Smartcard-Web-Server

Approved Version 1.1.2 – 27 Sep 2012

Open Mobile Alliance

OMA-TS-Smartcard_Web_Server-V1_1_2-20120927-A

Contents

Figures

1. Scope

Open Mobile Alliance (OMA) specifications are the result of continuous work to define industry-wide interoperable mechanisms for developing applications and services that are deployed over wireless communication networks.

This specification defines the interfaces to an HTTP server in a smart card (i.e. Smart Card Web Server), which is embedded in a mobile device (e.g. SIM, (U)SIM, UICC, R-UIM, CSIM). These interfaces cover the following aspects:

- The URL to access the Smart Card Web Server (SCWS)
- The transport protocol that is used to enable the communication between HTTP applications in the device and the Smart Card Web Server
- The HTTP profile that the Smart Card Web Server needs to implement
- A secure remote administration protocol for the Smart Card Web Server
- User, or principal, authentication with the Smart Card Web Server and related security protocols

It is important to note that the Smart Card Web Server can be administrated only by the smart card issuer (e.g. Mobile Network Operator) or a delegated authorized entity. This clearly sets the scope of ownership and roles for the remote administration and services that are deployed via the Smart Card Web Server.

2. References

2.1 Normative References

[3GPP TS 31.103] "TS 31.103 Technical Specification Group Core Network and Terminals; Characteristics of the IP Multimedia Services Identity Module (ISIM) application", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/31_series/31.103/

[3GPP TS 51.011] "TS 51.011 Technical Specification Group Terminals; Specification of the Subscriber Identity Module-Mobile Equipment (SIM - ME) interface", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/51_series/51.011/

[3GPP2 C.S0023-C] "Removable User Identity Module for Spread Spectrum Systems", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0023,
URL: http://www.3gpp2.org/

[3GPP2 C.S0065] "cdma2000 Application on UICC for Spread Spectrum Systems", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0065,
URL: http://www.3gpp2.org/

[ASN.1] Abstract Syntax Notation One (ASN.1), International Telecommunication Union (ITU),
URL: http://www.itu.int/ITU-T/studygroups/com17/languages

[C.S0078] "Secured Packet Structure for CDMA Card Application Toolkit (CCAT) Applications", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0078,
URL: http://www.3gpp2.org/Public_html/specs/alltsgscfm.cfm

[C.S0079] "Remote APDU Structure for CDMA Card Application Toolkit (CCAT) Applications", 3rd Generation Partnership Project 2 (3GPP2), Technical Specification 3GPP2 C.S0079,
URL: http://www.3gpp2.org/Public_html/specs/alltsgscfm.cfm

[DER] ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), International Telecommunication Union (ITU),
URL: http://www.itu.int/ITU-T/studygroups/com17/languages

[HTML 4.0.1] HyperText Markup Language, "HTML 4.01 Specification", W3C Recommendation,
URL: http://www.w3.org/TR/1999/REC-html401-19991224/

[HTTP over TLS] "Hypertext Transfer Protocol over TLS protocol", RFC 2818, May 2000,
URL: http://www.ietf.org/rfc/rfc2818.txt

[HTTP/1.1] "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999,
URL: http://www.ietf.org/rfc/rfc2616.txt

[ISO7816-4] "Information technology - Identification cards - Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange"

[PSK-TLS] "Pre-Shared Key Cipher suites for Transport Layer Security (TLS)", RFC 4279,
URL: http://www.ietf.org/rfc/rfc4279.txt?number=

[RFC1630] "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web",
URL: http://www.ietf.org/rfc/rfc1630.txt

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997,
URL: http://www.ietf.org/rfc/rfc2119.txt

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF", D. Crocker, Ed., P. Overell, November 1997,
URL: http://www.ietf.org/rfc/rfc2234.txt

[RFC2617] "HTTP Authentication: Basic and Digest Access Authentication",

URL: http://www.ietf.org/rfc/rfc2617.txt?number=2617

[RFC3546] "Transport Layer Security (TLS) Extensions",
URL: http://www.ietf.org/rfc/rfc2119.txt

[RFC3986] "Uniform Resource Identifier (URI): Generic Syntax",
URL: http://www.ietf.org/rfc/rfc3986.txt

[RFC4785] "Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)",
URL: http://www.ietf.org/rfc/rfc4785.txt

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures,
URL: http://www.openmobilealliance.org/

[SCWS-RD] "Smartcard Web Server Requirements", Open Mobile Alliance™, OMA-RD_Smartcard_Web_Server-V1_0,
URL: http://www.openmobilealliance.org/

[TLS] "Security Transport Protocol", RFC 2246, January 1999,
URL: http://www.ietf.org/rfc/rfc2246.txt

[TS 31.102] "TS 31.102 Technical Specification Smart Cards; Characteristics of the Universal Subscriber Identity Module (USIM) application", 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/31_series/31.102/

[TS 31.115] 3GPP TS 31.115: Secured packet structure for (U)SIM Toolkit applications, 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/31_series/31.115/

[TS 31.116] 3GPP TS 31.116: Remote APDU Structure for (U)SIM Toolkit applications, 3rd Generation Partnership Project (3GPP),
URL: http://www.3gpp.org/ftp/Specs/archive/31_series/31.116/

[TS 102 221] "TS 102 221 Smart Cards; UICC-Terminal interface; Physical and logical characteristics", European Telecommunications Standards Institute (ETSI),
URL: http://www.etsi.org

[TS 102 223] "TS 102 223 Technical Specification Smart Cards; Card Application Toolkit (CAT)", R7 or higher, European Telecommunications Standards Institute (ETSI),
URL: http://www.etsi.org

[WAPCert] "WAP Certificate and CRL Profiles", WAP Forum™, WAP-211-WAPCert,
URL: http://www.openmobilealliance.org/

[XML 1.0] Extensible Markup Language (XML) 1.0 (Third Edition),
URL: http://www.w3.org/TR/2004/REC-xml-20040204/

[XML Media Type] XML Media Types,
URL: http://www.rfc-editor.org/rfc/rfc3023.txt

2.2 Informative References

[OMA-DICT] "OMA Dictionary", Open Mobile Alliance™, OMA-Dictionary-V2_1,
URL: http://www.openmobilealliance.org/

[SCWS WID] Smartcard web server work item (WID 92)

[WAPWAE] "Wireless Application Environment Specification", Open Mobile Alliance™, OMA-WAP-WAESpec-V2_3,
URL: http://www.openmobilealliance.org/

[WP HTTP] "Wireless Profiled HTTP", WAP Forum™, WAP-229-HTTP-20010329-a,
URL: http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

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

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

3.2 Definitions

Application authentication	An application that is invoked by the SCWS and that may generate dynamic content can implement its own user or principal authentication scheme. We call this authentication “Application authentication”.
BIP	Bearer Independent Protocol as defined in ETSI [TS 102 223].
BIP gateway	BIP implementation in the terminal as defined in [TS 102 223].
Browser	A program used to view (x) HTML or other media type documents.
CSIM	A Cdma2000 Subscriber Identify Module is an application defined in [3GPP2 C.S0065] residing on the UICC to register services provided by 3GPP2 mobile networks with the appropriate security.
HTTPS	A short term for HTTP over TLS.
ISIM	An IP Multimedia Services Identity Module is an application defined in [3GPP TS 31.103] residing in the memory of the UICC, providing IP service identification, authentication and ability to set up Multimedia IP Services.
Network Operator	An entity that is licensed and allocated frequency to operate a public mobile wireless telecommunications network for the purpose of providing publicly available commercial services.
Proactive UICC session	A “Proactive UICC session” is a sequence of related CAT commands and responses which start with the status response '91XX' (proactive command pending) and ends with a status response of '90 00' (normal ending of command) after Terminal Response as defined in [TS 102223].
ProactiveHandler	A ProactiveHandler is a smart card entity that is in charge of managing Proactive UICC sessions. Only one Proactive UICC session can be active at a given time.
R-UIM	A Removable User Identity Module is a standalone module defined in [3GPP2 C.S0023] to register services provided by 3GPP2 mobile networks with the appropriate security.
SCWS proactive session	A “SCWS proactive session” is a proactive UICC session that has been opened by a SCWS and is maintained by a SCWS.
SIM	A Subscriber Identity Module is a standalone module defined in [3GPP TS 51.011] to register services provided by 2G mobile networks with the appropriate security.
Smart card	This is a portable tamper resistant device with an embedded microprocessor chip. A smart card is used for storing data (e.g. access codes, user subscription information, secret keys etc.) and performing typically security related operations like encryption and authentication. A smart card may contain one or more network authentication applications like the SIM (Subscriber Identification Module), USIM, R-UIM (Removable – User Identification Module), CSIM (CDMA SIM).
Smart card application	An application that executes in the smart card.
Smart card issuer	The entity that gives/sales the smart card to the user (e.g. network operator for a SIM card).
Terminal (or device)	A voice and/or data terminal that uses a Wireless Bearer for data transfer. Terminal types may include (but are not limited to): mobile phones (GSM, CDMA, 3GSM, etc.), data-only terminals, PDAs, laptop computers, PCMCIA cards for data communication and unattended data-only terminals (e.g., vending machines).
UICC	UICC is the smart card defined for the ETSI standard [TS 102 221]. It is a platform to resident applications (e.g. USIM, CSIM or ISIM).

URI	Uniform Resource Identifiers (URI, see [RFC1630]) provides a simple and extensible means for identifying a resource. URI syntax is widely used to address Internet resources over the web but is also adapted to local resources over a wide variety of protocols and interfaces.
URL	The specification is derived from concepts introduced by the World-Wide Web global information initiative, whose use of such objects dates from 1990 and is described in "Universal Resource Identifiers in WWW", [RFC1630]. The specification of URLs (see [RFC1738]) is designed to meet the requirements laid out in "Functional Requirements for Internet Resource Locators".
User	Person who interacts with a user agent to view, hear or otherwise use a resource.
USIM	A Universal Subscriber Identity Module is an application defined in [3GPP TS 31.102] residing in the memory of the UICC to register services provided by 3GPP mobile networks with the appropriate security.
Web Page	A document viewable by using a web browser or client application which is connected to the page server.
Web server	A server process running on a processor, which sends out web pages in response to HTTP requests from browsers.

3.3 Abbreviations

APDU	Application Protocol Data Units
CSIM	CDMA SIM
IP	Internet Protocol
OMA	Open Mobile Alliance
PSK-TLS	Pre-Shared Key TLS
R-UIM	Removable User Identity Module
SCWS	Smart Card Web Server
TCP	Transmission Control Protocol
TLS	Transport Layer Security
(U)SIM	(Universal) Subscriber Identity Module

4. Introduction

A Smart Card Web Server (SCWS) is an HTTP server that is implemented in a smart card, embedded in the mobile device (e.g. SIM, (U)SIM, UICC, R-UIM, CSIM). It allows network operators to offer state of the art smart card based services to their customers by using the widely deployed [HTTP/1.1] protocol.

This solution integrates well in the Internet and the OMA architecture. The main scope of the specification is to allow a local communication between a WEB browser running in the terminal and a Web Server running in the smart card. It also covers remote administration of the smart card web server by authorized entities (i.e. card issuer or a delegated entity). This new HTTP interface is a logically separated communication channel from those that already exist today between the terminal and the smart card (e.g. using protocols that are defined in [ISO7616-4], [TS 102 221] and [TS 102 223]). It enables HTTP applications in the terminal to communicate with the smart card independently from the current telecom based communication between these two entities. This new communication channel can be used by Mobile Network Operators to deploy services for the benefit of their subscribers.

A Smart card-URI is used in order to communicate with the web server that is embedded in the smartcard (SCWS). We limit our discussion to smart card platforms such as (U)SIM (Subscriber Identification Module), UICC and R-UIM (Removable – User Identification Module), CSIM in a mobile phone.

4.1 Version 1.0

The Smart Card Web Server 1.0 defines all the main requirements of an HTTP server implemented in a smart card, allowing an HTTP client running in the terminal (e.g. the browser) to access resources stored in the smart card. The content delivered by the SCWS can be static resources but also be generated by a smart card application. The SCWS 1.0 also defines the remote administration of the Smart Card Web Server by an authorized entity.

4.2 Version 1.1

The Smart Card Web Server 1.1 enabler is a set of optimisations of the Smart Card Web Server 1.0 enabler and therefore does not introduce any new requirement or any change into the architecture. This enabler therefore refers to the requirements and architecture documents of the Smart Card Web Server 1.0 enabler.

The Smart Card Web Server 1.1 improves the Smart Card Web Server 1.0 mainly to optimize the remote management of the SCWS from different trusted entities. Each authorized entity is able to control what content and which smart card applications can be accessed under a given URI.

The Smart Card Web Server 1.1 also clarifies the cache management to improve the efficiency of the exchanges with the HTTP application in the terminal.

The Smart Card Web Server 1.1 has been updated to manage any type of resources allowing a SCWS implementation to be future proof using the `Content-Type`, `Content-Encoding` and `Content-Language` headers defined by the administration server.

The following other optimizations have been included:

- Deletion of a whole directory
- Management of multiple audit commands in the same administration request
- Addition of a cipher suite for PSK-TLS requesting only a signature
- Management of a default page when "`abs_path`" is "/"

The following clarifications have been added:

- Behaviour when the card memory is full
- Behaviour when the SCWS doesn't support persistent connections

Finally the Smart Card Web Server 1.1 specification clarifies the expected behaviour of the SCWS and of the Remote Administration server to ensure compatibility with former versions of the SCWS enabler.

5. Smart Card Web Server URL

The SCWS URL SHALL take the form of an HTTP URL as defined in [HTTP/1.1]:

http_URL = "http:" "//" host [":" port] [abs_path ["?" query]]

The optional <query> is a sequence of one or more <name>=<value> pairs separated by a '&' character. The SCWS SHALL support URLs with a length of at least 1024 characters and SHALL support abs_path with a length of at least 128 characters.

6. Support for dynamic content

Applications in the smart card, that are registered to the SCWS and are identified by the URL, can be triggered by the SCWS. An application performs a specific task and may dynamically create content and return it to the SCWS. It SHALL be possible to invoke a smart card application to the SCWS by using a path defined by the card issuer. Parameters for the application can be passed in the URL or in the request body. It SHALL be possible to apply any syntax in the data part of the URL as long as it conforms to the URI specification [RFC3986]. For example, the parameters in the URL start with the '?' character and are being formatted as a series of name=value pairs, separated by the '&' character.

A remote administration server can also trigger these applications (see 13.SCWS remote administration).

The following specification defined 2 types of applications: interception applications and content providing applications.

6.1 Security processing

Prior to searching for an application as described in section 6.2.1 and 6.3.1 the SCWS SHALL verify the security constraint fixed by protection set (see section 13.1.6.3) and take the adequate procedure if they are not fulfilled (e.g. send error or request authentication). The SCWS security policy is applied before the application invocation but the application can also apply an additional security policy in a proprietary way (like form based authentication).

6.2 Interception application

An application can be registered as an interception application in order to intercept requests, e.g. for capturing request information (e.g. logging application). Interception applications are passive applications which never reply to the request and rely on other part of the SCWS for the request processing (Content providing application or static resources).

6.2.1 Triggering

Upon receipt of a client request, after security processing, the SCWS SHALL determine if any interception application is available to capture the request and trigger it. The following URL path mapping rules are used:

- The SCWS finds all interception applications that are mapped to the exact request URL "abs_path".
- The SCWS recursively finds all applications that are mapped to the path-prefix. This is done by stepping along the path tree a directory at a time, using the '/' character as a path separator.

The SCWS SHALL apply this search until no more interception applications fulfil the triggering condition for the incoming request. Then the SCWS SHALL continue the procedure as explained in section 6.3.1.

The SCWS SHALL use case-sensitive string comparisons for matching. Characters other than those in the "reserved" and "unsafe" sets (see RFC 2396) are equivalent to their ""%" HEX HEX" encoding.

The invocation order of interception applications is not significant and is left to implementation.

Interception applications which intend to capture a large number of requests (e.g. mapped on "/*") SHOULD be designed to have a minimal performance impact on the SCWS.

Please refer to the section 13.1.6.9 on how to map requests to an application.

6.2.2 Processing

When invoked, an interception application should have access to the following data:

- Request URI
- HTTP-version (e.g. HTTP 1.0 or HTTP 1.1)
- Method (e.g. GET, POST)

- Headers fields (e.g. request headers fields, general headers fields etc.)

And it should not be permitted to access to:

- Request Message body

And it should not be able to set the response:

- Status-code
- Headers
- Message body

6.3 Content providing application

An application can be registered as a content providing application in order to perform a specific task and dynamically respond to incoming requests. Providing content is the default behaviour of an application registered to the SCWS.

6.3.1 Triggering

Upon receipt of a client request, after interception application processing, the SCWS SHALL determine if an application is available to serve the request. The following URL path mapping rules are used in order. The first successful match is used with no further matches attempted:

- The SCWS tries to find an application that is mapped to the exact request URL “abs_path”.
- If not found the SCWS tries recursively to find an application that is mapped to the longest path-prefix. This is done by stepping along the path tree a directory at a time, using the ’/’ character as a path separator.
- If no application is found the SCWS attempts to serve an appropriate content for the resource requested (e.g. static resource or error).

The SCWS SHALL use case-sensitive string comparisons for matching. Characters other than those in the “reserved” and “unsafe” sets (see RFC 2396) are equivalent to their “"%" HEX HEX” encoding.

Please refer to the section 13.1.6.9 on how to map a request to an application.

6.3.2 Processing

When invoked, an application that is registered to the SCWS, should have access to the following data:

- Request URI
- HTTP-version (e.g. HTTP 1.0 or HTTP 1.1)
- Method (e.g. GET, POST)
- Headers fields (e.g. request headers fields, general headers fields etc.)
- Message body

The application should be able to return the following data to the SCWS:

- Status-code
- Headers
- Message body

The SCWS SHALL integrate this data in the response to the HTTP client.

6.4 Examples - Informative:

If we have following content in a SCWS:

- A static resource “/index.html”
- An interception application registered as “logger” mapped to prefix path “/*”
- An interception application registered as “app_counter” mapped to prefix path “/app/*”
- A content providing application registered as “bank” mapped to resource path“/app/bank/bank”
- A static resource “/app/bank/banner.gif”
- A content providing application registered as “storage” mapped to prefix path “/app/storage/*”

The following requests may trigger the given applications:

- “/index.html/” first trigger “logger” then SCWS return “/index.html” static resource.
- “/app/bank/bank?account=1” first trigger “logger”,”app_counter” then trigger “bank” for response.
- “/app/bank/banner.gif” first trigger “logger”,”app_counter” then SCWS return “/app/bank/banner.gif” static resource.
- “/app/storage/mypictures/picture1.jpg” first trigger “logger”,”app_counter” then trigger “storage” for response.

7. IP Address and Port Numbers for local communication

This section deals with the IP addresses that a local HTTP application, running in the terminal, uses in order to connect to the SCWS. The IP address depends on the local transport protocol as described in the following sections. Two protocols are specified: The Bearer Independent protocol (BIP) or directly native TCP/IP if the smart card implements a TCP/IP stack.

7.1 Using the BIP transport protocol

If the smart card does not have its own IP address and does not directly support TCP/IP, the BIP gateway in the terminal is used as a protocol converter. Then, the TCP/IP protocol is used between the HTTP application in the terminal and the BIP gateway, and the BIP protocol is used between the BIP gateway and the smart card.

The loopback IP Address 127.0.0.1 SHALL be used by the HTTP application in the terminal to address the BIP gateway. This address is also named “localhost” on some systems. Mnemonic names may be used if the implementation can apply adequate security measures on the association of the mnemonic and the IP address, otherwise the IP address 127.0.0.1 should be used.

7.1.1 Port Numbers

The BIP gateway SHALL listen on the ports opened by the SCWS to manage the requests from HTTP applications in the terminal. The SCWS SHALL open a port for HTTP requests and MAY open a port for the HTTP over TLS requests.

