTHENTICATION_REQUIRED`
		`sendError(407)`
				`HTTPResponse(`
				` statusCode= 407`
				`)`

	i	`// SC_REQUEST_TIMEOUT`
		`sendError(408)`
				`HTTPResponse(`
				` statusCode= 408`
				`)`

	j	`// SC_CONFLICT`
		`sendError(409)`
				`HTTPResponse(`
				` statusCode= 409`
				`)`

	k	`// SC_GONE`
		`sendError(410)`
				`HTTPResponse(`
				` statusCode= 410`
				`)`

	l	`// SC_LENGTH_REQUIRED`
		`sendError(411)`
				`HTTPResponse(`
				` statusCode= 411`
				`)`

	m	`// SC_PRECONDITION_FAILED`
		`sendError(412)`
				`HTTPResponse(`
				` statusCode= 412`
				`)`

	n	`// SC_REQUEST_ENTITY_TOO_LARGE`
		`sendError(413)`
				`HTTPResponse(`
				` statusCode= 413`
				`)`

	o	`// SC_REQUEST_URI_TOO_LONG`
		`sendError(414)`
				`HTTPResponse(`
				` statusCode= 414`
				`)`

	p	`// SC_UNSUPPORTED_MEDIA_TYPE`
		`sendError(415)`
				`HTTPResponse(`
				` statusCode= 415`
				`)`

	q	`//`
		`SC_REQUESTED_RANGE_NOT_SATISFIABLE`
		`sendError(416)`
				`HTTPResponse(`
				` statusCode= 416`
				`)`

	r	`// SC_EXPECTATION_FAILED`
		`sendError(417)`
				`HTTPResponse(`
				` statusCode= 417`
				`)`

5		**Normal use case** : Return a Server Error 5xx code		

	a	`// SC_INTERNAL_SERVER_ERROR`
		`sendError(500)`
				`HTTPResponse(`
				` statusCode= 500`
				`)`

	b	`// SC_NOT_IMPLEMENTED`
		`sendError(501)`
				`HTTPResponse(`
				` statusCode= 501`
				`)`

	c	`// SC_BAD_GATEWAY`
		`sendError(502)`
				`HTTPResponse(`
				` statusCode= 502`
				`)`

	d	`// SC_SERVICE_UNAVAILABLE`
		`sendError(503)`
				`HTTPResponse(`
				` statusCode= 503`
				`)`

	e	`// SC_GATEWAY_TIMEOUT`
		`sendError(504)`
				`HTTPResponse(`
				` statusCode= 504`
				`)`

	f	`// SC_HTTP_VERSION_NOT_SUPPORTED`
		`sendError(505)`
				`HTTPResponse(`
				` statusCode= 505`
				`)`

6		**After writeStatusCode**
		
		`writeStatusCode(200);`
		
		`sendError(403);`
				`HTTPResponse(`
				` statusCode= **403**`
				`)`

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
7	**HTTP_CODE_UNKNOWN** sendError(0) sendError(-1) sendError(1000) sendError(99) sendError(102) sendError(199) sendError(207) sendError(299) sendError(308) sendError(399) sendError(418) sendError(499) sendError(506) sendError(600) sendError(700) sendError(800) sendError(900)	uicc.scws.ScwsException with reason codes HTTP_CODE_UNKNOW is thrown. Catch the exception and build a page with status code = 204	HTTPResponse (statusCode = "204")

8	**finalizeHeader called previously**
	
	writeStatusCode(200)
	finalizeHeader();
	
	sendError(403);
	
	appendContent([OK],0,2);
	flush();
		Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= **200**
			Header("**Content-Length**")= "**2**"
			contentBody= "**OK**"
			)

9	**appendContent called previously**
	
	writeStatusCode(200)
	appendContent([OK],0,2);
	
	sendError(403);
	
	flush();
		Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= **200**
			Header("**Content-Length**")= "**2**"
			contentBody= "**OK**"
			)

10	**flush called previously**
	
	writeStatusCode(200)
	appendContent([OK],0,2);
	flush();
	
	sendError(403);
		uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= **200**
			Header("**Content-Length**")= "**2**"
			contentBody= "**OK**"
			)

11	**sendError called previously**
	
	sendError(500);
	
	sendError(403);
		uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= **500**
			)

6.1.5.11 Method setContentType

Test Area Reference: Api_1_Hrs_Scot.

