3GPP TSG-T (Terminals) Meeting #21
Frankfurt, Germany
17 - 19 September, 2003

TP-030169

Agenda Item: 5.2.3

Source: T2

Title: Change Request on MExE

Document for: Approval

Spec	CR	Rev	Rel	Subject	Cat	Vers-Current	Vers-New	T2 doc	Workitem
23.057	122	-	Rel-6	Correcting figure and table numbering	D	6.1.0	6.2.0	T2-030400	TEI6

3GPP TSG-T2 #22
Cambridge, UK
25 -29 August 2003

T2-030400

CR-Form-v7

CHANGE REQUEST

⌘ **23.057** **CR** **122** ⌘ **rev** **-** ⌘ Current version: **6.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: UICC apps⌘ ☐ ME ☒ Radio Access Network ☐ Core Network ☐

Title:	⌘	Correcting figure and table numbering
Source:	⌘	T2
Work item code:	⌘	TEI6
Date:	⌘	01/07/2003
Category:	⌘	**D**
Release:	⌘	Rel-6

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change: ⌘ To realign the numbering of the figures and the tables with the new ordering of the chapters. Also remove the dublication of the figure 15!

Summary of change: ⌘ Changed the following figure numbers throughout the specification:
15 -> 3
9 -> 4
7 -> 5
8 -> 6
15 -> 7
10 -> 8
11 -> 9
12 -> 10
13 -> 11
14 -> 12
3 -> 13
4 -> 14
5 -> 15
6 -> 16

And changed the following table numbers throughout the specification:
2 -> 1
3 -> 2
1 -> 3
10 -> 4
6 -> 5
7 -> 6
8 -> 7
9 -> 8

		4 -> 9 5 -> 10
Consequences if not approved:	⌘	Very low readability, and the possibility for misinterpretation in implementations.

Clauses affected:	⌘	Whole specification			
		Y	**N**		
Other specs affected:	⌘		**X**	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

5 Generic MExE functions (excluding security)

5.1 User profile

Support of the user profile is optional.

NOTE: The user profile is not yet specified in an interoperable way. Support of the user profile will be defined when it has been fully specified in a fully interoperable way.

The user profile (which may consist of sub user profiles for a user) contains the characterisation of the MExE device as defined by the user and service provider. Further, it is also possible for multiple users of a MExE device to each have their own user profiles. The user profile is not unique to the MExE device, 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 3GPP 22.101 [14] for further details on the user profile.

5.1.1 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 table 5~~6~~ "Security domains and actions" in the Security clause 6.3 "MExE executable permissions" for further details on user profile activation.

The user's characterisation of the MExE device 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 shall be securely managed by the MExE device, and stored in a secure area of the MExE device (either (U)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 may also be stored in the network, however only with the user permission.

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

5.1.2 Support of the user profile

The user profile acts as a repository (which is always available in the MExE device) defining the MExE device 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) [22].

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 clause 5.2 "Capability and content negotiation".

The following user preference information is supported by UAProf [17]. A MExE device shall support the following property in table 12 "Mandatory UAProf properties" of the UAProf schema for user preference information.

Table 12: Mandatory UAProf properties

Attribute	Description	Resolution Rule	Type	Sample Values
AcceptDownloadableSoftware	Indicates the user's preference on whether to accept downloadable software	Locked	Boolean	"Yes", "No"

It is recommended that a MExE device supports the following UAProf properties in table 23 "Recommended UAProf properties":

Table 23: Recommended UAProf properties

Attribute	Description	Resolution Rule	Type	Sample
CcppAccept-Language	User's preference for document language. Property value is a list of natural languages, where each item in the list is the name of a language as defined by RFC 1766 [46].	Append	Literal (Bag)	"zh-CN", "en fr"
PreferenceForFrames	User's preference for displaying frames	Locked	Boolean	"Yes", "No"
WapPushMsgPriority	User's settings for WAP Push message priorities	Locked	Literal	"critical", "low", "none"

Also, there is in UAProf [17] support for indicating MExE device's capabilities related to UI features, e.g. capability for displaying images or frames, as well as capability information about input and output methods.

5.1.3 User interface personalisation

Support of user interface personalisation as detailed in this clause is optional.

The MExE device interface consists of the buttons, menus, screens and MMI as designed and provided by the MExE device manufacturer; the nature of this MExE device interface is naturally evolving, MExE device 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 MExE device. This specification does not place any requirements or limitations on the individual manufacturers' MExE device interface.

The MExE MMI, in turn, is the interface available to the user to support MExE services and functionality on the MExE device. The nature of the MExE MMI interface, like the normal MExE device 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 MExE device interface, the normal MExE device interface with modifications, a different interface to the normal MExE device 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 MExE device, require to be recorded. The second part refers to the MExE device's support of the user's preferences for the interface, within the capabilities of an MExE device. 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, e.g. CcppAccept-Language and/or PreferenceForFrames), and thereby provide portability of the user's preferences.

5.1.3.1 MExE user interface personalisation

Personalisation of the user interface offers the MExE Service Environment and or the user, the ability to inform the MExE device of the desired extent of personalisation. All support of the user interface personalisation is optional, not mandatory on any class of MExE device, and subject to the capabilities of the MExE device. Depending on the capability of the MExE device, 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.

5.1.3.2 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.

5.1.4 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.

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.

5.2 Capability and content negotiation

Support of capability negotiation for all MExE devices is mandatory, while support of content negotiation is optional.

Interaction between the MExE device and the MSE for WWW/WAP browsing and service discovery 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 device and the MSE supports:

- Capability negotiation

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

  Capability negotiation represents the mechanism by which the MExE device 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 MExE device to inform the MSE of its capabilities. The MExE device may be informed by the MSE of its use of the MExE device's capabilities. The MExE device may also spontaneously inform the MSE of its capabilities (i.e. following a change in MExE support, such as removal of MExE device from a docking station with its keyboard, mouse and monitor). A subset of characteristics which may be transferred between the MExE device and the MSE during the capability negotiation are identified in clause 5.2.1 "Capability negotiation characteristics".

- Content negotiation

  Content negotiation represents the means by which the MExE device 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 device 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 by using CC/PP framework.

The content negotiation transferred between the MExE device and the MSE is identified in clause 5.2.3 "Client content capability report" onwards.

5.2.1 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 MExE device. If the capabilities change, then the degree of support of the user's preferences may change too.

The capability negotiation between the MExE device and the MSE, as shown in figure 2 "Model of user profile and capability relationship", contains those user preferences which the MExE device is able to support.

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

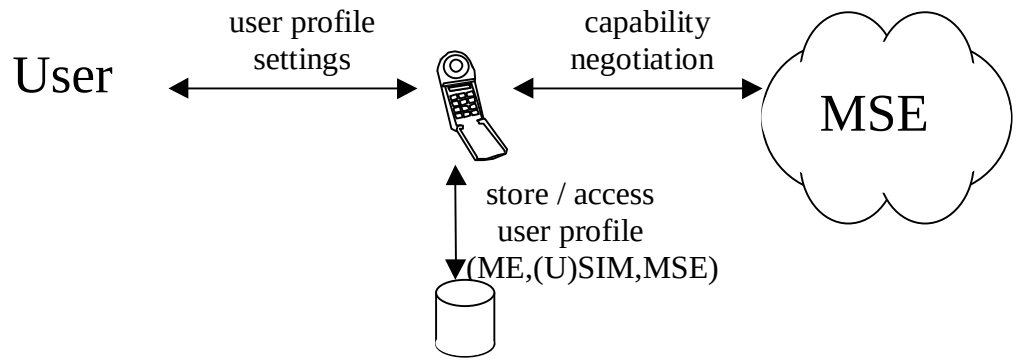


Figure 2: Model of user profile and capability relationship

5.2.2 Capability negotiation characteristics

The method for capability negotiation is based on the Composite Capability/ Preferences Profiles (CC/PP) specification made by W3C, [16]. The properties and the actual schema, table 3~~1~~"UAProf properties supported by MExE", is based on the WAP UAProf specification [17]. The CC/PP 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) [37] in CC/PP allows for interoperable encoding of the profile metadata in XML[36] and supports multiple vocabularies to provide for future extensibility. The WAP UAProf is based on the CC/PP framework. The purpose of the UAProf outlined in this document is to specify:

- an RDF based schema and vocabulary for CC/PP in the context of the 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 URL(s) where the server may fetch the properties. An MSE may, or may not, use the client capability information.

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 device shall support the properties in the UAProf schema for capability negotiation defined in table 3~~1~~ "UAProf properties supported by MExE" as "mandated properties".

It is recommended that MExE device supports the properties defined in the table 3~~1~~"UAProf properties supported by MExE" as "recommended properties". It is not required that a MExE device shall send all the "recommended properties", when sending a request, however it should be possible for the MExE device to send one or more of the "recommended properties", with user permission.

The mandatory and recommended properties in table 3~~1~~"UAProf properties supported by MExE" are specified in WAP UAProf [17].

Support of the properties of the UAProf schema in this specification shall not be limited to those listed in table 3~~1~~ "UAProf properties supported by MExE". A MExE device may support any other properties from WAP UAProf specification [17].

