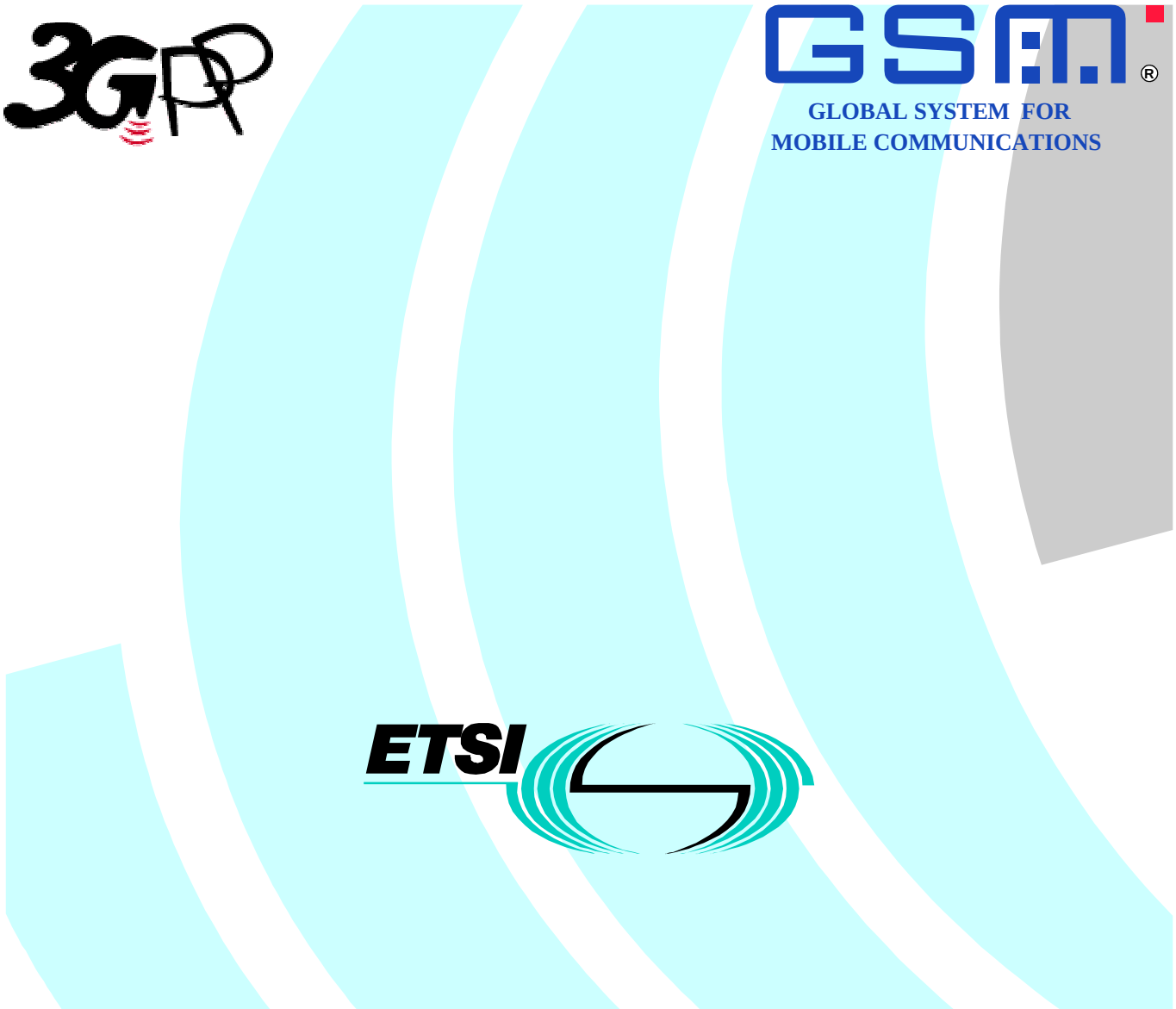
ETSI TS 123 057 V3.1.1 (2000-03)

Technical Specification

Digital cellular telecommunications system (Phase 2+) (GSM);
Universal Mobile Telecommunications System (UMTS);
Mobile Station Application Execution Environment (MExE);
Functional description;
Stage 2
(3G TS 23.057 version 3.1.1 Release 1999)



Reference
RTS/TSGT-0223057UR1

Keywords
GSM, UMTS

ETSI

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

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

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

Intellectual Property Rights

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

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

Foreword

This Technical Specification (TS) has been produced by the ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under www.etsi.org/key.

Contents

Foreword

This Technical Specification (TS) has been produced by the 3rd Generation Partnership Project (3GPP).

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

Version x.y.z

where:

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

1 Scope

The present document defines the stage 2 and stage 3 description of the Mobile Station Application Execution Environment (MExE). Stage 2 identifies the functional capabilities and information flows needed to support the service described in stage 1.

The present document includes information applicable to network operators, service providers and terminal, switch and database manufacturers.

The present document contains the core functions for a Mobile Station Application Execution Environment (MExE) which are sufficient to provide a complete service.

MExE uses a number of technologies to realise the requirements of the stage 1 description (3G TS 22.057). The present document describes how the service requirements are realised with the selected technologies. The TS is devised into sections each covering the aspects relating to particular MExE technologies, it is intended that this specification will evolve along with the MExE technologies. A generic section of the specification covers areas of MExE common to all technologies.

2 References

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] GSM 01.04: "Digital cellular telecommunications system (Phase 2+); Abbreviations and acronyms".

[2] 3G TS 22.057: "MExE Stage 1 Description".

[3] Personal Java 1.1.1, Sun Microsystems http://java.sun.com/products/personaljava/spec-1-1-1/index.html

[4] JavaPhone API version 0.9, http://java.sun.com/products/javaphone/.

[5] JTAPI 1.2, Sun Microsystems http://www.java.sun.com.

[6] Wireless Application Protocol (WAP) version 1.1 http://www.wapforum.org.

[7] vCard – The Electronic Business Card Exchange Format – Version 2.1, The Internet Mail Consortium (IMC), September 1996, http://www.imc.org/pdi/vcard-21.doc.

[8] vCalendar – The Electronic Calendaring and Scheduling Exchange Format – Version 1.0, The Internet Mail Consortium (IMC), September 1996, http://www.imc.org/pdi/

[9] Hypertext Transfer Protocol – HTTP/1.1, IETF document RFC2068, http://www.w3.org/Protocols/rfc2068/rfc2068

[10] Java Mail API version 1.0.2, http://www.java.sun.com

[11] 3G TR 22.170: "Universal Mobile Telecommunications System (UMTS); Service aspects; Provision of Services in UMTS - The Virtual Home Environment".

[12] 3G TS 22.121: "Universal Mobile Telecommunications System (UMTS); Provision of Services in UMTS - The Virtual Home Environment: Stage 1".

[13] ISO 639 International Standard - codes for the representation of language names.

[14] 3G TS 22.101: "Universal Mobile Telecommunications System (UMTS); Service Aspects; Service Principles".

[15] CC/PP Exchange Protocol based on HTTP Extension Framework; W3C http://www.w3.org/TR/NOTE-CCPPexchange

[16] Composite Capability/Preference Profiles (CC/PP):A user side framework for content negotiation; Available at W3C web pages.

[17] UAProf Specification http://www1.wapforum.org/member/wg/wag/Activities/ActivityUAPROF/index.htm

[18] JDK 1.1 security http://www.javasoft.com/products/jdk/1.1/docs/guide/security/index.html

[19] Java 2 security http://www.javasoft.com/products/jdk/1.2/docs/guide/security/index.html

[20] Java security tutorial http://java.sun.com/docs/books/tutorial/security1.2/overview/index.html

[21] OCF 1.1.: "Smartcard API specified by OpenCard Consortium http://www.opencard.org

[22] RFC 1738 Uniform Resource Locators (URL) http://www.w3.org/pub/WWW/Addressing/rfc1738.txt

[23] The MD5 Message Digest Algorithm", Rivest, R., RFC 1321, April 1992. URL: ftp://ftp.isi.edu/in-notes/rfc1321.txt

[24] ISO/IEC 10118-3 1996: "Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions".

[25] IETF RFC 2368: "The mailto URL scheme".

[26] ITU-T Recommendation X.509: "Information technology – Open Systems Interconnection – The Directory: Authentication framework".

[27] GSM 11.11: "Digital cellular telecommunications system (Phase 2+); Specification of the Subscriber Identity Module – Mobile Equipment (SIM-ME) interface".

[28] 3G TS 23.107: "3rd Generation Partnership Project; Technical Specification Group Services and system Aspects QoS Concept and Architecture (3G TS 23.107)".

[29] 3GPP TS 24.007: "3rd Generation Partnership Project; Technical Specification Group Core Network; Mobile radio interface signalling layer 3; General Aspects (3G TS 24.007)".

[30] 3GPP TS 24.008: "3rd Generation Partnership Project; Universal Mobile Telecommunications System; Mobile radio interface layer 3 specification, Core Network Protocols – Stage 3 (TS 24.008)".

[31] 3GPP TS 23.060: "3rd Generation Partnership Project; Technical Specification Group Core Network; Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Service Description; Stage 2 (3G TS 23.060)".

[32] PKCS #15 "Cryptographic Token Information Standard" version 1.0, RSA Laboratories, April 1999
URL: ftp://ftp.rsa.com/pub/pkcs/pkcs-15/pkcs15v1.doc

[33] RFC 2510 Internet X.509 Public Key Infrastructure January 1999.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document the following definitions apply:

administrator: The administrator of the MExE MS is the entity which has the control of the third party trusted domain, and all resources associated with the domain. The administrator of the device could be the user, the operator, the manufacturer, the service provider, or a third party as designated by the owner of the device.

best effort QoS (Quality of Service): The best effort QoS refers to the lowest of all QoS traffic classes. If the guaranteed QoS cannot be delivered, the bearer network delivers the QoS which can also be called best effort QoS [28].

certificate: An entity that contains the issuer's public key, identification of the issuer, identification of the signer, and possibly other relevant information. Also, a certificate contains a signed hash of the contents. The signer can be a 3rd. party other than the issuer.

delivered QoS: Actual QoS parameter values with which the content was delivered over the lifetime of a QoS session [28].

fine grain: Refers to the capabilities of the Java security system to allow applications, sections of code or Java classes to be assigned permissions to perform a specific set of privileged operations. The smallest programming element that can be given permission attributes is a Java class [19].

key pair: Key pairs are matching private and public keys. If a block of data is encrypted using the private key, the public key from the pair can be used to decrypt it. The private key is never divulged to any other party, but the public key is available, e.g. in a certificate.

negotiated QoS: In response to a QoS request, the network shall negotiate each QoS attribute to a level that is in accordance with the available network resources. After QoS negotiation, the bearer network shall always attempt to provide adequate resources to support all of the negotiated QoS profiles [31].

personal certificate: This is a certificate loaded by the user or a user application which is limited to the application that it is intended for, and is not a MExE Certificate. E.g. an e-mail application could load certificates for its usage. Personal certificates are out of scope for MExE.

phonebook: A phonebook is a dataset of personal or entity attributes. The simplest form is a set of name-number pairs as supported by GSM SIMs.

MExE: MExE (Mobile station application Execution Environment) is defined in detail in this document, but the scope of MExE does not include the operating system, or the manufacturer’s execution environment.

MExE certificate: This is a certificate used in the realisation of MExE security domains. A MExE Certificate can be used to verify downloaded MExE executables. Use of the word "certificate" in this document implies a MExE certificate. Other varieties of certificate will be explicitly qualified as a e.g. "Personal Certificate".

MExE executable: An executable is an applet, application, or executable content, which conforms to the MExE specification and may execute on the ME.

MExE Java VM: This is a standard Java virtual machine used to execute MExE Java applets and applications.

MExE native library: This is a downloaded native library that can be accessed by MExE executables.

MExE-SIM: A SIM that is capable of storing a security certificate that is accessible using standard mechanisms.

owner: An owner of the MExE MS. An owner could be a user, operator (e.g. where the MS is obtained as part of a subscription and the cost of the MS is subsidised), service provider, or a third party (e.g. the MS is owned by the user’s company and this company wishes to control how the MS is used).

power up event: An abstract event that occurs when the MExE MS is cold started (i.e. switched on).

QoS session: Lifetime of PDP context. The period between the opening and closing of a network connection whose characteristics are defined by a QoS profile. Multiple QoS sessions may exist, each with a different QoS profile [28].

QoS profile: A QoS profile comprises of a number of QoS parameters. A QoS profile is associated with each QoS session. The QoS profile defines the performance expectations placed on the bearer network [28].

requested QoS: A QoS profile is requested at the beginning of a QoS session. QoS modification requests are also possible during the lifetime of a QoS session [28], [31].

sandbox: A sandbox is a safe area to run Java code. Untrusted Java code executing in a sandbox has access to only certain resources [18].

service: A service (which may consist of an application or applet, and its related content) is a set of functions offered to a user by an organisation, and may be performed on the MExE MS and/or remotely.

service name: An identifier associated with a service, which could be a string, a fully qualified Java class name, a unique URI or other identifier.

session: The period between the launching of a MExE executable and its execution termination. A WAP-session is established between the mobile and the WAP Gateway. The duration of a WAP-session can range from a second to years. The WAP-session can be associated with a particular subscription in the WAP Gateway.

signature: "Signing" is the process of encrypting a hash of the data using a private key. If the signature can be decrypted using the public key, then the signature is valid.

signed JAR file: Archives of Java classes or data that contain signatures that also include a way to identify the signer in the manifest. (The Manifest contains a file which has attributes defined in it.)

subscribed QoS: The network will not grant a QoS greater than that subscribed. The QoS profile subscription parameters are held in the HLR. An end user may have several QoS subscriptions. For security and the prevention of damage to the network, the end user cannot directly modify the QoS subscription profile data [31].

user: The user of the MExE MS.

Further definitions specific to MExE are in GSM given in 3G TS 22.057 (MExE stage 1) [2].

3.2 Abbreviations

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

API	Application Programming Interface
APDU	Application protocol data unit
CA	Certification Authority
CC/PP	Composite Capability/Preference Profiles
Diff-serv	Differentiated Services
CGI	Common Gateway Interface
CCM	Certificate Configuration Message
CP-Admin	Certificate Present (in the MExE SIM) - Administrator
CP-TP	Certificate Present (in the MExE SIM) - Third Party
DHCP	Dynamic Host Configuration Protocol
GSM	Global System for Mobile Communication
GPRS	General Packet Radio Service
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transport Protocol Secure (https is http/1.1 over SSL, i.e. port 443)
IETF	Internet Engineering Task Force
IP	Internet Protocol
JNDI	Java Naming Directory Interface
JTAPI	Java Telephony Application Programming Interface
JAR file	Java Archive File
MMI	Man-Machine Interface
MSE	MExE Service Environment
OCF	OpenCard Framework
QoS	Quality of Service
PDP	Packet Data Protocol
RDF	Resource Description Format
RFC	Request For Comments
SAP	Service Access Point
SMS	Short Message Service
TLS	Transport Layer Security
TP	Third Party
UDP	User Datagram Protocol
UE	User Equipment
UI	User Interface
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
USSD	Unstructured Supplementary Service Data

WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WSP	Wireless Session Protocol
WTA	Wireless Telephony Applications
WTAI	Wireless Telephony Applications Interface
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
WWW	World Wide Web

Further abbreviations are given in 3G TS 22.057 (MExE stage 1) [2] and GSM 01.04 [1].

4 Generic MExE aspects

This section defines the common aspects of all MExE compliant devices, independent of MExE technology.

Considering the wide and diverse range of current and future technology and devices that (will) use wireless communication and provide services based thereon a one-size-fits-all approach is unrealistic. Instead the present document categorises devices by giving them different MExE classmarks. In this specification two MExE classmarks are defined:

- MExE classmark 1 - based on WAP (Wireless Application Protocol) [6] - requires limited input and output facilities (e.g. as simple as a 3 lines by 15 characters display and a numeric keypad) on the client side, and is designed to provide quick and cheap information access even over narrow and slow data connections.
- MExE classmark 2 - based on Personal-Java [3] - provides and utilises a run-time system requiring more processing, storage, display and network resources, but supports more powerful applications and more flexible MMIs. MExE Classmark 2 also includes support for MExE classmark 1 applications (via the WML browser.)

Future classmarks may require other Java-packages, APIs, and/or support of other features such as speech-recognition, video-I/O with online (de)-compression, minimal storage requirements, high-speed local communication, etc. but these are subject to future standardisation efforts.

Content negotiation allows for flexible choice of formats available from a server or adaptation of a service to the actual classmark of a specific client device.

Bi-directional capability negotiation between the MExE Service Environment and MExE device (including MExE classmark), supports the transfer of capabilities between the client and the server.

4.1 MExE classmark 1 (WAP environment)

The WAP forum has proposed designs for both the transport protocols on the wireless leg of the end-to-end connection (based on the Wireless Datagram Protocol (WDP), the Wireless Transaction Protocol (WTP), Wireless Transport Layer Security (WTLS) and Wireless Session Protocol (WSP)), as well as the client-side application environment, which revolves around a Wireless Markup Language (WML) browser supporting a Wireless Markup Language Script (WMLScript).