6.1.5.11.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void setContentType(short type) throws ScwsException
```

6.1.5.11.1.1 Normal execution

- CRRN1: Sets the content type of the HTTP response.

6.1.5.11.1.2 Parameter Errors

- CRRP1: if parameter type is not correct according to `SCWSConstants`, a `uicc.scws.ScwsException` with reason codes `UNKNOWN_KEYWORD_ID` is thrown.

6.1.5.11.1.3 Context errors

- CRRC1: if the HTTP header was already finalized, a uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
- CRRC2: if appending the header variable would cause an overflow of the response buffer, a uicc.scws.ScwsException with reason codes BUFFER_OVERFLOW is thrown.
- CRRC3: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC4: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.11.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Scot.java.

Test Applet: Api_1_Hrs_Scot_1.java.

Cap File: Api_1_Hrs_Scot.cap.

6.1.5.11.3 Test coverage

CRR number	Test case number
N1	1, 2, 3, 4, 5
P1	6
C1	7, 8
C2	-
C3	9
C4	10

6.1.5.11.4 Test procedure

Test Case				
Id		**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1		**All success type**		

	a	`// CONTENT_TYPE_IMAGE_GIF`
		`writeStatusCode(200)`
		`setContentType(4);`
		`finalizeHeader();`
		`appendContent([OK],0,2);`
		`flush();`
				`HTTPResponse (`
				` statusCode= "200"`
				` Header("Content-Type")= "image/gif"`
				` Header("Content-Length")= "2"`
				` contentBody= "OK"`
				`)`

	b	`// CONTENT_TYPE_IMAGE_JPEG`
		`writeStatusCode(200)`
		`setContentType(3);`
		`finalizeHeader();`
		`appendContent([OK],0,2);`
		`flush();`
				`HTTPResponse (`
				` statusCode= 200`
				` Header("Content-Type")= "image/jpeg"`
				` Header("Content-Length")= "2"`
				` contentBody= "OK"`
				`)`

	c	`// CONTENT_TYPE_IMAGE_PNG`
		`writeStatusCode(200)`
		`setContentType(5);`
		`finalizeHeader();`
		`appendContent([OK],0,2);`
		`flush();`
				`HTTPResponse (`
				` statusCode= 200`
				` Header("Content-Type")= "image/png"`
				` Header("Content-Length")= "2"`
				` contentBody= "OK"`
				`)`

	d	`// CONTENT_TYPE_TEXT_HTML`
		`writeStatusCode(200)`
		`setContentType(1);`
		`finalizeHeader();`
		`appendContent([OK],0,2);`
		`flush();`
				`HTTPResponse (`
				` statusCode= 200`
				` Header("Content-Type")= "text/html"`
				` Header("Content-Length")= "2"`
				` contentBody= "OK"`
				`)`

	e	`// CONTENT_TYPE_TEXT_PLAIN`
		`writeStatusCode(200)`
		`setContentType(2);`
		`finalizeHeader();`
		`appendContent([OK],0,2);`
		`flush();`
				`HTTPResponse (`
				` statusCode= 200`
				` Header("Content-Type")= "text/plain"`
				` Header("Content-Length")= "2"`
				` contentBody= "OK"`
				`)`

2		**Call in first**
		`setContentType(2);`
		`writeStatusCode(200)`
		`finalizeHeader();`
		`appendContent([OK],0,2);`
		`flush();`
				`HTTPResponse (`
				` statusCode= 200`
				` Header("Content-Type")= "text/plain"`
				` Header("Content-Length")= "2"`
				` contentBody= "OK"`
				`)`

3		**Call after status line**
		`writeStatusCode(200)`
		`setContentType(2);`
		`finalizeHeader();`
		`appendContent([OK],0,2);`
		`flush();`
				`HTTPResponse (`
				` statusCode= 200`
				` Header("Content-Type")= "text/plain"`
				` Header("Content-Length")= "2"`
				` contentBody= "OK"`
				`)`

4		**Call after appendHeaderVariable**
		`writeStatusCode(200)`
		`appendHeaderVariable(`
		`  headerKeywordNameId= 1,`
		`  value= [header1header2],`
		`  valueOffset= 0,`
		`  valueLength= 7)`
		`setContentType(2);`
		`finalizeHeader();`
		`appendContent([OK],0,2);`
		`flush();`
				`HTTPResponse (`
				` statusCode= 200`
				` Header("Accept")= "header1"`
				` Header("Content-Type")= "text/plain"`
				` Header("Content-Length")= "2"`
				` contentBody= "OK"`
				`)`

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

5	**Multiple call**
	
	writeStatusCode(200)
	
	setContentType(5);
	setContentType(4);
	setContentType(3);
	setContentType(2);
	setContentType(1);
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
			HTTPResponse (
			statusCode= **200**
			Header("**Content-Type**")= "**text/html**"
			Header("**Content-Length**")= "**2**"
			contentBody= "**OK**"
			)

6	**UNKNOWN_KEYWORD_ID**
	
	writeStatusCode(200)
	
	setContentType(0);
	
	setContentType(-1);
	
	setContentType(6);
	
	setContentType(-2);
	
	setContentType(2);
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
		uicc.scws.ScwsException with reason codes UNKNOWN_KEYWORD_ID is thrown.
		uicc.scws.ScwsException with reason codes UNKNOWN_KEYWORD_ID is thrown.
		uicc.scws.ScwsException with reason codes UNKNOWN_KEYWORD_ID is thrown.
		uicc.scws.ScwsException with reason codes UNKNOWN_KEYWORD_ID is thrown.
		
		Catch all exceptions and continue the procedure.
			HTTPResponse (
			statusCode= 200
			Header("**Content-Type**")= "**text/plain**"
			Header("**Content-Length**")= "**2**"
			contentBody= "OK"
			)

7	**finalizeHeader called previously**
	
	writeStatusCode(200)
	finalizeHeader();
	
	setContentType(2);
	
	appendContent([OK],0,2);
	flush();
		Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= **200**
			Header("**Content-Length**")= "**2**"
			contentBody= "**OK**"
			)

8	**appendContent called previously**
	
	writeStatusCode(200)
	finalizeHeader();
	appendContent([OK],0,2);
	
	setContentType(2);
	
	flush();
		Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse (
			statusCode= **200**
			Header("**Content-Length**")= "**2**"
			contentBody= "**OK**"
			)

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

9	**flush called previously**
	
	`writeStatusCode(200)`
	`setContentType(2);`
	`appendContent([OK],0,2);`
	`flush();`
	
	`setContentType(1);`
		`uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= 200`
			` Header("Content-Type")= "text/plain`
			` Header("Content-Length")= "2"`
			` contentBody= "OK"`
			`)`

10	**sendError called previously**
	
	`sendError(500)`
	
	`setContentType(2);`
		`uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse (`
			` statusCode= 500`
			`)`

6.1.5.12 Method writeStatusCode

Test Area Reference: Api_1_Hrs_Wstc.

6.1.5.12.1 Conformance Requirement

The method with following header shall be compliant to its definition in the API.