Table 31: UAProf properties supported by MExE

Mandated Properties				
Attribute	Description		Type	Sample
MexeClassmarks	List of supported MExE classmarks (Note 1)		Literal (bag)	"1", "2", "3"
MexeSpec	The first two digits of the MExE Specification version that the MExE device conforms to		Literal	"3.3", "4.1"
MexeSecureDomains	Indicates whether the device supports the MExE security domains		Boolean	"Yes", "No"
Recommended Properties				
Vendor	MExE device vendor		Literal	"Lexus", "Ford"
Model	MExE device model number		Literal	"Mustang 90", "Q10"
SoftwareNumber	The number of the MExE device specific software.		Literal	"1.0", "2.7.0"
ScreenSize	The size of the MExE device's screen in units of pixels.		Dimension	"160x160", "640x480"
ScreenSizeChar	Size of the MExE device's screen in units of characters (based on the standard font).		Dimension	"12x4", "16x8"
ColorCapable	Whether the MExE device display supports colour		Boolean	"Yes", "No"
AudioInputEncoder	List of audio input encoders supported by the MExE device		Literal (bag)	"G.711"
VideoInputEncoder	List of video input encoders supported by the MExE device		Literal (bag)	"MPEG-1", "MPEG-2", "H.261"
PointingResolution	Type of resolution of the pointing accessory supported by the MExE device		Literal	"Character", "Line", "Pixel"
CcppAccept-Language	List of preferred document languages		Literal (bag)	"zh-CN" "en fr"
Keyboard	Type of keyboard supported by the MExE device as an indicator of ease of text entry.		Literal	"Disambiguating", "Qwerty", "PhoneKeypad"
SupportedBearers	List of bearers supported by the MExE device.		Literal (Bag)	"GPRS", "GUTS", "SMS", CSD", "USSD"
JavaPlatform	List of Java platforms and profiles installed on the device		Literal (Bag)	"Pjava/1.1.3-compatible", "MIDP1.0-compatible", "J2SE/1.0-compatible"
Proposed New Recommended Property				
CLIPlatform (Note 2)	List of CLI profiles installed on the device		Literal (Bag)	"CLI-Compact/1.0-compatible", "CLI-Standard/1.0-compatible

NOTE 1: In pre-release 4.0.0 specifications the attribute MexeClassmark (as opposed to MexeClassmarks) which was a literal (as opposed to as Literal, Bag) indicating only one MExE classmark was notified.
NOTE 2: The property name "CLIPlatform" is proposed as a placeholder. Once a decision has been made the final property name will be proposed to UAProf for UAProf approval.

Generally, the combination of user profile and MExE device 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 VideoInputEncoder information the user's profile controls if and when VideoInputEncoder 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 device and the MSE, CC/PP method is used with the schema defined in the WAP UAProf working group.

5.2.3 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].

5.2.4 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.

5.2.5 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.

5.3 Provisioning and management of services

Support of management of services as detailed in this clause is mandatory.

The MExE device shall be capable of supporting services in a standard (WAP or Java) execution environment independently of the MExE device manufacturer. Service provisioning provides a standardised method for a MExE device to discover and install services. It includes download and installation of the service's client application. Once discovered and delivered, services are managed by the user, under the principles stated in 3GPP TS 23.227 [48]. Management of services provides the user with the capability to:

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

on his MExE device.

5.3.1 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. To be able to interactively discover the services via standard mechanisms such as WSP or HTTP, a MExE device should feature a browser which is a common tool for service discovery. The request, and transfer of information on MExE services from the MExE server is supported 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 receiving information on MExE services, is the use of an application on the MExE device 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.

5.3.2 Service transfer

The standardisation of the transferral of MExE services to a MExE device is outside the scope of this specification.

Examples of possible ways of supporting service transfer are from a MExE server or from another user MExE device (e.g. using wireless and standard protocols and mechanisms such as HTTP, FTP, proprietary protocols and mechanisms, via a serial link, infrared, Bluetooth data exchange, etc.).

The above examples are not exhaustive. Regardless of the means of transfer, all services are required to conform with the security requirements in clause 8 "Security".

5.3.3 Service installation and configuration

Installation of a service may result in changes to the MExE device user interface using icons, browsers or menus as applicable depending on the capability of the MExE device to support them. The name of the installed service 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. After installation, the service may be configured. Configuration of the service includes setting the user permissions that apply to the service (e.g. blanket permission for call origination). Configuration may be performed automatically based on the user profile.

The user controls whether a service transferred to the MExE device is automatically configured and installed in the MExE device. If automatic configuration and/or installation is enabled, the user is notified once it is completed. In the event that there is no authorisation for the automatic installation and/or configuration of a transferred service, the user is notified.

Subsequent user modification of a service's configuration (e.g. by modification of user profile settings) shall take effect at the earliest possible opportunity thereafter.

5.3.4 Service management

The MExE device shall support the ability to determine which services are transferred to, resident, installed or executing on the MExE device. The information relating to the services shall include the name as a minimum and the version number if available.

The user controls which services are permitted or denied to be transferred, resident, installed, configured or executing on the MExE device via the user profile, e.g. AcceptDownloadableSoftware. The user profile permits characteristics such as security level, identification of specific services etc. to manage services on the MExE device.

5.3.5 Service termination

A MExE device shall support the termination of services.

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.

5.3.6 Service deletion

A MExE device shall support the deletion of services.

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

5.4 User control of application connections

Support of the user control of application connections is mandatory and shall follow the principles and requirements stated in 3GPP TS 23.227 [48].

This clause addresses the generic aspects of connection control supported by both WAP and Java classmark MExE devices.

In order to allow the user to maintain control over connections on his MExE device 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 device. The user shall be able to obtain information about all connections associated with applications on the MExE device (e.g. requesting information, being informed by the MExE device etc.). Behaviour of the application following termination or suspension of its connection is undefined.

The specific support of connection control by WAP classmark MExE devices is identified in subsequent clause 7.3.1 "Call control", the security aspects of connection control are identified in clause 8 "Security", and the user control of connection authorisation is identified in clause 5.1 "User profile".

5.4.1 Journaling of network events

Support of the journaling of network events is mandatory.

To support the user in monitoring (potentially chargeable) network events initiated by services in the MExE environment, the MExE device shall maintain a record of network events initiated by MExE executables on the MExE device.

Network events for the purposes of journaling, are defined as events which result in the origination of connections by a service in the MExE environment of the MExE device. Examples of such events (any (potentially chargeable) network event initiated by services in the MExE environment) 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 length, format and longevity of the journal is undefined and subject to manufacturers' discretion.

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

5.4.2 User notification

Support of user notification is optional.

It is recommended that the MExE 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 MExE 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 MExE device is performing network actions.

5.5 Quality of service

Support of Quality of Service is optional.

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 device MMI. A MExE executable may have several QoS streams open simultaneously.

When the MExE execution environment supports QoS, 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.

Clause 9 "Quality of Service" defines the necessary functions for a MExE device to accommodate QoS management and provisioning. QoS management may be available directly to the MExE executables themselves, or to the MExE environment.

5.5.1 Introduction

Support of quality of service for MExE devices supporting bearers defined by QoS as defined in this clause is optional.

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 device supporting bearers defined by QoS – see figure 3~~15~~ "Logical MExE device QoS manager elements". Non QoS aware MExE executables shall operate with the defined QoS by the user or the network.