HTTP and HTTP over TLS already have default TCP port numbers reserved (80 and 443) and it must remain possible for the hosting device to run its own HTTP services using these ports.

For the SCWS, HTTP SHALL be addressed using the TCP port number 3516 and HTTP over TLS SHALL be addressed using the TCP port number 4116. Both ports are already reserved by IANA. Port 3516 is reserved as “smartcard Port” and port 4116 as “smartcard-TLS”.

7.1.2 Sample URLs

It SHALL be possible to address any SCWS resource with URLs. Such resources may be an xHTML file, an image file or any other content as a static file or dynamically generated on-the-fly.

As an example, a resource called "foobar.xhtml" with the path "/pub/files" corresponds to these URLs:

http://127.0.0.1:3516/pub/files/foobar.xhtml

https://127.0.0.1:4116/pub/files/foobar.xhtml

The following URLs may trigger an application and contain parameters, which are specific for the addressed application:

http://127.0.0.1:3516/cgi/start?launch

http://127.0.0.1:3516/cgi/SSO?account=username&otherparam=123

https://127.0.0.1:4116/apps/display?df=7F01&ef=3F01&record=01&offset=50&length=10

http://127.0.0.1:3516/apps/show?title=This%20is%20the%20front%20page

7.2 Using the TCP/IP protocol

If the smart card has its own IP address and directly supports TCP/IP and the terminal supports direct IP addressing of the smart card, then the TCP/IP protocol SHALL be used between the HTTP application in the terminal and the SCWS on the smart card.

7.2.1 Port Numbers

The port numbers for accessing the SCWS shall be the default HTTP ports: 80 for HTTP and 443 for HTTP over TLS.

7.2.2 Sample URLs

Same as the examples above in [7.1.2] but the smart card now has an IP address allocated to it, e. g.

http://<smart card IP address>/pub/files/foobar.xhtml

https:// <smart card IP address>/cgi/display?df=7F01&ef=3F01&record=01&offset=50&length=10

Where <smart card IP address> is the IP address that is given to the smart card.

7.3 Remote connection to the SCWS

Connection to the SCWS from remote applications (i.e. not running in the terminal) is not supported in this version.

8. Local transport protocols

The SCWS responds to HTTP requests from HTTP applications (e.g. browsers) in the terminal. The HTTP requests and responses are exchanged over a local transport protocol between the smart card and the terminal. The local transport protocol provides the basic functionality of data exchange between the SCWS and the terminal.

NOTE: In order to conform to the (U)SIM constraints defined in the [TS 102 223] the processing time of the SCWS and of the potential triggered applications should not exceed 2 seconds for each consecutive processing unit in order to allow interception and treatment of (U)SIM commands from the Device (e.g. network authentication requests). Please refer to 13.3.2.8 for more information that is related to BIP implementation.

8.1 The BIP transport protocol

The BIP protocol is specified by ETSI SCP [TS 102 223] and enables the smart card to communicate with external entities over standardised protocols, including TCP/IP. For the SCWS only the TCP/IP protocol is used. Only the related features are described in the next paragraphs.

According to [TS 102 223] the smart card can open a BIP TCP data channel with the terminal and ask for the following:

- Client mode: A smart card application wants to communicate with a remote server over TCP and the destination server is identified with an IP address. When the BIP channel is opened the terminal behaves as a gateway communicating over TCP/IP with the remote server and locally with BIP commands with the smart card.
- Server mode: The smart card is a server allowing TCP applications in the terminal to connect to it on a TCP port number. When the BIP channel is opened the terminal shall listen on the localhost IP address (e.g. 127.0.0.1 for IPv4) at the TCP port given in the command and forward incoming / outgoing data on this port to / from the UICC.

The two above features are used to implement the SCWS. The Client mode is used for SCWS remote administration while the Server mode is used for the operation of the SCWS itself (browsing the SCWS).

The BIP functionality, used in the SCWS specification, can be depicted in the figures in B.5 and is described for information only. The detailed functionality is specified in [TS 102223].

8.1.1 SCWS usage of BIP

When the SCWS starts, it may open several BIP channels in “TCP, UICC in server mode”. The terminal MUST support at least two opened BIP channels in “TCP, UICC in server mode”. One BIP channel is for HTTP and another is for HTTP over TLS. The terminal MAY support additional opened BIP channels “TCP, UICC in server mode”. The Terminal MUST automatically enable the BIP channel “TCP, UICC in server mode”, independently of network connection and without the need for special configuration by the user. The SCWS may start after a Terminal Profile event. The SCWS SHOULD keep the number of opened BIP channels to the minimum whenever possible.

The SCWS may also open an additional BIP channel “TCP, UICC in client mode” for administration as defined in chapter [13]. The terminal MUST thus support an additional opened BIP channel “TCP, UICC in client mode” besides the BIP channels for “TCP UICC in server mode” as defined above. In this specification we call “BIP Gateway” the BIP implementation in the terminal as defined in [TS 102 223].

Each BIP channel in UICC server mode SHALL be opened by sending the “Open channel related to UICC server mode” command to the terminal as described in [TS 102 223]. The HTTP BIP channel shall be opened to listen on TCP port 3516. The HTTP over TLS BIP channel shall be opened to listen on TCP port 4116.

As a result the BIP gateway in the terminal shall listen on the indicated TCP ports and shall inform the SCWS when a client connects to these TCP ports. The BIP gateway shall manage the exchange of data between the SCWS and the connecting HTTP application as described in [TS 102 223].

When an HTTP application in the terminal connects to the SCWS (via the BIP gateway) and data is exchanged, the SCWS MAY open an additional BIP channel on either ports (i.e. 3516 or 4116), in order to allow another HTTP application in the terminal to connect to the SCWS (via the BIP gateway) on the same TCP port number. This will allow the SCWS to communicate with several terminal HTTP applications at the same time, since each BIP channel is used to communicate with only one application at a time.

The SCWS SHALL implement HTTP and SHOULD implement HTTP over TLS. The SCWS SHALL be able to communicate over HTTP and HTTP over TLS at the same time.

8.1.1.1 SCWS and other CAT applications concurrency

A "SCWS proactive session" is a proactive UICC session that has been opened by a SCWS and is maintained by a SCWS (see [Definitions] section). A SCWS proactive session consists of a sequence of the following APDU commands:

FETCH ⇆ '90 00'

TERMINAL RESPONSE ⇆ '91 XX'

A SCWS proactive session can use either "TCP, UICC in Client Mode" or "TCP, UICC in Server Mode" when the used transport protocol is BIP (see chapter 8.1).

A SCWS proactive session, where the SCWS uses the ProactiveHandler, SHALL be interruptible in the case when another CAT Application has requested the proactive handler.

In this case the SCWS may have to suspend its proactive session with the terminal, depending of operating priorities, and let the other application take over and start its proactive UICC session. Once the CAT Application(s) release the ProactiveHandler the SCWS will be resumed by the CAT.

NOTE: The processing priorities can be set by applet installation parameters. It is up to the card issuer to decide which processing priority shall be assigned to the SCWS in relation to other CAT Applications (e.g. other Java Card™ Toolkit applets). There may be CAT Applications on the UICC with higher or lower processing priorities with respect to the SCWS processing priority.

NOTE: In order to conform to the (U)SIM constraints defined in the [TS 102 223] the processing time of the SCWS and of the potential triggered applications should not exceed 2 seconds. If the execution time exceeds this limit a MORE TIME command should be issued between the consecutive processing units.

8.1.1.1.1 Example

During an active SCWS BIP session an ENVELOPE (SMS PP Data Download) is sent by the terminal to the UICC. The incoming ENVELOPE command contains instructions to trigger a CAT Application that should display a notification message on the terminal screen. This notification message should be displayed as soon as possible by the terminal. Waiting for the end of the SCWS BIP session may not be acceptable for displaying this notification message.

Since the SCWS uses the ProactiveHandler at that moment the triggered CAT application registers itself to the EVENT_PROACTIVE_HANDLER_AVAILABLE event. When the TERMINAL RESPONSE, which belongs to the FETCH command of the SCWS proactive session, has been sent by the terminal to the UICC the SCWS suspends its proactive session with the terminal and let the other application take over and start its Proactive UICC session. Once the CAT Application finishes the SCWS is resumed by the CAT.

ME		UICC
ENV (Data available)	⇆	91 XX
FETCH (Receive data)	⇆	90 00
TR (a HTTP request to the UICC)	⇆	91 XX
		SCWS computes the HTML response
FETCH (Send data)	⇆	Data (1st part of huge HTML response)

ME		UICC
		90 00
TR (Send data ok)	⇆	91 XX
FETCH (Send data)	⇆	Data (2nd part of huge HTML response)
		90 00
TR (Send data ok)	⇆	91 XX
...	...	...
FETCH (Send data)	⇆	Data (Nth part of huge HTML response)
		90 00
ENV (SMS PP Data Download)	⇆	90 00, because the SMS PP DD cannot be processed immediately. However the SCWS makes the Proactive Handler available for the other CAT application **after the next TR**. Then the SMS PP DD can be processed by the UICC actually without significant delay.
TR (Send data ok)	⇆	**91 YY**
FETCH (Display Text)	⇆	90 00
TR (Display Text ok)	⇆	91 XX (the BIP data transfer continues)
FETCH (Send data)	⇆	Data ((N+1)th part of huge HTML response)
		90 00
TR (Send data ok)	⇆	91 XX
...	...	...
FETCH (Send data)	⇆	Data (Last part of huge HTML response)
		90 00
TR (Send data ok)	⇆	90 00

8.2 TCP/IP transport Protocol

If the smart card supports TCP/IP then the HTTP application in the terminal SHALL directly communicate with the SCWS without a BIP gateway. In this case the SCWS SHALL listen on the default port 80 for incoming HTTP requests and SHOULD listen on port 443 for incoming HTTP over TLS requests.

9. SCWS HTTP Profile

In order to ensure interoperability of the SCWS an HTTP profile, that the SCWS needs to implement, is defined. This profile identifies some HTTP features as mandatory in the context of a SCWS implementation, however if the HTTP 1.1 standard itself mandates additional features they should also be implemented. The SCWS MUST at least be a conditionally compliant implementation of an HTTP server as specified in [HTTP/1.1] (i.e. implement all the MUST level requirements but not all the SHOULD level requirements).

9.1 Request-URI

The Request-URI SHALL fulfil the rules for SCWS URIs defined in this document (chapter [5]).

9.2 HTTP Methods

The following table lists optional and mandatory HTTP methods for the SCWS:

Method	Supported	Additional comments
OPTIONS	Optional	
GET	Mandatory	Mandated for HTTP 1.1 server implementations
HEAD	Mandatory	Mandated for HTTP 1.1 server implementations
POST	Mandatory	Support for forms in user interface
PUT	Mandatory	Support for remote administration
DELETE	Mandatory	Support for remote administration
TRACE	Optional	
CONNECT	Optional	

Specific actions on reception:

When receiving an incoming request that is not supported, the SCWS shall respond with an HTTP response message with Status-Code = 405 (Method not allowed).

If the SCWS does not support conditional and partial GET commands it SHALL return the whole page.

When receiving a GET command whose "`abs_path`" is "/", the SCWS shall convert it into a GET on a default page, for example on "/index.html".

9.3 HTTP Authentication headers

The following Headers MUST be implemented in order to Support HTTP Authentication as specified in section [10]

- Authorisation request header.
- WWW-Authenticate response header.

9.4 Status Code Definitions

9.4.1 Server Error 5xx

9.4.1.1 590 Memory Full

The server is currently unable to handle the request because the card memory is full.

9.5 Implementation notes - informative

9.5.1 Caching

To improve SCWS performance use of response caches based on Entity Tag Validators MUST be supported.

- The conditional GET based on the If-Match and If-None-Match request-header MUST be supported by SCWS.
- The ETag response-header MUST be generated by SCWS, a simple implementation is to increment a short value on each PUT admin request.

Exemple:

```
--------------------------------->
GET /index.html HTTP/1.1CRLF
Host: 127.0.0.1:3516 CRLF
CRLF

<---------------------------------
HTTP/1.1 200 OK
Content-Type: text/html CRLF
Content-Length: 4 CRLF
Etag: 0001
CRLF
Hello

--------------------------------->
GET /index.html HTTP/1.1CRLF
Host: 127.0.0.1:3516 CRLF
If-None-Match: 0001 CRLF
CRLF

<---------------------------------
HTTP/1.1 304 Not Modified CRLF
Content-Length: 0CRLF
```

SCWS will generally be a clock-less HTTP server (as already permitted by the IETF [HTTP/1.1]).

This means that it will have a specific handling of caching based on dates validators, and generally will ignore them by systematically sending a complete response.

It is also expected that due to limited resources, the SCWS will generally be a light weight HTTP server and will normally implement a minimal set of HTTP /1.1 features. This means that it will have a specific handling of content negotiation related headers (Accept, Accept-Encoding, Accept-Language), and generally will ignore them.

9.5.2 Connection

In order to improved interoperability with handset behaviour it is recommended for SCWS server that does not support persistent connections to include the "close" connection general header in every http response.

Example:

```
--------------------------------->
GET /index.html HTTP/1.1CRLF
Host: 127.0.0.1:3516 CRLF
CRLF

<---------------------------------
HTTP/1.1 200 OK
Content-Type: text/html CRLF
Content-Length: 4 CRLF
Connection: close CRLF
CRLF
Hello
```

10.User or principal authentication to the SCWS

If the SCWS requires an access condition which has not been fulfilled it shall provide means to fulfil this security conditions (e.g. it may perform a request to the user, or principal to enter a user name and password).

The authentication is performed between the client application and the SCWS and MAY use standard HTTP authorization exchange mechanisms specified in IETF [RFC 2617]. The SCWS MUST support Basic authentication and MAY support Digest authentication as defined in IETF [RFC 2617].

An application that is invoked by the SCWS can also implement its own authentication scheme. This authentication may be based on user name and password or other means. As an example the invoked card application can display a form that asks for a user name and password and then captures the entered data.

11.Security Protocols

11.1 Transport Layer Security (TLS)

TLS (Transport Layer Security) [TLS] provides a secure and reliable transport mechanism between two communicating parties. It provides confidentiality and integrity protection for the transport used. It can also provide unilateral or mutual authentication depending on the implementations. TLS works in a client-server model, where the initiator is called the Client and the responder is called the Server. In most cases, a TLS client can authenticate a TLS server using a public key certificate. Mutual authentication is possible using public key certificates or with pre-shared keys using PSK-TLS.

A SCWS remote administration server acting as an HTTPS server SHALL implement [HTTP over TLS] using [PSK-TLS].

A SCWS administration agent acting as an HTTPS client SHALL implement [HTTP over TLS] using [PSK-TLS].

The implementation of [HTTP over TLS] using [PSK-TLS] is defined in sub-section 11.1.1.

The SCWS acting as a local HTTPS server SHOULD implement [HTTP over TLS] using public key as defined in sub-section 11.1.2 and MAY implement [HTTP over TLS] using [PSK-TLS].

The implementation of [HTTP over TLS] using public key is defined in sub-section 11.1.2.

11.1.1 PSK-TLS

PSK-TLS is used when a symmetric key is shared between the SCWS and the connecting principal (e.g. a remote administration server). How shared keys are provisioned in the smart card and the connecting principle is beyond the scope of this specification.

11.1.1.1 Supported Cipher Suites for PSK-TLS

The SCWS MUST support all of the following cipher suites:

- TLS_PSK_WITH_3DES_EDE_CBC_SHA [PSK-TLS]
- TLS_PSK_WITH_AES_128_CBC_SHA [PSK-TLS]
- TLS_PSK_WITH_NULL_SHA [RFC4785]

11.1.2 Public Key Pair and device certificate

The SCWS SHOULD use a public key pair, stored in a secure area, and SHALL allow the usage of these keys only to the TLS implementation or to authorized card applications, as defined by the card issuer internal security policy. The SCWS SHOULD also embed a device certificate for the public key. The device certificate shall be provisioned by the card issuer and be signed by a trusted authority of the card issuer. The public key pair and device certificate SHALL be used for server authentication in TLS (i.e. TLS class 2 authentication).

11.1.2.1 Supported Cipher Suites

If the SCWS use a public key pair and device certificate then it MUST support all of the following cipher suites:

- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA

11.1.2.2 Server Authentication

The SCWS MUST support server authentication using TLS 1.0 and SHALL use the WAP profiled X.509 server certificate [WAPCert].

11.1.3 Supported TLS extensions

It may be desirable for the SCWS to negotiate a smaller maximum fragment length due to memory limitations or bandwidth limitations. This extension enables the usage of the following fragment length (when the default value is 2^14):

```
2^9(1), 2^10(2), 2^11(3), 2^12(4), (255)
```

- The card administration agent MAY use the Maximum Fragment Length Negotiation as defined in [RFC3546].
- The administration server SHALL support the Maximum Fragment Length Negotiation as defined in [RFC3546] and SHALL accept fragment length down to the minimum of 512 bytes.
- A HTTP client connecting to the SCWS MAY use the Maximum Fragment Length Negotiation as defined in [RFC3546].
- The SCWS SHALL support Maximum Fragment Length Negotiation as defined in [RFC3546] and SHALL accept fragment length down to the minimum of 512 bytes. If the client does not negotiate, the SCWS SHALL accept TLS fragment length with the predefined length of 16 KB.

11.1.4 Session Resume

The SCWS SHOULD support the session resume as defined in TLS. The longer session life (e.g., 12 hours) SHOULD be used. The guidelines on the session resume as documented in TLS 1.0 SHOULD be respected.

12. Access Control Policy (ACP)

This section defines a complementary and yet optional security feature that adds an additional access control to the SCWS from within the terminal itself. It is called the ACP Enforcer (Access Control Policy Enforcer) and is aimed to control the access to the SCWS for terminal applications. Its main purpose is to provide protection against denial of service attacks on the SCWS and the misuse of the SCWS by malicious terminal applications. The ACP Enforcer is especially useful in terminals that allow the user to freely download and install applications in the terminal (e.g. terminals with so-called open operating systems). For example the user may download and install a malicious application in the terminal that could try to block access to the SCWS or potentially ask the user for sensitive information such as passwords or secrets required to access personal information in the SCWS.

The Access Control Policy (ACP) is a data object that the device, implementing an ACP Enforcer, can retrieve from the smart card. An ACP Enforcer MAY be implemented by terminals that implement a trusted execution environment (as defined by external standardization fora). The ACP data object defines the following possible access rights:

- Allow access to the SCWS to terminal applications that are trusted by the handset manufacturer
- Allow access to the SCWS for terminal applications that belong to the operator trust level, if supported by the terminal.
- Allow access to the SCWS for terminal applications that belong to an enterprise trust level, if supported by the terminal.
- Allow access to the SCWS to all trusted terminal applications in the handset
- Allow access to the SCWS to some trusted terminal applications in the handset that are identified by the hash of the signing certificate
- Allow access to all terminal applications

The Access Control Policy Enforcer may enforce access restrictions to the SCWS by blocking access to the relevant TCP ports (i.e. used by the SCWS) for certain local terminal applications within the terminal. If no ACP can be retrieved from the smart card then all terminal applications can access the SCWS.

12.1 ACP retrieval

Since the ACP is not confidential it can be accessed via the HTTP protocol itself. The device SHALL retrieve the ACP by sending the following HTTP GET command:

```
Get /config/acp HTTP/1.1 CRLF
Host: <hostname or IP address> CRLF

CRLF
```

If the terminal supports ACP enforcement it SHALL retrieve and apply the ACP immediately after the initialization phase of the SCWS transport protocol with the smart card. The terminal MUST block access to the SCWS for all terminal applications before the retrieval and enforcement of the ACP rules that are described in it. After the retrieval of the ACP from the SCWS the terminal MUST apply the rules and MUST block access to the SCWS for all non-authorized terminal applications. If the terminal is not able to interpret the ACP (e.g. ACP is malformed) it MUST block access to the SCWS for all terminal applications.