4.2 MExE classmark 2 (Personal Java environment)

This classmark characterises Java enabled devices with telephony specific extensions.

4.3 High level architecture

The following architectural model shows an example of how a GSM network uses standardised transport mechanisms to transfer MExE services between the MS and the MExE service environment, or to support the interaction between two MSs executing a MExE service. The same architectural model can be applied in 3G networks as well.

The MExE service environment could, as shown in Figure 1,consist of several service nodes each providing MExE services that can be transferred to the MS using standard Internet protocols. The MExE service environment may also

include a proxy server to translate content defined in standard Internet protocols into their wireless optimised derivatives.

For the versatile support of MExE services, the network shall provide the MS with access to a range of bearer services on the radio interface to support application control and transfer from the MExE service environment.

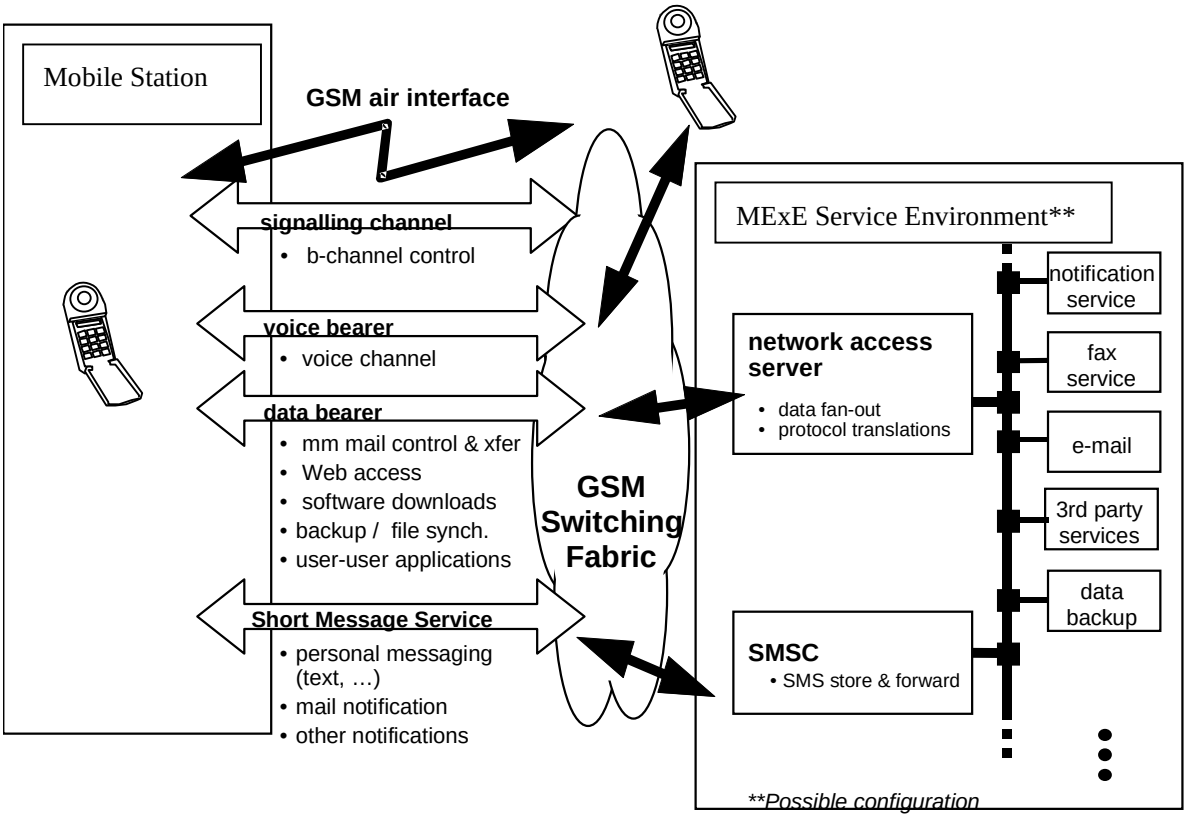


Figure 1: Generic MExE architecture

4.4 Capability and content negotiation

Interaction between the MExE MS and the MSE shall be supported by the use of the hypertext transfer protocol HTTP/1.1 [9], or an HTTP/1.1 derived protocol (e.g. WSP as defined in Wireless Application Protocol [6]). Communication between the MExE MS and the MSE supports:

- Capability negotiation

  The MExE MS connects to the MSE by using HTTP/1.1 or an HTTP/1.1 derived protocol. Capability negotiation between the MExE MS and the MSE only takes place for the first time after the MExE MS has connected to the MSE, and the MSE is informed about the MExE MS. Without this first initial contact from the MExE MS, the MSE has no knowledge of the MExE MS, and thereafter the MSE may connect to the MExE MS by using HTTP/1.1 or an HTTP/1.1 derived protocol.

  Capability negotiation represents the mechanism by which the MExE MS and the MSE interact to inform each other of the specific mechanisms, capabilities and support which each is able to provide or support within the scope of a MExE service interaction. The capability negotiation normally takes place prior to any content transfer between the two entities.

  Capability negotiation is used by the MSE to request the MS' MExE capabilities, to which a response may be returned by the MExE MS. Information is normally requested by the MSE and supplied by the MExE MS, however the MExE MS may also be informed by the MSE of its current view of the MExE MS's capabilities. The MExE MS may also spontaneously inform the MSE of its capabilities without initially being requested to send them (i.e. following a change in MExE support, such as removal of MExE MS from a docking station with its keyboard, mouse and monitor). The characteristics which may be requested and transferred between the MExE MS and the MSE during the capability negotiation are identified in subclause 4.4.1 Capability negotiation characteristics.

- Content negotiation

  Content negotiation represents the means by which the MExE MS and the MSE inform each other of the requested and available form of content. If needed, the content negotiation may take place following capability negotiation between the two. The methods for content negotiation are the basic HTTP/1.1. or WSP methods explained in [9] and [6].

  Content negotiation is used to select the best representation of an entity when there are multiple representations of the entity available from the MSE. The entity (e.g. a service, an image, etc) is located behind a URI, and the application in the MExE MS connects to the URI by using HTTP/1.1 or an HTTP/1.1 derived protocol. The best representation of an entity can be decided by the server (server-driven negotiation) or by the client application (agent-driven negotiation).

Both the capability and the content negotiation has the same purpose: to optimise the content according to client's capabilities. The term "content negotiation" has been used e.g. in the HTTP specification and the HTTP/1.1. and the WSP contain headers to perform the content negotiation. However, the capability negotiation in MExE aims at extending the basic HTTP and WSP methods for content negotiation. MExE terminal is free to use both the existing HTTP/WSP content negotiation methods and the new MExE capability negotiation methods.

The content negotiation transferred between the MExE MS and the MSE is identified in subclause 4.4.2 Client Capability Report onwards.

4.4.1 Capability negotiation characteristics

Method for capability negotiation is based on the CC/PP specification made by W3C, [16]. The properties and the actual schema is based on the WAP UAProf group specification [17]. The Composite Capability/ Preferences Profiles framework is intended to provide an efficient mechanism for enabling enhanced content and service negotiation through a standardised format for user agent profiles. The use of Resource Description Framework (RDF) in CC/PP allows for interoperable encoding of the profile metadata in XML and supports multiple vocabularies to provide for future extensibility. WAP UAProf is based on the CC/PP framework. The purpose of the UAProf is to specify:

- an RDF based schema and vocabulary for CC/PP in the context of WAP UAProf that includes the class definitions and semantics of attributes described in a user agent profile, and
- guidelines for schema extensibility to support a composite profile that enables future additions to the vocabulary and schema.

Not all capabilities have to be reported in the request to the server but instead, the client may point to an URL where the server may fetch the properties.

The generic set of capabilities which may be negotiated between the client and the server consists of the subsequently identified properties in the UAProf schema, [17]. A MExE terminal shall support (but not be limited to) the following properties in the UAProf schema for capability negotiation:

- MexeClassmark;
- MexeSpec;
- Vendor;
- Model;
- Screensize;
- ScreenSizeChar;
- ColorCapable;
- AudioInputEncoder;
- VideoInputEncoder;
- PointingResolution;

- CcppAccept-Language;
- Keyboard;
- SoftwareNumber;
- SupportedBearers.

It is not required that a MExE terminal shall send all the above properties together when sending a request, however it shall be possible for the MExE terminal to send one or more of the above properties if subsequently requested by the server, with user permission.

Generally, the combination of user profile and ME logic will determine the information sent in the capability negotiation from the MExE device to the MExE Service Environment. As an example, for the support of location information the user's profile controls if and when location information may be sent to the MExE Service Environment (e.g. never sent, always sent, only after user confirmation).

The capability negotiation process shall be used by the client to permit transfer of capabilities from the client to the server. By transferring its capabilities, the client will support efficient use of resources both over the radio interface as well as in the client or server. Capability negotiation shall be performed prior to transfer over the radio interface to verify as far as possible the ability of the client to support any services to be downloaded.

In order to transfer the capability information between the MExE MS and the MSE, CC/PP method is used with the schema defined in the WAP UAProf working group.

4.4.2 CC/PP over WSP (classmark1)

In classmark 1 the CC/PP is carried over by using CC/PP over WSP, [17].

4.4.3 CC/PP over HTTP (classmark2)

In classmark 2 the CC/PP is carried over by using CC/PP over HTTP, [15] or CC/PP over WSP, [17].

4.4.4 Client content capability report

The client may perform content negotiation capabilities to the server by using appropriate HTTP/1.1 or WSP request headers. The following methods are available for content negotiation:

- Client software (product): `User-Agent` header;
- MIME media types: `Accept` header;
- Character set: `Accept-Charset` header;
- Content encoding: `Accept-Encoding` header;
- Language: `Accept-Language` header.

There is no need for MExE to specify any tokens for content negotiation, as these headers are already defined in HTTP and WSP. The formats for these headers are specified in [9] and [6].

Example:

The following HTTP request reports that the name of client software is "GSM-Phone" version "1.0". The client accepts both compiled WML and compiled WMLScript, and supports both the English and Swedish as languages.

```
GET / HTTP/1.1
Host: www.company.com
User-Agent: GSM-Phone/1.0
Accept: application/x-wap.wmlc, application/x-wap.wmlscriptc
Accept-Language: en, sv
...
```

The basic format of the `User-Agent:` header is: `User-Agent: software-name/version.` A comment can be attached enclosed in parentheses to give more specific information. For example, operating system, display size, supported software extensions, script libraries, etc. The format of the comment is, however, not specified in [9].

4.4.5 Server role in capability negotiation

The server may request the capabilities of a client whenever required, and shall enquire of the client's capabilities prior to making each transaction resulting in a set of transfers to the client; the characteristics which may be reported in the client capability report are identified in the list above.

In server-driven negotiation the server signals to the client that the response entity was selected from a set of available representation. To do this, the server attaches the `Vary:` response header in the response to the client. The `Vary:` header includes a list of request headers. For example:

```
HTTP/1.1 200 OK
Vary: User-Agent, Accept-Language
Content-type: application/x-wap.wmlc
Content-language: en
...
```

Indicates that the entity is available for multiple languages and user-agents. The selected entity is the English version.

4.4.6 Client-driven negotiation

If the server cannot specify an optimal version for the client basing on the CC/PP sent over to the server, the server may also indicate to client which type of versions are available and let the client make the decision. This method is already available in HTTP1.1. In client-driven negotiation the client selects the best representation after having received an initial response from the server. The response from the server is a `300 Multiple Choices` response and contains a list of available representations. The selection of the available representations may be done automatically by the client application or by the (human) user from a menu.

It is noted that there is an implicit overhead of (at least) one additional round-trip delay with client-driven negotiation. The client-driven negotiation will always require an additional request/response iteration, due to the fact that the initial response from the server to the client's initial request may be a `300 Multiple choices` response, or an equivalent presentation of available choices. After the user has selected one of the available options, the client sends the request for the actual content to the server.

4.5 User profile

The user profile (which may consist of sub user profiles for a user) contains the characterisation of the MExE MS as defined by the user and service provider. Further, it is also possible for multiple users of a MExE MS to each have their own user profiles. The user profile is not unique to the MExE MS, and this clause identifies the usage and content of the user profile from a MExE perspective only, and does not identify the generic support of user profiles in general. Refer to UMTS 22.101 [14] for further details on the user profile.

4.5.1 Location of, access to, and security of, the user profile

As multiple user profiles may be defined, the user is able to set up or receive calls/connections associated with different user profiles simultaneously by securely activating a user profile (with each user profile being associated with at least one unique identifier). Refer to the Security clause for further details on user profile activation.

The user's characterisation of the MExE MS in the user profile may be modified at any time by the user and the service provider, and changes affected at the earliest possible opportunity.

The security clause shall apply to all user profiles at all times, whether activated or not

The user profile is securely managed by the MExE MS, and stored in a secure area of the MExE MS (either SIM or ME). The service provider may also retain the user profile in the network for service optimisation. User private data in the user profile should not be stored in the network.

The support of more than one user profile is not mandatory.

4.5.2 User profile and capability negotiation relationship

The user profile contains the user's preferences. Support of the user's preferences will depend on the capabilities of the device. If the capabilities change, then the degree of support of the user's preferences may change too.

The capability negotiation between the MExE terminal and the MSE, as shown in Figure 2, contains those user preferences which the device is able to support.

In this way the MSE will serve a MExE terminal with the lowest common denominator of the users preferences, the terminal capabilities and the provided service characteristics and support the user's preferences to the maximum degree.

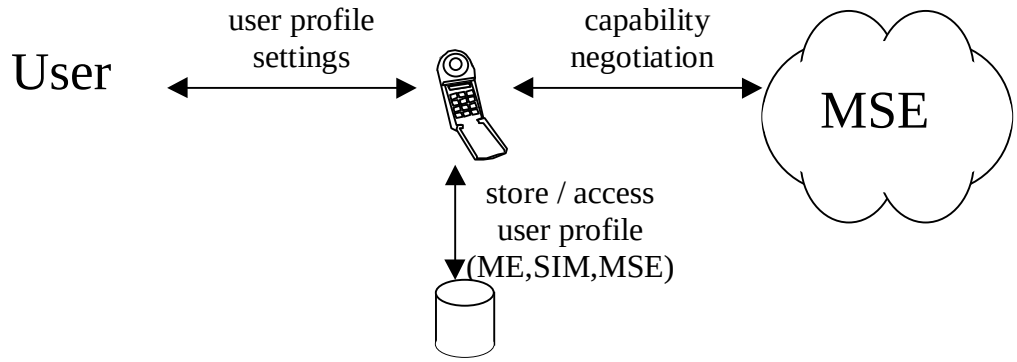


Figure 2: Model of user profile and capability relationship

4.5.3 Support of the user profile

The user profile acts as a repository (which is always available in the MExE MS) defining the MExE MS behaviour.

MExE preferences and personalisation are supported in the user profile (e.g. UMTS portability and support of VHE defined in [12] and other 22-series specifications), which in turn is based on the Composite Capability/Preference Profile (CC/PP) specification from W3C [16].

MExE preferences and personalisation may not only be recorded directly in the user profile as supported by CC/PP (the direct referencing mechanism), but may also be retrieved from a URL (the indirect referencing mechanism).

Generally, the user profile's CC/PP framework provides the mechanism for the standardised format of preferences, and its use of Resource Description Framework (RDF) permits the interoperable encoding of MExE preferences and personalisation. Future extensions will be supported by the W3C mechanism, allowing for evolution and development of MExE preferences and personalisation.

The set of preferences which are supported in the user profile consists of the following:

- user interface personalisation

  the user's personalisation of the user interface.

- service personalisation and management

  the user's generic service management information.

The coding and presentation of the above characteristics in the user profile is defined by the Composite Capability/Preference Profile (CC/PP) specification from W3C [16], and referenced by the MExE capability negotiation in subclause 4.4.

The following user preference information is supported by UAProf [17]. A MExE terminal shall support (but not be limited to) the following properties in the UAProf schema for user preference information:

- CcppAcceptLanguage User's preference for document language
- DownloadableSoftwareSupport User's preference for accepting downloadable software
- PreferenceForFrames User's preference for displaying frames

- PushMsgPriority User's settings for WAP Push message priorities

Also, there is support for indicating terminal's capabilities related to UI features, e.g. capability for displaying images or frames, as well as capability information about input and output methods.

E.g. the following preference information is for future consideration:

- Maximum size and time of transfer and other preferences related to transferring the content.
- User's preferences for input/output methods and other preference parameters related to user interface management.
- User's preferences for memory usage.
- Service-related parameters (eg. voice mail numbers, etc.).

4.6 User interface personalisation

The MS interface consists of the buttons, menus, screens and MMI as designed and provided by the MS manufacturer; the nature of this MS interface is naturally evolving, MS specific and proprietary to the individual manufacturers of the industry. This interface is the one normally seen by the user in normal operation of his MS. This specification does not place any requirements or limitations on the individual manufacturers' MS interface.