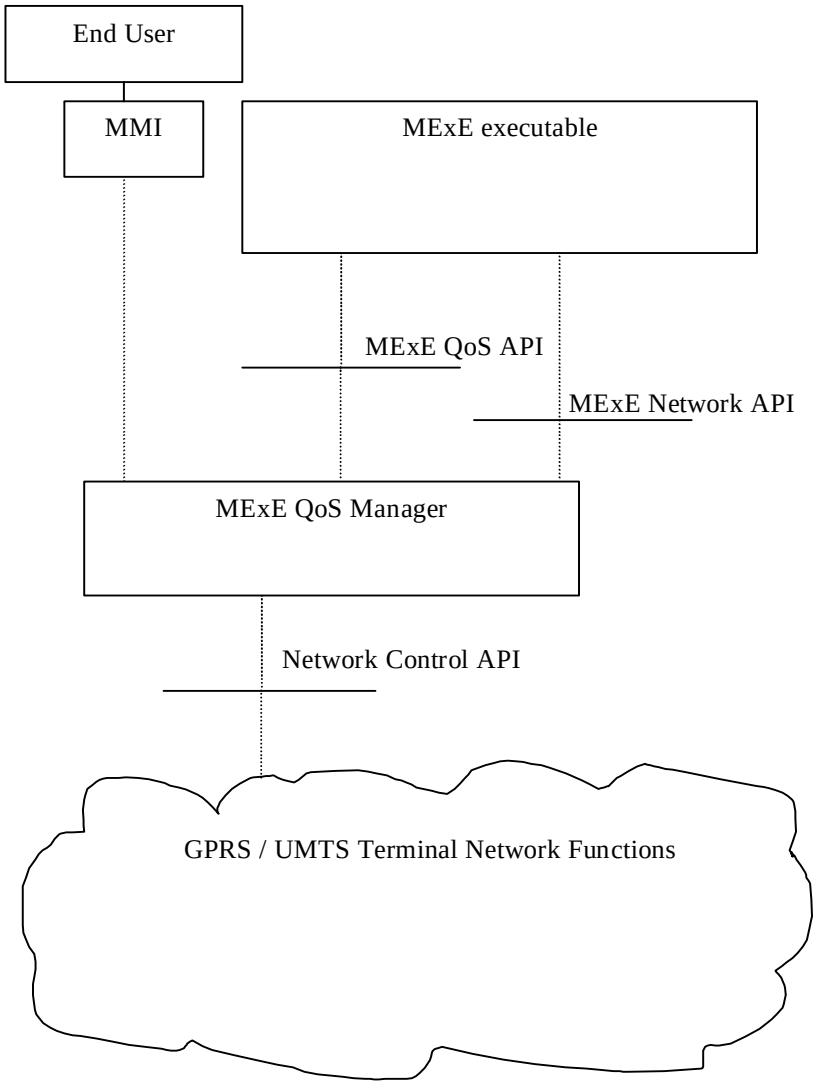


Figure 3~~15~~: Logical MExE device QoS manager elements

5.5.2 MExE QoS support

In the logical architecture depicted in figure 3~~15~~ "Logical MExE device QoS manager elements", a conceptual entity, a MExE QoS manager exists between the MExE executable and the Network Control API. A QoS API for MExE executables is provided and an API to the network is provided. The MExE QoS functions accommodate standard methods of end to end QoS provisioning.

For a MExE device supporting bearers defined by QoS, it is recommended that the MExE device shall support the following basic QoS operations:

- The end user should be able to manage the QoS directly via the MMI.

For MExE devices supporting bearers defined by QoS, the MExE device shall optionally 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;

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.

5.5.3 MExE QoS manager

As a conceptual entity, 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 MExE device, and provide functions such as:

- insert additional QoS signalling parameters;
- 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 MExE device's MMI.

5.5.4 Network control API

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

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

5.5.5 MExE QoS API

The MExE 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 1).

NOTE 1: The FLOWSPEC parameters, defined by the IETF Integrated Services Working Group, provide the QoS information required by QoS capable network elements.

Table 4~~10~~ "Example parameters" shows the set of example parameters.

Table 4~~10~~: 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 2: 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 clause 5.5.6 "Sources of Bearer Service Parameters".

5.5.6 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 MExE device capabilities;
- The MExE device 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.

5.5.7 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.

5.5.8 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.

5.6 Charging

Support of charging is outside the scope of MExE standardisation.

The following informative clauses provide a brief overview of the charging possibilities enabled by MExE.

5.6.1 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).

6 Generic MExE Security

6.1 Introduction

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 clause 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 device;
- the secure storage of these permissions (and permission type as defined in clause 6.5 "User permission types");
- 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 3GPP TS 22.057 [2] and is as follows (there is no implied hierarchy):

- MExE Security Operator Domain (MExE executables authorised by the HPLMN operator, as described in clause 6.3.1 "MExE executable permissions for operator, manufacturer and third party security domains");
- MExE Security Manufacturer Domain (MExE executables authorised by the ME manufacturer, as described in clause 6.3.1 "MExE executable permissions for operator, manufacturer and third party security domains");
- MExE Security Third Party Domain (trusted MExE executables authorised by trusted third parties, as described in clause 6.3.1 "MExE executable permissions for operator, manufacturer and third party security domains");
- MExE Untrusted Area. Untrusted MExE executables are not permitted to execute in a security domain (i.e. Operator domain, Manufacturer domain or Third Party domain) and execute in the Untrusted area, and have very reduced privileges as described in clause 6.3.2. "MExE executable permissions for untrusted MExE executables".

A MExE device shall support either all three security domains or no domains. If the security domains are not supported, then all applications shall be untrusted. The MExE device shall not support any subset of the three security domains. Support of the MExE Untrusted area is mandatory.

6.2 MExE executable integrity

If the 3 MExE security domains defined in clause 6.1 "Generic security" are not supported, then the pre-verification of MExE executables at launch time described in this clause is optional.

A potential threat is that MExE executables may be securely authenticated at the time of download, but tampered with or corrupted prior to being launched. Further a certificate may be compromised or expired. Authentication of a MExE executable at the time of download does not ensure that the MExE executable has not been modified when it is subsequently launched. Furthermore, authentication of a MExE executable at the time of launch does not ensure that the MExE executable is not modified during execution. Similarly, verification of the certificate at the time of download may not ensure that the certificate is valid at time of application launch, and verification of the certificate at the time of launch does not ensure that the certificate remains valid during execution.

Therefore, the MExE device shall ensure application integrity immediately prior to application execution.

Application integrity is defined as the state in which:-

- application code has not been modified since authentication; and
- the certificate containing the root public key is checked and known to be valid.

The mechanism by which the device preserves integrity is an implementation detail, dependant on the application storage mechanism and access. Examples of mechanisms that contribute to such application integrity could include :

- Storage of applications in a memory area that cannot be compromised on the device;
- Preventing launch of the application when the MExE device becomes aware that the certificate is invalidated;
- Full signature verification prior to each application invocation (see clause 6.2.1 “Full signature verification”);
- Optimised pre-launch signature verification (see clause 6.2.2 “Optimised pre-launch signature verification”);
- Periodic full signature verification by separate process during application execution.

The list of examples is not exhaustive and any other mechanisms ensuring application integrity may be equally considered.

A MExE device may furthermore ensure that the application code has not been modified during application execution.

6.2.1 Full signature verification

Full signature verification assumes that the procedure of validation for downloaded MExE executables and certificates is used. For more details see clause 6.7 “Certification and Authorization Architecture”.

6.2.2 Optimised pre-launch signature verification

This is an optional feature which is used to eliminate the potentially excessive overhead of checking a signature again after initial full certificate verification has already been performed.

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 device, 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 MExE device implementation.

6.3 MExE executable permissions

Support of MExE executable permissions as detailed in this clause is mandatory.

6.3.1 MExE executable permissions for operator, manufacturer and third party security domains

The following table 5~~6~~ "Security domains and actions" specifies the permissions of operator, manufacturer and third party security domains in the order of restriction.

The actions listed in the security table 5~~6~~ "Security domains and actions" 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) The 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 clause 6.5 "User permission types".

Untrusted MExE executables are not permitted access to any actions which access the phone functionality (phone functionality includes all the actions in table 5~~6~~ "Security domains and actions") except for the exceptions identified in clause 6.3.2 "MExE executable permissions for untrusted MExE executables".

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 5~~6~~ "Security domains and actions" shall be categorised into one of the groups in the security table 5~~6~~ "Security domains and actions" by comparing its action against the groups in order as they are listed in the table 5~~6~~ "Security domains and actions". 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. For example, 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. Support of core software download, which allows updating the ME radio, characteristics and properties by changing the core software in the ME (e.g. a new CODEC may be loaded into a ME, a new air interface, etc.)
3. (U)SIM smart card low level access includes functions, which allow communications at the transport service access point (send and receive application protocol data unit).
4. 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.
5. Network property access includes functions, which enable the management of operator-related data parameters and network settings.
6. Network services access includes all functionalities which result in or need interaction via the operator's network.
7. User private data access includes all functionalities which relate to management, reading or modifying of data that the user has stored in the MExE device including user preferences.
8. MExE security functions access includes all functionalities which, through an API relate to certificate handling in the MExE device; end to end encryption, signed content, hashing, access to public, private, secret keys stored in the MExE device or in a smart card.
9. Application access includes the functionalities which relate to launch provisioned functionality, MExE executables, external executables ((U)SIM tool kit application,…) usage.
10. Lifecycle management includes the functionalities which are needed for installing or removing MExE executables in the MExE device.
11. Terminal data access includes the functions which relate to accessing terminal data, i.e. not user data.
12. Peripheral access includes the functionalities related to peripherals other than user interface peripherals usage through a high level software application interface.
13. Input output user interface access includes the functionalities related to the user interface and user notification means usage.

Table 56: Security domains and actions

Actions	MExE Security Domains		
	Operator	**Manufacturer**	**Third Party**
Device core function access Start/stop radio Turn on/off device Write time and/or date Activate a user profile Modify a user profile	No		
Support of Core Software Download e.g. Update ME software	No	Yes	No
(U)SIM smart card low level access[11] Send APDU Slot management (power on/off, reset, port lock…)	No		
[11] – Access to (U)SIM is provided using more high level API as phonebook, application launching			
Network Security access Run algorithm Verify CHV/2 or UNBLOCK CHV/2 Activate/deactivate CHV Modify CHV/2	No		
Network property access Get IMSI Get home network Select network	Yes	No	
Network services access Initiate a voice/data connection [3] Accept a voice/data connection [3] Call forward [4] Multiparty call [4] Call deflection [4] Explicit call transfer [4] Terminate an existing connection Hold an existing connection Resume an existing connection Send point-point message (e.g. SMS, USSD) [4] Query network status Get signal level Get call list 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 clause 6.13.3 "Determining the administrator of the MExE device"			
User private data access [1] Read Write Get properties Delete Get Location Information Read stored SMS Delete stored SMS Modify user preferences	 Yes[2] Yes[2] Yes[2] Yes[2] Yes[2] Yes[2] Yes[2] Yes[7]		
[1] – User private data includes user files, phonebook, MSISDN, etc located on the MExE device. [2] – The user shall be able to specify data access permissions within the capabilities of the MExE 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.			