```
void writeStatusCode(short code)
                     throws ScwsException
```

6.1.5.12.1.1 Normal execution

- CRRN1: Write an HTTP status code into the outgoing buffer.

6.1.5.12.1.2 Parameter Errors

- CRRP1: if parameter `code` is unknown with respect to the HTTP 1.1 protocol, a `uicc.scws.ScwsException with reason codes` HTTP_CODE_UNKNOWN is thrown.

6.1.5.12.1.3 Context errors

- CRRC1: if the HTTP header was already finalized, a uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
- CRRC2: if the writeStatusCode has already been invoked, a uicc.scws.ScwsException with reason codes STATUS_LINE_ALREADY_SET is thrown.
- CRRC3: if the appendHeaderVariable has already been invoked, a uicc.scws.ScwsException with reason codes STATUS_LINE_ALREADY_SET is thrown.
- CRRC4: if the flush method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.
- CRRC5: if the sendError method was called previously, a uicc.scws.ScwsException with reason codes HTTP_RESPONSE_ALREADY_SENT is thrown.

6.1.5.12.2 Test area files

Specific triggering: None:

Test Source: Test_Api_1_Hrs_Wstc.java.

Test Applet: Api_1_Hrs_Wstc_1.java.

Cap File: Api_1_Hrs_Wstc.cap.

6.1.5.12.3 Test coverage

CRR number	Test case number
N1	1, 2, 3, 4, 5, 12
P1	6
C1	7
C2	8
C3	9
C4	10
C5	11

6.1.5.12.4 Test procedure

Test Case				
Id		**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	**Normal use case** : Return a basic page with an Informational status code (1xx)			

	a	// SC_CONTINUE
		writeStatusCode(100)
				Not testable

	b	// SC_SWITCHING_PROTOCOLS
		writeStatusCode(101)
				Not testable

2	**Normal use case** : Return a basic page with a Successful status code (2xx)			

	a	// SC_OK
		writeStatusCode(200)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 200
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	b	// SC_CREATED
		writeStatusCode(201)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 201
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	c	// SC_ACCEPTED
		writeStatusCode(202)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 202
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	d	// SC_NON_AUTHORITATIVE_INFORMATION
		writeStatusCode(203)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 203
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	e	// SC_NO_CONTENT
		writeStatusCode(204)
				HTTPResponse(
				statusCode= 204
				Header("Content-Length")=
				null // not present in the
				response.
				)

	f	// SC_RESET_CONTENT
		writeStatusCode(205)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 205
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	g	// SC_PARTIAL_CONTENT
		writeStatusCode(206)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 206
				Header("Content-Length")= 2
				contentBody= "OK"
				)

Test Case				
Id		**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
3	**Normal use case** : Return a basic page with a Redirection status code (3xx)			

	a	// SC_MULTIPLE_CHOICES
		writeStatusCode(300)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 300
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	b	// SC_MOVED_PERMANENTLY
		writeStatusCode(301)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 301
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	c	// SC_FOUND
		writeStatusCode(302)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 302
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	d	// SC_SEE_OTHER
		writeStatusCode(303)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 303
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	e	// SC_NOT_MODIFIED
		writeStatusCode(304)
				HTTPResponse(
				statusCode= 304
				)

	f	// SC_USE_PROXY
		writeStatusCode(305)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 305
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	g	// SC_TEMPORARY_REDIRECT
		writeStatusCode(307)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 307
				Header("Content-Length")= 2
				contentBody= "OK"
				)

4	**Normal use case** : Return a basic page with a Client error status code (4xx)			

	a	// SC_BAD_REQUEST
		writeStatusCode(400)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 400
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	b	// SC_UNAUTHORIZED
		writeStatusCode(401)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 401
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	c	// SC_PAYMENT_REQUIRED
		writeStatusCode(402)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 402
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	d	// SC_FORBIDDEN
		writeStatusCode(403)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 403
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	e	// SC_NOT_FOUND
		writeStatusCode(404)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 404
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	f	// SC_METHOD_NOT_ALLOWED
		writeStatusCode(405)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 405
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	g	// SC_NOT_ACCEPTABLE
		writeStatusCode(406)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 406
				Header("Content-Length")= 2
				contentBody= "OK"
				)

Test Case				
Id		**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

	h	// SC_PROXY_AUTHENTICATION_REQUIRED
		writeStatusCode(407)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 407
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	i	// SC_REQUEST_TIMEOUT
		writeStatusCode(408)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 408
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	j	// SC_CONFLICT
		writeStatusCode(409)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 409
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	k	// SC_GONE
		writeStatusCode(410)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 410
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	l	// SC_LENGTH_REQUIRED
		writeStatusCode(411)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 411
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	m	// SC_PRECONDITION_FAILED
		writeStatusCode(412)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 412
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	n	// SC_REQUEST_ENTITY_TOO_LARGE
		writeStatusCode(413)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 413