The MExE MMI, in turn, is the interface available to the user to support MExE services and functionality on the MS. The nature of the MExE MMI interface, like the normal MS interface described above, is not standardised in any way, to allow for manufacturer innovation, cater for evolving market needs, and permit manufacturer differentiation. The MExE MMI, depending on different manufacturer implementations, may consist of the normal MS interface, the normal MS interface with modifications, a different interface to the normal MS interface, or some combinations thereof etc. MExE services operate within, and using the capabilities of, the MExE MMI.

User interface personalisation consists of two parts. The first part refers to the user's ability to request, and verify, the preferred changes to the user interface; thus the user's preferences, as supported by the specific MS, require to be recorded. The second part refers to the MExE MS's support of the user's preferences for the interface, within the capabilities of an MS. By defining the user interface personalisation to consist of two stages, the preferences which have been recorded by the user may be transferred (as part of the user profile, eg. CcppAcceptLanguage and/or PreferenceForFrames), and thereby provide portability of the user's preferences.

4.6.1 MExE user interface personalisation

Personalisation of the user interface offers the MExE Service Environment and or the user, the ability to inform the MExE MS of the desired extent of personalisation. All support of the user interface personalisation is optional, not mandatory on any class of MS, and subject to the capabilities of the MS. Depending on the capability of the MS, the personalisation may be fully supported, partially supported, interpreted or ignored.

Personalisation of the user interface is not restricted to modifying the appearance of the MMI, but also the modification of MMI parameters (e.g. programming of the voicemail number). The user's personalisation of the interface is retained as part of the user profile.

4.6.2 Support of MExE user interface personalisation

MExE user interface personalisation is supported via the preferences in the user profile, which in turn is based on the Composite Capability/Preference Profile (CC/PP) specification from W3C [16].

User interface personalisation may not only be reported in the CC/PP request to the server (the direct referencing mechanism), but indeed the client may point to a URL (the indirect referencing mechanism) from where the user interface personalisation preferences may be retrieved.

Generally, the user profile's CC/PP framework provides the mechanism for the standardised format of preferences, and its use of Resource Description Framework (RDF) permits the interoperable encoding of user interface personalisation. Future extensions will be supported by the W3C mechanism, allowing for evolution and development of MExE user interface personalisation.

4.7 Management of services

The MExE ME shall be capable of supporting services in a standard (WAP or Java) execution environment independently of the MExE ME manufacturer. Management of services provides the user with the capability to:

- discover services;
- configure services ;
- control the transfer and execution of services;
- terminate or suspend executing services

on his MExE MS.

4.7.1 Service discovery

A MExE user is able to request (or be informed about), the range of MExE services available from the MExE server to which it is connected. Support for the request, and transfer, of information on MExE services from the MExE server is primarily provided by the use of the capability negotiation mechanism.

All services available in the network continue to be available to the user, in addition to MExE services.

An example of an alternative means of possibly receiving information on MExE services, is the use of an application on the MExE MS which the user interrogates to provide services information (from various sources), and which in turn then obtains such information and presents it to the user. Such an example is not subject to standardisation.

4.7.2 Service configuration

- The user controls whether a service transferred to the MExE MS is also automatically configured in the MS. Configuration of a service may result in changes to user interface using icons, browsers or menus as applicable depending on the capability of the MExE MS to support them. The service name may be contained in the package in which it was received (i.e. a JAR file or script), assigned by the user during configuration, or some other means. If service automatic configuration is enabled, the user is notified of the automatic service configuration once it is completed.
- In the event that there is no authorisation for the automatic configuration of a transferred service, the user is notified that it was not configured.
- The user controls which services may be automatically configured.
- Subsequent user modification of a service's configuration (e.g. by modification of use profile settings) shall take effect at the earliest possible opportunity thereafter.

4.7.3 Service management

- The MExE MS shall support the ability to determine which services are transferred to, resident, configured or executing on the MS. The information relating to the services includes (as a minimum) the name and version.
- The user controls which services are permitted or denied to be transferred , resident, configured or executing on the MExE MS via the user profile, eg. DownloadableSoftwareSupport. The user profile permits characteristics such as security level, identification of specific services etc. to manage services on the MExE MS.

4.7.4 Service termination

- A service may terminate by itself, or be terminated by the provider of the service or by the user. The user is in charge of the service, except when the service provider may appropriately control the service as part of user support.
- The mechanism for terminating a service is out of scope of standardisation and shall be provided on a service by service basis by the provider of the service.

4.7.5 Service deletion

- A service may be deleted (i.e. removed) from the MExE with the authorisation of the user. The deletion may be initiated by the authoriser of the service or by the user.

4.8 Virtual home environment

Virtual Home Environment (VHE) (see [11] and [12]) is defined as a concept for personalised service portability across network boundaries and between terminals. MExE is identified by VHE as one of the mechanisms which may be used to support VHE.

VHE presents the same look and feel depending on the capabilities of the serving network and the capabilities of the terminal in use, with any limitations in the terminal being indicated to the user when the terminal is changed. Services that are available/unavailable will be displayed in a way that is familiar to the user no matter what class of terminal is used.

With the general ability of the MExE requirements and functionality to support VHE requirements, MExE shall support the Virtual Home Environment system concept.

The characteristics of the VHE (to reflect any user or home environment modification of the user's VHE) shall be stored as part of the user profile. Access and modification of the user profile) to support:

- identification of subscribed services;
- service personalisation;
- user interface personalisation;

shall be supported by the MExE APIs (when available).

4.9 User control of application connections

- This subclause addresses the generic aspects of connection control supported by both WAP and Java classmark MExE MSs.
- In order to allow the user to maintain control over connections on his MExE MS and the ability to initiate connections, the user shall be able to terminate or suspend any active connection associated with an application in the MExE environment of the MExE MS. The user shall be able to obtain information on all connections associated with applications on the MExE MS. Behaviour of the application following termination or suspension of its connection is undefined.
- The specific support of connection control by WAP and Java classmark MExE MSs is identified in subsequent subclauses, the security aspects of connection control are identified in the security subclause, and the user control of connection authorisation is identified in the user profile subclause.

4.10 Journalling of network events

To support the user in monitoring and maintaining a record of (potentially chargeable) network events initiated by services in the MExE environment, it shall be possible for the user to request the MExE MS to maintain a record of network events initiated by services on the MExE MS. Support of such journalling is mandatory.

Network events for the purposes of journalling, are defined as events which result in the origination of voice or data connections by a service in the MExE environment of the MExE MS. Examples of such events are:

- Sending an SMS message;
- Sending an USSD message;
- Initiating a circuit switched connection;
- Initiating a packet switched connection;

- Sending data over a packet switched connection.

The above list is not exhaustive, but includes any (potentially chargeable) network event initiated by services in the MExE environment.

The user shall be able to activate and deactivate the journalling of network events in a secure manner. The length, format and longevity of the journal is undefined and subject to manufacturers' discretion.

The journal shall be managed by the ME, and not be accessible by MExE executables.

4.10.1 Journalling requirements

The terminal shall support the logging of network events. The user shall determine whether logging is in operation or not and the events that are logged. The size and format of the log is not the subject of this specification.

4.11 User notification

It is recommended that the device should clearly display an indicator whenever network activity is in progress.

Ideally, this would be an icon on the phone's screen which is displayed whenever the device is sending/receiving SMS, USSD, data call, voice call, or packets.

However, there are certain cases when this indicator need not be displayed, especially if it is obvious by some other means that the device is performing network actions.

4.12 Quality of Service

Quality of Service (QoS) [28] is seen by the end user as a measure of the amount of network resources given to an application by the underlying network. The network may employ a number of QoS mechanisms, but the end user / MExE executable is not involved in these. The end user / MExE executable requires an interface into the network QoS through a visible set of standard parameters.

A QoS aware MExE executable may request a QoS from the network at the beginning of a QoS session. Changes in the level of QoS provided shall be notified to the end user / MExE executable. An end user may request a change in the QoS through the MExE MS MMI. A MExE executable may have several QoS streams open simultaneously.

The MExE executable shall be able to dynamically request a change in the level of QoS at connection setup request or subsequently during the connection. The end user / MExE executable may receive a rejection to a QoS modification request, upon which the end user / MExE executable must be notified.

The end user's service level QoS subscription parameters are stored in the network, they identify the maximum permissible QoS that a user may negotiate with the network. Several QoS subscriptions may be possible for one user. MExE is neither aware nor able to determine or modify the end user's service level QoS subscriptions.

For MExE devices supporting bearers defined by QoS, the MExE execution environment shall support QoS management. QoS management may be available directly to the MExE executables themselves, or to the MExE environment.

5 WAP MExE devices

WAP MExE devices shall be based on the WAP specifications [6]. In addition to the base specifications in [6], further developments made in the WAP specifications shall form part of this MExE specification.

WAP MExE devices shall implement the WAP version as specified in reference [6], or a later version, under the condition that the version of WAP is backward compatible with the version specified in reference [6].

The existing WAP specification covers security, creation and transfer of WAP executables and content, access, and execution.

5.1 High level architecture

The WAP architecture provides a scaleable and extensible environment for application development for mobile communication devices. This is achieved through a layered design of the entire protocol stack.

The key features of WAP include:

- Markup language (WML) and a script language (WMLScript) designed to create applications on the small displays of handheld devices. WML does not assume a QWERTY keyboard and a mouse is available for user input. Unlike the flat structure of HTML documents, WML documents are divided into a set of well defined units of user interactions. One unit of interaction is called a card, and services are created by letting the user navigate back and forth between cards from one or several WML documents. WML has a smaller set of markup tags that makes it more appropriate to implement in handheld devices, than, say, HTML.

- Light-weight protocol stack to minimise the required bandwidth and to guarantee that a maximum number of wireless network types can run WAP applications. For example, GSM SMS/USSD, circuit switched data (CSD), and GPRS.

- A framework for Wireless Telephony Applications (WTA) allows access to telephony functionality such as call control, phone book and messaging from within WMLScript scripts. This allows operators to develop telephony applications integrated into WML/WMLScript services.

Since WAP is based on a scalable layered architecture, each layer can develop independently of the others. This makes it possible to switch onto new bearers, to use new transport protocols, without major changes in the other layers.

5.2 Optionality

Mandatory and optional components of WAP are specified in the WAP specifications. Services and applications shall be able to determine the presence of optional parts of the functionality.

5.3 Call control

WAP telephony services are written in WML and WMLScript. The WAP Telephony API (WTAI) exposes telephony functions to service authors as a set of libraries. The WTAI function libraries can be accessed from WML as URIs, and from WMLScript as script functions. The following libraries have been specified:

- Public library
  This includes functions that are available in all networks, and can be provided by any third party service provider; and not only the network operator. The user must acknowledge the function before it is carried out. Functions have been specified, which can be used e.g. to initiate a mobile originated call, send DTMF tones and add phonebook entry.

- Network Common library
  This includes functions that are available in all networks, and can be provided only by the network operator. E.g. functions for advanced call control, accessing the phonebook, and sending and reading network text (SMS) have been specified.

- Network Specific library
  Functions that are only available in certain types of networks, and can be provided only by the network operator. For GSM, e.g. functions for call reject, call hold, call transfer, multiparty, getting location information and sending USSD have been specified.

The WML and WMLScript author uses the WTAI libraries to create web services for mobile phones with telephony capabilities.

Call control shall be performed using WTA authenticated scripts.

5.4 Local phonebook

WAP Telephony API (WTAI) is used to access the information stored in the phonebook on the ME or the SIM. Phonebook entries consist of name, number and identity. Phonebook entries can be read, written, deleted, and searched for.

5.5 Services

WAP is a general purpose application based on World Wide Web (WWW) technologies and philosophies. Many services can be provided to both WAP clients and traditional WWW clients, from the same server. Services are created based on the same information space. The major difference is the user interface. The user interface of WAP services is realised by the Wireless Markup Language, WML [6], and has a menu tree oriented structure, instead of the traditional flat structure of HTML pages.

Typical WAP services provided to mobile phones may include (this list is not exhaustive):

- News
- Weather information
- Package Tracking
- Stocks
- Telephony Services
- Time Tables
- Access to corporate databases
- Sports

5.5.1 User interface

The user interface of WAP services is realised by the Wireless Markup Language, WML [6]. WML does not define the user interface itself, the implementation of the browser defines how the WML data is presented to the user (e.g. hyperlinks are blue and underlined). The script language, WMLScript [6], may be used to enhance the standard browsing and presentation facilities of WML with behavioural capabilities, and to access the device and its peripheral functionality.

5.5.2 Access points

Services may be hosted on standard HTTP servers and can be created with proven technologies; CGI, Java Servlets. URLs are used to address services.

The WAP network topology is shown in Figure 3.

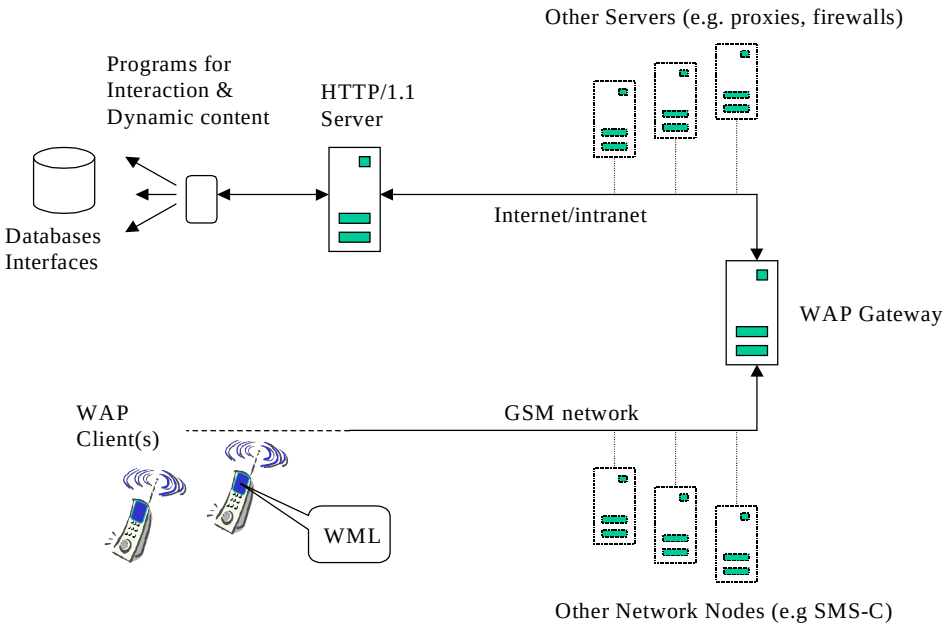


Figure 3: WAP network topology.

Mobile phones access services by sending a request with a URI to the WAP gateway. The URI is used to identify the origin server on which the service is available. The request is sent from the mobile phone by the WAP protocols over one of the available bearer networks. The WAP Gateway is a WAP to HTTP/1.1 proxy that translates the WAP request into an HTTP/1.1 request (from binary form to text). The HTTP/1.1 request is passed on to the server identified by the URI.

The HTTP server may have multiple access points to various databases and other services available in the infrastructure network. Once the request has been serviced a response is sent back to the WAP Gateway, which in turn translates it into a WAP response (from text to binary form) and sends it down to the mobile phone.

Note that WAP does not specify anything "behind" the WAP Gateway. However it is assumed that the origin server is an HTTP/1.1 server, and that the WAP Gateway has access to the TCP/IP network on which the origin server is hosted.

5.5.3 Transferring

The core of WSP [6] is a binary version of the Hypertext Transfer Protocol - HTTP/1.1 [9]. The core function of WSP is the same as for HTTP/1.1. A client sends a request to the server using an appropriate request method with a URI and information about the client. The server responds with a status code and possibly (if success) the requested content.

There is a differentiation between an origin server and a WSP server. The origin server is where the content is stored, and the WSP server is where the WSP session terminates. The WSP server is also typically the WAP gateway.

In addition to the basic HTTP/1.1 function, WSP has some functions that can not be found in HTTP/1.1, they are:

- Session Establishment and Management
  Before a request is sent, the WSP client can establish a session with the server. During session establishment the client and server exchange static headers. The header are cached for the duration of the session, thus they need to be sent in every single request within the session. Static headers may be: `Accept` headers, `User-agent` header, etc. In addition, capabilities such as supported optional protocol functions, the maximum service data unit the protocol can handle, the maximum number of simultaneously outstanding requests, supported header code pages, etc. can also be exchanged during session establishment.

- Header encoding
  WSP is using a compact binary header encoding to minimise the number of bytes sent over the air.

- Asynchronous transactions
  WSP allows for multiple asynchronous transactions, that is, unordered transactions.

- Transaction Abort
  WSP support abortion of an outstanding transaction.

- Datagram transport
  WSP together with the helper protocol Wireless Transaction Protocol, WTP [9], can run over a datagram transport such as SMS or UDP. The WDP can also be used for non-IP bearers.