The following is an example of an HTTP response to the above GET command:

```
HTTP/1.1 200 OK CRLF

[Content-Type: application/octet-stream CRLF]
[Content-Length: xxxx CRLF] or [Transfer-Encoding: chunked CRLF]
CRLF

ACP as binary data
```

The ACP data is the binary DER [DER] encoding of the ACP ASN.1 data object described below.

12.2 ACP data objects

The ACP ASN.1 [ASN.1] data objects are described below. A description of the role of each data object follows afterwards.

--Access Control Policy

```
ACP ::= SEQUENCE {
            trustedAppInformation      TrustInformation,
            SelectedTrustedApplications             CertHashList     OPTIONAL
}


TrustInformation ::= BIT STRING {
            allApplications                         (0),
            allTrustedApplications                  (1),
            trustedByManufacturer                   (2),
            trustedByOperator                       (3),
            trustedByEnterprise                     (4),
            selectedTrustedApps                     (5),
            ... -- for future extensions
}
CertHashList  ::=  SEQUENCE SIZE (1..MAX) OF CertHash
CertHash ::= OCTET STRING  --SHA1 Hash of Entire Certificate
```

Trusted certificates that indicate various levels and granularity of trusted terminal applications may already be present in the device. If the smart card contains additional certificates they MUST be read as described in [SCProv] and added to the list of existing trusted certificates.

The bits in the TrustInformation type indicate which terminal applications are allowed to access the SCWS and are defined as follows:

- The allApplications bit is asserted when access to the SCWS is allowed by all terminal applications without restriction.
- The allTrustedApplications bit is asserted when access to the SCWS is allowed by signed terminal applications that have been properly validated to any one of the trusted certificates.

- The trustedByManufacturer bit is asserted when access to the SCWS is allowed to terminal applications trusted by the terminal manufacturer (e.g. signed applications that have been properly validated to trusted certificate(s) associated with the manufacturer, or by other means).
- The trustedByOperator bit is asserted when access to the SCWS is allowed to signed terminal applications that have been properly validated to trusted certificate(s) associated with the operator trust level, if this trust level is supported by the terminal.
- The trustedByEnterprise bit is asserted when access to the SCWS is allowed to signed terminal applications that have been properly validated to trusted certificate(s) associated with the "Enterprise trust level", if this trust level is defined and supported by the application's runtime environment of the terminal.
- The selectedTrustedApps bit is asserted to indicate that access to the SCWS is allowed by signed terminal applications that have been properly validated to a trusted certificate (or certificates) as indicated in the SelectedTrustedApplications structure.

The SelectedTrustedApplications type indicates that access to the SCWS is allowed by signed terminal applications that have been properly validated to one of the trusted certificates whose hash is explicitly indicated in this structure. If a trusted certificate corresponding to the indicated hash cannot be found, the signed terminal application SHALL not have access to the SCWS.

The certificate hash is a SHA-1 hash calculated over the DER encoding of the complete certificate.

13.SCWS remote administration

The SCWS administration is the ability to upload new data (e.g. xHTML pages), delete data and change configuration parameters for the SCWS. The commands are sent using the "Administration protocols" that are described below. All these protocols implement end-to-end authentication, integrity and confidentiality. Only an authorized administration entity can administrate the SCWS.

It is assumed that a correctly authenticated administration entity can use all the administration commands listed in this chapter. The PUT and DELETE HTTP commands SHOULD NOT be accessible for a non authenticated client or for an authenticated client that does not have admin privileges.

Pages that only the admin server is allowed to read MUST be protected e.g. using the protection set mechanism described in 13.1.6.2.

The "/SCWS/admin" URL SHALL be accessible only to authorized administration entities e.g. using the protection set mechanism described in 13.1.6.2.

The SCWS SHALL NOT terminate an HTTP request response session with a terminal HTTP client application upon receiving an HTTP administration command.

13.1 Administration commands

The SCWS administration commands are HTTP commands as described below. These commands are sent to the smart card with the administration protocols as described in [13.3]. This chapter describes the commands only and does not deal with how they are sent to the SCWS. The following chapters describe the administration protocols themselves.

The following HTTP requests are used for administrating the SCWS:

PUT	Is used to install or update a page on the SCWS
DELETE	Is used to delete a page from the SCWS
POST	Is used to send special administration commands to the SCWS May be used to send HTTP commands that pass some parameters to dynamic applications registered to the SCWS.
GET	Is used by the admin server to retrieve a page from the SCWS Is used by an admin server to audit the SCWS (see Appendix G SCWS content audit) May be used to retrieve data from a dynamic application registered to the SCWS
HEAD	Is used to retrieve the header of a page on the SCWS

The URIs in PUT-requests, DELETE-requests, GET-requests and HEAD-requests must be an absolute path, i. e. start with "/". The POST requests contain special commands, which are parsed by a special admin-component within the SCWS.

If the full administration protocol is used, PUT, DELETE, GET and HEAD commands SHALL be rejected if the card administration agent involved in the exchanges is not allowed to access the URI requested in the request.

If the URI of the HTTP command is assigned to a dynamic content application then the command is processed by the application itself and the response to the command depends on the implementation of the dynamic content application.

13.1.1 PUT

The PUT HTTP request is used to install or update a page on the SCWS.

- The URL to be installed/updated must be an absolute path (starting with the root of the SCWS)
- The value of the `Content-Type`, `Content-Encoding` and `Content-Language` headers present in the request shall be stored by the SCWS, and shall be included in the response to a GET command on this resource.
- The entity-header-field "Content-Length" must be used. The value is the length of the message-body in bytes as a decimal number.
- The value of the Cache-control header present in the request shall be stored by the SCWS, and shall be included in the response to a GET command on this resource.

Example:

```
PUT /example.html HTTP/1.1 CRLF
Content-Type: text/html; charset=utf-8 CRLF
Content-Length: 16 CRLF
Content-Language: en-gb CRLF
Content-Encoding: gzip CRLF
Host: anything CRLF
CRLF
<html>xxx</html>
```

Then the response to a GET command on `/example.html` shall be as follows:

```
HTTP/1.1 200 OK CRLF
Content-Type: text/html; charset=utf-8 CRLF
Content-Language: en-gb CRLF
Content-Encoding: gzip CRLF
Content-Length: 16 CRLF
CRLF
<html>xxx</html>
```

Example with Cache-Control:

```
PUT /example.html HTTP/1.1CRLF
Content-Type: text/html CRLF
Content-Length: 5 CRLF
Cache-Control: no-cache CRLF
Host: anything CRLF
CRLF
<html>xxx</html>
```

Then the response to a GET command on `/example.html` shall be as follows:

```
HTTP/1.1 200 OK CRLF
Content-Type: text/html CRLF
Cache-Control: no-cache CRLF
Content-Length: 16 CRLF
CRLF
<html>xxx</html>
```

13.1.2 DELETE

The DELETE request is used to delete a page or a directory from the SCWS. The URL to be deleted must be an absolute path (starting with the root of the SCWS). If the path identifies a directory, this directory and its content shall be deleted.

Example:

```
DELETE /faq/abc HTTP/1.1  CRLF
Host: anything CRLF

DELETE /pub/files/ HTTP/1.1  CRLF
Host: anything CRLF
```

13.1.3 POST

The POST request is used to send special administration commands to the SCWS as described below.

- The URL must be "/SCWS/admin"
- The entity-header-field "Content-Type" must be used. The value is "application/x-www-form-urlencoded".
- The entity-header-field "Content-Length" must be used. The value is the length of the message-body in bytes as a decimal number
- The message-body, in this case called "admin-body", contains data that is interpreted by the admin-component of the SCWS. See 13.1.5 for the definition of the admin-body.

Example:

```
POST /SCWS/admin HTTP/1.1  CRLF
Content-Type: application/x-www-form-urlencoded CRLF
Content-Length: xx CRLF
Host: anything CRLF
CRLF
admin-body  ;the message-body of the POST request is called admin-body and

            ;contains one or several admin commands
```

The POST command MAY also be used to send HTTP commands that pass some parameters to applications that are invoked by the SCWS. In this case the POST request shall include the URL of the invoked application and contain the parameters in the POST command body:

- The URL must be the URL of the invoked application
- The message-body contains data that is interpreted by the invoked application.

Example:

```
POST /myApplication HTTP/1.1  CRLF
Content-Type: application/x-www-form-urlencoded CRLF
Content-Length: xx CRLF
Host: anything
CRLF
data1=abc&data2=xyz
```

13.1.4 GET

The GET request is used by the Admin Server to read a page of the SCWS. The URL to be read must be an absolute path (starting with the root of the SCWS).

Example:

```
GET /config/content-version   HTTP/1.1    CRLF
Host: anything CRLF CRLF
```

The GET request is also used to audit the SCWS content (see Appendix G).
The GET request may be used to retrieve data from a dynamic application registered to the SCWS. The response depends on the implementation of the dynamic content application

13.1.5 HEAD

The HEAD request is identical to GET except that SCWS MUST NOT return a message-body in the response. The URL to be read must be an absolute path (starting with the root of the SCWS).

Example:

```
 HEAD /example   HTTP/1.1    CRLF
Host: anything CRLF CRLF
```

13.1.6 Special admin commands that are used within a POST request

Special Admin-Commands are transported within the message-body of POST-requests. The SCWS must be able to process an admin-body with a length of at least 1024 Bytes. If more than 1024 bytes are needed for administration, the Admin Server in the network should transport the admin commands within several POST requests. The admin commands are handled by the admin server application, to which only authorized principals can have access (as provisioned by the card issuer).

The commands always start with a command attribute (i.e. cmd=commandName) followed by one or several attribute value pairs as parameters. These command pairs are formatted according to the W3C recommendation [HTML 4.0.1]. This means that the attribute name is separated from the attribute value by '=' and attribute name/value pairs are separated from each other by '&'. All unsafe characters in names and values are escaped: Space characters in names and values are replaced by '+', and reserved characters are escaped as described in [RFC3986]. Non-alphanumeric characters are replaced by '%HH', a percent sign and two hexadecimal digits representing the ASCII code of the character. All attributes names and values are case sensitive and the order of attribute value pairs is not relevant (if not otherwise specified).

If there are several commands in one POST request, these are also separated by '&', see the following example.

If a parameter name in a command is not known or if a parameter name has a wrong parameter value the command MUST be aborted. To recognize constants they are written with quotes in this specification, but in the commands themselves they appear without quotes. For example "on" refers to the string "on" but in the command invocation it appears without the quotes: http=on.

If the full administration protocol is used and if a parameter of the administration command is an URI or a path, the command SHALL be rejected if the card administration agent involved in the exchanges is not allowed to access this URI or this path.

Example:

cmd=set&http=on&cmd=pr&psid=ps1&protocol=http

```
cmd=set&http=on&cmd=pr&psid=ps1&protocol=http
```

In this example two commands are sent. The name of the first command name is `“set”` with the attribute value pair `“http=on”.` The name of the second command is “pr” with two attribute-value pairs.

Some parameter/value pairs are mandatory in an admin command; others are optional (as indicated in the tables).

13.1.6.1 Configure the SCWS

This command allows the administrator to set parameters in the SCWS. Command name is: set

Parameter name	Value	mandatory/ optional	minimum supported length
http	"on" – turn the HTTP server operational mode on "off" – turn the HTTP server operational mode off	o	
https	"on" – turn the HTTPS server operational mode on "off" – turn the HTTPS server operational mode off	o	
uri	The URI that indicates the default page to associate with the abs_path : "/"	o	128 characters

Operational mode is distinguished from administration mode. The communication with remote administration application is always enabled and is done on an authenticated secure channel as described in [13].

- If HTTP or HTTPS is set to Off, the SCWS SHALL abort current communication and stop listening on the relevant TCP port
- If HTTP or HTTPS is set to On the SCWS SHALL start listening on the relevant TCP port.

The following error codes can be used (see 13.2.2):

- 0002 wrong parameter
- 0003 wrong value
- 0005 missing parameter or value

Example: cmd=set&https=off&http=on&uri=/index.html

13.1.6.2 Define a resource protection set

This command is used to define or update a set of security parameter that need to be satisfied to access a protected resources. A URI sub-tree can be linked to a protection set (see"pr" command). The protection set, to which a URI is linked, defines how to access this URI (or sub-tree).

Pages can be protected with HTTPS, or with basic or digest authentication or with both. The 'protocol' parameter defines whether https is needed or whether http is sufficient. If a realm is defined, the user must be authenticated by basic or digest authentication. All users added to this "psid" are allowed to access the URI with the correct authentication with name and password.

Command name is: dps

Parameter name	Value	mandatory/ optional	minimum supported length

psid	Protection Set Identifier	m	32 characters
protocol	"http" – HTTP protocol "https" – HTTPS protocol "admin" – Remote Administration Protocol	m	
realm	The realm-name as defined in [RFC2617]	o	32 characters
auth	"basic" – Require Basic or digest authentication (if supported by the SCWS) to authenticate the connecting principal "digest" – Require Digest authentication to authenticate the connecting principal	conditional: must be used if the parameter realm is present	

The protocol value "admin" specifies that only the remote administration server (or the SCWS itself) can access this protection set. It should be used to protect internal configuration resources (for example admin server settings specified in 13.3.2.3) or to hide some pre-issued resources waiting to be published.

If a protection set already exists with the given "psid" the SCWS SHALL update it. If the updated protection set no longer includes a realm parameter all users previously added to this protection set SHALL be automatically removed from the protection but not deleted from the SCWS (see 0).

If a protection set with auth=digest is defined but the SCWS does not implement Digest Authentication, the returned error code shall be "wrong value: auth=digest".

The following error codes can be used (see 13.2.2):

- "0002 wrong parameter"
- "0003 wrong value"
- "0005 missing parameter or value"

Examples:

```
1. cmd=dps&psid=ps1&protocol=http&realm=Your%20pages&auth=basic

2. cmd=dps&psid=ps2&protocol=https

3. cmd=dps&psid=ps3&protocol=https&realm=Your%20secured%20pages&auth=digest

4. cmd=dps&psid=ps4&protocol=admin
```

The first example defines (or changes) a protection set internally named "`ps1`" using HTTP with basic (or digest) user authentication. The realm name used to prompt the user will be "Your pages".

The second example defines (or changes) a protection set internally named "`ps2`" using HTTPS with server authentication (no user authentication).

The third example defines (or changes) a protection set internally named "`ps3`" using HTTPS with user `digest` authentication. The realm name used for prompt the user will be "Your secured pages".

The forth example defines (or changes) a protection set internally named "ps4" only accessible by SCWS remote administration server under authenticated administration protocol or internally by the SCWS itself.

Note:

A resource protection set which uses a realm with authentication and which has no authorized user registered (see 13.1.6.7) will always refuse access to its content.

13.1.6.3 Protect a URI sub-tree with a protection set

This command is used to protect a URI or sub-tree with a protection set. This means that access to this sub-tree is granted only if authentication succeeded according to the parameters in the given protection set. A URI sub-tree can be linked to at most one protection set. If a resource URI doesn't inherit any protection set from an upper URI sub-tree, no access restrictions apply. In this case it defaults to http with no authentication.

Command name is: pr

Parameter name	**Value**	**mandatory/ optional**	**minimum supported length**
uri	The URI that indicates the resource or the URI sub-tree to protect	m	128 characters
psid	ID of a protection set	o	32 characters

The SCWS MUST support that this command can be sent even before the resource (referenced by URI) is uploaded to the SCWS.

The "uri" parameter SHALL be interpreted in the following way:

- A string beginning with a '/' character and ending with a '/*' suffix is used for URI sub-tree protection.
- All other strings are used for protecting a resource which exactly matches the given URI "abs_path".

Using URI sub-tree defines kind of "inheritance" rules for protected resources:

- A resource is always protected by the protection set associated to the closest upper URI.
- If no matching protection set is found then the resource MUST be granted free access (i.e. http with no authentication)

If somewhere within the defined sub-tree, there already exist another resource linked to another protection set (subtree with a longer "uri"), this resource MUST remain linked to its protection set, else it MUST be linked to the new one (See example 1 and 2).

If an URI is already linked to a protection set, then this operation is an update. In this case all the resources protected by the previous protection set in the sub-tree MUST be linked to the new protection set. (See example 3)

If the parameter "psid" is not present and the "uri" sub-tree was previously linked to a protection set, this operation is a removal operation. The previous link to a protection set is removed but the protection set of the closest matching URI (up the path) MUST be used instead, (See example 4). If no matching protection set is found then the resource MUST be granted free access (i.e. http with no authentication).

The SCWS SHALL use case-sensitive string comparisons for matching.

The following error codes can be used (see 13.2.2):

- "0002 wrong parameter"
- "0004 referenced data not found" (e.g. psid is not defined)
- "0005 missing parameter or value"

Examples:

Let assumed that we have the protection set "ps1", "ps2" and "ps3" as defined in the previous chapter (see 13.1.6.2) and the following resources present in the SCWS

```
/index.html => free access
/bank/bank.html => free access
/bank/secret/private.html => free access
```

1°) After command "*cmd=pr&uri=/bank/*&psid=ps2*" the resources have the following access condition:

```
/index.html => free access
/bank/bank.html => https access (ps2)
/bank/secret/private.html => https access (ps2)
```

2°) Then after command "*cmd=pr&uri=/bank/secret/*&psid=ps3*" the resources have the following access condition:

```
/index.html => free access
/bank/bank.html => https access (ps2)
/bank/secret/private.html => https access for digest authenticated user of "ps3"
```

3°) Then after command "*cmd=pr&uri=/bank/*&psid=ps1*" the resources have the following access condition:

```
/index.html => free access
/bank/bank.html => http access for basic authenticated user of "ps1"
/bank/secret/private.html => https access for digest authenticated user of "ps3"
```

4°) Then after command "*cmd=pr&uri=/bank/secret/**" the resources have the following access condition:

```
/index.html => free access
/bank/bank.html => http access for user of "ps1" granted by basic
/bank/secret/private.html => => http access for basic authenticated user of "ps1"
```

13.1.6.4 Delete a resource protection set

This command deletes a resource protection set in the server.

Command name is: delps

Parameter name	Value	mandatory/ optional	minimum supported length
psid	Protection Set Identifier	m	32 characters

When a resource protection set is deleted, all resources previously protected by this protection set SHALL inherits the protection set from the upper URI sub-tree, if no matching protection set is found then the resource SHALL be granted a free access (i.e. http access with no authentication). This behaviour is the same as described in the removal operation (no "psid") of the command "pr" (see 13.1.6.3).

This command SHALL NOT delete the users associated to this resource protection set. A separate command is used to delete a user (see 13.1.6.6).

The following error codes can be used (see 13.2.2):

- "0002 wrong parameter"
- "0003 wrong value"
- "0004 referenced data not found" (e.g. protection id is not defined)
- "0005 missing parameter or value".

Example:

If we have:

- A protection set "ps_a" to protect the uri prefix "/a" with basic authentication access by user "u1"
- A protection set "ps_b" to protect the uri prefix "/a/b" with basic authentication access by user "u1" and "u2"
- A resource "/a/b/index.html" (so under "ps_b" control)

If the following command is used:

```
"cmd=delps&psid=ps_b"
```

Then the resource "/a/b/index.html" is now protected by "ps_a". By side effect the user "u2" can no longer access "/a/b/index.html" but is still defined on the SCWS and ready to be added to any protection set.

13.1.6.5 Define user

This command defines a new user in the SCWS. This user definition is unique in the whole server, and can be used in several realms of different protection sets (see 13.1.6.7). A user is a pair of login and password and is used to support basic and digest authentication as specified by [RFC2617].

Command name is: dusr

Parameter name	Value	mandatory/ optional	minimum supported length
user	User's login name	m	32

			characters
pwd	User's password	m	32 characters
user-admin	"true" or "false"	o	

If the specified user already exist the SCWS SHALL update it with its new password value. In this case all existing associations to protection sets remain unchanged. If the user-admin parameter is set to true then user administration commands are authorized (see annex H). If this parameter is not present the default value is "false".

The SCWS MUST support that this command can be sent even before any resource protection set is defined on the card.