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	o	// SC_REQUEST_URI_TOO_LONG
		writeStatusCode(414)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 414
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	p	// SC_UNSUPPORTED_MEDIA_TYPE
		writeStatusCode(415)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 415
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	q	// SC_REQUESTED_RANGE_NOT_SATISFIABLE
		writeStatusCode(416)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 416
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	r	// SC_EXPECTATION_FAILED
		writeStatusCode(417)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 417
				Header("Content-Length")= 2
				contentBody= "OK"
				)

5	**Normal use case** : Return a basic page with a Server error status code (5xx)			

	a	// SC_INTERNAL_SERVER_ERROR
		writeStatusCode(500)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 500
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	b	// SC_NOT_IMPLEMENTED
		writeStatusCode(501)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 501
				Header("Content-Length")= 2
				contentBody= "OK"
				)

	c	// SC_BAD_GATEWAY
		writeStatusCode(502)
		appendContent([OK],0,2);
		flush();
				HTTPResponse(
				statusCode= 502
				Header("Content-Length")= 2
				contentBody= "OK"
				)

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

d	// SC_SERVICE_UNAVAILABLE
	writeStatusCode(503)
	appendContent([OK],0,2);
	flush();
			HTTPResponse(
			statusCode= 503
			Header("Content-Length")= 2
			contentBody= "OK"
			)

e	// SC_GATEWAY_TIMEOUT
	writeStatusCode(504)
	appendContent([OK],0,2);
	flush();
			HTTPResponse(
			statusCode= 504
			Header("Content-Length")= 2
			contentBody= "OK"
			)

f	// SC_HTTP_VERSION_NOT_SUPPORTED
	writeStatusCode(505)
	appendContent([OK],0,2);
	flush();
			HTTPResponse(
			statusCode= 505
			Header("Content-Length")= 2
			contentBody= "OK"
			)

6	**HTTP_CODE_UNKNOWN** writeStatusCode(0) writeStatusCode(-1) writeStatusCode(1000) writeStatusCode(99) writeStatusCode(102) writeStatusCode(199) writeStatusCode(207) writeStatusCode(299) writeStatusCode(308) writeStatusCode(399) writeStatusCode(418) writeStatusCode(499) writeStatusCode(506) writeStatusCode(600) writeStatusCode(700) writeStatusCode(800) writeStatusCode(900)	uicc.scws.ScwsException with reason codes HTTP_CODE_UIINKNOW is thrown. Catch ALL exceptions and build a page with status code = 204	HTTPResponse (statusCode = "204")

7	**header already finalized**
	
	writeStatusCode(200)
	finalizeHeader();
	
	writeStatusCode(500)
	
	appendContent([OK],0,2);
	flush();
		Uicc.scws.ScwsException with reason codes HEADER_ALREADY_FINALIZED is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse(
			statusCode= 200
			Header("Content-Length")= 2
			contentBody= "OK"
			)

8	**writeStatusCode already invoked**
	
	writeStatusCode(200)
	
	writeStatusCode(500)
	
	finalizeHeader();
	appendContent([OK],0,2);
	flush();
		Uicc.scws.ScwsException with reason codes STATUS_LINE_ALREADY_SET is thrown.
		Catch the exception and continue the procedure.
			HTTPResponse(
			statusCode= 200
			Header("Content-Length")= 2
			contentBody= "OK"
			)

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

9	**appendHeaderVariable already invoked**
	`writeStatusCode(200)`
	`appendHeaderVariable(`
	`  headerKeywordNameId= 10,`
	`  value= [test],`
	`  valueOffset= 0,`
	`  valueLength= 4`
	`)`
	`writeStatusCode(500)`
	`finalizeHeader();`
	`appendContent([OK],0,2);`
	`flush();`
		`Uicc.scws.ScwsException` with reason codes `STATUS_LINE_ALREADY_SET` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse(`
			` statusCode= 200`
			` Content-Length: 2`
			` Content-Type: test`
			` contentBody= "OK"`
			`)`

10	**flush called previously**
	`writeStatusCode(200)`
	`finalizeHeader();`
	`appendContent([OK],0,2);`
	`flush();`
	`writeStatusCode(500)`
		`Uicc.scws.ScwsException` with reason codes `HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse(`
			` statusCode= 200`
			` Content-Length: 2`
			` contentBody= "OK"`
			`)`

11	**sendError called previously**
	`sendError(500)`
	`writeStatusCode(200)`
		`Uicc.scws.ScwsException` with reason codes `HTTP_RESPONSE_ALREADY_SENT` is thrown.
		Catch the exception and continue the procedure.
			`HTTPResponse(`
			` statusCode= 500`
			`)`

12	**writeStatusCode not called**
	`appendContent([OK],0,2);`
	`flush();`
		Default status code is 200 .	`HTTPResponse(`
			` statusCode= 200`
			` Content-Length: 2`
			` contentBody= "OK"`
			`)`

6.2 SCWS Runtime Environment

6.2.1 Applet state

6.2.1.1 Invokation of applets not in state selectable

Test Area Reference: Api_2_Aps_Ivns.

6.2.1.1.1 Conformance Requirement

The SCWS Framework implementation shall be compliant to the following statement (TS 102 588 [2], clause 4.3):

- Only an Applet that is in selectable state can be invoked by the SCWS.

6.2.1.1.1.1 Normal execution

- CRRN1: Only an applet in selectable state can be invoked by the SCWS.
- CRRN2: If the applet is in non selectable state, then the SCWS does not invoke this one and returns an HTTP response with a error status code = SC_NOT_FOUND whatever the HTTP method.

6.2.1.1.1.2 Parameter Errors

- n/a.

6.2.1.1.1.3 Context errors

- n/a.

6.2.1.1.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_2_Aps_Ivns.java.

Test Applet: Api_2_Aps_Ivns_1.java.

Cap File: Api_2_Aps_Ivns.cap.

6.2.1.1.3 Test coverage

CRR number	Test case number
N1, N2	1
N1, N2	2
N1, N2	3
N1, N2	4
N1, N2	5
N1, N2	6
N1, N2	7

6.2.1.1.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

1	**Invoke an applet LOCKED / UNLOCKED with HTTP methods: POST**
	
	1. Send POST HTTP REQUEST (1) to *Api_2_Aps_Ivns_1*
	
	2. Lock the applet *Api_2_Aps_Ivns_1* (non selectable state)
	
	3. Send POST HTTP REQUEST (2) to *Api_2_Aps_Ivns_1*
	
	4. Unlock the applet *Api_2_Aps_Ivns_1* (selectable state)
	
	5. Send POST HTTP REQUEST (3) to *Api_2_Aps_Ivns_1*
	
	Applet source code in doPost()
	
	`response.writeStatusCode(ScwsConstants.SC_NO_CONTENT);`
			Check the HTTP
			
			RESPONSE (1):
			`HTTPResponse ( statusCode = "204" )`
			
			RESPONSE (2):
			`HTTPResponse ( statusCode = "404" )`
			
			RESPONSE (3):
			`HTTPResponse ( statusCode = "204" )`

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