Actions	MExE Security Domains		
	Operator	**Manufacturer**	**Third Party**
MExE security functions access			
Install a certificate for a given domain	Yes [5]		
Uninstall a certificate for a given domain			
Replace a certificate for a given domain	Yes [5]		
Data encryption API			
Verify a signature API	Yes [5]		
Compute a digital signature API			
Hash a content API	Yes		
Non repudiation API	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			
Get application list	Yes[8]		
Launch an application	Yes[8]		
Get application status	Yes[8]		
Stop, suspend, resume an application	Yes[9]		
[8] – ME provisioned functionality access is limited to manufacturer domain. (U)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	Yes		
Install a MExE Executable			
Uninstall a MExE executable			
Terminal data access			
Get manufacturer software version	Yes		
Read time and date	Yes		
Peripheral access	Yes		
Sound generation to speaker (e.g. via stream)			
Set speaker volume			
printer access			
Monitor the power state			
Change the power state			
Activate/ access Serial port (RS232, IrDA, Bluetooth, USB …) access			
Activate/access Parallel port			
Activate/access Smart card other than (U)SIM card (Send APDU, Slot management)			
Input output User interface access			
Input device (keyboard, mouse …)	Yes[10]		
Output device (display)	Yes[10]		
Output notification device(smart icon, sound, light, vibrator …)	Yes		
[10] – Access request requires no user permission.			

The lists in the groups in table 5~~6~~ "Security domains and actions" 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.

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

6.3.2 MExE executable permissions for untrusted MExE executables

When the Security Domains are not supported then all executables are untrusted and they execute in the untrusted area for which the executable permissions are defined as follow in table 6~~7~~ "Executable permissions for untrusted MExE executables".

In order to facilitate untrusted MExE executables having some limited access to MExE device functionality beyond their very limited privileges, some of the access permissions in the previous table 5~~6~~ "Security domains and actions" are extended to untrusted MExE executables and described in table 6~~7~~ "Executable permissions for untrusted MExE executables" as well as in clause 6.11 "Separation of I/O streams".

The untrusted MExE executables permitted to use these facilities shall be MExE executables the user has downloaded him or herself, and not be MExE executables that have been pushed to the user. MExE executables on the MExE device due to the user having visited a particular Web site are considered to be MExE executables that the user had downloaded him or herself.

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

Table 6~~7~~: Executable permissions for untrusted MExE executables

	Classmark 1	Classmark 2	Classmark 4	Classmark 3
User Interface	An untrusted, uninstalled MExE executable (e.g. an applet) can access the user interface output and input 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 session) requires user permission. An installed untrusted MExE executable shall only be able to access the user interface output and input 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).			Untrusted MExE executables can access the user interface output and input without the user permission.
File, Persistent Data	File access is not permitted for untrusted MExE executables.			
	But, untrusted MExE executables can access files only in the MExE executable's own directory.			But, persistent data may be stored via the MIDP record management system (stores are shared between MIDlets in the same MIDlet Suite).
Transmission over the Access Network	Untrusted MExE executables shall be able to exchange data, voice, HTTP requests, etc. over the Access Network under the following conditions: The recipient of a transmission (e.g. a phone number, a URL, a server name, etc.) shall be presented to the user for permission by a provisioned functionality of the MExE device itself, even if this recipient was already presented by the executable (this facility would support, for example, "click to dial" buttons/links in untrusted MExE executables). It shall not be possible for an application to use a transmission channel that it did not initiate (except for MIDlets within the same MIDlet suite).			
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 device.			
Add Phonebook Entry	Untrusted MExE executables shall be able to add a phonebook entry (i.e. name and number only) under the following conditions: The name and the number to be added shall be displayed to the user for permission by a provisioned functionality of the MExE device and not by the MExE executable itself. The phonebook entry shall not be added without user permission. The function shall not be able to modify or delete any phonebook entry.			
Executable Interaction	Executable interaction is not permitted for untrusted MExE executables (except for MIDlets within the same MIDlet suite).			

NOTE: The functionality of "Generate DTMF tones" and "Add Phonebook Entry" is not supported by the MIDP at the moment.

6.4 Handling of MExE executables when their valid root public key is not available

This clause considers the effect on MExE executables when the root public key of a secure domain (e.g. operator, manufacturer, third party) is no longer available (e.g. when the UICC is being physically removed, or the root public key is no longer valid).

6.4.1 Launching of MExE executables when their valid RPK is not available

It shall not be possible to launch a MExE executable to run in a security domain unless the root public key of that security domain is available and valid.

6.4.2 Currently executing secure MExE executables when their valid RPK is no longer available

On detection that the valid root public key of a secure domain is no longer present, the MExE device shall permit MExE executables currently executing in the secure domain controlled by that root public key to continue executing. Furthermore, if the same RPK is available again, the executable is allowed to keep on executing. However, if a different RPK is validated, the currently running MExE executables (under the old RPK) in that secure domain shall be terminated.

6.5 User permission types

Support of user permission types is mandatory.

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 7~~8~~ "User Permissions". Support single action permission is mandatory, but support of blanket permission and session permission is optional.

Any request for user permission as described in table 7~~8~~ "User Permissions" must display a user friendly name identifying the signer of the corresponding MExE executable, if available. 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) shall be made available to the user upon request. If an application, for which user permission is being sought, is untrusted, the fact that the application is untrusted shall be at least visually indicated to the user, if the MExE device is capable of visual indication, whenever user permission is sought. Other means of indication are additionally permitted. If the MExE device is not capable of visual indication, or is not designed for use by a human user, other means of indication shall be used.

The MExE device shall allow user control of permissions relating to all action groups listed in the table 5~~6~~ "Security domains and actions" that are required by the MExE executable and supported by the MExE device.

Multiple action group permissions may be controlled in a single user action on the MExE device regardless of the permission type as listed in table 7~~8~~ "User Permissions". In such case, these action group permissions shall be made explicit to the user.

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

Table 78: 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.

6.6 Root Public keys

If the 3 MExE security domains defined in clause 6.1 "Generic security" are not supported, then the root public key management described in this clause is optional.

The definition of the secure mechanism in this clause to mark as valid a root public key certificate on the ME, is out of the scope of this specification.

6.6.1 Operator root public key

The ME may support secure storage for one or more certificates, each of which contains an operator root public key. The ME shall support the use and management of a certificate containing an operator root public key stored on the MExE-(U)SIM and in the ME. The ME shall behave according to clause 6.6.1.2 "ME actions on SIM insertion and/or power up". For support of public key management on the SIM and the USIM refer to 3GPP TS 51.011 [27] and 3GPP TS 31.102 [39] respectively. The certificate contains a root public key generated either by the operator, or by a CA trusted by the operator.

If the MExE device 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 the MExE executables 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 MExE device, 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 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 MExE device at any one time. A valid operator root public key on the (U)SIM shall always have precedence over any operator root public key on the ME. Any operator root public key(s) on the ME shall be marked invalid when a valid operator root public key is present on the (U)SIM.

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 MExE device (either ME or MExE-(U)SIM) and is marked as trusted.

6.6.1.1 Caching of root public keys

The ME shall behave as if it reads the operator root public key from the secure area every time the ME needs the key to verify a signature. Examples of the secure area include an area on a (U)SIM or a secure, persistent area on the ME.

If the ME uses a mechanism for caching public keys, it shall do so in a way that maintains the integrity of the secure area and is consistent with the keys stored in the secure area. With the exception of improved performance, the operation of the device using cached public keys must be indistinguishable from that of a device that reads the key from the secure area every time it uses the key for verification.

No cached version of a key may exist beyond the expiration or termination of the key in the secure area. For example, if the ME caches a root public key held on the (U)SIM, the ME shall purge the cache when the (U)SIM application is stopped (or the SIM card is withdrawn).

6.6.1.2 MExE device actions on detection of valid (U)SIM application and/or power up

This clause defines the sequence of actions on identification by the MExE ME that a valid SIM card, or USIM application on the UICC, has been detected (e.g. through insertion of (U)SIM card, power up of MExE device etc.). More specifically, these actions relate to the enabling or disabling of the operator domain and the status of the operator applications on the ME.

The requirements in this clause ensure that the operator domain on the ME belongs to the same operator as the operator that issued the valid (U)SIM application (if detected) in the MExE device and, if there is an operator root public key (ORPK) on the MExE-(U)SIM, that trusted operator applications on the MExE device were verified using that ORPK.

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

On power up the MExE device shall behave as dictated by figure 4~~9~~ "MExE device behaviour on power up" below.