- Push
  WSP supports the push of data from server to client. This can be done within and outside of a session. It can be done with and without acknowledgement from the client. Push of indications down to mobile phones is an essential function many wireless applications.

5.5.3.1 WSP and HTTP/1.1 Proxy Function

The WAP Architecture is a client-proxy-server architecture. The client is typically a mobile phone, the data gateway is the WAP Gateway and the server is the origin server (a standard HTTP server). The WAP Gateway translates the binary WSP header into text formatted HTTP/1.1 headers and passes them on to the origin server. In the opposite direction the WAP Gateway translates the text formatted HTTP/1.1 header into binary WSP headers. If the WAP Gateway receives a header it does not recognise it simply passes it on as an unknown header. Unknown headers that are not part of the WSP Header Code page or Extended code pages (negotiated at session establishment) are sent in plain text for the client to interpret as best it can.

6 Java MExE devices

Java MExE devices shall be based on the MExE API for Java, which defines the required and optional components of Java APIs that shall be used to realise a MExE compliant device.

The MExE API for Java primarily defines the functions available to a Java-based MExE device such that services (specified in the form of Java classes and interfaces) can control such a device in a standardised way.

Many aspects of the MExE API specification are optional. Services and applications shall be able to determine the presence of optional parts of the functionality. When optional parts of the functionality are implemented, the MExE API shall be supported.

6.1 High level architecture

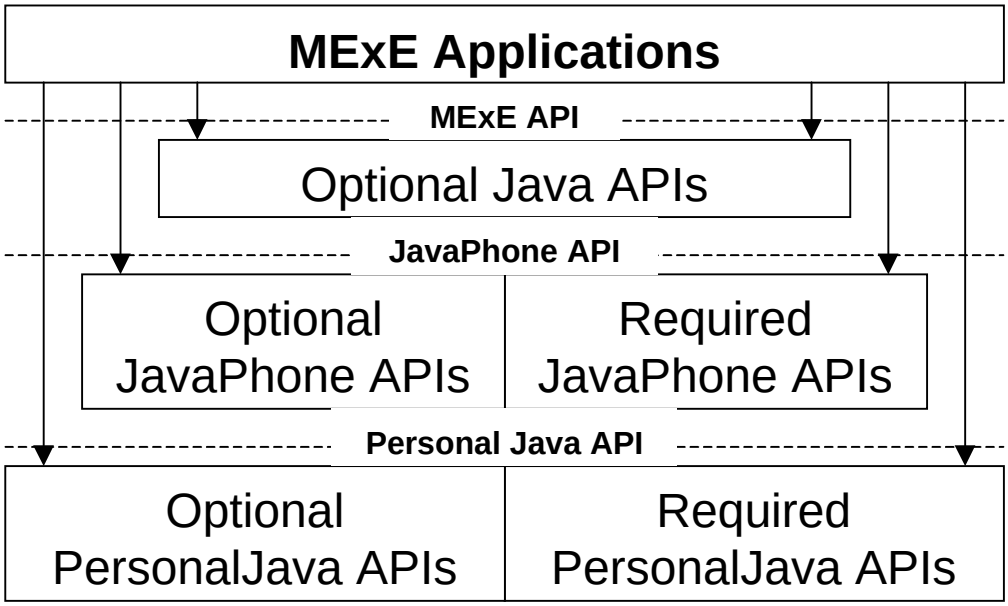


Figure 4: Basic functional architecture of a Java MExE device

The functional architecture of a Java MExE device is shown in Figure 4. Java applets, applications, and services access functionality via the MExE API for Java. The MExE API is based on a combination of optional Java APIs approved by Sun Microsystems and the Wireless Profile of the JavaPhone API [4] as defined by the JavaPhone Expert Group. The JavaPhone API is based on the PersonalJava API [3] defined by Sun Microsystems.

6.2 High level functions

6.2.1 Optionality

The use of Java encourages development of modular interfaces and minimal required functionality. Additional functionality is provided by optional APIs specified in terms of the Java language. In general, optionality is specified in terms of Java packages. Packages are containers for the highest level of functionality in the Java language. In some cases, optionality is specified in terms of Java classes and interfaces. Classes and interfaces are elements contained inside packages.

The following Table 1 specifies the Sun Microsystems defined optionality of the Wireless Profile of the JavaPhone APIs. Within some of the packages, certain classes and methods may be individually specified as optional by the JavaPhone API specification.

Where a mandatory package is identified, it is implicit that any packages called by that mandatory package are also mandatory.

Table 1: Optionality of the Wireless Profile of the JavaPhone APIs

JavaPhone API	Java package	Optionality
Addressbook	Javax.pim.addressbook	Mandatory
User Profile	Javax.pim.userprofile	Mandatory
Calendar	Javax.pim.calendar	Mandatory
Network	Java.net	Mandatory
Datagram	Javax.net.datagram	Mandatory
Power Monitor	Javax.power.monitor	Mandatory
Power Management	Javax.power.management	Optional
Install	Javax.install	Optional
Communications	Java.comm	Optional
SSL	Javax.net.ssl	Optional
JTAPI Core Package	Javax.telephony	Mandatory
JTAPI Core Capabilities Package	Javax.telephony.capabilities	Mandatory
JTAPI Core Events Package	Javax.telephony.events	Mandatory
JTAPI Call Control Package	Javax.telephony.callcontrol	Optional
JTAPI Call Control Capabilities Package	Javax.telephony.callcontrol.capabilities	Optional
JTAPI Call Control Events Package	Javax.telephony.callcontrol.events	Optional
JTAPI Phone Package	Javax.telephony.phone	Optional
JTAPI Phone Capabilities Package	Javax.telephony.phone.capabilities	Optional
JTAPI Phone Events Package	Javax.telephony.phone.events	Optional
JTAPI Mobile Package	Javax.telephony.mobile	Mandatory
	Java.math	Optional
	Java.rmi	Optional
	Java.rmi.dgc	Optional
	Java.rmi.registry	Optional
	Java.rmi.server	Optional
	Java.security	Optional
	Java.security.interfaces	Optional
	Java.sql	Optional
	Java.io	Optional

6.2.2 Required and optional PersonalJava APIs

Java MExE devices shall support the PersonalJava specification [3]. The PersonalJava APIs provide a standardised and readily implementable execution environment as a means for applications, applets, and content:

- to access and personalise the user interface via the java.awt packages;
- to utilise both Internet and Intranet connections via the java.net package.

6.2.3 Required and optional JavaPhone APIs

The JavaPhone APIs extend the PersonalJava APIs to provide functionality unique to telephony devices. Java MExE devices shall support the Wireless Profile of the JavaPhone API specification [4]. Java MExE devices shall support all APIs specified as required by the Wireless Profile in the JavaPhone API specification. All APIs that are optional in the Wireless Profile shall be optional in Java MExE devices.

6.2.3.1 Application installation

Java MExE devices shall support the following JAR file manifest entries (as described in the JavaPhone specification) as described below:

- Implementation-Title

  the Implementation-Title shall be used in any textual description of the application which is displayed in the UI element used to launch the application. E.g. the text displayed with an icon.

- Main-Icon

if icons are used as elements to launch the application, then the icon file within the JAR file named by the Main-Icon attribute shall be displayed, and may be scaled if desired.

- Main-Class and Class-Path

  when the application is launched, the MExE Java VM shall be supplied with the classpath and shall call the main() method in the class named by the Main-Class attribute.

6.2.3.2 Power

Java MExE devices shall support the Power Monitor package (javax.power.monitor) as specified by the JavaPhone API to access the power level of the device and receive notifications concerning changes in power states.

Note that the Power Monitor package does not specify the minimum required events that should be generated under certain circumstances. A MExE Java device shall at least implement the following event generation:

- BatteryCritical

  shall be generated when the battery is at a critically low level.

- BatteryNormal

  shall be generated when the battery is no longer low.

All the other event generation should be supported by the implementation.

6.2.4 Required and optional MExE APIs

A Java MExE device shall not be required to support any other Java APIs.

A Java MExE device may optionally support any other Java APIs which comply with the MExE security requirements in Table 3, such as:

- OCF SmartCard API OpenCard, available from [21]. If the ME supports smartcards other than the SIM, and the smartcard is open to 3rd party applications, then the opencard.core.terminal section of the OpenCard API may be used to access the card.

6.2.5 Mandated services and applications

6.2.5.1 WAP browser support

To provide backward compatibility to MExE classmark 1, i.e. allow access to services designed for MExE classmark 1 devices, classmark 2 devices must feature a pre-installed or pre-loaded WAP browser that is capable of rendering at least the following content formats:

- tokenised WML documents ("WML decks");
- WMLscript bytecode;
- A WAP service in a MExE classmark 2 MS shall execute in the same manner as it executes in a MExE classmark 1 MS;
- A WAP service in a MExE classmark 2 MS shall execute in the same manner as it executes in a MExE classmark 1 MS.

Other WML formats (such as textual WML documents or textual WMLscripts) are optional.

The pre-installed/pre-loaded WAP browser may be upgraded, replaced or extended by transferring, a replacement, extension or plug-in mechanism to the MS. Depending on user preferences identified in the user profile and the terminal capabilities, the pre-installed or pre-loaded WAP browser may be overwritten or the new browser stored in a different location.

6.2.5.2 Network protocol support

Support for network protocols in Java MExE devices is specified in the following Table 2:

Table 2: Support for network protocols

Protocol	Optionality
HTTP/1.1 [9]	Mandatory
HTTPS	Mandatory
Gopher	Optional
ftp	Optional
mailto [25]	Mandatory
File	Optional

6.2.6 Datagram recipient addressing

The Datagram API (as specified by JavaPhone) may resolve server/service names using a name resolution service. The MExE Java implementation shall at least be able to resolve names using the addressbook.

The implementation of the Datagram API shall support use of the addressbook to resolve names: The addressbook entries shall be searched for items whose name matches the server/service name by the Datagram API implementation in order to resolve names to actual addresses. It shall look for an item of type "FN" with a value equal to the server name, using the following syntax:

server_name: *mostchars
e.g. "The PIZZA Hut".

Then it shall search for aggregate fields named by the service name concatenated to "X-DATAGRAM-", the field name which is named according to the following syntax:

field_name = "X-DATAGRAM-" service_name

service_name = *mostchars

Then it shall use the value of the aggregate attributes which use this field. These strings shall specify in order, the preferred and available protocols and addresses for the named server/service. The string value in each X-DATAGRAM… field shall be formatted as so (inspired by RFC 1738 [22]):

address = primary_name "{" primary_addr "}" *[secondary_name "{" secondary_addr "}"]
primary_name = wdp_name | udp_name | sms_name | any_name

primary_addr = internet_addr | phone_number | port | httpurl | *unreserved

secondary_name = sms_name | url_name | sms_center_name | ip_name | any_name

wdp_name = "WDP"

udp_name = "UDP"

sms_name = "SMS"

url_name = "URL"

sms_center_name = "CENTER"

ip_name = "IP"

any_name = 1*alphadigit

secondary_addr = internet_addr | phone_number | port | httpurl | *unreserved

internet_addr = hostname | hostnumber

phone_number = *phonechar

httpurl = "http://" host ["/" hpath ["?" search]]

```
host          =  hostname | hostnumber
hostname      =  *[ domainlabel "." ] toplabel
domainlabel   =  alphadigit | alphadigit *[ alphadigit | "-" ] alphadigit
toplabel      =  alpha | alpha *[ alphadigit | "-" ] alphadigit
hostnumber    =  digits "." digits "." digits "." digits
port          =  digits
hpath         =  hsegment *[ "/" hsegment ]
hsegment      =  [ "~" ] *[ uchar | ";" | ":" | "@" | "&" | "=" ]
search        =  *[ uchar | ";" | ":" | "@" | "&" | "=" ]
lowalpha      =  "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
                 "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
                 "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
                 "y" | "z"
hialpha       =  "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
                 "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
                 "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
alphadigit    =  alpha | digit

alpha         =  lowalpha | hialpha
phonechar     =  "+" | digit | "#" | "*" | "C" | "D" | "c" | "d"
digits        =  1*digit
digit         =  "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                 "8" | "9"
safe          =  "$" | "-" | "_" | "." | "+"
extra         =  "!" | "*" | "'" | "(" | ")" | ","
hex           =  digit | "A" | "B" | "C" | "D" | "E" | "F" |
                 "a" | "b" | "c" | "d" | "e" | "f"
escape        =  "%" hex hex
unreserved    =  alpha | digit | safe | extra
uchar         =  unreserved | escape
mostchars     =  unreserved | " "
```

As a minimum, the following transport/bearer combinations shall be supported if the device supports the bearer/transport combination:

Transport/bearer combination	Value of primary_name	Syntax of primary_addr	Value of secondary_name	Syntax of secondary_addr
WDP over SMS	WDP	port	SMS	phone_number
SMS	SMS	phone_number		
WDP over HTTP	WDP	port	URL	httpurl
UDP over IP	UDP	port	IP	internet_addr

For example:

```
WAP Datagram over SMS:  "WDP=358,SMS={+358503583862}"
SMS:                    "SMS={+358503583862}"
WDP over HTTP:          "WDP={1234},URL={http://somewhere.on.the.web/path/name}"
UDP over IP:            "UDP={1234},IP={147.23.120.2}"
```

If the service centre number is specified for SMS, then the secondary_name shall be "CENTER" and secondary_addr shall be the phone number. If it is not present, then it shall be derived from the default service center.

7 Charging

7.1 Generic charging support

The standard GSM/UMTS charging records contain information sufficient to associate bearer usage and SMS/USSD messages with a subscriber.

Third party service providers and/or service providers may define charging regimes for MExE services (e.g. on a MExE or WAP server), however they are outside the scope of standardisation.

7.2 WAP charging support

The WAP protocol suite in [6], with upgrades as identified in this specification, does not specify mechanisms for charging (e.g. charging records) or subscription management. WAP is bearer independent and is running as an application on top of the bearer network. However the WAP architecture suggests that appropriate charging information can be collected in the WAP Gateway; the point of convergence for all WAP traffic.

The WAP security protocol can be used for authentication of the subscriber.

7.3 Java charging support

MExE Java devices do not require any additional specific charging (e.g. charging records) or subscription management. Java usage of network resources is bearer independent and runs as applications on top of the bearer network.

8 Security

8.1 Generic security

In order to manage the MExE and prevent attack from unfriendly sources or transferred applications unintentionally damaging the MExE device a security system is required. This section defines the MExE security architecture.

The basis of MExE security is:

- a framework of permissions which defines the permissions transferred MExE executables have within the MExE MS;
- the secure storage of these permissions (and permission type as defined in subclause 8.3);
- conditions within the execution environment that ensure that MExE executables can only perform actions for which they have permission.

The MExE permissions framework is defined in 3G TS 22.057 and is as follows (there is no implied hierarchy):

- MExE Security Operator Domain (MExE executables authorised by the HPLMN operator);
- MExE Security Manufacturer Domain (MExE executables authorised by the terminal manufacturer);
- MExE Security Third Party Domain (trusted MExE executables authorised by trusted third parties);
- Support for the three domains is mandatory;
- Untrusted MExE executables are not in a specific domain, and have very reduced privileges as described in subclause 8.2.

8.2 MExE executable permissions

The following Table 3 specifies the permissions of operator, manufacturer and third party security domains in the order of restriction.

The actions listed in the security Table 3 are generic actions. These actions can only be performed by MExE executables via application programming interfaces (APIs) (which are intrinsically part of the MExE implementation) .tThe security restrictions shall apply to MExE executables whether the API functionality is called directly or indirectly by the MExE executable.Explicit user permission is required for all actions by MExE executables in all domains. Types of user permission are defined in subclause 8.3

Untrusted MExE executables are not permitted access to any actions which access the phone functionality (phone functionality includes all the actions in Table 3) except for the exceptions identified in 8.2.1.

Actions available using interfaces giving access to the phone functionality (either in existence at the time of approval of this specification or not) that are not listed in the security Table 3 shall be categorised into one of the groups in the security Table 3 by comparing its action against the groups in order as they are listed in the Table 3. If an action can be categorised into a more restrictive group near the top of the table, then it shall not be again categorised into another, less restrictive, group further down in the table. E.g if a new action eventually results in forwarding a call, it shall be categorised into Network access. If the action is totally new, it shall be categorised into some of the groups by comparing its functionality to the group description below and by comparing with the list of actions listed in the table within the group.

1. Device core function access includes functions, which are an essential part of the phone functionality .

2. SIM smart card low level access includes functions, which allow communications at the transport service access point (send and receive application protocol data unit).

3. Network security access includes all functionalities which relate to CHV, CHV2, UNBLOCK CHV and UNBLOCK CHV2 (verification, management, reading or modifying), GSM authentication, GSM ciphering.

4. Network property access includes functions, which enable the management of operator-related data parameters and network settings.

5. Network services access includes all functionalities which result in or need interaction via the operator´s network.

6. User private data access includes all functionalities which relate to management, reading or modifying of data that the user has stored in the MS including user preferences.