The following error codes can be used (see 13.2.2):

- "0002 wrong parameter"
- "0003 wrong value"
- "0005 missing parameter or value"

Example:

```
Cmd=dusr&user=James&pwd=XYZ675
```

13.1.6.6 Delete user

This command deletes a user in the server and removes it from every protection set which reference it.

Command name is: delusr

Parameter name	**Value**	**mandatory/ optional**	**minimum supported length**
user	User's login name	m	32 characters

This command SHALL automatically remove the user from every protection set which reference it and then delete it.

The following error codes can be used (see 13.2.2):

- "0002 wrong parameter"
- "0003 wrong value"
- "0004 referenced data not found" (e.g. user is not defined)
- "0005 missing parameter or value"

Example:

```
cmd=delusr&user=James
```

13.1.6.7 Add user

This command adds a user in a resource protection set that defines a realm.

Command name is: addusr

Parameter name	Value	mandatory/ optional	minimum supported length
user	User's login name	m	32 characters
psid	Protection Set Identifier	m	32 characters

If the given resource protection set doesn't define a realm name with it associated authentication algorithms, the SCWS SHALL return an error code "wrong value: psid =xxxx".

The following error codes can be used (see 13.2.2):

- "0002 wrong parameter"
- "0003 wrong value"
- "0004 referenced data not found" (e.g. psid or user is not defined)
- "0005 missing parameter or value"

Example:

```
cmd=addusr&user=James&psid=ps3
```

13.1.6.8 Remove user

This command removes a user from a resource protection set that defines a realm.

Command name is: rmvusr

Parameter name	Value	mandatory/ optional	minimum supported length
user	User's login name	m	32 characters
psid	Protection Set Identifier	m	32 characters

This command SHALL NOT delete the user even if it no more referenced by any protection set.

The following error codes can be used (see 13.2.2):

- "0002 wrong parameter"
- "0003 wrong value"

- "0004 referenced data not found" (e.g. psid or user is not associated to this psid)
- "0005 missing parameter or value"

Example:

```
cmd=rmusr&user=James&psid=ps3
```

13.1.6.9 Map requests to application

This command is used to map a registered application to incoming request by providing a path triggering condition.

Command name is: map

Parameter name	**Value**	**mandatory/ optional**	**minimum supported length**
appid	Registered application identifier	m	32 characters
path	Application path triggering condition	m	128 characters
intercept	"true" or "false"	o	

The parameter "appid" is the identifier used by the application when it has been registered to the administration agent.

The parameter "intercept" set to "true" marks the application as an interception application (see section 6.2). If it is not present the default value is "false" (i.e. content providing application).

The "path" parameter SHALL be interpreted in the following way:

- A string beginning with a '/' character and ending with a '/*' suffix is used for URL prefix path mapping.
- All other strings are used for exact matches only.

An application can be registered on several different paths.

A content providing application (i.e. not in intercept mode) SHOULD not be mapped on a path used by a static resource as this static content becomes unreachable and useless (applications have priority against static resources).

The SCWS SHALL return a "0003 wrong value" error if the path parameter value was already used by an application which is not in interception mode or if it is reserved by the SCWS (i.e "/SCWS/admin" URL)..

If the full administration protocol is used, the command SHALL be rejected if the card administration agent involved in the exchanges is not allowed to map the requested application (i.e. application not registered to this admin agent).

Please refer to section 6 about application triggering for more information.

The following error codes can be used (see 13.2.2):

- "0002 wrong parameter"
- "0003 wrong value"
- "0004 referenced data not found" (e.g. "appid" is not registered)
- "0005 missing parameter or value"

Examples:

Map the application registered as “application1” on exact request “`/app/application1`”

```
cmd=map&appid=application1&path=/app/application1
```

Map the application registered as “application2” on each request starting with a url prefix “`/app/application2/*`”

```
cmd=map&appid=application2&path=/app/application2/*
```

Map the application registered as “logger” in order to intercept all request to the SCWS.

```
cmd=map&appid=logger&path=/*&intercept=true
```

13.1.6.10 Unmap requests to application

This command is used for un-map request to an application.

Command name is: unmap

Parameter name	**Value**	**mandatory/ optional**	**minimum supported length**
appid	Registered application identifier	m	32 characters
path	Application path triggering condition	o	128 characters

The parameter “appid” is the identifier used by the application when it has been registered to the administration agent.

If “path” is not present all mappings to this application SHALL be removed otherwise only the given “path” SHALL be unmapped.

The following error codes can be used (see 13.2.2):

- “0002 wrong parameter”
- “0003 wrong value”
- “0004 referenced data not found” (e.g. “appid” is not registered or “path” was not previously mapped)
- “0005 missing parameter or value”

Example:

```
cmd=unmap&appid=application1
```

13.2 SCWS Responses to Administration commands

This chapter describes the responses of the SCWS to the received administration commands. Responses are sent using the “Administration protocols” that are described in the chapter13.3. There are two categories of responses, as described in the sub-chapters below.

13.2.1 Responses to HTTP commands

As described above, HTTP commands are used for the administration of the SCWS. These commands are: PUT, DELETE, POST, and GET. For the lightweight administration protocol these commands MUST NOT be pipelined. The SCWS SHALL reply with an HTTP Response for each HTTP Request that was sent by the remote administration application.

If the PUT HTTP command can not be performed because the card memory is full, the HTTP error code 590 SHALL be returned.

13.2.2 Responses to admin commands within the POST command

Special administration commands are sent within the post command as described in [13.1.5]. In the body of the POST command several admin commands can be sent. In order to indicate to the remote administration application which command was successfully executed the Pragma Header MUST be used.

The Pragma Header field in the response SHALL contain the number of the last command that was successfully executed by the SCWS. If a command was not successfully executed all commands after it SHALL be ignored and not executed by the SCWS. The Pragma header field SHALL be the following:

```
Pragma: cmd=CmdNum
```

where `CmdNum` is the last command number, encoded as a decimal integer, that was successfully executed by the SCWS.

If, for example, the POST command contains 14 admin commands and the 13th command was rejected by the SCWS, the SCWS would include the `Pragma: cmd=12` in the HTTP response and ignore the commands that appear after the rejected command. The remote administration application will thus be able to detect that only the first twelve admin commands were successfully executed and resend a new HTTP POST command with some new admin commands. If no admin command was executed, the SCWS SHALL include Pragma: cmd=0 in the HTTP response.
If the HTTP POST request itself failed the SCWS SHALL return the relevant error code (for example malformed HTTP request).

If all internal admin commands were executed successfully then the HTTP response status code SHALL be “204”.

If at least one of the internal admin commands was not executed the HTTP response status code SHALL be “403” and the `Pragma: cmd=CmdNum` SHALL be included in the response as defined above. The body of the response SHALL be the MIME type text/plain containing the error string (without the quotes) of the command that could not be executed. The following result-strings are defined. For some cases the erroneous parameter must be included in the result string, (see examples).

The error result string shall have following format: <**error_code**> [<**error_text**>[: <**error context**>]]

- <**error_code**>: decimal integer with four digits. (Mandatory)
- <**error_text**>: error explicit description (optional)
- <**error_context**>: error textual context (optional, but can be present only if <error_text> is present)

The following errors are defined:

Error code	**Error text**	**Error Context**	**Description**
“0001”	“command not supported”	“*<cmd>*”	An unknown *<cmd>* command. (example: “0001 command not supported: xyz”)
“0002”	“wrong parameter”	“*<param>*”	An unknown parameter *<param>* related to the given command. (example: “0002 wrong parameter: xyz”)

"0003"	"wrong value"	"<*param*>=<*val*>"	A parameter value given in the command is not supported. (example: "0003 wrong value: http=xyz")
"0004"	"referenced data not found"	"<*param*>=<*val*>"	A parameter <*param*> value <*val*> reference an entity which does not exist. (example: "0004 referenced data not found: psid=xyz")
"0005"	"missing parameter or value"	"<*param*>" "<*value*>" "<*cmd*>"	A mandatory parameter/value <*param*><*value*> for the command is missing (example: "0005 missing parameter or value: psid")
"0006"	"no resource"	*NA*	The command could not be fulfilled due to a lack of resource (available memory, static limitations...). This issue may be resolved by freeing the needed resources. (example: "0006 no resource")
"00FF"	"unknown error"	"<*cmd and parameters*>"	Any other error not covered by previous codes (e.g.: Internal error or malformed command) (example: "00FF unknown error: <cmd and parameters>")

The following example shows the response when all admin commands in the HTTP POST request were successfully executed:

```
HTTP/1.1 204 No Content CRLF
CRLF
```

The following example shows the response when only part of the admin commands in the HTTP POST request were successfully executed (in this example only the 12 first commands):

```
HTTP/1.1 403 Forbidden CRLF
Content-Type: text/plain CRLF
Content-Length: 20 CRLF
Pragma: cmd=12
CRLF CRLF


0001 command not supported: sett
```

13.3 Administration protocols

13.3.1 Lightweight Administration Protocol

The lightweight administration protocol can be used for sending short administration commands for setting or changing a small number of configuration parameters for the SCWS. It is suitable for the exchange of a small amount of data between the administration application and the SCWS. For example a "DPS" command (for setting a protection set) may be sent with only one SMS. These commands are sent via SMS as described below.

The SCWS can be administered via current OTA protocols. The OTA message that contains an administration command is formatted according to [TS 31.115] and [TS 31.116] (or [C.S0078] and [C.S0079] for 3GPP2) with authentication, integrity protection and sequence numbering (i.e. card shall reorder each part of Concatenated SMS if needed).

The data payload of the envelope SMS PP Data Download command SHALL be the administration command(s) as described in [13.1]. The TAR of the envelope SMS PP Data Download command must be the TAR of the SCWS card application.

The SCWS application SHALL respond with a proof of receipt, if demanded in the received message as defined in [TS 31.115] (or [C.S0078] for 3GPP2), to indicate that the message was well received, well formatted and was correctly secured. The secure data field shall contain the proof of execution that is the HTTP response for the sent HTTP administration commands.

The total internal allocated size for incoming OTA messages, that contain administration commands, SHALL be at least 512 bytes. The total internal allocated size of outgoing OTA messages that contain the response to the administration commands SHALL be at least 512 bytes.

13.3.2 Full Administration Protocol

The full administration protocol is used for doing a full blown administration of the SCWS which may include uploading new pages, deleting pages and changing configuration parameters for the SCWS. It is suitable for the exchange of a large amount of data between the administration application and the SCWS.

The full administration protocol can also be used for securely exchanging or updating data with applications registered to the SCWS as described in 13.1.3. This may be useful for securely updating data used by these applications or for securely retrieving data from them.

The full administration protocol enables the use of a standard web server as a remote administration server implementation. The full administration protocol (and its card administration agents) has the following characteristics:

- Based on a reliable and efficient end to end connected transport protocol: TCP/IP (eventually over BIP)
- Based on an industry standard security layer: TLS
- A card administration agent is a real HTTP 1.1 Client and is in charge to manage connection establishment between a remote administration server and the SCWS.
- Card administration agents are able to encapsulate and transparently transport any HTTP exchange between a remote administration server and the SCWS.
- Card administration agents are responsible of retry and reconnection management in case of communication break down.
- Card administration agents can be triggered either by external events (e.g. SMS) or by internal events (internally generated by the card) and initiate a connection to a remote administration server.
- Each directory (and its content) under the root of the SCWS is owned by a card administration agent.
- A card administration agent can only administrate its own URIs.

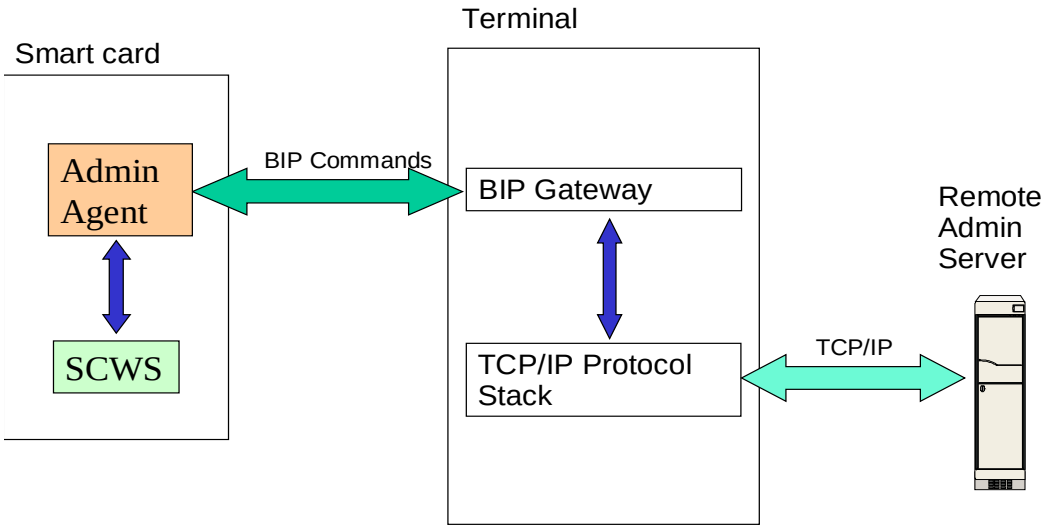

Figure 1: Remote SCWS administration using BIP

13.3.2.1 Administration session flow and behaviour using BIP

The Full Administration Protocol SHALL be implemented with BIP client mode as defined in [TS 102 223]. For this purpose the smart card SHALL include one or several administration agents that SHALL connect to a remote administration server by opening a BIP TCP channel in client mode, as defined in [TS 102 223]. An administration agent SHALL perform the following tasks:

1. Open a BIP channel in TCP client mode with the relevant remote administration server.
2. Use PSK-TLS over this TCP channel to enable mutual authentication, confidentiality and integrity (using one of the cipher suites that are defined in chapter 11.1.1). How shared keys are provisioned in both sides is beyond the scope of this specification.
3. After the TLS communication channel is established the card administration agent SHALL send an HTTP POST command (as defined in [13.3.2.6]) in order to get the first admin command (as defined in [13.1]):
4. When receiving the HTTP POST from the card administration agent the remote administration server SHALL send an HTTP response (as defined in [13.3.2.7]) which encapsulates an HTTP administration command dedicated to the SCWS itself (as defined in [13.1]).
5. When receiving the HTTP response for the above HTTP POST command the card administration agent SHALL forward it to the SCWS.
6. The SCWS SHALL consider this channel as authenticated by the card administration agent. If the card administration agent is allowed to access the URI involved in the command it SHALL process the delivered administration command.
7. After processing the delivered administration command the SCWS SHALL deliver the HTTP response back to the card administration agent.

8. The card administration agent SHALL submit the HTTP response from the SCWS in a new POST request to the remote administration server over the TLS secure channel.

9. The remote administration server SHALL send the next administration command to the card administration agent over the TLS secure channel or send a final response requesting the end of the remote administration session in the POST response.

10. If the card administration agent receives a final response from the remote administration server, it SHALL close the TLS channel and afterwards close the BIP channel.

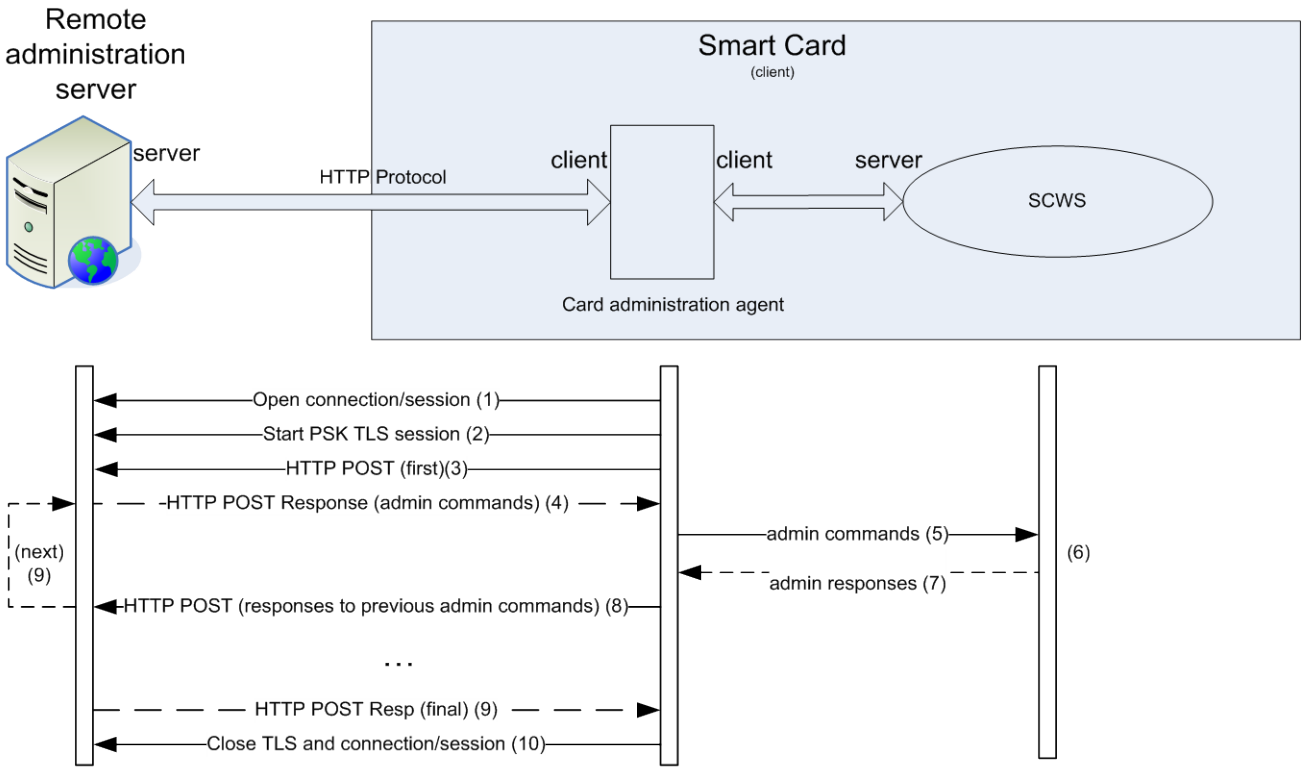

Figure 2: Administration session flow

13.3.2.2 Administration session flow and behaviour using TCP/IP

If the smart card implements a TCP/IP stack and can establish a direct TCP/IP connection to the remote administration server (without using BIP) the session flow and behaviour is the same as described in [13.3.2.1] but with a direct TCP/IP connection.

13.3.2.3 Admin Server Settings

To connect to the admin server the smart card Admin Agent can retrieve needed parameters from a resource (e.g. a file or data object), accessible via a URL, and managed by the card issuer (or delegated authorized entity).

A configuration resource shall have the following characteristics:

- The resource “Content-type” SHALL be “application/vnd.oma-scws-config”
- The resource body SHALL be an octet stream with a complete “Admin Agent configuration parameters” structure (see 13.3.2.9.3)

- The resource SHALL be manageable with administrative commands like PUT request.

Several configuration resources could be present in the SCWS but one of them must be set as the default one.

A configuration resource can be created or overwritten by the admin server with a PUT request. The default resource URL SHALL be " /scws-admin-agent/default-resource".

The following example illustrates how a PUT command is used for updating a configuration resource identified by the URL "/scws-admin-agent/default-resource":

```
PUT /scws-admin-agent/default-resource HTTP/1.1 CRLF
Host: anything CRLF
Content-Type: application/vnd.oma-scws-config CRLF
Content-Length: xxxx CRLF
CRLF
83 xx //Admin Agent configuration parameters

  84 xx //Connection parameters (with all parameters needed for open the remote
connection)

  85 xx //Security parameters ( with PSK identity and Card Key-Identifier)

  86 xx //Retry Policy parameters (with retry counter and retry delay)

    87 xx // retry failure SMS MO (with TP Destination Address)

  89 xx //Agent HTTP POST parameters

    8A xx //Administration Host parameter

    8B xx //Agent ID parameter

    8C xx //Administration URI parameter
```

13.3.2.4 Remote triggering of an administration session

To start an administration session an administration agent in the smart card needs to be triggered in order to start the administration session. A remote administration server, or a delegated authorized entity, SHALL trigger the administration session by sending a secure SMS to the smart card to tell an administration agent to start an administration session.