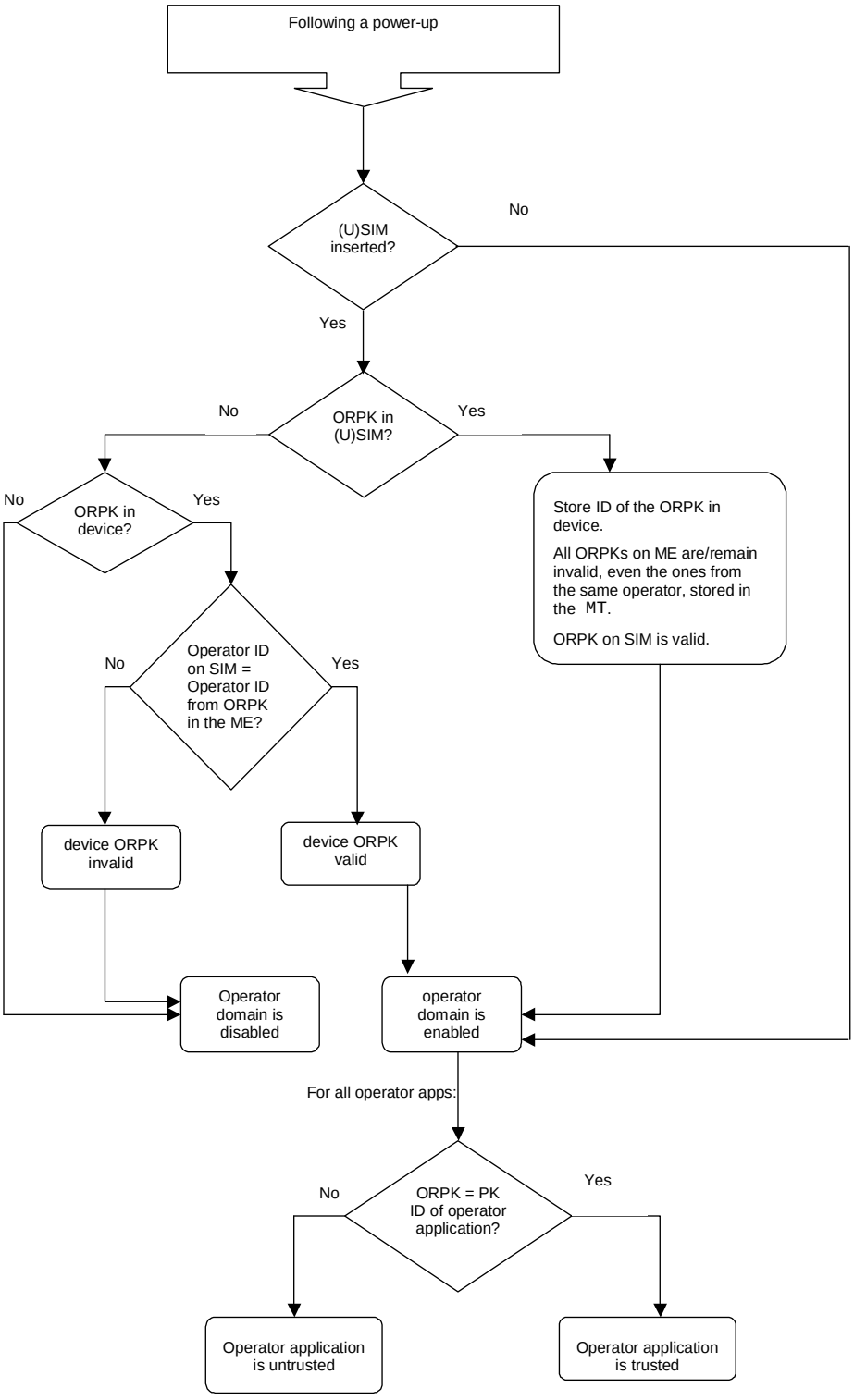


Figure 49: MExE device behaviour on power up

Note that the procedure in Figure 49 "MExE device behaviour on power up" checks for a match between the Operator ID on the (U)SIM and the Operator ID from the ORPK in the ME. Currently, one mechanism for defining the Operator ID on the (U)SIM is through use of the MCC+MNC. As an additional note, on DCS1900, the MCC+MNC is 6 digits, but elsewhere it is 5 digits. The MExE device needs to know how many digits to use. However, this is outside the scope of this specification. The implementations of MExE devices need to establish agreements on using the MCC+MNC as the Operator ID on the (U)SIM. Likewise, the implementations of MExE devices need to establish agreements on how to define the Operator ID belonging to the ORPK.

The ME shall only read the ORPK from the MExE-(U)SIM when required and shall not store a ORPK from the MExE-(U)SIM on the ME in a manner inconsistent with that detailed in clause 6.6.1.1 "Caching of root public keys".

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.

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

6.6.2 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 MExE device, or by a CA trusted by the manufacturer of the MExE 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 the MExE executables will be excluded from the manufacturer domain and marked as untrusted.

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

The Manufacturer shall put a root public key and optionally its corresponding disaster-recovery key in the ME 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 ME and is marked as trusted.

The manufacturer, and only the manufacturer, may use a secure mechanism to mark as valid/invalid a certificate containing the manufacturer root public key on the ME. It shall only be possible to use this mechanism to mark a certificate containing a new manufacturer root public key on the ME as valid, when all manufacturer root public keys are marked as invalid.

There shall be no more than one valid manufacturer root public key on the ME at any one time. Any other manufacturer root public key(s) on the ME device shall be marked invalid when a different manufacturer root public key is marked as valid on the ME.

6.6.3 Third party root public key

The ME shall support secure storage for at least one certificate containing a third party root public key. The ME shall support the use and management of certificates containing Third Party root public keys stored on the MExE-(U)SIM and in ME. For support of public key management on the SIM and the USIM refer to 3GPP TS 51.011 [27] and 3GPP TS 31.102 [39] respectively. The MExE device 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 Third Party 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 clause 6.10 "Certificate management" 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 MExE device.

The third party domain administrator, i.e. the Administrator (user or other body) shall be able to enable and disable Third Party root public keys by using CCM, see clause 6.10.1 "Certificate configuration message (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 clause 6.10 "Certificate management" for the management of Third Party root public keys.

6.7 Certification and authorisation architecture

If the 3 MExE security domains defined in clause 6.1 "Generic security" are not supported, then the certificate and authorisation architecture described in this clause is optional.

In order to enforce the MExE security framework a MExE device 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 device may want to authenticate content from many sources, a public key based solution is mandatory. Before trusting MExE executables, the MExE device will therefore check that the MExE executable was signed with a private key, for which the MExE device has the corresponding public key. The corresponding public key held in the MExE device must either be a root public key (securely installed in the MExE device, e.g. at manufacture) or a signed public key provided in a certificate. The MExE device 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 clause 6.7.1 "Certification requirements". An example authorisation and certification process is described in clause 6.7.3 "Example certification process".

6.7.1 Certification requirements

A MExE device 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 device, say, at the time of manufacture.

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

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

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

A MExE device shall support at least one level of certificate under operator, manufacturer or Third Party root public keys. The MExE device shall support at least one level of certificate chain analysis in a signed content package, as shown in figure 5~~7~~ "Trust hierarchy".

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 MExE device and if the certificate being verified has not expired.

Public keys shall not be shared between domains.

6.7.1.1 MExE terminal requirements for certificate processing

A MExE device shall support the processing of X509 certificates profiled in the “WAP Certificate and CRL Profile” [47] together with additional requirements defined in the MExE specification, see clause 6.9.1.1 “X509 version 3”. The certificate chain depth is still mandated to be one level only, as mentioned in clause 6.7.1 “Certificate requirements” and indicated in figure 5~~7~~ "Trust hierarchy".

A MExE device shall support the SHA1WithRSA signature algorithm. The object identifier value can be found in [49]. A MExE device may also support other signature algorithms.

MExE devices may also support the processing of other certificate formats.

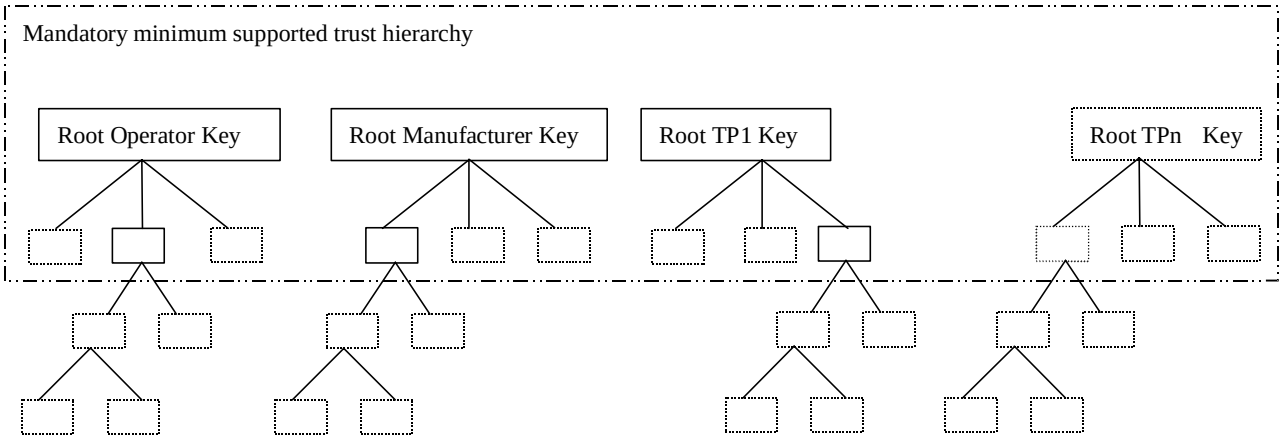


Figure 57: Trust hierarchy

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).