7. MExE security functions access includes all functionalitiess which, through an API relate to certificate handling in the MS, end to end encryption, signed content, hashing, access to public, private, secret keys stored in the MS or in a smart card.

8. Application access includes the functionalities which relate to launch provisioned functionality, MExE executables, external executables (SIM tool kit application,…) usage.

9. Lifecycle management includes the functionalities which are needed for installing or removing MExE executables in the MS.

10. Terminal data access includes the functions which relate to accessing terminal data, i.e. not user data.

11. Peripheral access includes the functionalities related to peripherals other than user interface peripherals usage through a high level software application interface.

12. Input output user interface access includes the functionalities related to the user interface and user notification means usage.

Table 3: Security domains and actions

	MExE Security Domains		
Actions	**Operator**	**Manufacturer**	**Third Party**
Device core function access 1. Start/stop radio 2. Turn on/off device 3. Write time and/or date 4. Activate a user profile 5. Modify a user profile	No		
SIM smart card low level access[11] 1. Send APDU **2.** Slot management (power on/off, reset, port lock…)	No		
[11] *– Access to SIM is provided using more high level API as phonebook, application launching*			
Network Security access 1. Run algorithm **2.** Verify CHV/2 or UNBLOCK CHV/2 **3.** Activate/deactivate CHV **4.** Modify CHV/2	No		
Network property access 1. Get IMSI 2. Get home network 3. Select network	Yes	No	
Network services access 1. Initiate a voice/data connection [3] 2. Accept a voice/data connection [3] 3. Call forward [4] 4. Multiparty call [4] 5. Call deflection [4] 6. Explicit call transfer [4] 7. Terminate an existing connection 8. Hold an existing connection 9. Resume an existing connection 10. Send point-point message (e.g. SMS, USSD) [4] 11. Generate DTMF 12. Query network status 13. Get signal level 14. Get call list 15. QoS management	Yes		Yes [6]
[3] *– A network connection may be via any supported bearer service* [4] *– Multiparty, deflection, and explicit call transfer shall be permitted only to numbers explicitly supplied by the user to the MExE Executable. Modification of call forward numbers stored in the network shall only be permitted to numbers explicitly supplied by the user to the operator.* [6] *– The Third Party domain's permission to access the networking action depends on the provisioning mechanism as described in subclause 8.15*			
User private data access [1] 1. Read 2. Write 3. Get properties 4. Delete 5. Get Location Information 6. Read stored SMS 7. Delete stored SMS 8. Modify user preferences	Yes[2] Yes[2] Yes[2] Yes[2] Yes[2] Yes[2] Yes[2] Yes[7]		

Actions	MExE Security Domains		
	Operator	**Manufacturer**	**Third Party**
[1] – *User private data includes user files, phonebook, etc located on the MS.* [2] – *The user shall be able to specify data access permissions within the capabilities of the device. It is not applied to user preferences* [7] – *Trusted applications only have permission to modify user preferences, and not to activate or de-activate them. The user shall be able to specify for each domain, the preferences that applications in that domain can access. All other preferences shall not be accessible to that domain. The default shall be that there is no access. Single action user permission is the only type of user permission that shall be possible for changes to User Preferences.*			
MExE security functions access 1. Install a certificate for a given domain 2. Uninstall a certificate for a given domain 3. Replace a certificate for a given domain 4. Data encryption API 5. Verify a signature API 6. Compute a digital signature API 7. Hash a content API 8. Non repudiation API	Yes [5] Yes [5] Yes [5] Yes Yes Yes Yes Yes		
[5] – *Only the organisation whose public key is certified (or the organisation that certified the public key) can add, delete or replace a particular certificate.*			
Application access **1.** Get application list **2.** Launch an application **3.** Get application status **4.** Stop, suspend, resume an application	Yes[8] Yes[8] Yes[8] Yes[9]		
[8] – *Device* provisioned functionality *access is limited to manufacturer domain.* *SIM tool kit application access is limited to operator domain.* *MExE executable access is limited to MExE executable issued by the same issuer (identify by the certificate) of launched MExE executable* [9] – *Access is limited to MExE executable which launch the application. But the end user, shall have a way to stop the launched application, MExE environment may stop the launched application or launched application may stop itself.*			
Lifecycle management 1. Install a MExE Executable 2. Uninstall a MExE executable	Yes		
Terminal data access 1. Get manufacturer software version 2. Read time and date	Yes Yes		
Peripheral access 1. Sound generation to speaker (e.g. via stream) 2. Set speaker volume 3. printer access 4. Monitor the power state 5. Change the power state **6.** Activate/ access Serial port (RS232, Irda, Bluetooth, USB …) access **7.** Activate/access Parallel port 8. Activate/access Smart card other than SIM card (Send APDU, Slot management)	Yes		

Actions	MExE Security Domains		
	Operator	Manufacturer	Third Party
Input output User interface access			
1. Input device (keyboard, mouse …)	Yes[10]		
2. Output device (display)	Yes[10]		
3. Output notification device(smart icon, sound, light, vibrator …)	Yes		
[10] *– Access request no user permission.*			

The lists in the groups in Table 3 are not exhaustive, and other actions which are of the same category shall be included in the group for the purposes of requesting user permission.

8.2.1 MExE executable pPermissions for untrusted MExE executables

Subclause 8.2 identifies the permissions for MExE executables in the 3 domains (operator, MS manufacturer and Third Party). The permissions do not apply to untrusted MExE executables which are not permitted to execute within the domains.

In order to facilitate untrusted MExE executables having some limited access to MExE MS functionality beyond their very limited privileges, the following specific access permissions in Table 3 are extended to untrusted MExE executables:

- User interface

  An untrusted, uninstalled MExE executable (e.g. an applet) can access the user interface output (display) and input (keyboard, mouse, ..) without user permission, but the sending of user data to a server to which the MExE executables has a session connection (e.g. as part of a browser sesssion) requires user permission.

  An installed untrusted MExE executable shall only be able to access the user interface output (display) and input (keyboard, mouse, ..) with user permission. (Clearly, for the usability of untrusted MExE executables such as games, blanket user permission should be sought and given, and this is permissible.)

- File

  File access is not permitted for untrusted MExE executables, except that untrusted MExE executables can access files only in the MExE executable's own directory.

- Initiate a voice/data connection

  Untrusted MExE executables shall be able to make calls under the following conditions.

  In addition to an untrusted MExE executable possibly displaying the number to be called to the user, the number to be called shall be presented to the user for permission by a provisioned functionality of the MExE MS and not by the MExE executable itself. (This facility would support, for example, "click to dial" button/links in an untrusted MExE executable, and a MExE MS provisioned functionality then represents the number to the user for confirmation.)

- Generate DTMF

  Untrusted MExE executables shall be able to generate DTMF tones under the following conditions.

  An untrusted MExE executable is only permitted to send DTMF tones in a currently active call. The request to generate DTMF tones in the currently active call, shall result in the characters which the tones represent being presented to the user for permission by a provisioned functionality of the MExE MS.

The untrusted MExE executables permitted to use the above facilities shall be MExE executables the user has downloaded himself, and not be MExE executables that have been pushed to the user. MExE executables/applets on the MExE MS due to the user having visited a particular web site are considered to be MExE executables that the user had downloaded himself.

Untrusted MExE executables shall not be permitted access to any other functions.

8.2.2 Separation of I/O streams

There shall be strict separation of the user interface input and output streams between different MExE executables, i.e. it shall not be possible for one MExE executable to access the user interface input or output of another MExE executable. In particular, it shall not be possible for an untrusted MExE executable to access the user interface input and output destined for or proceeding from a trusted MExE executable. (This requirement is to prevent a long lived malicious MExE executable from eavesdropping upon on interfering with the user to MExE executables communications, for instance PINs, of a trusted MExE executable).

8.3 User permission types

The term "user permission" is defined to mean that the user can give permission for a specific action in one of the ways defined in Table 4. Support of blanket permission and single action permission is mandatory, but support of session permission is optional.

All prompts for user permission as described in Table 4 must display a user friendly name identifying the signer of the corresponding MExE executable, if available. The user shall be able to request to see the "subject" field of the certificate of the signer ("subject" here refers to the "subject" fields of WTLS and X.509 certificates and an equivalent field for any other format of certificate). If an application, for which user permission is being sought, is untrusted, the fact that the application is untrusted shall be visually indicated to the user whenever user permission is sought.

The user shall be prompted for user permission relating to all action groups listed in the Table 3 that are required by the MExE executable. If a prompt for permission relates to more than one action, e.g. networking and user data, then it shall list the individual action group permissions which will be granted, though the action group permissions can all be granted with a single user action. This condition applies to any prompts relating to user permissions in Table 4.

Note that blanket permission cannot be used for uninstalled MExE executables e.g. applets, WMLS.

Table 4: User Permissions

	User Permissions		
Permission Type	Description	Invocation	Revocation
blanket permission	The user gives blanket permission to the MExE executable for the specified action, and the MExE executable subsequently uses the user's original permission for the identified subsequent actions whenever the MExE executable is running.	Typically such permission would be given at MExE executable configuration or run time.	The blanket permission maybe revoked by the user at any time. The user permission no longer applies once the MExE executable has been removed.
session permission	The user gives permission to the MExE executable for the specified action during a specific run time session of an MExE executable, and the MExE executable subsequently uses the user's permission for the identified subsequent actions whilst the MExE executable session is still running.	Typically such permission would be given at MExE executable run time.	The session permission maybe revoked by the user at any time. The user permission no longer applies once the MExE executable run time session has terminated.
single action permission	The user gives a single permission to the MExE executable for the specified action; if the MExE executable subsequently wishes to repeat the action it must again request the user's permission for the identified subsequent action.	Typically such permission would be given at MExE executable run time.	The user permission no longer applies once the action has terminated.

8.4 Certification and authorisation architecture

In order to enforce the MExE security framework a MExE capable MS is required to operate an authentication mechanism for verifying downloaded MExE executables. A successful authentication will result in the MExE executable being trusted; and able to be executed in a security domain (as determined by the root public key of its certification tree).

As the MExE MS may want to authenticate content from many sources, a public key based solution is mandatory. Before trusting MExE executables, the MExE MS will therefore check that the MExE executable was signed with a private key, for which the MExE MS has the corresponding public key. The corresponding public key held in the MS must either be a root public key (securely installed in the MS, e.g. at manufacture) or a signed public key provided in a certificate. The MExE MS must be able to verify certificates, i.e. have the public key (as a root key or in a certificate) corresponding to the private key used to sign the certificate. Support of certificate chains is therefore mandatory.

The requirements on authorisation and certification are given in subclause 8.4.1. An example authorisation and certification process is described in subclause 8.4.2.

8.4.1 Certification requirements

A MExE MS cannot verify certified MExE executables of a particular domain unless it has a root public key for that particular domain.

Root public keys shall be securely installed in the MExE MS, say, at manufacture.

It is recommended that a "disaster recovery" root public key be securely installed on the terminal, to be used to install new root public keys when all other root public keys on the terminal are invalid.

Third Party Domain root public keys will typically be installed along with and integrated into the MExE ME browser, as is done for PC-based browsers.

A MExE executable can only be verified if the MExE MS contains a valid root or certified public keys corresponding to the private key used to sign the MExE executable.

A MExE MS shall support at least one level of certificate under operator, manufacturer or Third Party root public keys. The MExE MS shall support at least one level of certificate chain analysis in a signed content package, as shown in Figure 5.

A certificate (other than one containing a root public key) shall only be considered valid if the signature on the certificate is verified by a valid public key (root or contained in a certificate) already present on the MS and if the certificate being verified has not expired.

Public keys shall not be shared between domains.

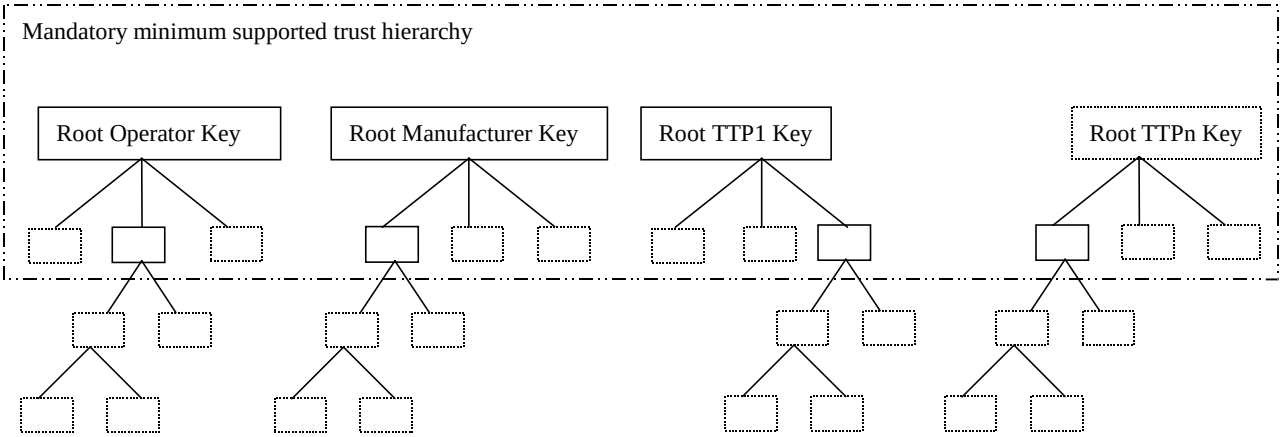


Figure 5: Trust hierarchy

The boxes below the root keys represent individual public key certificates. The solid boxes represent the minimum MExE, and the dotted boxes represent possible further support for public key certificates (either at the first or subsequent levels).

8.4.2 Example certification process

The following processes might be followed in order to securely download a Third Party application to a MExE MS.

Root public keys for a number of Certification Authorities (CAs) are installed in the ME, along with the ME browser, at manufacture. These root public keys can be used to verify certificates for Third Party MExE executables.

1. A third party software developer generates a private and public key pair (or obtains such a pair from a CA).
2. The third party software developer obtains a certificate for the public key from a CA. The certificate contains the developer public key, signed with the private key of the CA.
3. The 3rd party software developer adds all the certificates required in the key chain in the JAR.
4. The MExE MS downloads a MExE executable of the third party software developer.
5. The MExE MS verifies the certificate using the root public key, contained in the browser, of the relevant CA, and extracts the third party software developer public key and may store it in the certificate store for future use.
6. The MExE MS verifies that the MExE executable was signed using the private key corresponding to the third party software developer public key and installs or rejects the MExE executable accordingly.

8.5 Root Public keys

8.5.1 Operator root public key

The ME shall support secure storage for at least one certificate containing an operator root public key. The ME shall support the use and management of an operator root public key on the SIM. The certificate contains a root public key generated either by the operator, or by a CA trusted by the operator. The ME shall get the operator root public key from the secure area every time it needs to verify a signature, rather than cache the root public key for use in subsequent verifications.

If the MS does not contain a valid operator root public key, then the certificate chain to MExE executable previously executing in the Operator Domain will be invalid, and they will be excluded from the operator domain.

The user shall not be able to add or delete any type of operator public key (root or contained in a certificate).

Optionally, the operator may install a corresponding disaster-recovery root public key stored in the MS, enabling the operator to use a secure mechanism (involving the disaster-recovery key) to replace the certificate containing the standard operator root public key. It shall not be possible to use the disaster recovery operator root public key to replace the standard operator root public key unless both public keys are from the same operator.

There shall be no more than one valid operator root public key on the MS (excluding the disaster recovery root public key).

An application signed by an operator shall not be able to execute in the Operator Domain unless the root public key of that operator is installed in the MS (either ME or SIM) and is marked as trusted.

8.5.1.1 ME actions on SIM insertion and/or power up.

The requirements in this subclause ensure that the operator domain on the ME belongs to the same operator as the operator that issued the SIM inserted in the ME and, if there is an operator root public key (ORPK) on the SIM, that trusted operator applications on the terminal were verified using that ORPK.

The ME shall support the use and management of an Operator root public key (ORPK) on the SIM.

Editor's note: This line not to apply to release 98 spec

On power up of the terminal, the terminal shall behave as dictated by Figure 6 below.