The OTA message that triggers an administration session is formatted according to [TS 31.115] and [TS 31.116] (or [C.S0078] [C.S0079] for 3GPP2) with authentication, integrity protection and sequence numbering (i.e. card shall reorder each part of Concatenated SMS if needed). The TAR of the envelope SMS PP Data Download command must be the TAR of the administration agent in the smart card.

The administration agent SHOULD be able to register several administration requests.

The administration agent SHALL return SIM BUSY status (9300) if it is not able to immediately store the administration request. This must be done prior to verifying [TS 31.115] security in to order avoid security counter de-synchronisation.

The administration agent will respond with a proof of receipt, if demanded in the received message as defined in [TS 31.115], (or [C.S0078] for 3GPP2) to indicate that the message was well received, well formatted and was correctly secured.

The Remote Administration Server SHALL include a Remote Administration Request data structure (see 13.3.2.9.1) in the data payload of the envelope SMS PP Data Download.

If a parameter (e.g. a TLV) is present in the configuration resource and also in the data payload of the envelope SMS PP Data Download then the parameter in the SMS SHALL be used.

If any parameter (e.g. a TLV) is not present in the data payload of the envelope SMS PP Data Download the card Admin Agent must use the parameter value from the indicated configuration resource. If it is not present in the indicated configuration resource then the Admin Agent must use the parameter value in the default configuration resource.

Upon the reception of the above triggering command the administration agent SHALL start the administration session as described in [13.3.2.1].

The following are examples for the payload of a triggering SMS:

Use default configuration resource:

81 00 *//Remote Administration Request*

Use default configuration resource but with a different admin URI: "/otherurl"

81 0F *//Remote Administration Request*

83 0D *//Admin Agent configuration parameters*

89 0B *// Agent HTTP POST parameters*

8C 09 2F 6F 74 68 65 72 75 72 6C *//Administration URI parameter*

Use configuration resource "/scws-admin-agent/config-resource " but with a different admin URI: "/otherurl"

81 27 *//Remote Administration Request*

82 20 73 63 77 73 2D 61 64 6D 69 6E 2D 61 67 65 6E 74 2F 63 6F 6E 66 69 67 2D 72 65 73 6F 75 72 63 65 *// Configuration resource URL:* "/scws-admin-agent/config-resource "

83 0D *//Admin Agent configuration parameters*

89 0A *// Agent HTTP POST parameters***8C** 09 2F 6F 74 68 65 72 75 72 6C *//Administration URI parameter:* "/otherurl"

13.3.2.5 Auto and local triggering of an administration session

The admin session can be triggered by a local event (time based, new IMEI etc.) or by another card application. In this case the parameters to use to connect to the remote administration server shall be the default ones or parameters read from a configuration resource linked to the event that triggered the admin session.

13.3.2.6 HTTP POST request of card administration agent

The POST request is used by the card administration agent to fetch administrative commands for the SCWS and to transmit the result from the preceding administrative commands.

13.3.2.6.1 Request Format

The POST request SHALL have the following format:

```
POST <URI> HTTP/1.1 CRLF
Host: <Administration Host> CRLF
X-Admin-Protocol: oma-scws-admin-agent/<scws-version> CRLF
X-Admin-From: <Agent ID (as defined in triggering event)> CRLF
[X-Admin-Resume: true]
[Content-Type: application/vnd.oma-scws-http-response CRLF]
```

```
[Content-Length: xxxx CRLF] or [Transfer-Encoding: chunked CRLF]
CRLF
[body-with-previous-scws-commands-responses]
```

- If a X-Admin-Next-URI response field is present in the previous response (see description in [13.3.2.7]) then the card administration agent SHALL use it. Otherwise the last used URI MUST be used. The First URI to be used is defined by the triggering event. The card administration agent MUST be able to handle URI with a length of 1024 bytes. The remote administration server MAY use query parameters in the URI for session management purpose.
- The "Host" value SHALL be the Administration Host parameter defined by the triggering event (see 13.3.2.7).
- The card administration agent SHALL use the "X-Admin-Protocol" request headers with the given version to enable backward compatibility with evolution of this standard, current scws-version is 1.1.1
    - A card-administration agent that sends a request message that includes scws-version of "1.1.1" MUST at least comply with the mandatory features of this specification. Applications that are at least complying with the mandatory features of this specification SHOULD use a scws-version of "1.1.1" in their messages.
    - Compatibility with previous version
        - SCWS 1.1.1 card-administration-agent MUST accept message from remote-admin-server with a lower version number.
- The card administration agent SHALL use the "X-Admin-From" request headers with the "Agent ID" defined by the triggering event (see 13.3.2.9.11).
- If this session is resumed from a previous interrupted session, the card administration agent SHALL use the "`X-Admin-Resume`" extension-header with the value "true" in the first POST request of the resumed session. The "`X-Admin-Resume`" extension-header MUST not be used in the following POST requests.
- If a response from a previous admin command is to be sent, the card administration agent SHALL forward the SCWS response using:
    - "Content-Type" entity header with the value "application/vnd.oma-scws-http-response".
    - "Content-Length" entity header with the exact length of the body in bytes or "Transfer-Encoding" general header with the value "chunked" (see [HTTP/1.1]). The card administration agent may use the chunk encoding if it is not able to compute the total length of the SCWS response.
    - A body with the complete response of the previous administration command. The chuncked Transfer-Encoding may be used.

In order to keep the backward compatibility with former versions of SCWS (e.g. scws-version is 1.0 or 1.1), the card administration agent SHOULD manage requests with the following header "From", "User-Agent", "SCWS-Resume".

13.3.2.6.2 Examples:

First request of a new session:

```
POST /downloadmanager/meteo?cmd=1 HTTP/1.1 CRLF
Host: 172.96.0.1 CRLF
X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF
X-Admin-From: 8939010012751002010 CRLF
CRLF
```

Next request with preceding response:

```
POST /downloadmanager/meteo?cmd=5 HTTP/1.1 CRLF
Host: 172.96.0.1 CRLF
X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF
```

```
X-Admin-From: 8949020012751002010 CRLF

Content-Type: application/vnd.oma-scws-http-response CRLF
Content-Length: 27 CRLF
CRLF
HTTP/1.1 204 NO CONTENT CRLF
CRLF
```

Next request with preceding response with chunk encoding:

```
POST /downloadmanager/meteo?cmd=5 HTTP/1.1 CRLF
Host: 172.96.0.1CRLF
X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF
X-Admin-From: 8991200012751002010 CRLF

Content-Type: application/vnd.oma-scws-http-response CRLF
Transfer-Encoding: chunked CRLF
CRLF
B CRLF                                  // first chunk
HTTP/1.1 20
10 CRLF                                 // second chunk size CRLF
4 NO CONTENT CRLF
CRLF
0 CRLF                                  // last chunk
CRLF
```

First request of a resumed session:

```
POST /downloadmanager/meteo?cmd=12 HTTP/1.1 CRLF
Host: 172.96.0.1 CRLF
X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF
X-Admin-From: 8991200012751002010 CRLF
X-Admin-Resume: true CRLF
CRLF
```

13.3.2.7 HTTP POST response of remote administration server

The POST response is used by the remote administration server to transmit the next administrative commands to the SCWS through the administration agent and possibly to inform about the next URI that must be used to request the following admin command.

13.3.2.7.1 Response Format

The POST response SHALL have the following format:

```
HTTP/1.1 200 OK CRLF [or HTTP/1.1 204 No Content CRLF]
X-Admin-Protocol: oma-scws-remote-admin/<scws-version> CRLF
[X-Admin-Next-URI: <next-URI> CRLF]
[Content-Type: application/vnd.oma-scws-http-request CRLF]
[Content-Length: xxxx CRLF] or [Transfer-Encoding: chunked CRLF]
CRLF
[body-with-scws-command-request]
```

- The remote administration server SHALL use a successful status (200 OK) if the response contains a body else it SHALL use the status 204 (No Content) if no entity-body is send.

- The remote administration server SHALL use the "X-Admin-Protocol" request headers with the given version to enable backward compatibility with evolution of this standard, current scws-version is 1.1.1.
    - A remote-administration-server that sends a response message that includes scws-version of "1.1.1" MUST at least comply with the mandatory features of this specification. Applications that are at least complying with the mandatory features of this specification SHOULD use a scws-version of "1.1.1" in their messages.
    - Compatibility with previous version:
        - SCWS 1.1.1 Remote administration server MUST respond with a message in the same scws-version used by the card-administration-agent and adapt response body with scws-command-request compatible with the scws-version.
- If a "`X-Admin--Next-URI`" extension header is present in the response, the card administration agent SHALL use the given URI in the next POST request. If this header field is present and if the body is empty then the card administration agent SHALL generate a POST request to the given URI without forwarding anything to the SCWS. The URI SHALL respect the same constraints as the initial triggering URI (abs_path,query,length,…) and SHALL be available on the same server where the connection has been initiated (the session communication channel remains the same).
- If no "`X-Admin-Next-URI`" extension header is present and if the body is empty (i.e. 204), the administration session shall be successfully closed.
- If a the remote administration server has remaining administration commands to forward to the SCWS it SHALL use an entity body with:
    - "Content-Type" entity header with the value "application/vnd.oma-scws-http-request".
    - "Content-Length" entity header with the exact length of the body in bytes or "Transfer-Encoding" general header with the value "chunked" (see [HTTP/1.1]). The remote administration server may use the chunk encoding if it is not able to compute the total length of the SCWS response.
    - A body with HTTP administrative command to forward to the SCWS. The chunked Transfer-Encoding may be used.

In order to keep the backward compatibility with former versions of SCWS (e.g. scws-version is 1.0 or 1.1), the remote administration server SHOULD manage responses with the following header "User-Agent" and "SCWS-Next-URI".

13.3.2.7.2 Examples:

Response with an embedded PUT request for the SCWS:

```
HTTP/1.1 200 OK CRLF
X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF
Content-Type: application/vnd.oma-scws-http-request CRLF
Content-Length: 110 CRLF
CRLF
PUT /index.xhtml HTTP/1.1 CRLF
Host: anything CRLF
Content-Type: text/html CRLF
Content-Length: 18 CRLF
CRLF
<html>Hello</html>
```

Response using chunk encoding with an embedded PUT request for the SCWS (note: this is a didactical example, there is no need to use chunk for such small response):

```
HTTP/1.1 200 OK CRLF
X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF
Content-Type: application/vnd.oma-scws-http-request CRLF
Transfer-Encoding: chunked CRLF
```

```
CRLF
24 ;chunk size CRLF
PUT /index.xhtml HTTP/1.1 CRLF
Host: any CRLF
48 ;chunk size CRLF
thing CRLF
Content-Type: text/html CRLF
Content-Length: 18 CRLF
CRLF
<html>Hello</html> CRLF
0 ; last chunk CRLF
CRLF
```

Response with an embedded GET request for the SCWS and also with a next URI (incrementing query):

```
HTTP/1.1 200 OK CRLF
X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF
X-Admin-Next-URI: /downloadmanager/meteo?cmd=13 CRLF
Content-Type: /application/vnd.oma-scws-http-request CRLF
Content-Length: 43 CRLF
CRLF
GET /page1.html HTTP/1.1 CRLF
Host: anything CRLF
CRLF
```

Response with no body but a next URI:

```
HTTP/1.1 204 No Content CRLF
X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF
X-Admin-Next-URI: /otherdownloadmanager?cmd=1  CRLF
CRLF
```

Final response which close the administration session:

```
HTTP/1.1 204 No Content CRLF
X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF
CRLF
```

13.3.2.8 Retry management

As soon as an administration session has been triggered and accepted by the card administration agent, this agent is responsible for the connection to the remote administration server and for the accomplishment of the session.

This means that if a communication error, a power down or a network coverage loss occurs during the processing of the administration flow (see description in [13.3.2.1]) the administration agent should try to reconnect according to a card issuer specific retry policy.

The retry policy may include the following:

- An end condition (e.g. number of retries) to be used to avoid network congestion by stale or inconsistent remote administration request.
- A time or counter or an event based retry policy if the BIP or TCP/IP connection attempt fails (like network congestion).

If the TLS session establishment fails for security/authorisation reason the administration session SHALL be immediately discarded.

If a communication breakdown occurs after valid requests have been exchanged between the card administration agent and the remote administration server, the card administration agent SHALL always use the resume mode (see description in [13.3.2.6]).

The overall behaviour SHALL be based on the following rules:

- The Card Administration Agent will make several attempts for resuming the administration session (e.g. BIP or TCP/IP connection and opening an HTTP communication). The waiting period between two attempts and the maximum number of attempt is specified by the retry policy (e.g. triggering event 13.3.2.9.6).
- If the communication is re-established, the Card Administration Agent will try to resume the HTTP dialog by navigating to the last URL of this administration session (see hereafter for the detail for this URL navigation).
- At the opposite, if the maximum number of attempts has been reached the administration session request is then abandoned.

The retry scheme is shown in the following figure:

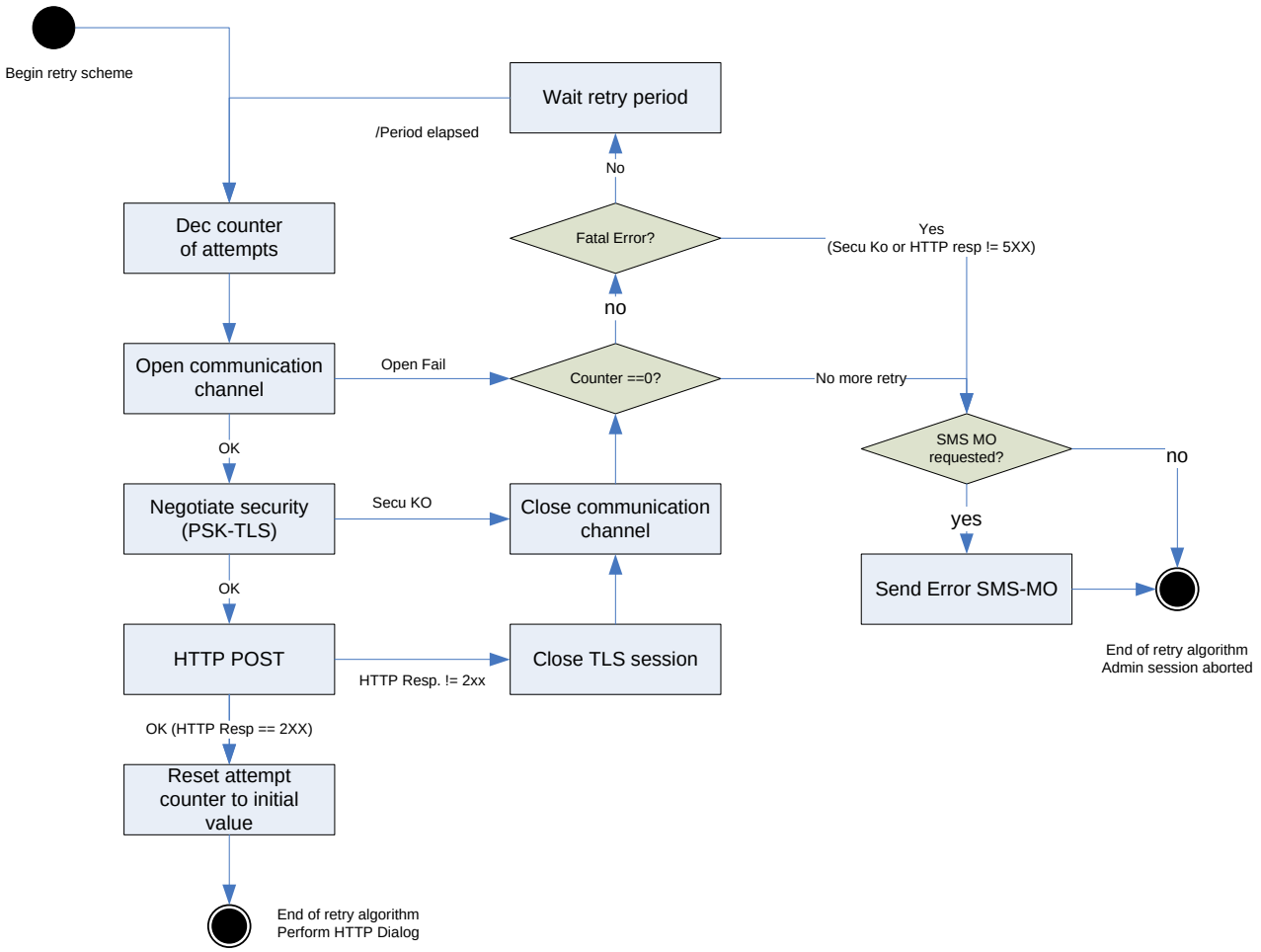

Figure 3: Retry algorithm

Where:

- 5xx HTTP responses are considered as errors (the connection is closed), but the retry schema should be applied (this might be a temporary error due to the remote administration server shutting down).

- A 2xx HTTP response received from the administration server means that the admin session is working. Retry counter should be reset to its initial value.
- Other HTTP response codes or errors during the security negotiation are considered to be fatal errors (no retry is performed).

If several administration requests are registered and need a retry, the card administration agent SHOULD handle these retries independently of each others (e.g. not block the other retry attempts if the current one is not successful).

If the administration session is abandoned (fatal error or maximum number of retry attempts is reached) and if the triggering event has required a failure report (see 13.3.2.9.7) than an SMS-MO SHALL be emitted.

The failure SMS-MO report SHALL be formatted according to [TS 31.115] (or [C.S0078] for 3GPP2)

The data payload in the failure SMS-MO report SHALL comply with the "Administration Failure report" format (see 13.3.2.9.8).

13.3.2.9 Tag descriptions

All the following Tags are defined using the same coding as the [TS 101 220] COMPREHENSION-TLV data objects and must be found in the data payload of the envelope SMS PP Data Download, in the indicated configuration resource or in the default configuration resource as described in [13.3.2.4].

Each TLV element may be found in the corresponding structure according to the following rules:

The M status indicates a mandatory entity in the TLV in which it is found (e.g. the tag of TLV, the retry counter of retry policy parameter)

The O status indicates an optional entity e.g. it could be absent from all configuration resources and also from the data payload of the envelope SMS PP Data Download.

The C status indicates a conditional entity e.g. it must be present at least in one of the configuration resources or in the data payload of the envelope SMS PP Data Download.

These Tags SHALL be provided in the given order in the description chapter.

13.3.2.9.1 Remote Administration request

The Remote administration request TLV is present in an SMS that triggers a remote administration session

Description	M/O/C	Length
Remote administration request tag	M	1
Length (A+B)	M	1,2 or 3
Configuration Resource URL	O	A
Admin Agent configuration parameters	C	B

- Configuration Resource URL:
  If not present the SCWS admin agent shall use default configuration resource defined by the card issuer.
- Admin Agent configuration parameters:
  These parameters have higher priority than the corresponding parameters in the used configuration resource.

13.3.2.9.2 Configuration Resource URL parameter

The Configuration Resource URL parameter TLV defines the configuration resource to use.

Description	M/O/C	Length
Configuration Resource URL parameter tag	M	1
Length (A)	M	1,2 or 3

URL	M	A

- URL:
  Absolute path URL of a configuration resource inside the SCWS (see [13.3.2.3]).
  The SCWS agent SHALL support URL Length of at least 1024 bytes.

13.3.2.9.3 Admin Agent configuration parameters

The Admin Agent configuration parameters TLV is used to configure an administration session between the admin agent and a remote administration server.