2	**Invoke an applet LOCKED / UNLOCKED with HTTP methods: GET**
	
	1. Send GET HTTP REQUEST (1) to *Api_2_Aps_Ivns_1*
	2. Lock the applet *Api_2_Aps_Ivns_1* (non selectable state)
	3. Send GET HTTP REQUEST (2) to *Api_2_Aps_Ivns_1*
	4. Unlock the applet *Api_2_Aps_Ivns_1* (selectable state)
	5. Send GET HTTP REQUEST (3) to *Api_2_Aps_Ivns_1*
	
	Applet source code in doGet()
	`response.writeStatusCode(ScwsConstants.SC_NO_CONTENT);`
			Check the HTTP
			
			RESPONSE (1):
			`HTTPResponse ( statusCode = "204" )`
			
			RESPONSE (2):
			`HTTPResponse ( statusCode = "404" )`
			
			RESPONSE (3):
			`HTTPResponse ( statusCode = "204" )`

3	**Invoke an applet LOCKED / UNLOCKED with HTTP methods: DELETE**
	
	1. Send DELETE HTTP REQUEST (1) to *Api_2_Aps_Ivns_1*
	2. Lock the applet *Api_2_Aps_Ivns_1* (non selectable state)
	3. Send DELETE HTTP REQUEST (2) to *Api_2_Aps_Ivns_1*
	4. Unlock the applet *Api_2_Aps_Ivns_1* (selectable state)
	5. Send DELETE HTTP REQUEST (3) to *Api_2_Aps_Ivns_1*
	
	Applet source code in doDelete()
	`response.writeStatusCode(ScwsConstants.SC_NO_CONTENT);`
			Check the HTTP
			
			RESPONSE (1):
			`HTTPResponse ( statusCode = "204" )`
			
			RESPONSE (2):
			`HTTPResponse ( statusCode = "404" )`
			
			RESPONSE (3):
			`HTTPResponse ( statusCode = "204" )`

4	**Invoke an applet LOCKED / UNLOCKED with HTTP methods : HEAD**
	
	1. Send HEAD HTTP REQUEST (1) to *Api_2_Aps_Ivns_1*
	2. Lock the applet *Api_2_Aps_Ivns_1* (non selectable state)
	3. Send HEAD HTTP REQUEST (2) to *Api_2_Aps_Ivns_1*
	4. Unlock the applet *Api_2_Aps_Ivns_1* (selectable state)
	5. Send HEAD HTTP REQUEST (3) to *Api_2_Aps_Ivns_1*
	
	Applet source code in doHead()
	`response.writeStatusCode(ScwsConstants.SC_NO_CONTENT);`
			Check the HTTP
			
			RESPONSE (1):
			`HTTPResponse ( statusCode = "204" )`
			
			RESPONSE (2):
			`HTTPResponse ( statusCode = "404" )`
			
			RESPONSE (3):
			`HTTPResponse ( statusCode = "204" )`

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

5	**Invoke an applet LOCKED / UNLOCKED with HTTP methods : PUT**
	
	1. Send PUT HTTP REQUEST (1) to *Api_2_Aps_Ivns_1*
	
	2. Lock the applet *Api_2_Aps_Ivns_1* (non selectable state)
	
	3. Send PUT HTTP REQUEST (2) to *Api_2_Aps_Ivns_1*
	
	4. Unlock the applet *Api_2_Aps_Ivns_1* (selectable state)
	
	5. Send PUT HTTP REQUEST (3) to *Api_2_Aps_Ivns_1*
	
	Applet source code in doPut()
	
	`response.writeStatusCode(ScwsConstants.SC_NO_CONTENT);`
			Check the HTTP
			
			RESPONSE (1):
			`HTTPResponse ( statusCode = "204" )`
			
			RESPONSE (2):
			`HTTPResponse ( statusCode = "404" )`
			
			RESPONSE (3):
			`HTTPResponse ( statusCode = "204" )`

6	**Invoke an applet LOCKED / UNLOCKED with HTTP methods : OPTIONS**
	
	1. Send OPTIONS HTTP REQUEST (1) to *Api_2_Aps_Ivns_1*
	
	2. Lock the applet *Api_2_Aps_Ivns_1* (non selectable state)
	
	3. Send OPTIONS HTTP REQUEST (2) to *Api_2_Aps_Ivns_1*
	
	4. Unlock the applet *Api_2_Aps_Ivns_1* (selectable state)
	
	5. Send OPTIONS HTTP REQUEST (3) to *Api_2_Aps_Ivns_1*
	
	Applet source code in doOptions()
	
	`response.writeStatusCode(ScwsConstants.SC_NO_CONTENT);`
			Check the HTTP
			
			RESPONSE (1):
			`HTTPResponse ( statusCode = "204" )`
			
			RESPONSE (2):
			`HTTPResponse ( statusCode = "404" )`
			
			RESPONSE (3):
			`HTTPResponse ( statusCode = "204" )`

7	**Invoke an applet LOCKED / UNLOCKED with HTTP methods : TRACE**
	
	1. Send TRACE HTTP REQUEST (1) to *Api_2_Aps_Ivns_1*
	
	2. Lock the applet *Api_2_Aps_Ivns_1* (non selectable state)
	
	3. Send TRACE HTTP REQUEST (2) to *Api_2_Aps_Ivns_1*
	
	4. Unlock the applet *Api_2_Aps_Ivns_1* (selectable state)
	
	5. Send TRACE HTTP REQUEST (3) to *Api_2_Aps_Ivns_1*
	
	Applet source code in doTrace()
	
	`response.writeStatusCode(ScwsConstants.SC_NO_CONTENT);`
			Check the HTTP
			
			RESPONSE (1):
			`HTTPResponse ( statusCode = "204" )`
			
			RESPONSE (2):
			`HTTPResponse ( statusCode = "404" )`
			
			RESPONSE (3):
			`HTTPResponse ( statusCode = "204" )`

6.2.1.2 Registration remains valid if applet is not in selectable state

Test Area Reference: Api_2_Aps_Regv.

6.2.1.2.1 Conformance Requirement

The SCWS Framework implementation shall be compliant to the following statement (TS 102 588, clause 4.2):

- If the Applet is in a non selectable state, its registration to the SCWS is still valid.

6.2.1.2.1.1 Normal execution

- CRRN1: If the Applet is in a non selectable state, its registrations to the SCWS is still valid.

6.2.1.2.1.2 Parameter Errors

- n/a.

6.2.1.2.1.3 Context errors

- n/a.

6.2.1.2.2 Test area files

Specific triggering: Unrecognized envelope

Test Source: Test_Api_2_Aps_Regv.java.

Test Applet: Api_2_Aps_Regv_1.java.

Cap File: Api_2_Aps_Regv.cap.

6.2.1.2.3 Test coverage

CRR number	Test case number
N1	1

6.2.1.2.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
Map an applet in non selectable state 1. Register *Api_2_Aps_Regv_1* to SCWS 2. Lock the applet *Api_2_Aps_Regv_1* (non selectable state) 3. Map the *Api_2_Aps_Regv_1* (1) 4. Unlock the applet *Api_2_Aps_Regv_1* (selectable state) 5. Send a HTTP REQUEST (2) to *Api_2_Aps_Regv_1* Applet source code in doPost() `response.writeStatusCode(ScwsConstants.SC_NO_CONTENT);`		*Api_2_Aps_Regv_1* mapping success (1) Check the HTTP RESPONSE (2): `HTTPResponse ( statusCode = "204" )`	

6.2.2 Response sending

All tests regarding sending of a response are implicitely covered by the test cases in clause 6.1.5.

6.2.3 Exception handling

6.2.3.1 No exception shall be propagated as HTTP error to the terminal

Test Area Reference: Api_2_Exh_Noex.

6.2.3.1.1 Conformance Requirement

The SCWS Framework implementation shall be compliant to the following statement (TS 102 588 [2], clause 4.4):

- Exceptions thrown by the invoked Applet shall not be propagated to the terminal, and the SCWS shall send an error status code according to HTTP 1.1[1]. In case of "chunked mode", some response data could have already been sent by the SCWS. In this case, no error shall be sent by the SCWS. In case of "fixed buffer size mode", the status code is always sent by the SCWS.

6.2.3.1.1.1 Normal execution

- CRRN1: If an exceptions thrown by the invoked Applet the SCWS send an error status code according to HTTP 1.1[1].