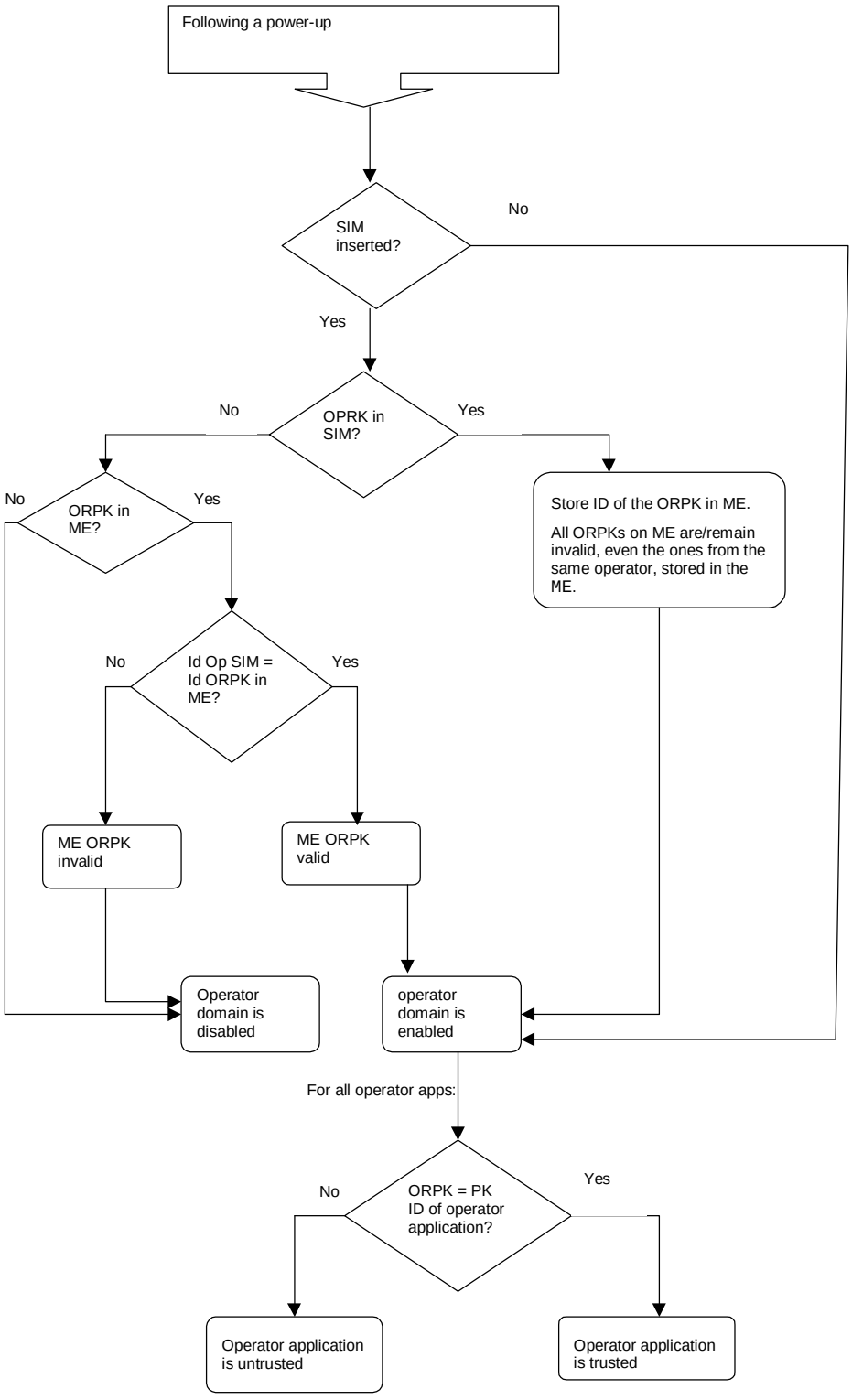


Figure 6: Terminal behaviour on power up

Editor's note: On DCS1900 the MCC+MNC is 6 digits, but elsewhere it is 5 digits. The ME needs to know how many digits to use. This problem may have been solved already. The identity of the root public key has to be defined.

The terminal shall only read the SIM ORPK from the SIM when required and shall not store a SIM ORPK on the terminal.

When an operator root public key stored on the ME is marked as invalid, all operator applications verified using that root public key or by certificates verified by a chain that terminates with that root public key, shall cease operation as soon as possible and shall be marked as untrusted.

8.5.1.2 ME actions on removal of the SIM

Removal of the SIM shall not cause the status (i.e. valid or invalid) of any operator root public key on the terminal to change.

If a SIM is removed from the ME (without another SIM being inserted), operator applications shall continue to execute in the operator domain.

8.5.2 Manufacturer root public key

The ME shall support secure storage for a certificate containing a manufacturer root public key. The certificate contains a root public key generated by the manufacturer of the device, or by a CA trusted by the manufacturer of the device.

If the ME does not contain a valid manufacturer root public key, then the certificate chain to MExE executable previously executing in the Manufacturer Domain will be invalid, and they will be excluded from the manufacturer domain.

The user shall not be able to add or delete any type of manufacturer public key (root or contained in a certificate).

The Mnufacturer shall put a root public key and optionally its corresponding disaster-recovery key in the device at the time of manufacture, and use a proprietary secure mechanism (e.g. using the disaster-recovery key) to replace the certificate containing the manufacturer root public key. It shall not be possible to use the disaster recovery manufacturer root public key to replace the standard manufacturer root public key unless both public keys are from the same manufacturer.

An application signed by a manufacturer shall not be able to run in the Manufacturer Domain unless the root public key of that manufacturer is installed in the MS and is marked as trusted.

There shall be no more than one valid manufacturer root public key on the MS (excluding the disaster recovery root public key).

8.5.3 Third party root public key

The ME shall support secure storage for at least one certificate containg a third party root public key. The ME shall support the use and management of Third Party root public keys on the SIM. The ME may contain root public key (s) generated by CA(s) implicitly trusted by the user. The user will be able to securely install (using a secure transport) or remove root public keys at any time using a system administrative tool.

The Manufacturer, Operator and Administrator may at their discretion, securely install certificates containing Third Party root public key(s) on behalf of the user, e.g. at the time of manufacture by the Manufacturer. See subclause 8.14 for details of Administrator control of Third Party certificate download.

If a Third Party public key is deleted or becomes invalid, then the certificate chain to MExE executables previously executing in the Third Party Domain certified by that public key will become "untrusted".

There may be any number of Third Party root public keys on the MS.

The third party domain administrator (user or other body) shall be able to enable and disable Third Party root public keys by using CCM. The process of adding/removing public keys and enabling/disabling public key are independent.

All third party certificates shall be subject to restrictions imposed by valid certificate configuration messages.

See subclause 8.8 for the management of Third Party root public keys on the SIM.

8.5.4 Administrator root public key

The ME shall support secure storage for a certificate containing an administrator root public key. The ME shall support the use and management of an Administrator root public key on the SIM. Only one administrator root public key shall be valid on the MExE MS.

The MExE MS shall support the administrator designation mechanism and the secure downloading of CCMs explained in subclause 8.8.

The user shall not be able to delete an administrator root public key or certificate.

The system shall support a mechanism (as part of a provisioned functionality and/or inherently part of the MExE implementation) allowing the owner of the MExE MS to manage the administrator root public key (including the download of a new administrator root public key) as defined in subclause 8.9.4. This mechanism shall be secure so that only the owner can use this functionality.

The administrator root public key can be downloaded to the MExE MS as described in subclause 8.10.4.

The terminal shall only read the SIM Administrator root public key from the SIM when required and shall not store the SIM Administrator root public key on the terminal.

See subclause 8.8 for the management of Administrator root public keys on the SIM.

The same root public key may be used for both the Administrator role and the operator or manufacturer domain. This facility does not imply any increased right of the manufacturer or operator to take the Administrator role.

If the same root public key is used for the operator domain and Administrator role and this root public key is stored on the SIM (see [27]), there shall be separate entries relating to each use of the root public key in the operator and administrator trusted certificate directory files. These entries in the operator and Administrator trusted certificate directory files may point to the same root public key in the certificate data file.

If the root public key to be shared is not stored on the SIM, then procedures relating to this are out of the scope of this specification.

8.6 Certificate management

The manufacturer may load initial third party certificates on the device. Downloaded certificates shall be verified by an existing trusted certificate and placed in the domain defined by the root public key at the top of the verification chain for the downloaded certificate.

The administrator root certificate shall be provided on the SIM if support for certificate storage on the SIM exists. For SIMs not having certificate storage the administrator root may be downloaded using the root download procedure described in subclause 8.9.2.

The actions that may be performed for a given certificate are:

- addition,
- deletion,
- mark un-trusted (un-trusted certificates cannot be used to verify applications or other certificates. This process may be preferred to certificate deletion as there is a chance that the certificate may become trusted again in the near future),
- mark trusted (marking as trusted is the process of allowing an untrusted certificate to come into use again),
- modify fine grain access permissions (proposed as a future enhancement).

The ability to perform these actions depend on the certificate type being modified as well as the access level of the entity performing the operation. Users may add a third party certificate as long as it is certified by an existing trusted certificate.

Using a provisioned functionality, users may delete Third Party certificates.

Table 4A: Allowed certificate types in signed packages

Signature on Package	Allowed Certificate types in package
Administrator	Third Party
Manufacturer	Administrator, Manufacturer, Operator, Third Party
Operator	Administrator, Operator, Third Party

8.6.1 Certificate extension for removal of network access

MExE defines the certificate extension (attribute) " access-Restriction". If the access-Restriction extension is present in a certificate used to verify the signature on a trusted application or in any certificate in the certificate chain used to verify that signature, then the application shall not be permitted the capabilities listed under "network service access" in the security table, (Table 3). This restriction applies irrespective of any user permission for network service access that may or may not be requested by the application and/or given by the user.

The extension prevents the trusted applications of developers who do not need network service access from writing applications that can perform network service access.

The support of this extension in the operator domain is mandatory. The support of this extension in the manufacturer and third party domains is optional.

The extension is defined for X.509v3 only. Support for WTLS, X9.68 certificate formats is for further study.

8.6.1.1 X.509 version 3

If MExE terminals support X.509v3 format in operator, manufacturer or third party domains, it shall support the X.509 version 3 access-Restriction extension.

X509 v3 provides a mechanism to define extensions. An Object identifier (OID) s defined for each private extension as defined in X509 [26]. The extension is defined to be within the ETSI Object Identifier (OID) name space.

This extension shall apply irrespective of the presence or otherwise of any other X.509 key usage or extended key usage field.

Normal use of the "critical" flag for extensions apply. That is, if this extension is marked as critical in the certificate used to verify the signature on the application or in any certificate in the chain used to verify the signature and this extension cannot be processed in the terminal then the certificate shall be considered invalid.

The syntax of the extension is defined in Annex C.

8.7 Certificate configuration message (CCM)

The MExE device shall use the CCM to determine the third party certificates (and only the Third Party certificates) that are trusted for use on the MExE MS. The CCM shall only be used to enable or disable third party certificates and can not be used to delete certificates. The CCM may be periodically fetched or downloaded to a MExE device by the Administrator to dynamically configure the third party list using the mechanisms defined in subclause 8.15.2. The Certificate Configuration Message shall be as shown in Figure 8. This message is essentially a simplified version of a certificate revocation list to satisfy a particular use case. More complex usage requires a full certificate revocation list.

The MExE device may additionally support other means of enabling/disabling root certificates.

8.7.1 CCM Numbering convention

Bits are grouped into octets. The bits of an octet are shown horizontally and are numbered from 0 to 7. Multiple octets are shown vertically and are numbered from 0 to n.

8.7.2 CCM Order of transmission

Frames are transferred in units of octets, in ascending numerical octet order (i.e., octet 0, 1, …, n-1, n). The order of bit transmission is specific to the underlying protocols used to transport the CCM.

8.7.3 CCM Field mapping convention

When a field is contained within a single octet, the lowest bit number of the field represents the lowest-order value. When a field spans more than one octet, the order of bit values within each octet progressively decreases as the octet number increases. In that part of the field contained in a given octet the lowest bit number represents the lowest-order value.

For example, a 16 bit number can be represented as shown in Figure 7.

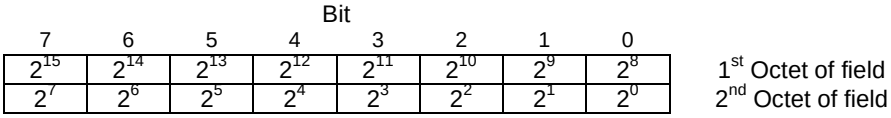


Figure 7: Field mapping convention

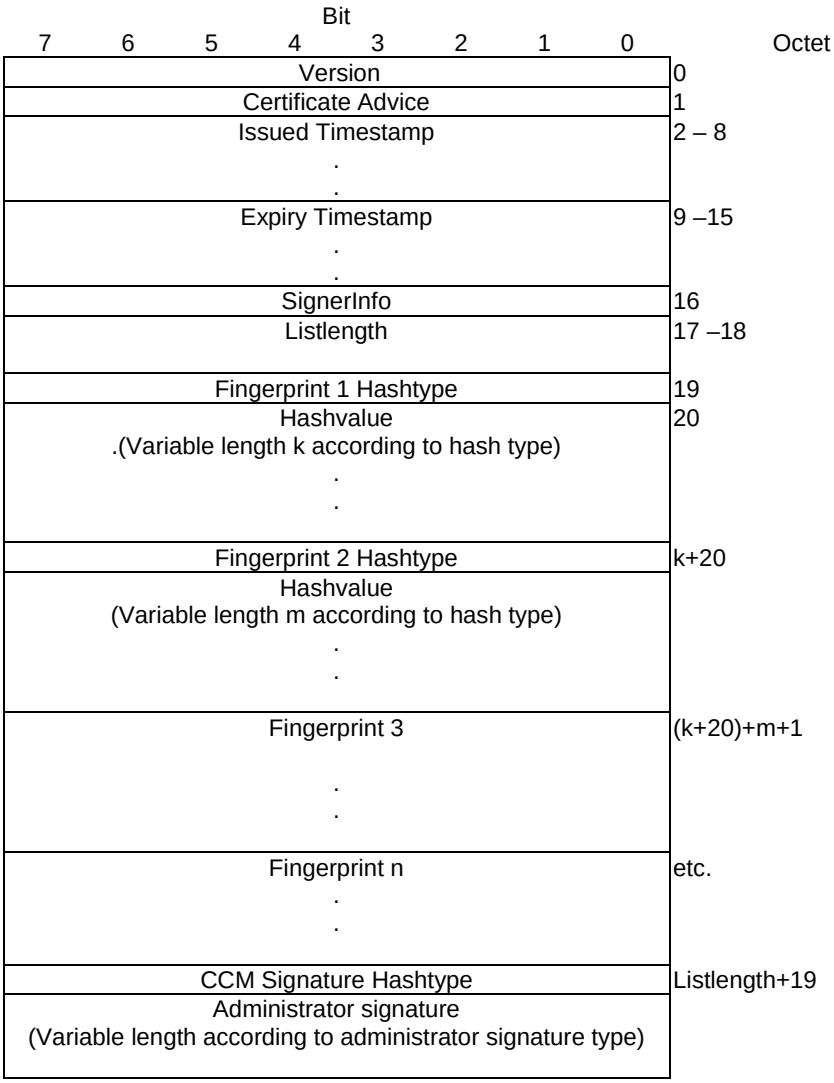


Figure 8: Format of a CCM

version = For MExE-98 the CCM format version is 0. All other values are reserved for future use.

certificateAdvice = enumerated { enable all present and future Third Party certificates (0), disable all present and future Third Party certificates (1), enable present list only (2),enable CCM list (3), disable CCM list (4) }. All other values are reserved for future use.

Issue and Expiry Timestamps = Fields used to identify the issue and expiry date of the CCM. The issue timestamp indicates a time before the current time of day (GMT) when a CCM message must be considered invalid. The expiry

timestamp (GMT) identifies the time when a CCM is to be deemed no longer valid. The receiver shall use these parameters to detect a replay attack. A MExE MS maintains information on the last valid CCM message received. A replay attack is an attacker replaying a previous valid CCM message to a MS in order to change the security settings. This is particularly dangerous for CCM messages used to enable certificates. Administrators should try and set the expiration time to be no longer than the next expected system update time of CCM information. CCM messages used to enable-all (rather than disable-all) certificates should be very short lived as the danger of these being used in a replay attack should be considered serious.

The encoding of time (GMT) shall be coded as an OCTET SEQUENCE of seven octets in length as follows:

Octet 0	1	2	3	4	5	Octet 6
Year		Month	Day	Hour	Minute	Second

Element	Size (bits)	Range
Year	16	$(0 – 65535)_{10}$
Month	8	$(1 – 12)_{10}$
Day	8	$(1 - 31)_{10}$
Hour	8	$(0- 23)_{10}$
Minute	8	$(0 – 59)_{10}$
Second (see note)	8	$(0 – 60)_{10}$
NOTE: The second field range includes the value 60 in order to accommodate leap seconds.		

For example, 1st January, 2001 00:00:30 would be encoded as: 07 d1 01 01 00 00 1E.

SignerInfo = one octet indicating the type of signer information for this CCM. The only currently defined value is device-admin = 0. In this case, no further signer information follows as it is implicit. All other values are reserved for future use.

listLength = The total length of the fingerprint list not including the final CCM signature. Shall be zero when certificateAdvice = enable-all or disable-all.

hashType = enumerated { signature (0), MD5 (1), SHA-1 (2) } All other values are reserved for future use.

hashLength = The number of octets output by the selected hash type (16 for MD5 [23] and 20 for SHA-1 [24]).