Description	M/O/C	Length
Admin Agent configuration parameters tag	M	1
Length (A+B+C+D)	M	1,2 or 3
Connection parameters	C	A
Security parameters	C	B
Retry Policy parameters	C	C
Agent HTTP POST Parameters	C	D

13.3.2.9.4 Connection parameters

The connection parameters TLV embeds all the needed parameters to establish a point to point TCP connection between the Administration Agent and the Remote administration server.

Description	M/O/C	Length
Connection parameters tag	M	1
Length (A)	M	1 or 2
Set of any comprehension TLV needed to open the TCP connection.	M	A

If the connection between the Admin Agent and the remote administration server is done over BIP, once merged with the configuration resource, the data SHALL contain all needed COMPREHENSION-TLV data objects that are defined for OPEN CHANNEL in [TS 102 223].

13.3.2.9.5 Security parameters

The security parameters TLV is used for configuring the TLS layer.

Description	M/O/C	Length
Security parameters tag	M	1
Length (1+A+1+B)	M	1, 2 or 3
Length of PSK-Identity	M	1
PSK-Identity	M	A
Length of Key-Identifier	M	1
Card Key-Identifier	M	B

- PSK-Identity:
  The PSK identity is an opaque string defined by [PSK-TLS]. It is used by the remote server to indicate the PSK identity to be used during the TLS session.
  The SCWS admin agent SHALL support a PSK Identity length of at least 32 bytes.

- Card Key-Identifier:
  The Card Key Identifier is an opaque string that is used internally in the smart card to find the key value to be used

in the TLS session
The SCWS admin agent SHALL support a Card Key-Identifier length of at least 32 bytes.

13.3.2.9.6 Retry policy parameters

The retry policy parameters TLV is used by the administration agent in case of administration session failure.

Description	M/O/C	Length
Retry policy tag	M	1
Length (2+5+A)	M	1
Retry counter	M	2
Retry waiting delay	M	5
Retry report failure SMS-MO	O	A

- Retry counter:
  Unsigned short initial value of the retry counter used by the retry policy.

- Retry waiting delay:
  Definition of the time to wait between two retries. This parameter is in the same format as the "timer" Comprehension TLV definition in [TS 102 223].
  Due to the clock-less and removable design of the smart card, the remote administration server shall only consider it as the minimum time the card administration agent has to wait and not the exact time when the session will be resumed.

13.3.2.9.7 Retry failure report SMS-MO

If this TLV is present then the administration agent SHALL send an SMS-MO in case of an abort of an administration request (as defined in section 13.3.2.8).

Description	M/O/C	Length
Retry failure report SMS-MO tag	M	1
Length (A+B+C)	M	1
TP-Destination Address parameter	O	A
Service Center Address	O	B
Alpha Identifier	O	C

- TP-Destination Address parameter (see 13.3.2.9.13):
  If no TP-Destination Address is present in this TLV, it SHALL be extracted from the TP-Originating Address of the trigger SMS.

- Service Center Address:
  The Service Center Address. This parameter is in the same format as the "address" Comprehension TLV definition in [TS 102 223].

- Alpha ID:
  The Alpha identifier to use during the administration failure report SMS-MO emit. This parameter is in the same format as the "Alpha Identifier" Comprehension TLV definition in [TS 102 223].

All the other parameters (as defined in [TS 23.040]), which are needed to submit the Retry failure report SMS-MO SHALL be extracted from the trigger SMS.

13.3.2.9.8 Administration failure report

The administration failure report TLV is present in the SMS-MO send to the administration server in case of an abort of an administration request (as defined in section 13.3.2.8).

Description	M/O/C	Length

Admin agent failure report Tag	M	1
Length (1+A+B)	M	1
Error code	M	1
Failure Report Contextual message.	O	A
Current Administration URI	O	B

- Error Code:
  Specifies the layer where the error occurred when the retry limit was reached (so only the last error is returned):
  - 0x01 : Communication establishment failed (e.g. Open Channel Error)
  - 0x02 : TLS Error
  - 0x03 : Connection lost (e.g. BIP Channel Error)
  - 0x04 : Remote administration server error

Failure Report Contextual message (see 13.3.2.9.14):
Contextual message gives some additional diagnostic information depending on the type of the error.

Current Administration URI:
Gives the current administration address used when the retry limit was reached. This TLV has the same format as the Administration URI parameter (see 13.3.2.9.12).

In order to avoid concatenation of SMS-MO the admin agent may truncate the value of the URI.

Example:

An administration session which fails in "404 Not Found" during the access of a very long Administration URI "/administration/remote/server/verylonguri?param=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxyyyyy" could generate the following truncated failure report:

08 7D *// Admin agent failure report*

04 // Remote administration server error

0E 03 34 30 34 *// Contextual message "404"*

0C 75 *//Administration URI parameter truncated to 117 bytes*
2F 61 64 6D 69 6E 69 73 74 72 61 74 69 6F 6E 2F 72 65 6D 6F 74 65 2F 73 65 72 76 65 72 2F 76 65 72 79 6C
6F 6E 67 75 72 69 3F 70 61 72 61 6D 3D 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78
78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78
78 78 78 78 78 78 78 78 78

13.3.2.9.9 Agent HTTP POST parameters

The Agent HTTP POST parameters TLV defines all parameters needed to send a POST request to the admin server.

Description	M/O/C	Length
Agent HTTP POST parameters tag	M	1
Length (A+B+C)	M	1,2 or 3
Administration Host parameter	C	A
Agent ID parameter	C	B
Administration URI parameter	C	C

13.3.2.9.10 Administration Host parameter

The Administration Host parameter TLV defines the Host header value to be used in the Admin Agent POST request (see 13.3.2.6.1).

Description	M/O/C	Length
Administration Host parameter Tag	M	1
Length (A)	M	1
Host	M	A

- Host:
  Host header value (as defined by [RFC 2616]) with the name of the administration server.
  The SCWS admin agent SHALL support a Host length of at least 32 bytes.

13.3.2.9.11 Agent ID parameter

The Agent Id parameter TLV defines the value to be used in the 'From' header in a POST request to an admin server (see 13.3.2.6.1).

Description	M/O/C	Length
Agent ID parameter Tag	M	1
Length (A)	M	1
From ID	M	A

- From ID:
  From header value (as defined by [RFC 2616]) with the ID of the admin Agent.
  The SCWS admin agent SHALL support a Host length of at least 32 bytes.

A good practice is to use the smart card ICCID to allow immediate identification of the card by the remote administration server. If the ICCID is used, it SHALL be formatted as a 19 or 20 byte long string (decimal digits, with check digit, without padding character). Note: the nibbles of the value extracted from the EF-ICCID [TS 102 221] are swapped)

13.3.2.9.12 Administration URI parameter

The Administration URI parameter TLV defines the URI value to be used in a POST request to an admin server (see 13.3.2.6.1).

Description	M/O/C	Length
Administration URI parameter Tag	M	1
Length (A)	M	1,2 or 3
URI	M	A

- URI:
  The URI value to set in the Admin Agent Post request.
  This URI SHALL be an absolute path (as defined by [RFC 2616]) and may contain query parameters.
  The SCWS admin agent SHALL support a URI length of at least 1024 bytes.

13.3.2.9.13 TP-Destination Address parameter

The TP-Destination Address parameter TLV defines the TP-Destination Address value to be used in SMS-MO in case of an abort of an administration request (as defined in section 13.3.2.8).

Description	M/O/C	Length
TP-Destination Address parameter Tag	M	1
Length (A)	M	1
TP-Destination Address	M	A

- The TP-Destination Address value as defined in [TS 23.040].

13.3.2.9.14 Failure Report Contextual message

The Failure Report Contextual message Tag TLV defines additional diagnostic information to be used in SMS-MO in case of an abort of an administration request (as defined in section 13.3.2.8).

Description	M/O/C	Length
Failure Report Contextual message Tag	M	1
Length (A)	M	1
Contextual message	M	A

Contextual message gives some additional diagnostic information depending on the type of the error (see 13.3.2.9.8):

- If error code = 0x01 : the Terminal Response of the Open Channel Command
- If error code = 0x02 : None
- If error code = 0x03 : Channel Status

If error code = 0x04: HTTP Error code returned by the remote administration server.

13.3.2.10 Tag values

Comprehension-TLV TAG	Description	Section
'81h'	Remote administration request tag	13.3.2.9.1
'82h'	Configuration resource URL parameter tag	13.3.2.9.2
'83h'	Admin agent configuration parameters tag	13.3.2.9.3
'84h'	Connection parameters tag	13.3.2.9.4
'85h'	Security parameters tag	13.3.2.9.5
'86h'	Retry policy parameters tag	13.3.2.9.6
'87h'	Retry failure report SMS-MO tag	13.3.2.9.7
'88h'	Admin agent failure report tag	13.3.2.9.8
'89h'	Agent HTTP POST parameters tag	13.3.2.9.9
'8Ah'	Administration Host parameter tag	13.3.2.9.10
'8Bh'	Agent ID parameter tag	13.3.2.9.11
'8Ch'	Administration URI parameter tag	13.3.2.9.12
'8Dh'	TP-Destination Address parameter tag	13.3.2.9.13
'8Eh'	Failure Report Contextual message	13.3.2.9.14

To keep the backward compatibility with remote admin servers implementing a lower version of the SCWS, a card admin agent should accept a remote administration triggering message containing Comprehension TLV Tags with 0xh values.

To keep the backward compatibility with card implementing a lower version of the SCWS, a remote admin server should accept administration session failure report SMS-MO containing Comprehension TLV Tag with 0xh values.

Appendix A. Change History (Informative)

A.1 Approved Version 1.1 History

Reference	Date	Description
Approved Version OMA-TS-Smartcard-Web-Server-V1_1	12 May 2009	Status changed to Approved by TP TP ref # OMA-TP-2009-0190- INP_Smartcard_Web_Server_V1_1_ERP_for_Final_Approval
Approved Version OMA-TS-Smartcard-Web-Server-V1_1_1	10 Sep 2010	Status changed to Approved by TP TP ref# OMA-TP-2010-0410-INP_SCWS_V1_1_1_ERP_for_Notification
Approved Version OMA-TS-Smartcard-Web-Server-V1_1_2	27 Sep 2012	Status changed to Approved by TP TP ref# OMA-TP-2012-0364-INP_SCWS_V1_1_2_ERP_for_Notification

Appendix B. Static Conformance Requirements (Normative)

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

B.1 SCR for SCWS Admin Client

Item	Function	Reference	Status	Requirement
SCWS-AC-001	Using BIP or TCP/IP	Section 13.3.2	M	SCWS-AC-002 OR SCWS-AC-003
SCWS-AC-002	Administration session over BIP	Section 13.3.2.1	O	
SCWS-AC-003	Administration session over TCP/IP	Section 13.3.2.2	O	
SCWS-AC-004	Admin-server settings: Default configuration resources	Section 13.3.2.3	M	
SCWS-AC-005	Triggering of the administration client by an SMS MO	Section 13.3.2.4	M	
SCWS-AC-006	Trigger the administration session by a local event	13.3.2.5	O	
SCWS-AC-007	Verify that all mandatory fields of HTTP POST request of card administration agent are present and valid	Section 13.3.2.6.1	M	
SCWS-AC-008	Retry management Resume mode	Section 13.3.2.8	M	
SCWS-AC-009	Retry management SMS-MO emitted if the administration session is abandoned	Section 13.3.2.8	M	
SCWS-AC-010	Support for all tags that are used for remote administration and retry management	Section 13.3.2.9	M	
SCWS-AC-011	Support for PSK-TLS	Section 11.1.1	M	
SCWS-AC-012	Support for Fragment Negotiation in TLS	Section 11.1.3	M	
SCWS-AC-013	Support for each authorized entities to be able to control what content and which smart card applications can be accessed under a given URI.	Section 13.3.2	M	
SCWS-AC-014	Support for TLS_PSK_WITH_3DES_EDE_CBC_SHA cipher suite	11.1.3	M	
SCWS-AC-015	Support for TLS_PSK_WITH_AES_128_CBC_SHA cipher suite	11.1.3	M	
SCWS-AC-016	Support for TLS_PSK_WITH_NULL_SHA cipher suite	11.1.3	M	
SCWS-AC-017	Compliance with HTTP 1.1	Section 13.3.2	M	

B.2 SCR for Remote SCWS Admin Server

Item	Function	Reference	Status	Requirement
SCWS-AS-001	Implement the “SCWS full administration protocol”	Section 13.3.2	M	
SCWS-AS-002	Support for PSK-TLS	Section	M	

Item	Function	Reference	Status	Requirement
		11.1.1		
SCWS-AS-003	Support for Fragment Negotiation in TLS	Section 11.1.3	M	
SCWS-AS-004	Trigger the administration session by sending a secure SMS to the smart card	Section 13.3.2.4	M	
SCWS-AS-005	Verify that all mandatory fields of HTTP POST response of remote administration server are present and valid	Section 13.3.2.7.1	M	
SCWS-AS-006	Support for Lightweight administration protocol	Section 13.3.1	M	
SCWS-AS-007	Support for TLS_PSK_WITH_3DES_EDE_CBC_SHA cipher suite	Section 11.1.3	M	
SCWS-AS-008	Support for TLS_PSK_WITH_AES_128_CBC_SHA cipher suite	Section 11.1.3	M	
SCWS-AS-009	Support for TLS_PSK_WITH_NULL_SHA cipher suite	Section 11.1.3	M	

B.3 SCR for SCWS Client

Item	Function	Reference	Status	Requirement
SCWS-C-001	Use the SCWS URL to access the SCWS	Section 5 Section 7	M	
SCWS-C-002	Implement the TLS protocol	Section 11.1	O	
SCWS-C-003	Implement the PSK-TLS protocol	Section 11.1.1	O	
SCWS-C-004	Implement HTTP Basic authentication	Section 10	O	
SCWS-C-005	Implement HTTP Digest authentication	Section 10	O	

B.4 SCR for SCWS Server

Item	Function	Reference	Status	Requirement
SCWS-S-001	SCWS URL with a length of at least 1024 characters is supported	Section 5 Section 7	M	
SCWS-S-002	Support for Dynamic Content	Section 6	M	
SCWS-S-003	Support for local transport protocols with the client device	Section 7 Section 8	M	SCWS-S-004 OR SCWS-S-005
SCWS-S-004	Support for the BIP transport protocol	Section 7.1 Section 8.1	O	SCWS-S-006
SCWS-S-005	Support for TCP/IP transport protocol	Section 7.2 Section 8.2	O	
SCWS-S-006	Support for local BIP channels with the connecting device	Section 8.1.1	O	
SCWS-S-007	Support for Concurrency with other CAT application	Section 8.1.1.1	M	
SCWS-S-008	Compliance with HTTP 1.1	Section 9	M	
SCWS-S-009	Implement HTTP GET Request	Section 9.2	M	
SCWS-S-010	Implement HTTP HEAD Request	Section 9.2	M	
SCWS-S-011	Implement HTTP POST Request	Section 9.2	M	

Item	Function	Reference	Status	Requirement
SCWS-S-012	Implement HTTP DELETE Request	Section 9.2	M	
SCWS-S-013	Support for HTTP Authorisation request header	Section 9.3	M	
SCWS-S-014	Support for HTTP WWW-Authenticate response header	Section 9.3	M	
SCWS-S-015	Support for HTTP Basic authentication	Section 10	M	
SCWS-S-016	Support for HTTP Digest authentication	Section 10	O	
SCWS-S-017	Implement the TLS protocol	Section 11.1	O	SCWS-S-020
SCWS-S-018	Support for server public key pair and certificate	Section 11.1.2	O	
SCWS-S-019	Implement the PSK-TLS protocol	Section 11.1.1	O	SCWS-S-020
SCWS-S-020	Support for Fragment Negotiation in TLS	Section 11.1.3	O	
SCWS-S-021	Deliver an ACP data object	Section 12.1	O	
SCWS-S-022	Support all SCWS administration commands	Section 13.1	M	
SCWS-S-023	Support for the lightweight administration protocol	Section 13.3.1	M	
SCWS-S-024	Support for UICC File access with URI	Appendix E	O	
SCWS-S-025	Support for SCWS Content Audit	Appendix G	O	
SCWS-S-026	Support for Pipelined administrative commands	Appendix H	O	
SCWS-S-027	Support for for self care administration	Appendix I	O	
SCWS-S-028	Implement HTTP PUT Request	Section 9.2	M	
SCWS-S-029	Support for directory DELETE	Section 13.1	M	
SCWS-S-030	Support for Content-Type, Content-Encoding, Content-Language request headers	Section 13.1	M	
SCWS-S-031	Support for Conditional GET based on the If-Match and If-none-Match request header related to caching mechanism	Section 9.5	M	
SCWS-S-032	Support for ETag response header related to caching mechanism	Section 9.5	M	
SCWS-S-033	Support for Session Resume as defined in TLS	Section 11.1.3	O	
SCWS-S-034	Support for Multiple Audit Commands	Appendix GG.5	O	

B.5 SCR for Device

This is the device that connects to the smart card (e.g. Mobile phone with a Web browser that accesses the SCWS).

Item	Function	Reference	Status	Requirement
SCWS-ME-001	Support for local transport protocols with the smart card	Section 7 Section 8	M	SCWS-ME-001 OR SCWS-ME-004
SCWS-ME-002	BIP transport protocol: IP Address of the smart card is 127.0.0.1 for local communication with the SCWS	Section 7	O	SCWS-ME-005 AND SCWS-ME-006 AND SCWS-ME-007
SCWS-ME-003	BIP transport protocol: Mnemonic names associated to the smart card IP Address 127.0.0.1	Section 7	O	

SCWS-ME-004	TCP protocol: Smart card has its own IP Address for local communication with the SCWS	7.2	O	
SCWS-ME-005	BIP transport protocol: Establish BIP channel when receiving the "Open channel related to UICC server mode" command	Section 8.1.1	O	
SCWS-ME-006	BIP transport protocol: Be able to open at least 2 BIP channels in "TCP, UICC in server mode"	Section 8.1.1	O	
SCWS-ME-007	BIP transport protocol Be able to open an additional BIP channel "TCP, UICC in client mode" for SCWS administration	Section 8.1.1	O	
SCWS-ME-008	ME implements a "trusted execution environment"	Section 12	O	SCWS-ME-009 AND SCWS-ME-010
SCWS-ME-009	Retrieve the ACP from the SCWS	Section 12.1	O	
SCWS-ME-010	Implement a SCWS ACP enforcer	Section 12	O	
SCWS-ME-011	Support for Conditional GET based on the If-Match and If-none-Match request header related to caching mechanism	Section 9.5	M	
SCWS-ME-012	Support for ETag response header related to caching mechanism	Section 9.5	M	

Appendix C. Bearer Independent protocol (BIP) (Informative)

The following two figures describe the basic functionality and usage of the Bearer Independent Protocol (BIP). It is depicted here for information only. For more details please refer to the ETSI SCP [TS 102 223] specification.