6.7.2 Certification administration requirements

For control of third party certificates, the MExE device supports storage of a certificate containing an administrator root public key as detailed in clause 6.13.1 “Administrator root public key”.

This certificate is managed separately from the hierarchy of Figure 57 “Trust Hierarchy” discussed in clause 6.7.1 “Certification requirements”. The administrator root public key in this certificate is primarily used for designating an administrator of the third party certificates. Note, the administrator root public key does not implicitly define a security domain, and is used in complement with the root public keys of the operator, manufacturer, and third party domains.

The relationship of the administrator certificate (and root public key) to the management of third party certificates is detailed in part of clause 6.10 “Certificate management”.

The relationship of the administrator certificate to the mechanism for determining if a third party certificate is trusted is detailed in part of clause 6.10.1 “Certificate configuration message (CCM)”.

Mechanisms for designating an administrator are detailed in clause 6.13.2 “Provisioned mechanism for designating administrative responsibilities and adding third parties in a MExE device”.

6.7.3 Example certification process

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

Root public keys for a number of Certification Authorities (CAs) are installed in the MExE device, along with the MExE device 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 device downloads a MExE executable of the third party software developer.

5. The MExE device 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 device 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.

All downloaded applications shall follow the procedure described in clause 6.7.4 "Certificate Chain Verification" in order to verify the application signature and the certificate chain. If the 3 security domains are not supported, the procedure described in the next clause is optional.

6.7.4 Certificate Chain Verification

This clause presents the procedure of validation of any downloaded MExE executable. It checks for the presence of the signature used to sign the application as well as the presence and integrity of all the certificates needed to successfully verify the signature. As a result, the application under scrutiny is deemed trusted or untrusted, i.e. will be allowed execution in one of the secure domains or in untrusted area, or otherwise the application will not be allowed to be executed and will be deleted. In any outcome of the verification, the user is notified about the result. The user also may wish to see certificate details if the application is allowed to be executed on the MExE device.

The MExE device shall follow "certificate verification" procedure as described below. The procedure shall contain at least the following logical phases (not necessarily in the order stated below):

Signature and Certificate Verification Supported: Checks whether signature and certificate verification procedure is supported on the MExE ME.

Executable with Signature and End Entity Certificate (note): Checks whether the executable contains a signature together with the corresponding end entity certificate.

Valid Application Signature (note): This phase comprises the following checks:

- Check if the signature and the end entity certificate formats are supported by the device. If this check fails, the application is classified as untrusted.

- Check if the signature algorithm is supported/known by the device. If this check fails, the application is classified as untrusted.

- Check if the signature can be cryptographically verified by using the accompanying end entity certificate . If this check fails, the application is not allowed execution and is deleted.

Complete set of Intermediate Certificates Available (note): Checks if all the necessary intermediate certificates (certificates between the RPK and the end entity certificate) are available.

Valid RPK on (U)SIM/ME: Checks if a valid RPK (not expired) exists on the (U)SIM or on the ME that could verify a certificate chain originating from the end entity certificate accompanying the application.

NOTE: These steps could include validation (e.g. expiration, revocation, etc.) checking by means of e.g. OCSP, SCVP, CRL-Consultation, and etc. The use of certificate revocation checking is recommended but is not mandated or defined in this specification.

Certificate Chain Cryptographically Verified: Checks if all the certificates from the end entity certificate to the RPK can be verified cryptographically. Certificate verification shall be performed according to the functional requirements given in clause 6.1 "Basic Path Validation" of RFC 2459 [43] excluding revocation checking.

Secure Domains Supported: Checks whether MExE ME supports secure domains.

Only if all the above checks are successful, the downloaded application is deemed trusted and is allowed to be executed in the designated trusted domain (operator, manufacturer, trusted third party). Otherwise, the application is either untrusted (execution in the untrusted area only is allowed) or deleted (execution is not allowed at all) as per the figure 6~~8~~ “Certificate Chain Verification Diagram” and as explained above. The executable shall only be designated into one of the trusted domains, and it shall be possible to verify the certificate chain unambiguously to one and only one root public key.

The MExE ME shall allow for a "user notification" procedure as described below.

It shall be possible to display certificate details to the user if requested, however, since the terminal might not have a display or might not be meant for a human user the methods presented in "user notification" section are not discussed any further in this specification. Figure 68 “Certificate Chain Verification Diagram” shows an example of the certificate chain verification procedure.

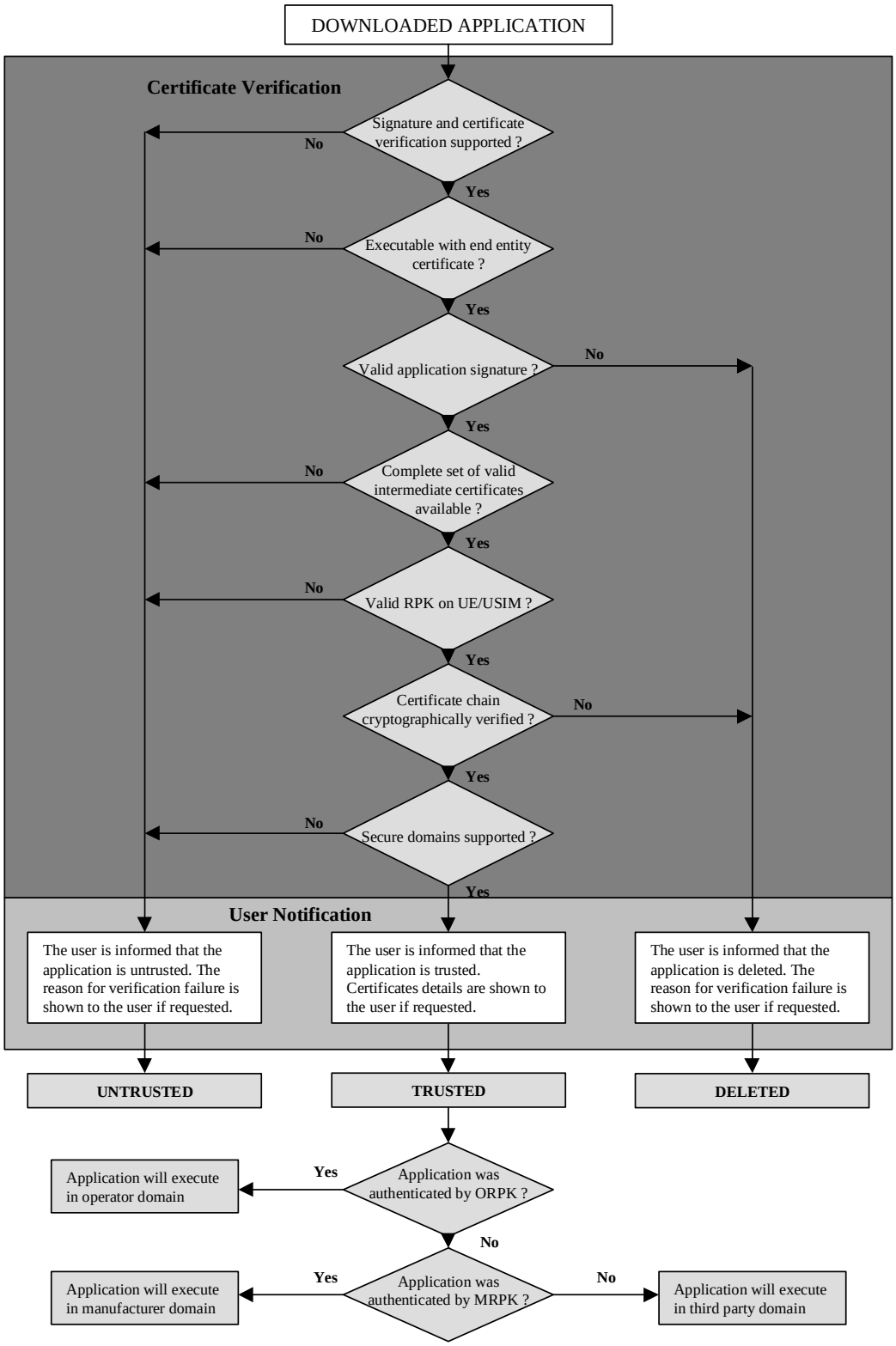


Figure 68: Certificate Chain Verification Diagram

6.8 Usage of Signed Content

6.8.1 Signed packages used for installation

If the 3 MExE security domains defined in clause 6.1 "Generic security" are not supported, then the signed packages used for installation, described in this clause, are optional.

The Java Archive (JAR) file format shall be supported on classmark 2 and 3 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`
The information contained within the manifest file is represented as so-called "name: value" pairs, where "name" is represented by `MExE-Implementation-Type`. Groups of name-value pairs are known as a "section", where sections are separated from other sections by empty lines.