The list entries shall contain certificate *fingerprints* in the form of hashes of the encoded signed certificates. The full hash output for the specified algorithm shall be used to generate the fingerprint. A list generator shall check to insure that no two list entries match when creating a list. For an X509v3 [26] or X9.68 (currently being drafted) certificate the fingerprint hash shall be computed over the ASN.1 encoded signed certificate object, first octet to last octet. For WTLS certificates the hash shall be computed over the signed WTLS certificate in network transmission format, first octet to last octet.

The signature type and length shall be indicated by the administrator certificate, which shall be present on the device. If no administrator certificate is on the device or the signature does not verify the message shall be rejected.

Upon receipt of a valid certificate configuration message the MExE device shall go through the third party certificate list, computing fingerprints if they are not stored with the certificate, enabling or disabling each certificate according to the following conditions:

- certificateAdvice is enable-all all Third Party certificates shall be enabled;
- certificateAdvice is disable-all all Third Party certificates shall be disabled;
- certificateAdvice is enable present list only enable all Third Party certificates currently on device, do not enable any future certificates (this option allow the list to be frozen at time of manufacture) until Administrator changes;
- certificateAdvice is enable-list if its fingerprint occurs in the CCM, it shall be enabled, otherwise it shall be disabled;

- certificateAdvice is disable-list if its fingerprint occurs in the CCM, it shall be disabled, otherwise it shall be enabled.

For future releases, the setting of fine grained permissions for each certificate is expected to be supported.

An implementation shall keep track of the domain that authorised a given application. If a CCM message is received while MExE applications are currently running the implementation shall check to ensure any applications no longer in a trusted domain have their permissions re-configured appropriately and actions that are no longer permissible are terminated.

8.7.1 Authorised CCM download mechanisms

The download of third party certificate lists by a remote administrator shall be performed by using a secure mechanism as defined below. The download mechanisms shall use HTTP over IP and/or the WAP Protocol. The URL from which the CCM is downloaded shall be in the administrator certificate if the CCM was not downloaded with the Administrator certificate. The format for storing the URL information with the certificate shall be as shown in Figure 9:

Urltype	**CharacterSet**	**UrlLength**	**URL**

Figure 9: CCM Message URL storage format

Urltype= one byte, enumerated {WAP (0), HTTP (1)}. All other values are reserved for future use

CharacterSet = one byte, Internet Assigned Numbers Authority assigned character set.

UrlLength = one byte unsigned integer, length of the URL in octets.

The format for storing the URL information in the certificate shall be defined as part of the enhanced administrator mechanism.

When the administrator is changed, then the CCM shall also be changed. If there is URL information with the certificate as described in Figure 9, then the new CCM shall be obtained using the URL. If the Administrator certificate was downloaded in a JAR file, the CCM shall be obtained from the same JAR file.

8.8 Provisioned mechanism for designating administrative responsibilities and adding third parties in a MExE MS

All applications in the Domain are to be signed by a key which shall be verified back to a Third Party root public key on the MExE MS. The Third Party root public keys shall be managed (e.g. addition/deletion/mark trusted/mark untrusted/change fine grained access privileges) by an administrator that is designated by the owner of the MExE MS using the MExE administrator provisioning mechanism. A mechanism is required to be provided to enable the owner of the device to dynamically assign an administrator. The mechanism shall support the following cases:

- the user is the owner;
- the owner is at a remote location. In this case the owner could be the operator, a service provider or a third party;
- the owner of the MExE-SIM wants to be a temporary administrator.

8.8.1 Determining the administrator of the MExE MS

The administrator of the MExE MS shall be determined by the logical process shown in the flowchart in Figure 10. During power-up the provisioned mechanism shall look for an administrator root public key that is stored on the ME.

- Administrator root public key is absent

  if the administrator root public key is absent, then the user shall automatically become the administrator of the MExE MS.

- administrator root public key is present

if an administrator root public key is present, this root public key shall be used for all remote administration authentication, implying that the owner of the administrator root public key is the administrator.

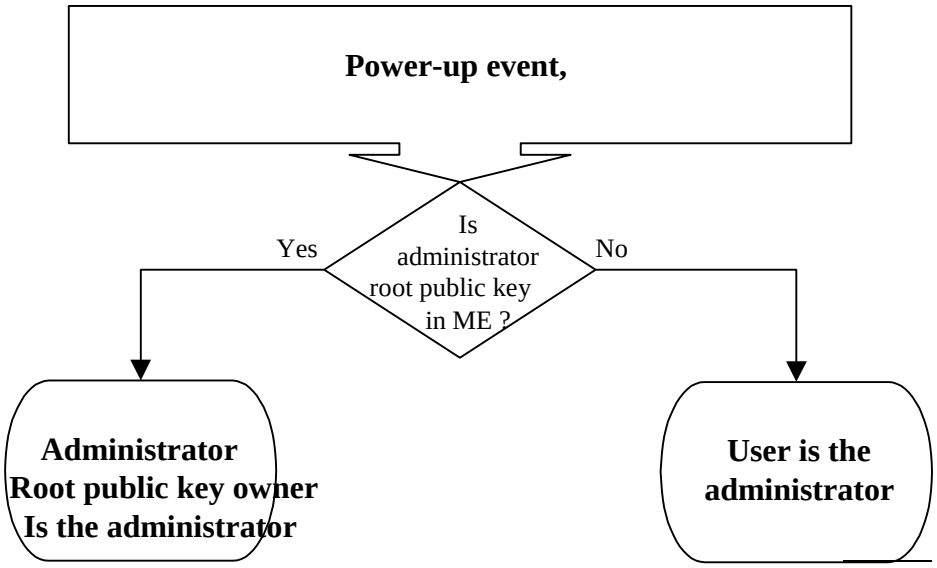


Figure 10: MExE Release 98 administrator mechanism

The rest of the mechanism is subsequently defined, however it is a future release implementation, see Figure 11. This future enhanced administrator Mechanism shall be initiated after a power-up event is processed or when a MExE-SIM is detected.

(The following subclauses assume that Third Party certificates can be added using the MExE-SIM, however Third Party certificates may be added using a non-SIM approach.)

8.8.1.1 Administrator of the MExE MS is the user

If the administrator is the user, then a check shall be made to determine whether there is a MExE-SIM. If a MExE-SIM is present, then a check shall then be made to determine whether there is a certificate in the MExE-SIM. The enhanced administrator Mechanism shall allow the MExE MS to determine (via a format) what type of certificate is present:

- certificate present - third party (CP-TP)

  A certificate present in the MExE-SIM shall be considered by the ME as a Third Party certificate, whilst that MExE-SIM is inserted in the ME. The user shall be queried to allow or disallow the certificate as a Third Party.

- certificate present - administrator (CP-Admin)

  If a temporary certificate is present in the MExE-SIM, the user shall be queried whether to allow the certificate on the MExE-SIM to take temporary control of the third party domain. By temporary control, it is meant that once the card is removed the administrator reverts back to the user administrator settings. The above mechanism implies that the previous configuration settings for the administrator shall be saved, so that they may be restored. If the user disallows the MExE-SIM certificate, the Third Party Domain shall not be able to use any of the network capabilities in the third party domain as identified in the network access section of the security Table 3.

If a certificate is not present on the MExE-SIM and the administrator is the user, the user shall continue to be the administrator and may make use of all functionality.

8.8.1.2 Administrator of the MExE MS is not the user

If the administrator is not the user, then a check is made to determine if there is a MExE-SIM. If a MExE-SIM is present, then a check is made to see if there is a certificate in the MExE-SIM. If a certificate is present in the MExE-SIM, then a comparison is made of the certificate's root public key on the MExE-SIM with the root public key on the ME for the following cases:

- Case (a): they are the same;

- Case (b): they are not the same, but the ME certificate is cross-certified with the MExE-SIM certificate (a cross-certificate exists on the ME);
- Case (c): they are not the same, but the ME certificate has a line of trust back to the MExE-SIM certificate domain;
- Case (d): they are not the same.

If the owner of the public key in the certificate on the MExE-SIM is to be a temporary administrator (CP-Admin), then in cases (a), (b) and (c), the temporary administrator shall be the owner of the CP-Admin root public key. In case (d), the Third Party domain shall not use any of the network capabilities in the third party domain as identified in the network access section of the security Table 3. If the certificate is to be a Third Party, then the certificate (CP-TP) shall be verified with the CCM and based on the content and permissions of the CCM, the certificate shall be added to the Third Party list or rejected.

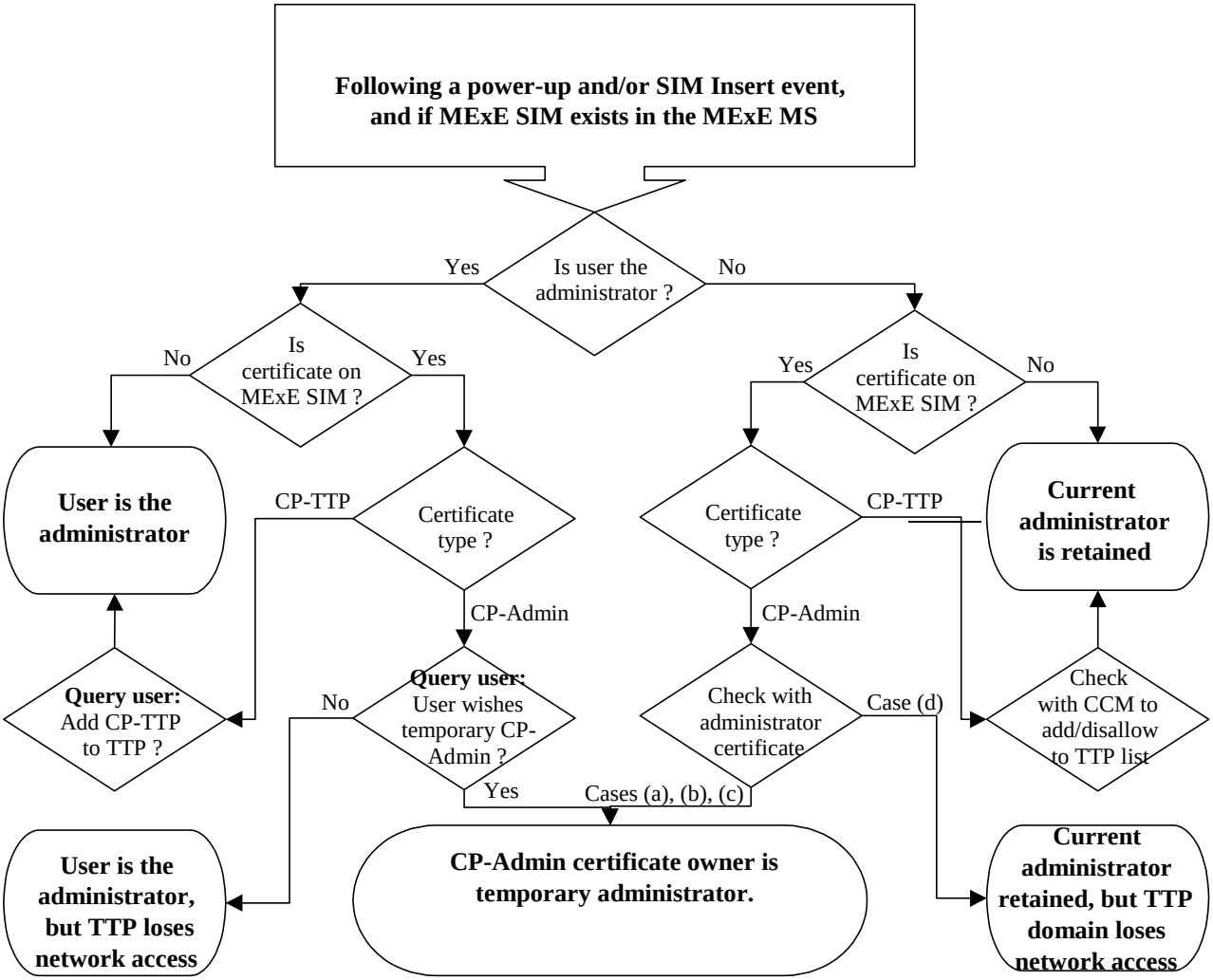


Figure 11: Enhanced administrator mechanism

8.9 Java security

There are two types of Java security [20]: sandbox, and fine grain.

The sandbox model [18] has just one domain; there is no concept of a *partly trusted* domain. The sandbox meaning of "trusted" means it is totally unrestricted to access all system resources.

Using the sandbox system, each MExE domain shall be implemented as running in a sandbox, configured with different privileges corresponding to those of the domain.

Using the fine grain Java security system [19], each MExE domain will be a set of constraints within which a Java fine grain security domain can be configured.

8.9.1 Java applet certification

Support for trusted applets is optional. Although a Java application shall be executed in a trusted domain if its certification can be validated, a Java Applet will not necessarily be executed in a trusted domain even if it does have a valid signature. It will be up to the implementers to decide if "trusted" Applets will be supported. (In certain implementations, all Applets may be always executed as "untrusted".)

8.9.2 Java application signature verification

The verification of the certification of the application or applet shall be performed as described in subclause 8.8.

8.9.3 Java loading native libraries

The MExE Java VM may be able to load native libraries that are intrinsically part of the ME implementation, and MExE native libraries. The MExE Java VM shall not load other native libraries.

8.10 Signed packages used for installation

The Java Archive (JAR) file format shall be supported on classmark 2 MExE devices for securely packaging objects that are to be downloaded and installed on the ME. The method for securely packaging objects for MExE classmark 1 devices may be referenced from the WAP specifications in a future release of this specification. A MExE device may support other proprietary means of downloading and installing objects.

The JAR file shall contain a manifest file that has at least the following attribute:

`MExE-Implementation-Type`
Whose value shall be either

- "**`MExENativeLibrary`**" in the case of a MExE Native Library (as described in 8.9.1);
- "**`TTPCertificate`**" in the case of a certificate containing a 3rd party root public key (as described in 8.9.2);
- "**`ManufacturerCertificate`**" in the case of a certificate containing a manufacturer root public key (as described in 8.9.2);
- "**`OperatorCertificate`**" in the case of a certificate containing an operator root public key (as described in 8.9.2);
- "**`AdminCertificate`**" in the case of an administrator certificate (as described in 8.9.2); or
- "**`CCM`**" in the case of a CCM (as described in 8.12); or
- *-free-format-value-* in the case of proprietary binaries or Java classes such as native DSP code, provisioned functionality upgrades and patches (as described in 8.9.2).

E.g.

```
MExE-Implementation-Type: MExENativeLibrary
```

See Figure 12. When a download of a JAR file is completed, the system installer shall read the manifest to determine what types of files are contained in the JAR, and install them appropriately.

Note that a signed package containing a library which does not have a manifest attribute "MExE-`Implementation-Type: MExENativeLibrary`" shall be considered to be some type of upgrade to libraries that are intrinsically part of the ME implementation rather than a "MExE native library". E.g.

`MExE-Implementation-Type: ManufacturerUpgrade (something.dll)`

(Recommended behaviour for the server is that it uses the capability information supplied from the ME to determine how to offer appropriate upgrades.)

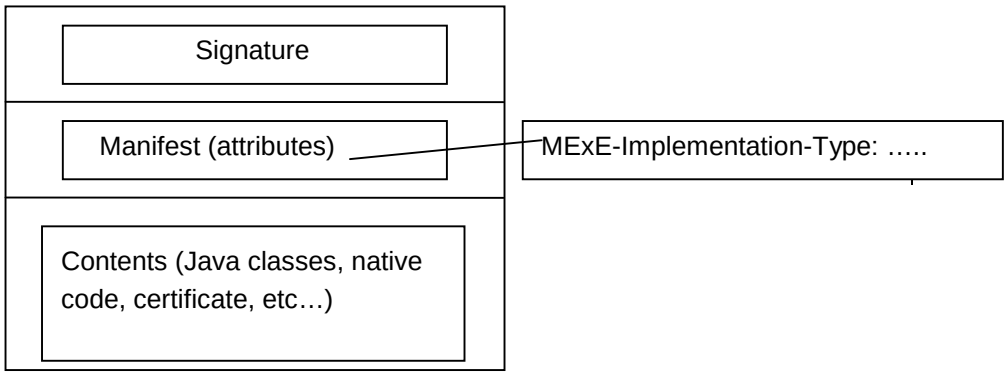


Figure 12: signed packages

8.10.1 Installing MExE native libraries

A signed native library whose signature verifies as describe in subclause 8.8 as belonging to the Manufacturer Domain may be installed as a "MExE native library".

A MExE native library may be called by a MExE executable, and shall not compromise the MExE security system.

Support of MExE native library signed package installation is optional.

8.10.2 Installation of root certificates in a signed data package

Root certificates in a signed package (whose signature verifies as described in subclause 8.9 to the Manufacturer root, Operator root, or the Administrator root), may be installed to the root public key store on the ME. Note that the certificate thus packaged does not necessarily belong to the manufacturer domain. The types of certificate that can be present and installed by packages are given in Table 4. The ME shall store the root public key as indicated by the certificate type.

When a certificate containing an Administrator root public key is thus contained in a signed package, the signed package (JAR) shall contain two files: the Administrator root public key and the CCM.