- CRRN2: In case of "chunked mode", if an exceptions thrown by the invoked Applet the SCWS and if some response data could have already been sent by the SCWS. Then, no error shall be sent by the SCWS.
- CRRN3: In case of "fixed buffer size mode", if an exceptions thrown by the invoked Applet the SCWS. Then, the status code is always sent by the SCWS.

6.2.3.1.1.2 Parameter Errors

- n/a.

6.2.3.1.1.3 Context errors

- n/a.

6.2.3.1.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_2_Exh_Noex.java.

Test Applet: Api_2_Exh_Noex_1.java.

Cap File: Api_2_Exh_Noex.cap.

6.2.3.1.3 Test coverage

CRR number	Test case number
N1	1
N2	2
N3	3

6.2.3.1.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

1	Applet throws an exception
	
	`Byte[] array = new byte[2]`
	`Array[3] = 0x00;`
		`java.lang. ArrayIndexOutOfBoundsException is thrown`	`HTTPResponse (`
			` statusCode = "500"`
			`)`

2	In "**chunked mode**", applet throws an Exception AFTER `writeStatusCode ()`
	
	`enableChunkMode()`
	`writeStatusCode(200)`
	
	`Byte[] array = new byte[2]`
	`Array[3] = 0x00;`
		`java.lang. ArrayIndexOutOfBoundsException is thrown`	`HTTPResponse (`
			` statusCode = "500"`
			`)`

3	In "**chunked mode**", applet throws an Exception AFTER `appendHeaderVariable ()`
	
	`enableChunkMode()`
	`writeStatusCode(200)`
	`appendHeaderVariable("a: B", 0, 4)`
	
	`Byte[] array = new byte[2]`
	`Array[3] = 0x00;`
		`java.lang. ArrayIndexOutOfBoundsException is thrown`	`HTTPResponse (`
			` statusCode = "500"`
			`)`

4	In "**chunked mode**", applet throws an Exception AFTER `appendContent()`
	
	`enableChunkMode()`
	`writeStatusCode(200)`
	`appendHeaderVariable("A: B", 0, 4)`
	`appendContent("OK", 0, 2)`
	
	`Byte[] array = new byte[2]`
	`Array[3] = 0x00;`
		`java.lang. ArrayIndexOutOfBoundsException is thrown`	`HTTPResponse (`
			` statusCode = "200"`
			` Header = "A: B`
			` Content = "OK"`
			`)`

5	In "**fixed buffer size mode**", applet throws an Exception AFTER `writeStatusCode ()`
	
	`writeStatusCode(200)`
	
	`Byte[] array = new byte[2]`
	`Array[3] = 0x00;`
		`java.lang. ArrayIndexOutOfBoundsException is thrown`	`HTTPResponse (`
			` statusCode = "500"`
			`)`

6	In "**fixed buffer size mode**", applet throws an Exception AFTER `appendHeaderVariable_ ()`
	
	`writeStatusCode(200)`
	`appendHeaderVariable("a: B", 0, 4)`
	
	`Byte[] array = new byte[2]`
	`Array[3] = 0x00;`
		`java.lang. ArrayIndexOutOfBoundsException is thrown`	`HTTPResponse (`
			` statusCode = "500"`
			`)`

7	In "**fixed buffer size mode**", applet throws an Exception AFTER `appendContent()`
	
	`writeStatusCode(200)`
	`appendHeaderVariable("A: B", 0, 4)`
	`appendContent("OK", 0, 2)`
	
	`Byte[] array = new byte[2]`
	`Array[3] = 0x00;`
		`java.lang. ArrayIndexOutOfBoundsException is thrown`	`HTTPResponse (`
			` statusCode = "500"`
			`)`

6.2.4 Response Header Management

6.2.4.1 Send status code indicating success

Test Area Reference: Api_2_Rhm_Scis.

6.2.4.1.1 Conformance Requirement

The SCWS Framework implementation shall be compliant to the following statement (TS 102 588 [2], clause 4.5):

- The following headers shall be added by the SCWS if not provided by the application:
  - Status line, indicating success status (200 – OK or 204 – No Content)

6.2.4.1.1.1 Normal execution

- CRRN1 : If an Applet returns data in the response body, the SCWS shall append status code 200 (OK) if not set by the Applet.
- CRRN2 : If an Applet returns no data in the response body, the SCWS shall append status code 204 (No Content) if not set by the Applet.

6.2.4.1.1.2 Parameter Errors

- n/a.

6.2.4.1.1.3 Context errors

- n/a.

6.2.4.1.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_2_Rhm_Scis.java.

Test Applet: Api_2_Rhm_Scis_1.java.

Cap File: Api_2_Rhm_Scis.cap.

6.2.4.1.3 Test coverage

CRR number	Test case number
N1	1
N2	2

6.2.4.1.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**

1	**Invoke an applet that returns data**
	finalizeHeader();
	appendContent([OK], 0, 2);
	flush();
			HTTPResponse(
			statusCode= 200
			Content-Length: 2
			contentBody= "OK"
			)

2	**Invoke an applet that returns no data**
	finalizeHeader();
	flush();
			HTTPResponse(
			statusCode= 204
			)

6.2.5 Availability of ProactiveHandler / ProactiveResponseHandler

6.2.5.1 Incoming Http request

Test Area Reference: Api_2_Pro_Inhr.

6.2.5.1.1 Conformance Requirement

The SCWS Framework implementation shall be compliant to the following statement (TS 102 588 [2], clause 4.6).

The ProactiveHandler shall be available for applets that are invoked by an incoming Http request (i.e. by one of the methods *doGet()*, *doPost()*, *doDelete()*, *doHead()*, *doOptions()*, *doPut()*, and *doTrace()*) in the same way as if it would be available for an applet which was triggered by the method processToolkit() in the current card state.

6.2.5.1.1.1 Normal execution

- CRRN1: ProactiveHandler is available for applets that are invoked by an incoming Http request.

6.2.5.1.1.2 Parameter Errors

- n/a.

6.2.5.1.1.3 Context errors

- n/a.

6.2.5.1.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source:	Test_Api_2_Pro_Inhr.java.
Test Applet:	Api_2_Pro_Inhr_1.java.
Cap File:	Api_2_Pro_Inhr.cap.

6.2.5.1.3 Test coverage

CRR number	Test case number
N1	1 – 7

6.2.5.1.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	Applet is triggered by doGet() Get reference to the ProactiveHandler.	Invocation of ProactiveHandler.getCapacity() returns a value greater than 0.	HTTPResponse (statusCode = "200")
2	Applet is triggered by doPost() Get reference to the ProactiveHandler.	Invocation of ProactiveHandler.getCapacity() returns a value greater than 0.	HTTPResponse (statusCode = "200")
3	Applet is triggered by doDelete() Get reference to the ProactiveHandler.	Invocation of ProactiveHandler.getCapacity() returns a value greater than 0.	HTTPResponse (statusCode = "200")
4	Applet is triggered by doHead() Get reference to the ProactiveHandler.	Invocation of ProactiveHandler.getCapacity() returns a value greater than 0.	HTTPResponse (statusCode = "200")
5	Applet is triggered by doOptions() Get reference to the ProactiveHandler.	Invocation of ProactiveHandler.getCapacity() returns a value greater than 0.	HTTPResponse (statusCode = "200")
6	Applet is triggered by doPut() Get reference to the ProactiveHandler.	Invocation of ProactiveHandler.getCapacity() returns a value greater than 0.	HTTPResponse (statusCode = "200")