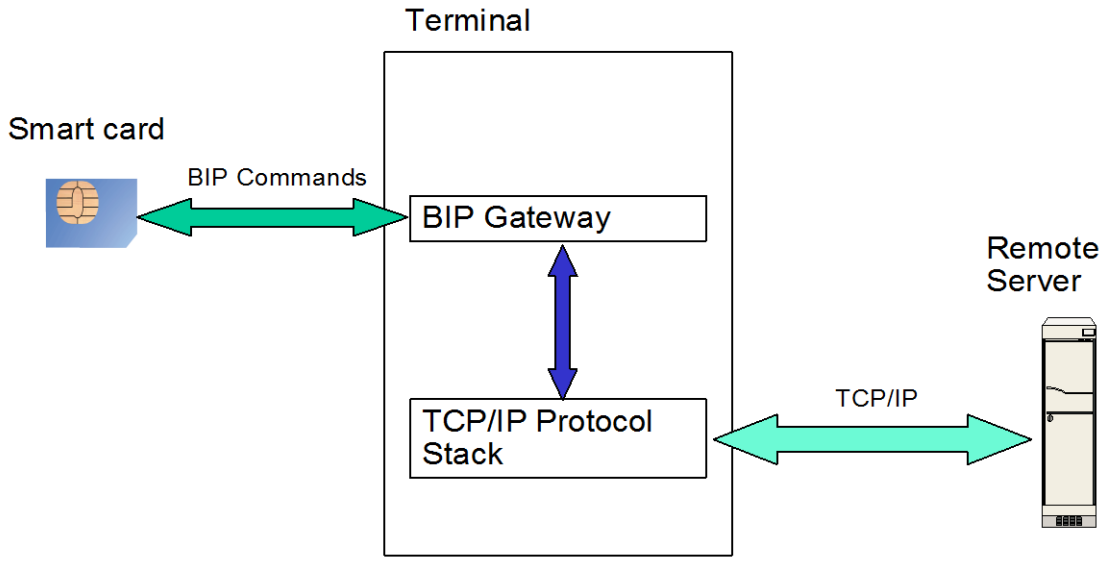

Figure 4: Usage of BIP in client mode

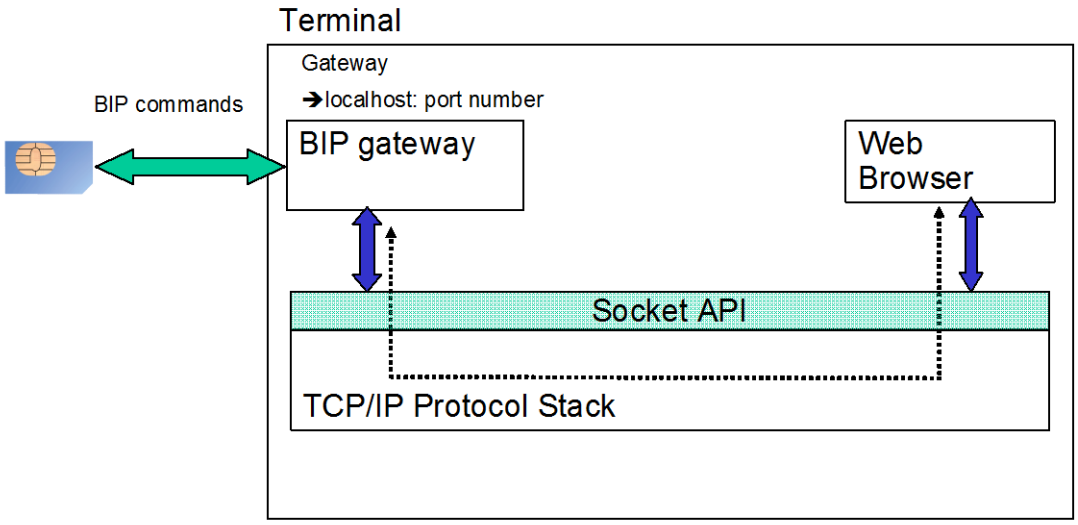

Figure 5: Usage of BIP in server mode

Appendix D. Overview of Transport Protocols (Informative)

Type	Usage	Comments
High-Speed Interface	Operational mode	Can be used for browsing web pages on the SCWS from a client through a direct TCP/IP connection on top of the UICC's High-Speed Interface. Please refer to chapter [7.2].
High-Speed Interface	Administrative mode	Can be used to transfer and manage web pages on the SCWS from an Administrative Entity through a direct TLS connection on top of a direct TCP/IP connection over the UICC's High-Speed Interface. Please refer to chapter [13.3.2.2].
BIP transport protocol type "TCP, UICC in server mode"	Operational mode	Can be used for browsing web pages on the SCWS from a client through a CAT BIP connection according to TS 102 223. Please refer to chapter [7.1].
BIP transport protocol type "TCP, UICC in client mode"	Administrative mode	Can be used to transfer and manage web pages on the SCWS from an Administrative Entity through a CAT BIP connection according to TS 102 223. Please refer to chapter [13.3.2.1].
Concatenated SMS	Administrative mode	Can be used to transfer and manage web pages on the SCWS from an Administrative Entity through an SMS channel. Used also to trigger a Full administration session of the SCWS. Please refer to chapter [13.3.1] and [13.3.2.4].

Appendix E. UICC File access with a URI (Normative)

A UICC implementing a file access with a URI MAY provide access to the legacy TS 102 221 file structure as described in this section. This access is useful in order to have standardized URI definitions for a remote administration of resources in the smart card. A SCWS that implements a UICC file access with a URI SHALL adhere to the specification in this annex.

The HTTP GET and PUT commands are used in order to read or update the relevant resource in the smart card. The reading or updating SHALL be possible only with the SCWS administration protocol defined in chapter [13] and abide to the relevant access conditions that are derived from the mutual authentication with the remote administration server. For the purpose of deriving the internal access conditions it is considered that the mapping of the admin server key (PSK-key) to an internal access right key (e.g. ADM1) is proprietary.

If the remote entity does not have the proper access/update conditions the request SHALL be rejected and the returned HTTP response SHALL indicate the relevant error code. Internally derived access rights are beyond the scope of this specification.

The path that is sent via the HTTP request SHALL start with `/adminUICC` and will be followed by the concrete path to the relevant resource.

The complete syntax of the path component for the resource in the card is depicted below. In this section, BYTE refers to an 8 bit field, HEX refers to a hexadecimal value coded on 4 bits, and CHAR represent an ASCII character. The symbol '|' represents a logical OR.

path	=	/adminUICC["/"sc_resource]
sc_resource	=	[path] | [AID]
path	=	[DF"/"]* EF
DF	=	2*[BYTE]
EF	=	2*[BYTE]
AID	=	16*[BYTE] ["/"path]
BYTE	=	2*[HEX]
HEX	=	"A" | "B" | "C" | "D" | "E" | "F"| "a" | "b" | "c" | "d" | "e" | "f" | DIGIT
DIGIT	=	"1" | "2" | "3" | "4" | "5" | "6"| "7" | "8" | "9" | "0"

E.1 Updating a file

Updating a file is done with the PUT command and can also be used for updating a portion of a file. The following possible query parameters are defined:

`record` – the number of the record to be updated. Record number starts from 1.

(applicable only for a linear fixed EF or cyclic EF)

`offset` – the offset from which the content should be read or updated

(applicable only for a transparent EF, linear fixed EF, cyclic EF)

`tag` – the tag in a BER-TLV structure EF

(applicable only for a BER-TLV structure EF)

E.2 Retrieving file content

Retrieving a file content is done with the Get command. It can also be used for retrieving a record from a record based file or a portion of a file or by using an offset and length from a binary file. It is also possible to read only a portion of a record by specifying an offset within a record. The following possible query parameters are defined. If no query parameter is specified, the complete resource shall be returned.

`record` – the number of the record to be read. Record number starts from 1.

(applicable only for a linear fixed EF or cyclic EF)

`offset` – the offset from which the content should be read.

(applicable only for a transparent EF, linear fixed EF, cyclic EF, or BER-TLV structure EF)

`tag` – the tag in a BER-TLV structure EF (applicable only for a BER-TLV structure EF)

`length` – length of the data to be retrieved in bytes.

(applicable to any EF)

E.3 Examples

E.3.1 Example 1

```
PUT /adminUICC/3F00/4F48    HTTP/1.1  CRLF
Content-Type: application/octet-stream  CRLF
Content-Length: 16 CRLF
Host: anything CRLF
CRLF
body_data…
```

In the above example the file located in `/3F00/4F48` is updated with the sent data, which overrides the previous data in the file. The length of the sent data is 16 bytes.

E.3.2 Example 2

```
PUT /adminUICC/3F00/4F01?record=2&offset=10   HTTP/1.1  CRLF
Content-Type: application/octet-stream  CRLF
Content-Length: 20 CRLF
Host: anything
CRLF
body_data…
```

In the above example record number 2 in the file `/3F00/4F01` is updated with the sent binary data but starting from offset of 10 bytes from the beginning of the record. The length of the sent data is 20 bytes.

E.3.3 Example 3

```
GET /adminUICC/1212121519976438256457986754273 4/12A1?length=50   HTTP/1.1    CRLF
Host: anything
```

In the above example the file located in `12A1` under the `1212121519976438256457986754273 4 ADF` is retrieved.

E.3.4 Example 4

```
GET /adminUICC/3F00/0512?record=3&length=50   HTTP/1.1    CRLF
Host: anything
```

In the above example the first 50 bytes from record number 3 are retrieved from the file located in /`3F00/0512.`

E.4 Error Codes

If the HTTP GET or PUT request itself failed the SCWS SHALL return the relevant HTTP error code (for example malformed HTTP request).

If the HTTP request was well understood but accessing or updating a file failed the HTTP response status code SHALL be

either "404" (when the resource was not found) or “403” . If the response status is "403" the body of the response SHALL be the MIME type text/plain containing an error string (without the quotes) of the request that could not be executed. For some cases the erroneous parameter must be included in the result string, (see examples). :

The error result string shall have following format: <**error_code**>[:<**error_text**>[:<**error context**>]]

- <**error_code**>: decimal integer with four digits. (Mandatory)
- <**error_text**>: error explicit description (optional)
- <**error_context**>: error textual context (optional, but can be present only if <error_text> is present)

Additional spaces between error code, error text and error context are allowed.

The following errors are defined:

Error code	**Error text**	**Error Context**	**Description**
“0002”	“wrong parameter”	*“<param>”*	An unknown parameter <*param*> related to the given command. Example: “0002:wrong parameter: lengtthh”
“0003”	“wrong value”	*“<param>=<val>”*	A parameter value given in the command is not supported. Example: “0003:wrong value: length=abc”
“0004”	“referenced data not found”	*“<param>=<val>”*	A parameter <*param*> value <*val*> reference an entity which does not exist (e.g. offset too high, record number not found, tag not found) Examples: “0004:referenced data not found:`record=2`”) "0004:referenced data not found:`offset=12`” "0004:referenced data not found:`tag=88`”
“00FF”	“unknown error”	*“<cmd and parameters>”*	Any other error not covered by previous codes (e.g.: Internal error or malformed command) Example: “00FF:unknown error:<cmd and parameters>”

"0020"	"security status not satisfied"		The security status is not sufficient (e.g. trying to update the IMSI may not be possible) Example: "0020:security status not satisfied"
"0021"	"file invalidated"		Updating the resource is not possible because the file associated with the given URL is invalidated Example: "0021:file invalidated"

The following example shows the response when the HTTP GET request refers to a resource that does not exist. Suppose that the request was `GET /adminUICC/3F00/0512?record=2`. In this error example either the file does not exist or the record within the file is out of range:

```
HTTP/1.1 403 Forbidden CRLF
Content-Type: text/plain CRLF
Content-Length: 60 CRLF
CRLF 0004:referenced data not found:record=2
```

The following example shows the response when the HTTP GET request contains an unknown parameter. Suppose that the request was `GET /adminUICC/3F00/0512?lengtthh=50.` This error example indicates that the “lengtthh” parameter is wrong (e.g. “length” should have been used):

```
HTTP/1.1 403 Forbidden CRLF
Content-Type: text/plain CRLF
Content-Length: 51 CRLF
CRLF 0002:wrong parameter:lengtthh
```

Appendix F. Security Considerations (Informative)

Although the smart card can be regarded as a secure hardware token, the following notes should be considered when offering SCWS based services:

- It cannot be expected that the smart card is always in a trusted environment or in a mobile handset (e.g. the smart card can be put in a card reader which is connected to a PC)
- It cannot be expected that the administration channel is always available (e.g. the smart card is not inserted in mobile phone or there is no network coverage)
- Even if an ACP enforcer is implemented in the device (e.g. mobile handset) it should be understood that the smart card may be moved to another environment without ACP enforcement functionality
- Protocols used in the SCWS specification (e.g. basic or digest authentication, TLS) may have their inherent security consideration and implementers should refer to the specification of each protocol and take them into consideration

Appendix G. SCWS content audit (Normative)

G.1 Generalities

The SCWS MAY provide administrative commands to allow a remote administration server to audit the SCWS content. This audit capability is useful when having multiple separated administration servers. It allows re-synchronisation between the remote servers and the SCWS, and to do diagnosis of the state of the SCWS in case of unexpected card behaviour. A SCWS that implements a SCWS content audit SHALL adhere to the specification in this annex.

Audit commands SHALL NOT be accessible by non authorized entity (i.e. only with admin authentication).

Audit commands SHALL be “safe” and “idempotent” meanings that they shall not take an action other than retrieval (see [HTTP/1.1]).

G.1.1 Content audit request format

SCWS content audit commands MUST be sent with HTTP GET as follows:

- The abs_path part of the URL MUST be “/SCWS/admin” (same URL as defined in section 13.1.3)
- The First Query parameter MUST always be “cmd” with a value defined in this appendix for selecting a specific audit command.
- All query parameters SHALL be case sensitive
- No message-body may be present.

Example:

```
GET /SCWS/admin?cmd=dir&uri=/meteo /HTTP/1.1  CRLF
Host: anything CRLF
CRLF
```

If the full administration protocol is used and if a query parameter of the administration command is an URI, the request SHALL be rejected if the card administration agent involved in the exchanges is not allowed to access this URI.

G.1.2 Content audit response format

SCWS content audit response SHALL be sent within the HTTP response with:

- The content type SHALL be “text/xml” (see [XML Media Type]).
- A message-body MUST be present.
- The message-body MUST be a well-formed XML validating the DTD (see [XML 1.0]) corresponding to the audit command specified by the “cmd” query parameter of the incoming request.
- Specific errors may be coded by using the error HTTP status code **403 Forbidden.**

The response to an audit request contains only data related to the administration agent involved in the exchange (e.g. protection sets or users defined for this administration agent, URIs accessible by this administration agent).

G.2 Static pages audit

G.2.1 Static pages list

G.2.1.1 Description

This command retrieves a list of all file names and directory names under the given directory *abs_path* URI.

This command is not recursive. Iteration on sub directories could be done for a complete exploration.

By directory and file we mean segments of a static resource abs_path as defined here:

```
static-resource-abs_path = “/”*( directory “/”) filename
 dir-sep = “/”
 directory = <any OCTET except “/” >
 filename = <any OCTET except “/” >
```

Note:

This directory and file description is a virtual namespace interpretation of static resources URI (fixes no constraint about the real SCWS static pages implementation of a file system).

G.2.1.2 Request query parameters

Name	Required	Value	Description
cmd	YES	**dir**	Select the dir command
uri	YES	*[absolute_path]*	`directory` URI

G.2.1.3 Response message

The command returns XML content with:

- Selected directory URI (tag <u>)
- Set of directory name (tag <d>) found directly under the selected directory.
- Set of file name (tag <f>) found directly under the selected directory.

The XML must validate the following DTD:

```
<!-- Root -->
<!ELEMENT r (u,d*,f*) >
<!-- Current URI -->
<!ELEMENT u (#PCDATA) >
<!-- Sub-directory name -->
<!ELEMENT d (#PCDATA) >
<!-- Sub-file name -->
<!ELEMENT f(#PCDATA) >
```

HTTP error status code 403 (Forbidden) SHALL be return, if:

- the given “uri” is incorrect (null or empty)
- the given “uri” doesn’t exist

G.2.1.4 Example [Informative]

The following request performed on a SCWS which contains the following static resources:

- /meteo/index.html
- /meteo/weather.html
- /meteo/images/sunny.gif
- /meteo/images/cloudy.gif

```
GET /SCWS/admin?cmd=dir&uri=/meteo HTTP/1.1  CRLF
Host: anything CRLF
CRLF
```

May obtain the following response

```
HTTP/1.1 200 OK CRLF
Content-Type: text/xml CRLF
Content-Length: XX CRLF
CRLF
<?xml version=”1.0” encoding=”UTF-8”?>
<r>
<u>/meteo</u>
<d>images</d>
<f>index.html</f>
<f>weather.html</f>
</r>
```

G.2.2 Static page properties

There is no need for a specific command to audit static page properties like size because the HTTP HEAD request will return all useful entity/general HTTP headers concerning this resource.

G.2.3 Memory

G.2.3.1 Description

This command retrieves the overall amount of used and available memory of the server for storing static pages using PUT command.

Note: Depending of internal implementation (memory allocation granularity, storage of headers and system information), it may not be possible to exactly compute the amount of free memory. So there is no success guaranty when trying to put a resource with a size close to free memory size.

G.2.3.2 Request query parameters

Name	Required	Value	Description
cmd	YES	**mem**	Select the mem command

G.2.3.3 Response message

The command returns XML content with:

- Integer value of the amount in byte of free memory available for SCWS static pages (tag <a>).

- Integer value of the amount of memory that is used by the SCWS for static pages (tag <u>).

The XML must validate the following DTD:

```
<!--Root -->
<!ELEMENT r (a,u) >
<!--Integer value of the remaining available memory for SCWS in bytes-->
<!ELEMENT a (#PCDATA) >
<!--Integer value of the memory used by the SCWS in bytes -->
<!ELEMENT u (#PCDATA) >
```

G.2.3.4 Example [Informative]

The following request performed on a SCWS with 1KB used on a total memory of 8KB (so 7KB free memory):

```
 GET /SCWS/admin?cmd=mem HTTP/1.1  CRLF
 Host: anything CRLF
 CRLF
```

May obtain the following response

```
HTTP/1.1 200 OK CRLF
Content-Type: text/xml CRLF
Content-Length: XX CRLF
CRLF
<?xml version=”1.0” encoding=”UTF-8”?>
<r>
<a>7168</a>
<u>1024</u>
</r>
```

G.3 Protection set audit

G.3.1 User's list

G.3.1.1 Description

This command gets all users owned by the current administration agent.

G.3.1.2 Request query parameters

Name	Required	Value	Description
cmd	YES	**lusr**	Select the command list user

G.3.1.3 Response message

The command returns XML content with:

- Set of user login (tag <l>).

The XML must validate the following DTD:

```
<!-- Root -->
<!ELEMENT r (l)* >
<!-- User login Name -->
<!ELEMENT l (#PCDATA) >
```

G.3.1.4 Example [Informative]

The following request performed on a SCWS with 3 users with the following logins “James”, “Mary” and “John”:

```
 GET /SCWS/admin?cmd=lusr HTTP/1.1  CRLF
 Host: anything CRLF
 CRLF
```

May obtain the following response

```
HTTP/1.1 200 OK CRLF
Content-Type: text/xml CRLF
Content-Length: XX CRLF
CRLF
<?xml version=”1.0” encoding=”UTF-8”?>
<r>
<l>James</l>
<l>Mary</l>
<l>John</l>
</r>
```

G.3.2 Resource Protection Set's list

G.3.2.1 Description

This command retrieves all resource protection sets owned by the current administration agent.

G.3.2.2 Request query parameters

Name	Required	Value	Description
cmd	YES	**lps**	Select the command

G.3.2.3 Response message

The command returns XML content with:

- Set of resource protection set id(tag <i>).

The XML must validate the following DTD:

```
<!-- Root -->
<!ELEMENT r (i)* >
<!-- resource protection set id -->
<!ELEMENT i (#PCDATA) >
```

G.3.2.4 Example [Informative]

The following request performed on a SCWS with 3 resource protection set with the following ids: “mypages”, “securearea” and “msecurepgs”.

```
 GET /SCWS/admin?cmd=lps HTTP/1.1  CRLF
 Host: anything CRLF
 CRLF
```

May obtain the following response

```
HTTP/1.1 200 OK CRLF
```

```
Content-Type: text/xml CRLF
Content-Length: XX CRLF
CRLF
<?xml version=”1.0” encoding=”UTF-8”?>
<r>
<i>ps1</i>
<i>ps2</i>
<i>ps3</i>
</r>
```

G.3.3 Resource Protection Set

G.3.3.1 Description

This command gets Resource Protection Set information.

G.3.3.2 Request query parameters

Name	Required	Value	Description
cmd	YES	**ps**	Select the command
psid	YES	*[ps_identifier]*	Protection set identifier

G.3.3.3 Response message

The command returns XML content with:

- Protection set identifier (tag <i>)
- Authorized protocol under this protection set (tag <p>)
- The Realm name (tag <n>)
- Authentication type (tag <t>)
- Set of registered user login (tag <l>).