The `MExE-Implementation-Type` value shall be one of the following:-

- "**`MExENativeLibrary`**"

  in the case of a MExE Native Library (as described in 8.3.2 "Installing MExE native libraries");

- "**`TTPCertificate`**"

  in the case of a certificate containing a 3rd party root public key (as described in 6.8.2 "Installation of root certificates in a signed data package");

- "**`ManufacturerCertificate`**"

  in the case of a certificate containing a manufacturer root public key (as described in 6.8.2 "Installation of root certificates in a signed data package");

- "**`OperatorCertificate`**"

  in the case of a certificate containing an operator root public key (as described in clause 6.8.2 "Installation of root certificates in a signed data package");

- "**`AdminCertificate`**"

  in the case of an administrator certificate, which shall consist of a section containing both the administrator certificate and a CCM (as described in clause 6.8.2 "Installation of root certificates in a signed data package"); or

- **"OrdinaryTTPCertificate"**

  in the case of a certificate or certificate list containing 3rd party public key(s). An example of a certificate list syntax can be found in [52]

- **"OrdinaryManufacturerCertificate"**

  in the case of a certificate or certificate list containing manufacturer public key(s). An example of a certificate list syntax can be found in [52]

- **"OrdinaryOperatorCertificate"**

  in the case of a certificate or certificate list containing operator public key(s). An example of a certificate list syntax can be found in [52]

- "**`CCM`**"

  in the case of a CCM (as described in clause 6.8.2 "Installation of root certificates in a signed data package"); 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 clause 6.8.3 "Installation of other signed data").

Refer to [42] for full details of how to encode the "name: value" pairs and "section" in a JAR manifest file.

See figure 7~~15~~ "Signed packages". 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 MExE device 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 MExE device to determine how to offer appropriate upgrades.)

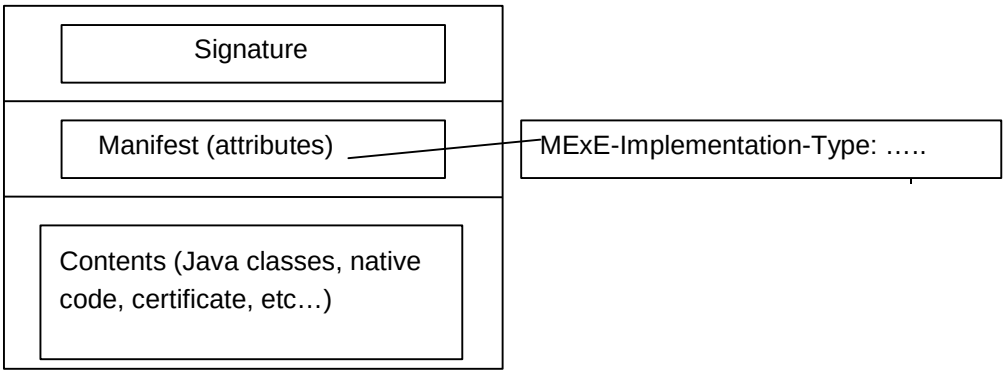


Figure 7~~15~~: Signed packages

6.8.2 Installation of root certificates in a signed data package

Root certificates in a signed package (whose signature verifies as described in clause 6.6 "Root Public keys" to the Manufacturer root, Operator root, or the Administrator root), may be installed to the root public key store on the MExE device. 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 8~~9~~ "Allowed certificate types in signed packages". The MExE device 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 (e.g. a JAR file in the case of Java based MExE classmarks) shall contain two files: the Administrator root public key and the CCM.

Table 8~~9~~: 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

6.8.3 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 clause 6.6.2 "Manufacturer root public key" 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.

6.9 Certificate Format

6.9.1 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 56 "Security domains and actions"). 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.509 version 3 only. Support for WTLS, X9.68 certificate formats is for further study.

6.9.1.1 X.509 version 3

The MExE certificate format as specified in clause 6.7.1.1 “MExE terminal requirements for certificate processing” shall support the X.509 version 3 access-Restriction extension.

X509 version 3 provides a mechanism to define extensions. An Object identifier (OID) is 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 MExE device then the certificate shall be considered invalid.

The syntax of the extension is defined in annex C “Access restriction certificate extension”.

6.10 Certificate management

If the 3 MExE security domains defined in clause 6.1 "Generic security" are not supported, then the certificate management described in this clause is optional. The manufacturer may load initial third party certificates on the ME. 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 (U)SIM if support for certificate storage on the (U)SIM exists (e.g. MExE-(U)SIM) or in the MExE device. For (U)SIMs not having certificate storage the administrator root may be downloaded using the root download procedure described in clause 6.13.4 "Administrator root certificate download mechanism".

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.

The Administrator may mark trusted/untrusted Third-Party certificates using Certificate Configuration Messages (see clause 6.10.1 “Certificate configuration message (CCM)”.

Users cannot add or delete any Operator or Manufacturer certificate containing a root public key.

An example of public key infrastructure certificate management protocols can be found in [33].

6.10.1 Certificate configuration message (CCM)

If the 3 MExE security domains defined in clause 6.1 "Generic security" are not supported, then the certificate configuration message described in this clause is optional.

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 device. 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 clause 6.10.1.4 "Authorised CCM download mechanisms".

The Certificate Configuration Message shall be as shown in figure 8~~10~~ "Format of a CCM". 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.

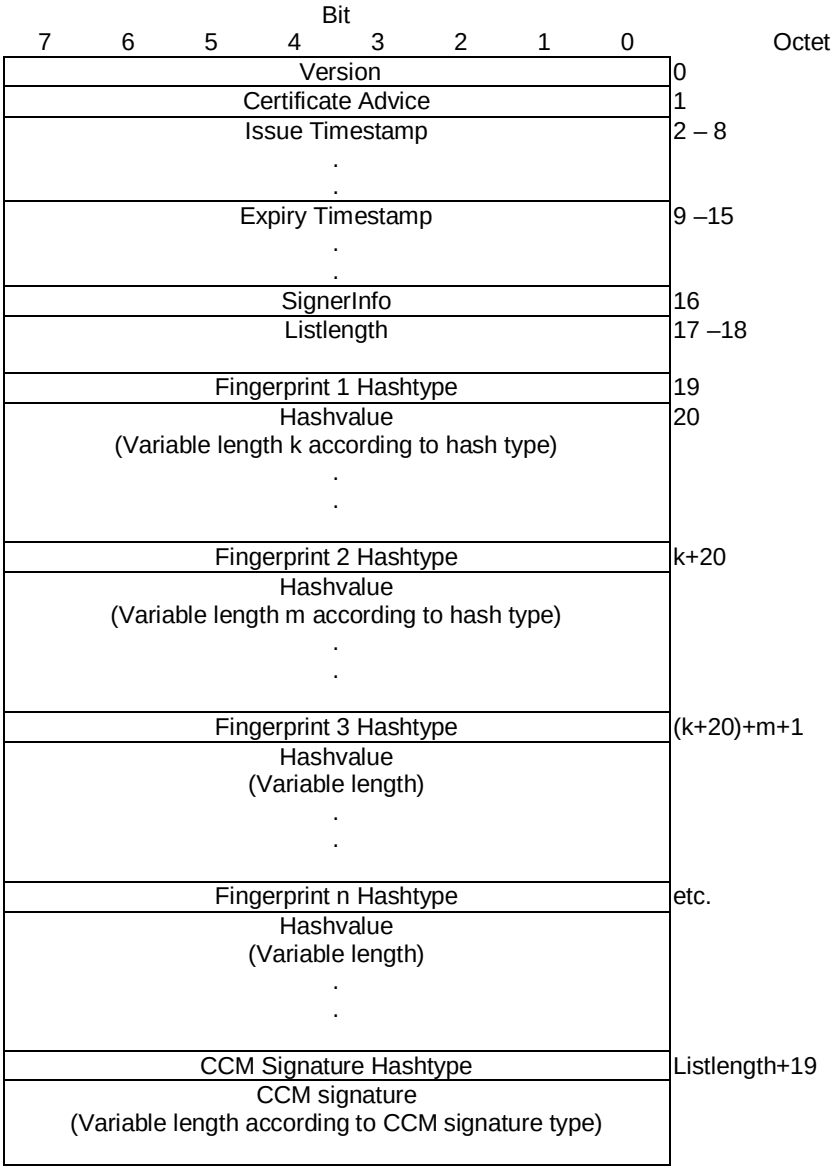


Figure 8~~10~~: Format of a CCM

Version = The CCM format version is 0. All other values are reserved for future use.

Certificate Advice = 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 device maintains information on the last valid CCM message received. A replay attack is an attacker replaying a previous valid CCM message to a MExE device 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, disable-all or enable present list.

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

The length of the Hashvalue field, number of octets output by the selected hash type, is 16 for MD5 [23] or 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 MExE device. If no administrator certificate is on the MExE device or if the signature is not verified, 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 and 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 MExE device, do not enable any future certificates (this option allows 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 executable. If a CCM message is received while MExE executables are currently executing, the implementation shall check to ensure that any executables no longer in the Third Party domain, have their permissions re-configured appropriately and actions that are no longer permissible are terminated.