7	Applet is triggered by doTrace() Get reference to the ProactiveHandler.	Invocation of ProactiveHandler.getCapacity() returns a value greater than 0.	HTTPResponse (statusCode = "200")

6.2.5.2 Available for complete Http response

Test Area Reference: Api_2_Pro_Avco.

6.2.5.2.1 Conformance Requirement

The SCWS Framework implementation shall be compliant to the following statement (TS 102 588 [2], clause 4.6).

If available, the ProactiveHandler shall be available for these applets until they have sent the complete Http response to the SCWS.

6.2.5.2.1.1 Normal execution

- CRRN1: In chunked mode, retrieve a reference to the ProactiveHandler after sending some chunks of data to the SCWS and issue a Provide Local Information (IMEI) command. Then send HttpResponse.flush(). The whole data sent in chunks to the SCWS shall be included in the SCWS response page.
- CRRN2: In fixed buffer mode fill the buffer until the second last byte, issue Provide Local Information (IMEI) command. Then append one more byte the to HTTPResponse object. The SCWS shall send the whole response page.
- CRRN3: In fixed buffer mode invoke HttpResponse.flush(). Afterwards get a reference to the ProactiveHandler and issue a proactive command.

6.2.5.2.1.2 Parameter Errors

- n/a.

6.2.5.2.1.3 Context errors

- n/a.

6.2.5.2.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_2_Pro_Avco.java.

Test Applet: Api_2_Pro_Avco_1.java.

Cap File: Api_2_Pro_Avco.cap.

6.2.5.2.3 Test coverage

CRR number	Test case number
N1	1
N2	2
N3	3

6.2.5.2.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	Get reference to the ProactiveHandler in chunked mode.	Data appended to the HttpResponsebuffer before invocation of ProactiveHandler is sent in the http response.	Prov.Loc.Info(IMEI) HTTPResponse (Content complete statusCode = "200")
2	Get reference to the ProactiveHandler in fixed buffer mode.	Data appended to the HttpResponsebuffer before invocation of ProactiveHandler is sent in the http response.	Prov.Loc.Info(IMEI) HTTPResponse (Content complete statusCode = "200")
3	Issue a proactive command after invoking flush()	Append Data to the HttpResponse Buffer, invoke flush(), then get a reference to the ProactiveHandler and invoke a proactive command.	HTTPResponse (Content complete statusCode = "200") Prov.Loc.Info(IMEI) or Prov.Loc.Info(IMEI) HTTPResponse (Content complete statusCode = "200")

6.2.5.3 Availability of ProactiveResponseHandler

6.2.5.3.1 Conformance Requirement

The SCWS Framework implementation shall be compliant to the following statement (TS 102 588 [2], clause 4.6).

The availability of the ProactiveResponseHandler shall depend on the availability of the ProactiveHandler as defined in TS 102 241 [12], except that it is available until the termination of the method that was invoked by the incoming Http request.

6.2.5.3.1.1 Normal execution

- Tested implicitely in clause 6.2.5.2.1.1.

6.2.5.3.1.2 Parameter Errors

- n/a.

6.2.5.3.1.3 Context errors

- Tested implicitely in clause 6.2.5.2.1.3.

6.2.5.3.2 Test area files

- n/a.

6.2.5.3.3 Test coverage

- n/a.

6.2.5.3.4 Test procedure

- n/a.

6.2.5.4 Triggering through the ToolkitInterface

Test Area Reference: Api_2_Pro_Tool.

6.2.5.4.1 Conformance Requirement

The SCWS Framework implementation shall be compliant to the following statement (TS 102 588 [2], clause 4.6).

Applets implementing the ToolkitInterface as defined in TS 102 241 [12] in addition to the SCWS interface will be triggered in the processToolkit() method upon reception of any Toolkit event.

6.2.5.4.1.1 Normal execution

- CRRN1: Trigger an applet registered to the SCWS by an http request. Within the appropriate doXXX method register to EVENT_PROACTIVE_HANDLER_AVAILABLE. After returning from doXXX the applet is triggered in the processToolkit method by this event and sets a flag. Then the applet is triggered a second time by doXXX and the flag is checked to be set.

6.2.5.4.1.2 Parameter Errors

- n/a.

6.2.5.4.1.3 Context errors

- n/a.

6.2.5.4.2 Test area files

Specific triggering: Unrecognized envelope:

Test Source: Test_Api_2_Pro_Tool.java.

Test Applet: Api_2_Pro_Tool_1.java.

Cap File: Api_2_Pro_Tool.cap.

6.2.5.4.3 Test coverage

CRR number	Test case number
N1	1

6.2.5.4.4 Test procedure

Test Case			
Id	**Description**	**API/SCWS Framework Expectation**	**APDU/HTTP Expectation**
1	Applet is triggered by doGet() Register to EVENT_PROACTIVE_HANDLER_AVAILABLE. When applet is triggered in processToolkit() set a flag. Trigger applet again by goGet() and check if flag is set.	Applet is triggerd in processToolkit() after returning from first doGet().	HTTPResponse (statusCode = "200") HTTPResponse (statusCode = "200")

6.2.5.5 Presence of CAT_TP multiplexing application

6.2.5.5.1 Conformance Requirement

The SCWS Framework implementation shall be compliant to the following statement (TS 102 588 [2], clause 4.6):

Implementation dependent on a central CAT_TP multiplexing application as defined in TS 102 225 [14] may be present in the card. It must not block the ProactiveHandler when an applet is invoked by an incoming Http request.

6.2.5.5.1.1 Normal execution

- implicitely tested by all tests in clause 6.2.4.

6.2.5.5.1.2 Parameter Errors

- n/a.

6.2.5.5.1.3 Context errors

- n/a.

6.2.5.5.2 Test area files

- n/a.

6.2.5.5.3 Test coverage

- n/a.

6.2.5.5.4 Test procedure

- n/a.

Annex A (normative): Class, methods and SCWSFramework tests acronyms

A.1 Smart Card Web Server part

ScwsExtensionRegistry	Ser
ScwsExtension	Set
ScwsExtensionService	Ses
HttpRequest	Hrq
HttpResponse	Hrs

A.1.1 Class ScwsExtensionRegistry

Method name	Acronyms
static void deregister()	Dreg
static void register()	Regi

A.1.2 Class ScwsExtensionService

Method Name	Acronyms
void doDelete()	Ddel
void doGet()	Dget
void doHead()	Dhea
void doOption()	Dopt
void doPost()	Dpos
void doPut()	Dput
void doTrace()	Dtra

A.1.3 HttpRequest interface

Method Name	Acronyms
short findAndCopyKeywordValue(byte[] headerKeywordName, short nameOffs, short nameLength, byte[] buffer, short bufferOffs, short maxLength)	Fckw_Bss_Bss
short findAndCopyKeywordValue(short keywordId, byte[] buffer, short bufferOffs, short maxLength)	Fckws_Bss