The XML must validate the following DTD:

```
<!-- Root -->
<!ELEMENT r (i,p,(n,t,l*)?) >
<!-- Protection set identifier -->
<!ELEMENT i (#PCDATA) >

<!-- Protocol used by the realm : 'http', 'https' or 'admin' -->
<!ELEMENT p (#PCDATA) >
<!-- Realm name -->
<!ELEMENT n (#PCDATA) >
<!-- Authentication type: 'basic' or 'digest' -->
<!ELEMENT t (#PCDATA) >
<!-- User login -->
<!ELEMENT l (#PCDATA) >
```

HTTP error status code 403 (Forbidden) SHALL be returned if:

- the psid is incorrect (null or empty)
- the psid doesn’t exist for the current administration agent

G.3.3.4 Example [Informative]

The following request performed on a SCWS with a protection set “mypages” (defined as the first example find in section 13.1.6.2) and with two authorized users named “James” and “Mary”:

```
GET /SCWS/admin?cmd=ps&psid=ps3 HTTP/1.1  CRLF
Host: anything CRLF
CRLF
```

May obtain the following response

```
HTTP/1.1 200 OK CRLF
Content-Type: text/xml CRLF
Content-Length: XX CRLF
CRLF
<?xml version=”1.0” encoding=”UTF-8”?>
<r>
<i>ps3</i>
<p>https</p>
<n>Your%20secured%20pages</n>
<t>digest</t>
<l>James</l>
<l>Mary</l>
</r>
```

G.3.4 Protected URI sub-tree list

G.3.4.1 Description

This command gets all URI sub-trees protected by a protection set owned by the current administration agent.

G.3.4.2 Request query parameters

Name	Required	Value	Description
cmd	YES	**lpr**	Select the command

G.3.4.3 Response message

The command returns XML content with:

- Set of URI sub tree protection set definition (tag <p>) with
    - The given protected URI sub-tree prefix (tag <u>).
    - The associated protection set id (tag <i>).

The XML must validate the following DTD:

```
<!-- Root -->
<!ELEMENT r (p*) >
<!— protected sub tree definition-->
<!ELEMENT p (i,u) >
<!-- Protection set identifier -->
<!ELEMENT i (#PCDATA) >
<!-- sub-tree uri -->
```

```
<!ELEMENT u (#PCDATA) >
```

G.3.4.4 Example [Informative]

The following request performed on a SCWS with 3 resource protection set with the following ids: "mypages", "securearea" and "msecurepgs".

```
 GET /SCWS/admin?cmd=lpr HTTP/1.1  CRLF
 Host: anything CRLF
 CRLF
```

May obtain the following response

```
HTTP/1.1 200 OK CRLF
Content-Type: text/xml CRLF
Content-Length: XX CRLF
CRLF
<?xml version="1.0" encoding="UTF-8"?>
<r>
<p><i>ps1</i><u>/mypages/</u></p>

<p><i>ps1</i><u>/otherpages/</u></p>

<p><i>ps2</i><u>/bank/</u></p>

<p><i>ps3</i><u>/bank/secret/</u></p>
</r>
```

G.4 Registered application audit

G.4.1 Registered application list

G.4.1.1 Description

This command retrieves the list of applications that are registered on an URI owned by the current administration agent.

G.4.1.2 Request query parameters

Name	Required	Value	Description
cmd	YES	**lapp**	Select the command

G.4.1.3 Response message

The command returns XML content with:

- A list of applications that are registered to URIs owned by the current administration agent (tag <a>):

The XML must validate the following DTD:

```
<!-- Root -->
<!ELEMENT r (a*) >

<!-- application identifier used for SCWS registration-->
<!ELEMENT a(#PCDATA) >
```

G.4.1.4 Example [Informative]

The following request performed on a SCWS with two registered applications with the following ids: "logger", "bank".

```
GET /SCWS/admin?cmd=lapp HTTP/1.1  CRLF
Host: anything CRLF
CRLF
```

May obtain the following response

```
HTTP/1.1 200 OK CRLF
Content-Type: text/xml CRLF
Content-Length: XX CRLF
CRLF
<?xml version="1.0" encoding="UTF-8"?>
<r>
<a>logger</a>
<a>bank</a>
</r>
```

G.4.2 Registered application information

G.4.2.1 Description

This command retrieves all information related to a given application registered to the administration agent.

G.4.2.2 Request query parameters

Name	Required	Value	Description
cmd	YES	**app**	Select the command
appid	YES	**[app_identifier]**	The application identifier

G.4.2.3 Response message

The command returns XML content with:

- Selected application identifier (tag <a>)
- List of intercept URI triggering condition (tag <i>) mapped on this application (see section 13.1.6.9).
- List of content providing URI triggering condition (tag <u>) mapped on this application (see section 13.1.6.9).

The XML must validate the following DTD:

```
<!-- Root -->
<!ELEMENT r (a,i*,u*) >
<!-- application identifier used for SCWS registration -->
<!ELEMENT a(#PCDATA) >
<!-- Interception application URI triggering condition-->
<!ELEMENT i(#PCDATA) >
<!-- Content providing application URI triggering condition-->
<!ELEMENT u(#PCDATA) >
```

G.4.2.4 Example [Informative]

The following request performed on a SCWS with an application registered as “advertising” which act in the same time as an interception and a content providing application:

- intercept all requests to resources under the URI sub-tree “/advertising/*”
- provide content for resource URI “/app/advertising/audit”

```
GET /SCWS/admin?cmd=app&appid=advertising HTTP/1.1  CRLF
Host: anything CRLF
CRLF
```

May obtain the following response

```
HTTP/1.1 200 OK CRLF
Content-Type: text/xml CRLF
Content-Length: XX CRLF
CRLF
<?xml version=”1.0” encoding=”UTF-8”?>
<r>
<a>advertising</a>
<i>/advertising/*</i>
<u>/app/advertising/audit</u>
</r>
```

G.5 Multiple audit commands

G.5.1 Description

Multiple SCWS audit commands MAY be send with HTTP GET as follows:

- The abs_path part the URL MUST be “/SCWS/admin” (same URL as defined in section 13.1.3).
- Each parameter/value(s) defining an audit command MUST be separated by a ‘&’ character.
- Each audit command MUST be separated by a ‘&’ character.
- All query parameters SHALL be case sensitive.
- No message-body may be present.

G.5.2 Response message

The XML content MUST contain the tag <b> to define several blocks of audit commands

The XML must validate the following DTD:

```
<!-- Root -->
<!ELEMENT r (b1,b2,..,bn)>
<!-- First command -->
<!ELEMENT b1(x1,..,xn)   >
<!-- Command parameter -->
<!ELEMENT x1 (#PCDATA)   >

.........................

<!-- Command parameter -->
<!ELEMENT xn (#PCDATA)   >
```

```
<!-- Second command -->
<!ELEMENT b2(y1,..,yn)   >

<!-- Command parameter -->
<!ELEMENT y1 (#PCDATA)   >

.........................

<!-- Command parameter -->
<!ELEMENT yn (#PCDATA)   >

.........................

<!-- Last command -->
<!ELEMENT b2(z1,..,zn)   >

<!-- Command parameter -->
<!ELEMENT z1 (#PCDATA)   >

.........................

<!-- Command parameter -->
<!ELEMENT zn (#PCDATA)   >
```

If at least one of the internal audit commands was not executed the HTTP response status code SHALL be “403” and the `Pragma: cmd=CmdNum` SHALL be included in the response as defined above. The body of the response SHALL be the MIME type text/plain containing the error string (without the quotes) of the command that could not be executed. The result-strings are defined in chapter 13.2.2

The Pragma Header field in the response SHALL contain the number of the last command that was successfully executed by the SCWS. If a command was not successfully executed all commands after it SHALL be ignored and not executed by the SCWS.

G.5.3 Example [Informative]

The following request performed a dir command example as defined in chapter G.2.1.4, a mem command example as defined in chapter G.2.3.4, a list user command as defined in chapter G.3.1.4 and a list protection set command as defined in chapter G.3.2.4.

```
 GET /SCWS/admin?cmd=dir&uri=/meteo&cmd=mem&cmd=lusr&cmd=lps /HTTP/1.1  CRLF
 Host: anything CRLF
 CRLF
```

May obtain the following response

```
HTTP/1.1 200 OK CRLF
Content-Type: text/xml CRLF
Content-Length: XX CRLF
CRLF
<?xml version=”1.0” coding=”UTF-8”?>
<r>

<b>
<u>/meteo</u>
<d>images</d>
<f>index.html</f>
<f>weather.html</f>

</b>
```

```
<b>
<a>7168</a>
<u>1024</u>
</b>
<b>
<l>James</l>
<l>Mary</l>
<l>John</l>
</b>
<b>
<i>ps1</i>
<i>ps2</i>
<i>ps3</i>
</b>
</r>
```

Appendix H. Pipelining over full administration protocol (Normative)

As explained in [HTTP/1.1] the pipelining allows a client to make multiple requests without waiting for each response, allowing a single TCP connection to be used much more efficiently, with lower elapsed time. This is particularly important when the network latency is high; this is usually the case on wireless based network.

In order to optimize the remote administration, the pipelining SHOULD be available over the full administration protocol.

H.1 Pipeline behaviour

As specified by [HTTP/1.1] the remote administration SHALL NOT pipeline request embedded into it POST response (see 13.3.2.7) until it's sure than the SCWS support persistence.

This means that if the remote administration server doesn't know the SCWS persistence capability, the first POST response of a full administration session SHALL embedded only one request. If the SCWS doesn't support pipeline it SHALL return "Connection: close" in it response. Otherwise the remote administration server could assume that pipeline is supported and MAY pipeline its Idempotent requests (see [HTTP/1.1] restriction about non-Idempotent sequences and methods).

If the Administration server pipelines several requests embedded into one POST response (see 13.3.2.7), the SCWS SHALL pipeline all the responses in the next administration agent POST request body (see 13.3.2.6). The order of the responses SHALL be the same than the requests (see [HTTP/1.1] for details).

H.2 Example - Informative:

Let's assume that during an administration session we needs to put 20 text files named "hello1.html" to "hello20.html".

H.2.1 Administration session without pipeline.

In this case the administration requires the 21 following POST request.

```
POST /downloadmanager/hello?cmd=1 HTTP/1.1 CRLF
Host: 172.96.0.1 CRLF
X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF
X-Admin-From: 8939010012751002010 CRLF
CRLF
```

```
HTTP/1.1 200 OK CRLF
X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF
X-Admin-Next-URI: /downloadmanager/hello?cmd=2 CRLF
Content-Type: application/vnd.oma-scws-http-request CRLF
Content-Length: 108 CRLF
CRLF
PUT /hello1.html HTTP/1.1 CRLF
Host: anything CRLF
Content-Type: text/html CRLF
Content-Length: 18 CRLF
CRLF
<html>Hello</html>
```

```
POST /downloadmanager/hello?cmd=2 HTTP/1.1 CRLF
Host: 172.96.0.1CRLF
X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF
X-Admin-From: 8991200012751002010 CRLF
Content-Type: application/vnd.oma-scws-http-response CRLF
Transfer-Encoding: chunked CRLF
CRLF
2E CRLF
HTTP/1.1 204 NO CONTENT CRLF Connection: close CRLF CRLF
```

CRLF 0 CRLF CRLF
HTTP/1.1 200 OK CRLF X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF X-Admin-Next-URI: /downloadmanager/hello?cmd=3 CRLF Content-Type: application/vnd.oma-scws-http-request CRLF Content-Length: 108 CRLF CRLF **PUT /hello2.html HTTP/1.1 CRLF** **Host: anything CRLF** **Content-Type: text/html CRLF** **Content-Length: 18 CRLF** **CRLF** **<html>Hello</html>**
[------ 18 POST request/response with one embedded PUT ------]
POST /downloadmanager/hello?cmd=21 HTTP/1.1 CRLF Host: 172.96.0.1CRLF X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF X-Admin-From: 8991200012751002010 CRLF Content-Type: application/vnd.oma-scws-http-response CRLF Transfer-Encoding: chunked CRLF CRLF 2E CRLF **HTTP/1.1 204 NO CONTENT CRLF Connection: close CRLF CRLF** CRLF 0 CRLF CRLF
HTTP/1.1 204 No Content CRLF X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF CRLF

H.2.2 Administration session on a with pipeline

In this case the administration requires the 3 following POST request.

POST /downloadmanager/hello?cmd=1 HTTP/1.1 CRLF Host: 172.96.0.1 CRLF X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF X-Admin-From: 8939010012751002010 CRLF CRLF
HTTP/1.1 200 OK CRLF X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF X-Admin-Next-URI: /downloadmanager/hello?cmd=2 CRLF Content-Type: application/vnd.oma-scws-http-request CRLF Content-Length: 108 CRLF CRLF **PUT /hello1.html HTTP/1.1 CRLF** **Host: anything CRLF** **Content-Type: text/html CRLF** **Content-Length: 18 CRLF** **CRLF**

<html>Hello</html>
POST /downloadmanager/hello?cmd=2 HTTP/1.1 CRLF Host: 172.96.0.1CRLF X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF X-Admin-From: 8991200012751002010 CRLF Content-Type: application/vnd.oma-scws-http-response CRLF Transfer-Encoding: chunked CRLF CRLF 1B CRLF **HTTP/1.1 204 NO CONTENT CRLF CRLF** CRLF 0 CRLF CRLF
HTTP/1.1 200 OK CRLF X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF X-Admin-Next-URI: /downloadmanager/hello?cmd=3 CRLF Content-Type: application/vnd.oma-scws-http-request CRLF Content-Length: 2160 CRLF CRLF **PUT /hello2.html HTTP/1.1 CRLF** **Host: anything CRLF** **Content-Type: text/html CRLF** **Content-Length: 18 CRLF** **CRLF** **<html>Hello</html>** *[------ 17X ------]* **PUT /hello20.html HTTP/1.1 CRLF** **Host: anything CRLF** **Content-Type: text/html CRLF** **Content-Length: 18 CRLF** **CRLF** **<html>Hello</html>**
POST /downloadmanager/hello?cmd=3 HTTP/1.1 CRLF Host: 172.96.0.1CRLF X-Admin-Protocol: oma-scws-admin-agent/1.1.1 CRLF X-Admin-From: 8991200012751002010 CRLF Content-Type: application/vnd.oma-scws-http-response CRLF Transfer-Encoding: chunked CRLF CRLF 1B CRLF **HTTP/1.1 204 NO CONTENT CRLF CRLF** CRLF *[------ 17X ------]* 1B CRLF **HTTP/1.1 204 NO CONTENT CRLF CRLF** CRLF 0 CRLF CRLF
HTTP/1.1 204 No Content CRLF X-Admin-Protocol: oma-scws-remote-admin/1.1.1 CRLF CRLF

Appendix I. User Self-Care Administration (Normative)

I.1 Generalities

The SCWS MAY provide administrative commands to allow the user to administrate himself personal data through the browser.

The scope of the user administrative commands is limited to the administration of features directly linked to the user such as the modification of his password. These operations may be possible directly from a web browser across HTML forms.

A SCWS that implements a user self-care administrative commands SHALL adhere to the specification defined in this appendix.

I.1.1 User self-care administration request format

All the specifics SCWS user self-care administration commands MUST be done with HTTP POST with:

- The abs_path part of the URL MUST be "/SCWS/user-admin"
- The entity-header-field "Content-Type" must be used. The value is "application/x-www-form-urlencoded".
- The entity-header-field "Content-Length" must be used. The value is the length of the message-body in bytes as a decimal number
- The message-body, in this case called "user-admin-command", contains data that is interpreted by the admin-component of the SCWS.
- All parameters SHALL be case sensitive.
- The first parameter SHALL be a "cmd" parameter with a command value as described in this appendix.
- Other parameters SHALL be described as specified according the given "cmd" value.

The access to the user self-care administrative URI "/SCWS/user-admin" SHOULD be permitted only over HTTPS protocol.

I.1.2 User self-care administration response format

The response to a user self-care administrative command:

- MUST use a response status 303 "see other" (as specified by the [RFC2616])
- MUST contain a "Location" header with an URI specified in the section describing the given "cmd" parameter behaviour.
- SHOULD contain a short hypertext note with a hyperlink to the URI specified by the "Location" header.

Example - Informative:

```
HTTP/1.1 303 See Other CRLF
Location: /redirection_uri CRLF
Content-Type: text/html CRLF
Content-Length: 68 CRLF
Host: anything CRLF
CRLF
<html><body><a href="/redirection_uri">Redirection</a></body></html>
```

I.2 Change password

I.2.1 Description

This command is used to change an existing password.

For security reason:

- This command is possible for a user account that was created with the “user-admin” property set to “true” (see 13.1.6.5)
- The previous password value must be presented.

I.2.2 Request query parameters

Name	Required	Value	Description
cmd	YES	**cpw**	Change the given password
login	YES	*[loggin_value]*	`Credential login value`
old	YES	*[old_pasword]*	`Old password value`
new	YES	*[new password]*	`New password value`
wrong	YES	*[wrong_uri]*	`Wrong Redirection URI`
ok	YES	*[succeed_uri]*	`Succeed Redirection URI`

The command shall be processed in the following way:

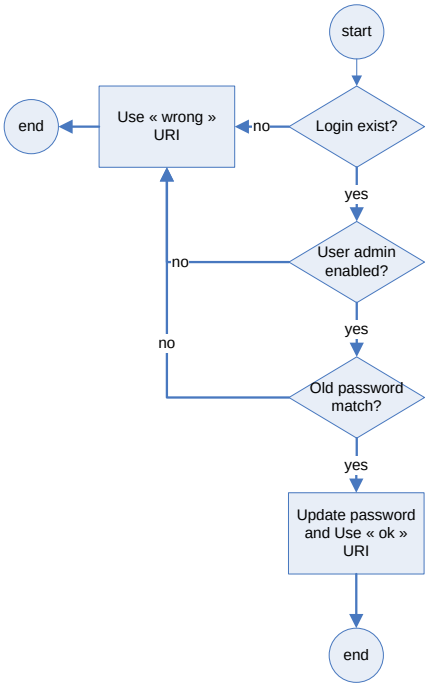

Figure 6: Change user password command

I.2.3 Response message

The following URI SHALL be used in the redirection response (303):

- *wrong_uri* if:
  - Given old password doesn't match.
  - Given login could not be administrated by user (not created with "user-admin" flag).
  - Given login doesn't exist.
  - The command could not be fulfilled.
- *succeed_uri* if command succeed.

I.2.4 Example [Informative]

If the following HTML form is used on a web browser:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">

<html>

  <head>

    <title>
```

```
      User self Administration
    </title>
  </head>
  <body>
    <h1>Change Password</h1>
    <form name="Change Password" action= "http://127.0.0.1:3516/scws/user-admin"
 method="POST" enctype= "application/x-www-form-urlencoded" id="ChangePassword">
      <p>
        <input type="hidden" name="cmd" value="cpw">
        Loggin:<br>
        <input type= "text" name="login"><br>
        Old Password:<br>
        <input type="password" name="old"><br>
        New Password:<br>
        <input type="password" name="new"><br>
        <input type="submit" value="Change password"><br>
        <input type="hidden" name="wrong" value="/wrong.html">
        <input type="hidden" name="ok" value="/admin_ok.html">
      </p>
    </form>
  </body>
</html>
```

The browser may emit the given request:

```
POST /SCWS/user-admin HTTP/1.1 CRLF
Content-Type: application/x-www-form-urlencoded CRLF
Content-Length: xx CRLF
Host: anything CRLF
CRLF
cmd=cpw&login=john&old=1234&new=4321&wrong=%2Fwrong.html& ok=%2Fadmin_ok.html
```

May obtain the following response:

```
HTTP/1.1 303 See Other CRLF
Location: /admin_ok.html CRLF
Content-Type: text/html CRLF
Content-Length: 77 CRLF
Host: anything CRLF
CRLF
<html><body><a href=”/admin_ok.html”>Change password result</a></body></html>
```