8.10.3 Installation of other signed data

A signed package of proprietary binaries or Java classes such as native DSP code, provisioned functionality upgrades and patches, whose signature verifies as described in subclause 8.8 as belonging to the Manufacturer Domain may be installed. The use of such binaries is outside the scope of MExE, but the manufacturer shall be responsible for ensuring that the integrity of MExE is not compromised.

Support of this feature is optional.

8.10.4 Administrator root certificate download mechanism

Devices supporting SIMs without certificates shall at least support the following procedure to download the administrator root certificate.

1. Upon sign-up with an administrator the user and administrator will make contact.

2. The administrator service centre will obtain any required information from the user and inform the user by SMS or other means of the location of the administrator root certificate.

3. The user will initiate the download of the Administrator root certificate using a signed package.

4. Once the procedure is complete the device shall compute the hash of the received Administrator certificate containing root public key.

5. The user will contact the administrator and enters on the device at least the first 8 bytes using decimal value of the hash of the Administrator root public key information provided by the administrator . The device compares the beginning of computed hash value and the abbreviated hash value entered by the user If these two values are the same ,the provisioning process will be complete. If the two values are different this shall be indicated to the user who should inform the administrator of this.

Alternative methods to download an administrator root certificate may be used where appropriate but must insure that the certificate is received by the device unaltered.

8.11 Pre-verification of applications

This is an optional feature added to eliminate the potentially excessive overhead of checking a signature each time an application is launched.

To use this process the MExE device shall create a hash of the executable object (executable object fingerprint) as if checking the signature. This shall be stored in a protected verified application list, along with indication of the domain permissions for the application. The hash used shall be the same type as that used for signing the object. When launching an application or downloading an applet, the hash shall be performed as for when computing the signature. The verified application list shall then be checked; if the hash value is present and the entry has not expired then the application or applet may execute. If no list entry exists for this object, or the entry has expired, the process shall then proceed with the full signature verification. Note that the lists for applications and applets should be separate and that an implementation determines management policy for the lists (e.g., ageing policy, which entries to delete when trying to add a new entry to a full list etc.). One restriction imposed that shall be enforced is that the maximum number of uses for an entry before it is marked invalid is limited to some maximum value.

In the event that a new CCM is received by the MExE MS, all verified application list entries shall be marked invalid unless some mechanism to determine the validity of an authorising certificate entry for each application is provided by the ME implementation.

9 Quality of Service

QoS aware MExE executables may be executing on the MExE device. To ensure correct operation with the QoS provisioning of the bearer network(s) the associated API's and the MExE QoS manager shall be supported by MExE MS supporting bearers defined by QoS – see Figure 13. Non QoS aware MExE executables shall operate with the defined QoS by the user or the network.

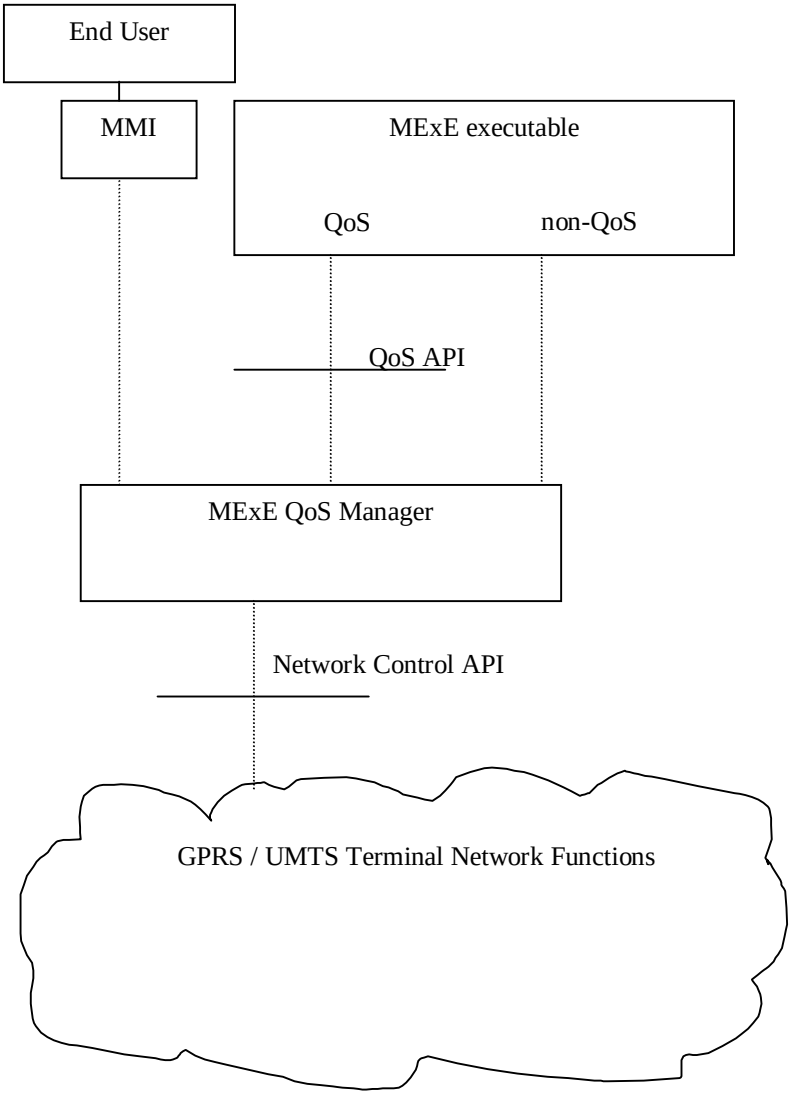


Figure 13: Logical MExE Terminal QoS manager elements

9.1 MExE QoS Support

A MExE QoS manager exists between the MExE executable and the Network Control API. To interface this, an API to the MExE executable is provided and another API to the network, see Figure 13. The MExE QoS functions accommodates standard methods of end to end QoS provisioning – e.g. differentiated services (Diff-serv).

For MExE devices supporting bearers defined by QoS, the MExE device shall support the following basic QoS operations:

- a mapping between the QoS requirements of the MExE executable and the network layer;
- MExE executables shall be able to indicate and interpret QoS values of the network via the MExE QoS Manager;
- MExE executables shall be able to modify the QoS dynamically;
- MExE executables shall be able to react to changes in the provided QoS;
- The end user shall be able to manage the QoS directly via the MMI.

MExE introduces two new elements to cater for QoS – the MExE QoS manager and the QoS API. The MExE QoS manager shall handle the fact that the network may not have QoS capabilities.

9.2 MExE QoS Manager

The MExE QoS manager is responsible for:

- Managing the QoS streams for MExE executables;
- Notification of the negotiated and delivered QoS to the end user / MExE executable.

The MExE QoS manager shall support the MExE QoS API according to the bearer supported by the device, and provide functions such as:

- insert additional QoS signalling parameters (e.g. Diff-serv);
- add the functionality of the MExE QoS API at best effort, if the network does not support it directly;
- translate between the QoS parameters from the MExE executable and those of the network;
- monitor the QoS delivered by the network and manage QoS requests between the MExE executable and the network;
- be informed by the MExE executable of the requested QoS traffic class ;
- be informed by the MExE executable of the lowest QoS traffic class which can be accepted by the MExE executable;
- attempt to re-negotiate the QoS if it falls below the lowest QoS traffic class.

The MExE QoS manager may request information from the network regarding the QoS available.

The MExE QoS manager does not need to know the end user's subscribed QoS, this is held within the network and used to validate a requested QoS level.

The MExE QoS manager may also be accessed through the device's MMI.

9.3 Network Control API

The network control API shall provide the QoS manager with access to the network specific QoS control (defined for GPRS/UMTS in [29] and [30].

The MExE QoS manager may perform some QoS control, if it is not provided in the network control.

9.4 QoS API

The QoS API provides the MExE executable with an interface to the QoS management. It does not require the MExE executable to have any knowledge of the underlying network, or how QoS is implemented in the network.

The QoS API shall provide the MExE executable with a standard set of parameters. Refer to [28] for details of these parameters (see note).

NOTE: The FLOWSPEC parameters, defined by the IETF Integrated Services Working Group, provide the QoS information required by QoS capable network elements.

Table 5 shows the set of example parameters.

Table 5: Example parameters

Parameter	Units	Type
Token Bucket Rate	bytes /sec	32-bit IEEE floating point number
Token Bucket Size	bytes	32-bit IEEE floating point number
Peak Data Rate	bytes/sec	32-bit IEEE floating point number
Minimum Policed Unit	bytes	32-bit integer
Maximum Packet Size	bytes	32-bit integer
Latency	micro secs	32-bit integer
Delay Variation	micro secs	32-bit integer
Service Type		service type

As a minimum the following three parameters shall be supported by the MExE QoS manager:

- Token Bucket Rate;
- Token Bucket Size;
- Peak Data Rate.

NOTE: The discussion of UMTS bearer service parameters as well as radio access bearer parameters is still going on. Especially the bitrate parameters and reliability parameter are under discussion [28].

If the MExE executable does not provide a full set of QoS parameters, then the MExE QoS manager shall the provide QoS parameters based on information available to it (e.g. from the MMI settings), see subclause 'Sources of UMTS Bearer Service Parameters'.

9.5 Sources of Bearer Service Parameters

A set of QoS parameters (QoS profile) specify the service provided to the user by the network. At bearer service establishment or modification different QoS profiles have to be taken into account. This is based on:

- The UE capabilities;
- The UE or the TE within the terminating network;
- A QoS profile in the QoS subscription (describes the upper limits);
- Default QoS profile (of the user or network);
- A Network specific QoS profile characterising for example the current resource availability or other network capabilities.

9.6 QoS Streams

Several MExE executables may be executing in the MExE device, each with a different QoS requirement. Also, a MExE executable may operate several QoS streams, each with different parameter settings. The MExE QoS manager within the MExE device shall be able to deal with each stream independently.

9.7 QoS Security

Only the end user, MExE executable or the network using a QoS stream should be able to modify the QoS of that stream.

Annex A (normative): MExE profile of PKCS#15

A.1 PKCS#15 certificate object attributes presentation

Details from PKCS#15[32] in this clause A.1 are for information only.

A.1.1 Object common attributes

Label — human readable label to describe the certificate

Flags — indicates whether the object is private (e.g. CHV authentication request), whether the object is read only.

Authentication object identifier properties — a cross-reference back to the authentication object, which describes the of a CHV, used to protect this object.

A.1.2 Certificate common attributes

identifier — the identifier is used for correlation between the public key contained in the certificate and the associated private key.

Authority — indicates whether the certificate is for an authority (i.e. CA or AA) or not.

Request identifier — used to search a certificate : Issuer and serial number SHA-1 hash, or issuer public key SHA-1 hash, or public key subject SHA-1 hash.

Thumbprint — used as secure way to verify that no one has tampered with a certificate: hash on to be signed certificate (internet). MExE uses the thumbprint to enable or disable a certificate through the certificate configuration message (CCM).

A.1.3 Certificate attributes

Type of certificate — indicates the type of certificate: WTLS, X509, SPKI, PGP, X9.68.

Value — direct value or indirect file path or URL.

MExE only supports storage of WTLS, X509, X9.68 certificates.

A.1.4 Specific X.509 Certificate attributes

For information See PKCS#15 [28].

A.2 MExE profile of PKCS#15

`PKCS15CommonObjectAttributes.label` must be present. The value content is unspecified.

`PKCS15CommonObjectAttributes.Flag` must be present. The value shall be private, not modifiable by ME.

`PKCS15CommonObjectAttributes.Authentication` must be present. The value shall be "CHV1". The certificates files are protected by CHV1, because MExE need also IMSI to manage domains availability.

`PKCS15CommonCertificateAttributes.Id` must be present. The value content is unspecified.

PKCS15CommonCertificateAttributes.Authority must be present if and only if certificate is a CA certificate. The value is true.

PKCS15CommonCertificateAttributes.RequestId must be at least present if certificate is an operator or third party root certificate. The value shall be the same as the ones used in the issuer/authority key identifier field of the certificates, provided by this issuer (as in RFC2459 document [33]). The aim of this attribute is to give a easy way to search a key issuer of a received certificate without reading all certificates content.

PKCS15CommonCertificateAttributes.Thumbprint must be at least present if certificate is a third party root certificate. The value shall be the same as the ones used in CCM. The aim of this attribute is to give a easy way to search a certificate with reference included in CCM message.

Domain attribute presence and value shall be added as soon as it will be available in PKC#15 v1.1.

PKCS15(type)CertificateAttributes.value must be present Value is a indirect file path (path, index, offset). Index and offset default value is 0.

Specific X509 attributes are not supported:

PKCS15X509CertificateAttributes.subject must not be present.

PKCS15X509CertificateAttributes.issuer must not be present.

PKCS15X509CertificateAttributes.serialNumber must not be present.

The ME shall recognise all optional present fields above. The ME shall accept and ignore all unused fields or new field extensions.

A.3 Coding and storage in SIM

See detail of file hierarchy and file properties in SIM document [27].

Since the domain attribute is not available in PKCS#15 v1.0, MExE creates one directory file for each trusted domain. If the domain attribute is available in the next PKCS#15 versions, for future new domains, MExE may create a common directory file.See abstract syntax definition and coding detail in PKCS#15 document [32].

The address of the certificate descriptor Elementary File is fixed.

According to PKCS#15 [32] subclause 7.6 The PKCS15Certificates type , the contents of a certificate descriptor Elementary File must be the *value* of the DER encoding of a **SEQUENCE OF PKCS15Certificate** (i.e. excluding the outermost tag and length bytes).

The address of the certificate data Elementary File is unspecified.

According to PKCS#15 [32] : subclauses 7.6.1 to 7.6.6, the certificate data value is coded according to the related certificate type (e.g. DER for X5.09, base64 for SPKI and PGP, WTLS network format for WTLS, DER or PER for X9.68).

Annex B (informative): PKCS#15 certificate objects ASN1 expanded syntax extract

```
{  -- sequence of certificate

x509Certificate,[0] x509AttributeCertificate,[1] spkiCertificate, [2] pgpCertificate,
[3] wtlsCertificate,[4] x9-68Certificate : {

        commonObjectAttributes {

            label  "" UTF8 string OPTIONAL,

            flags  {private (0), modificable (1)} bit string OPTIONAL, --

            authId  octet string OPTIONAL, --

        },

     CommonCertificateAttributes {

            iD      octet string,

            authority boolean default not an authority,

            requestId {

       idtype integer

       IdValueoctet string

                          pkcs15IssuerAndSerialNumber PKCS15KEY-IDENTIFIER::=

                                  {SYNTAX PKCS15-OPAQUE.&Type IDENTIFIED BY 1}

                                  -- As defined in RFC [CMS]

                          pkcs15SubjectKeyIdentifier PKCS15KEY-IDENTIFIER ::=

                                  {SYNTAX OCTET STRING IDENTIFIED BY 2}

                                  -- From x509v3 certificate extension

                          pkcs15IssuerAndSerialNumberHash PKCS15KEY-IDENTIFIER ::=

                                  {SYNTAX OCTET STRING IDENTIFIED BY 3}

                                  -- Assumes SHA-1 hash of DER encoding of IssuerAndSerialNumber

                          pkcs15SubjectKeyHash PKCS15KEY-IDENTIFIER ::=

                                  {SYNTAX OCTET STRING IDENTIFIED BY 4}

                                  -- Hash method defined in Section 7.

                          pkcs15IssuerKeyHash PKCS15KEY-IDENTIFIER ::=

                                  {SYNTAX OCTET STRING IDENTIFIED BY 5}

                                  -- Hash method defined in Section 7.

     } OPTIONAL,

     thumbprint   [0] OOBCertHash OPTIONAL, -- hash on to be signed certificate, used for
secure certificate identification as CCM using

        },

        [1] typeAttributes {
```

```
 value indirect : path : {
                path      octet string, -- '4331'H Reference by file identifier
          index    integer OPTIONAL, -- ‘XXXX’H offset in file
          [0] length  integer OPTIONAL, -- ‘XXXX’H length in file
            }
      -- other optional attributes are defined for X509 certificate
        }
    },
}
```

Annex C (normative): Access restriction certificate extension

access-Restriction extension

id-ETSI OBJECT IDENTIFIER ::= {ITU identified-organization (3) ETSI } ::= {ETSI}

id-mexe OBJECT IDENTIFIER ::= {ETSI MExE}

Id-mexe-accessRestriction ::= {id-mexe 1}

AccessRestriction ::= BIT STRING {

network_service_access (0),}

Annex D (informative): Change history

TSG	T-Tdoc	T2-Tdoc	CR	Rev	Ph	Subject	Cat	Version-Current	Version-New
T#7	TP-000024	T2-000047	001		R99	Corrections to WAP chapters	F	3.0.0	3.1.0
T#7	TP-000024	T2-000049	002		R99	QoS	F	3.0.0	3.1.0
						Editorial change by MCC		3.1.0	3.1.1

History

Document history		
V3.1.1	March 2000	Publication