int getContentLength()	Gcle
short getContentType()	Gcty
short getRequestHttpVersion()	Grhv
short readContent()	Rcon
short readContentType()	Rcty

A.1.4 HttpResponse interface

Method Name	Acronyms
`void appendContent()`	Acon
`void appendHeaderVariable(byte[] data, short offset, short length)`	Ahva_Bss
`void appendHeaderVariable(byte[] name, short nameOffset, short nameLength, byte[] value, short valueOffset, short valueLength)`	Ahva_Bss_Bss
`void appendHeaderVariable(short headerKeywordNameId, byte[] value, short valueOffset, short valueLength)`	Ahvas_Bss
`void enableChunkMode()`	Encm
`void finalizeHeader()`	Finh
`void flush()`	Flus
`int getRemainingResponseBufferSize()`	Grrs
`void reset()`	Rset
`void sendError()`	Sene
`void setContentType()`	Scot
`void writeStatusCode()`	Wstc

A.2 Acronyms for SCWS Framework tests

Method Name	Acronyms
Applet state	Aps
Exception handling	Exh
Response Header Management	Rhm
Availability of ProactiveHandler/ProactiveResponseHandler	Pro

A.2.1 Applet state

Test Area within the chapter	Acronyms
Invokation of applets not in state selectable	Ivns
Registration remains valid if applet is not in selectable state	Regv

A.2.3 Exception handling

Method Name	Acronyms
No exception shall be propagated as HTTP error to the terminal	Noex

A.2.4 Response Header Management

Method Name	Acronyms
Send status code indicating success	Scis

A.2.5 Availability of ProactiveHandler/ProactiveResponseHandler

Test Area within the chapter	Acronyms
Incoming Http request	Inhr
Available for complete Http response	Avco
Availability of ProactiveResponseHandler	Resp
Triggering through the ToolkitInterface	Tool
Presence of CAT_TP multiplexing application	Mult

Annex B (normative): Test file description

Every test source is written in JAVA™ and shall use methods defined in Annex C interfaces to communicate with the card.

In order to be more readable, data specified as method string parameters shall be presented in 4 blocks of 4 bytes per line. Every block is separated by a space character. Every string line is appended to previous one and shall be aligned. An example is provided in annex C.

Every test file shall start with a call to reset() method.

Annex C (normative): uicc.scws.test.util package and interfaces

See attached files:

- Annex_C_ScwsTestUtil.zip.
- Annex_C_ScwsInterfaces.zip.

NOTE: These files are contained in archive ts_102835v080100p0.zip which accompanies the present document.

Annex D (normative): Test Area files

See attached file:

- Annex_D_SourceCode.zip.

NOTE: This file is contained in archive ts_102835v080100p0.zip which accompanies the present document.

Annex E (informative): HTTP-Request and HTTP-Response handling

The HttpClient V3.1 [i.2] from the Apache Software Foundation is used to handle HTTP requests and HTTP responses.

It is permissible to use another framework for HTTP handling, in this case the interface *ScwsApplicationManagementService* has to be modified accordingly.

Annex F (informative): Bibliography

- IETF RFC 791: "Internet Protocol".

NOTE: Available at http://www.ietf.org/rfc/rfc791.txt.

- IETF RFC 2616: "Hypertext Transfer Protocol -- HTTP/1.1".

NOTE: Available at http://www.ietf.org/rfc/rfc2616.txt.

- IETF RFC 793: "Transmission Control Protocol".

NOTE: Available at http://www.ietf.org/rfc/rfc793.txt.

Annex G (informative): Change history

The table below indicates all changes that have been incorporated into the present document since it was placed under change control.

Change history								
Date	**Meeting**	**Plenary Doc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**Old**	**New**
						Creation of the specification		7.0.0
2010-07	SCP #45	SCP(10)0117	001	-	F	Test case 6.1.4.4 Method getContentType	7.0.0	7.1.0
		SCP(10)0117	002	-	F	Test case 6.1.5.4 ID 4 not in line with TS 102 588		
		SCP(10)0117	003	-	F	Test case 6.2.1.2 Registration remains valid if applet is not in selectable state		
		SCP(10)0117	004	-	F	Implementation of test case 6.1.4.3		
2010-10	SCP #46	SCP(10)0197r2	019	-	F	Several corrections for test case 6.1.3.7	7.1.0	7.2.0
		SCP(10)0226	005	-	F	Correction of implementation of test case 2		
		SCP(10)0227	006	-	F	Separation of tests included in Ap1_1_Hrq_Fckw. Test implementation correction		
		SCP(10)0228	007	-	F	Dynamically creation of a data buffer in response buffer size		
		SCP(10)0229	008	-	F	Illegal usage of method appendContent()		
		SCP(10)0230	009	-	F	Dynamic allocation of remaining response buffer		
		SCP(10)0231	010	-	F	Invalid import of used library		
		SCP(10)0232	011	-	F	Api_1_Ses_Ddel test correction		
		SCP(10)0233r1	012	-	F	Api_1_Ses_Dget test correction		
		SCP(10)0234	013	-	F	Api_1_Ses correction of test implementation		
		SCP(10)0235	014	-	F	Api_1_Ses_Dopt - Correction of test implementation		
		SCP(10)0236	015	-	F	Api_1_Ses_Dput -Corrected test implementation		
		SCP(10)0237	016	-	F	Api_1_Ses_Dpos - Correction of test implementation		
		SCP(10)0238	018	-	F	Removal of unused version information in Java files		
2011-09	SCP #51	SCP(11)0140r2	017	2	B	Addition of test cases introduced by TS 102 588 Rel-8	7.3.0	8.0.0
		SCP(11)0136	019	-	F	Corrections in AID coding (duplicate CR number)		
		SCP(11)0137	020	-	F	Correction of wrong comments		
		SCP(11)0138r1	021	1	F	Correct test implementation of method getRequestHttpVersion		
		SCP(11)0139	022	-	F	Correction of test execution and evaluation of test result		
		SCP(11)0141r1	023	1	F	Test for the support for HTTP response code 204		
2011-12	SCP #53	SCP(11)0350r1	024	1	F	CR 102 835 R7 #024r1: Class ScwsExtensionRegistry test correction	8.0.0	8.1.0
		SCP(11)0351r2	025	2	F	CR 102 835 R7 #025r2: Class ScwsExtensionService test corrections		
		SCP(11)0352r2	026	2	F	CR 102 835 R7 #026r2: Interface HttpRequest test corrections		
		SCP(11)0353r2	027	2	F	CR 102 835 R7 #027r2: Interface HttpResponse test corrections		
		SCP(11)0354r2	028	2	F	CR 102 835 R7 #028r2: SCWS Runtime Environment Applet state test corrections		
		SCP(11)0355r2	029	2	F	CR 102 835 R7 #029r2: SCWS Runtime Environment Exception handling test Corrections		
		SCP(11)0356r2	030	2	F	CR 102 835 R7 #030r2: SCWS Runtime Environment Response Header Management test corrections		

History

Document history		
V8.0.0	December 2011	Publication
V8.1.0	March 2012	Publication