6.10.1.1 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.

6.10.1.2 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.

6.10.1.3 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 9~~11~~ "Field mapping convention".

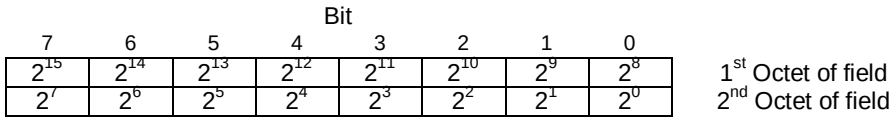


Figure 9~~11~~: Field mapping convention

6.10.1.4 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 10~~12~~ "CCM Message URL storage format":

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

Figure 10~~12~~: 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.

URL = a field where 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 10~~12~~ "CCM Message URL storage format", 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.

6.11 Separation of I/O streams

Support of the separation of I/O streams is mandatory.

Except for the MExE Classmark 3 executables (MIDlets) from the same MIDlet Suite, 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).

6.12 Core software download

Support of core software download is optional.

Core software download enables the MExE device radio, characteristics and properties to be updated by changing the software in the MExE device. E.g. a new CODEC may be loaded into a MExE device, a new air interface, etc. This process could include the transfer of executable code and software patches over the air.

This updating of core software (e.g. the Software Defined Radio (SDR) concept) can in principle be generically supported within the MExE framework by a MExE service that executes in the manufacturer security domain, and uses handset manufacturer proprietary APIs. Possible scenarios for the support of this functionality include:

- A MExE service that can be transferred to, and executed in, the manufacturer domain. The service would use manufacturer APIs to perform the software update, radio re-configuration, etc.
- A core software download application that executes in the manufacturers' domain that acts like a user agent in conjunction with a server to transfer software as needed or requested by the user. The core software download application uses manufacturer APIs to perform the software update, radio re-configuration, etc.

Similar functionality may be supported by a downloaded MExE service using manufacturer's OEM classes. All such OEM classes shall comply with the MExE security requirements in table 5~~6~~ "Security domains and actions" and table 6~~7~~ "Executable permissions for untrusted MExE executables".

The support of core software download functionality in a MExE device shall only be under the control of the MExE device manufacturer.

6.13 Administrator Concept

6.13.1 Administrator root public key

To help with the control of Third-Party certificates, the ME shall support secure storage for a certificate containing an administrator root public key. The ME shall support the use and management of a certificate containing an Administrator root public key stored on the MExE-(U)SIM and in the ME. The ME shall behave according to clause 6.13.3 “Determining the administrator of the MExE MS”. For support of public key management on the SIM and the USIM refer to 3GPP TS 51.011 [27] and 3GPP TS 31.102 [39] respectively.

A secure mechanism may be used to mark as valid/invalid a certificate containing the administrator root public key on the MExE device. It shall only be possible to use this mechanism to mark a certificate containing a new administrator root public key on the ME as valid, when all administrator root public keys are marked as invalid.

There shall be no more than one valid administrator root public key on the MExE device at any one time. A valid administrator root public key on the (U)SIM shall always have precedence over any administrator root public key on the ME. Any administrator root public key(s) on the ME shall be marked invalid when a valid administrator root public key is present on the (U)SIM.

The MExE device shall support the administrator designation mechanism explained in clause 6.13.2 "Provisioned mechanism for designating administrative responsibilities and adding third parties in a MExE device" and the secure downloading of CCMs explained in clause 6.10.1.4 “Authorised CCM download mechanisms”.

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 device to manage the administrator root public key (including the download of a new administrator root public key) as defined in clause 6.13.3.2.1 "Administrator of the MExE device is the user". 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 device as described in clause 6.13.4 "Administrator root certificate download mechanism".

If the Administrator root public key is stored in the (U)SIM, the ME shall only read the Administrator root public key from the MExE-(U)SIM when required and shall not store the Administrator root public key from the MExE-(U)SIM on the ME in a manner inconsistent with that detailed in clause 6.6.1.1 "Caching of root public keys".

See clause 6.10 "Certificate management" for the management of Administrator root public keys.

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 MExE-(U)SIM (see [27] and [39]), 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 (U)SIM, then procedures relating to this are out of the scope of this specification.

6.13.2 Provisioned mechanism for designating administrative responsibilities and adding third parties in a MExE device

If the 3 MExE security domains defined in clause 6.1 "Generic security" are not supported, then the administrator concept described in this clause is optional.

All applications in the Third-Party security domain are to be signed by a key which shall be verified back to a Third Party root public key on the MExE device. The Third Party root public keys shall be managed (e.g. addition/mark trusted/mark untrusted) by an administrator that is designated by the owner of the MExE device using the MExE administrator provisioning mechanism. A mechanism is required to be provided to enable the owner of the MExE 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-(U)SIM wants to be a temporary administrator.

6.13.3 MExE administrator determination mechanism

The administrator of the MExE device shall be determined by a two part logical process with the first part shown in the flowchart in figure 11~~13~~ "MExE administrator determination mechanism". The second part of the logical process is in figure 12~~14~~ "MExE administrator determination mechanism, for MExE-(U)SIM supporting third party certificates".

During power-up or MExE-(U)SIM insertion event, the provisioned mechanism shall look for an administrator root public key that is stored on the MExE-(U)SIM.

6.13.3.1 Determining the administrator of the MExE device

If an administrator root public key cannot be found on the MExE-(U)SIM, the provisioned mechanism shall look for one on the MExE device. This leads to the following two cases:

- 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 device.

- 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. Note that the owner of the administrator root public key could be the user.

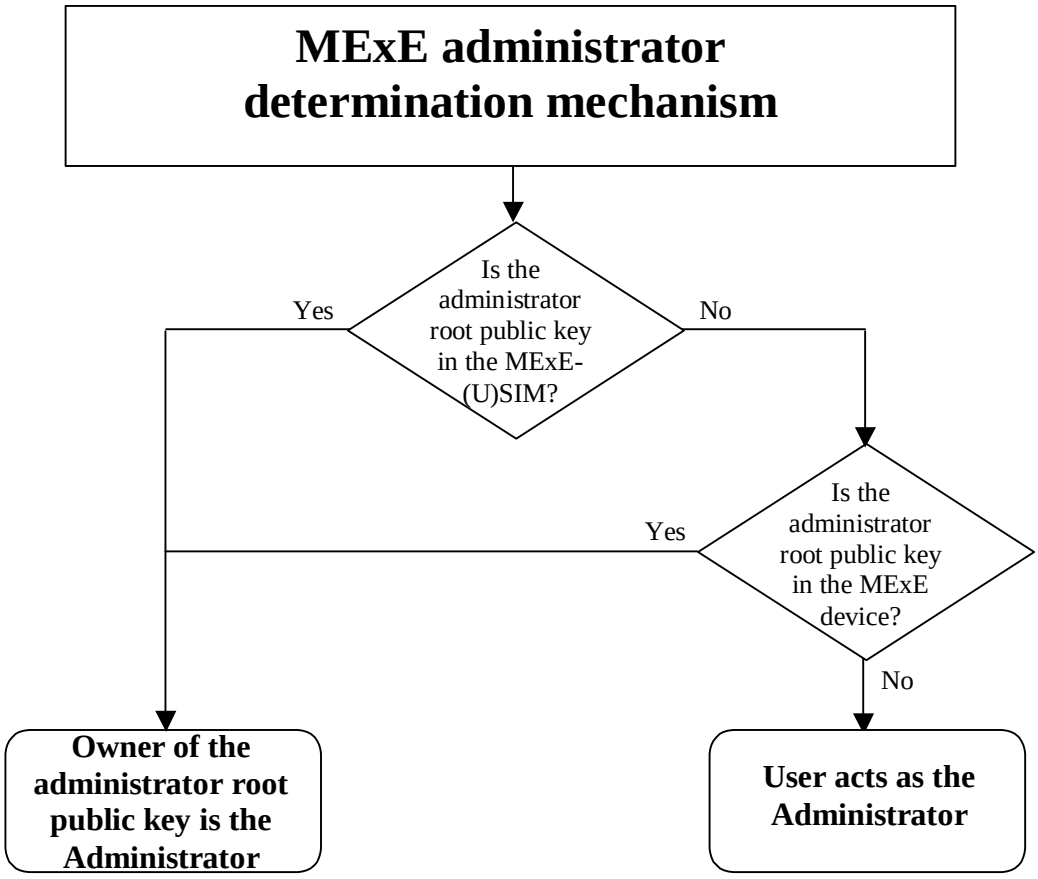


Figure 1113: MExE administrator determination mechanism

6.13.3.2 Determining the administrator of the MExE device, for MExE-(U)SIM supporting third party certificates

The second part of the administrator determination mechanism is subsequently defined (see figure 1214 "MExE administrator mechanism, for MExE-(U)SIM supporting third party certificates "), and shall be initiated after a power-up or MExE-(U)SIM insertion event is processed.

The following clauses 6.13.3.2.1 "Administrator of the MExE device is the user" and 6.13.3.2.2 "Administrator of the MExE device is not the user" assume that Third Party certificates can be added using the MExE-(U)SIM, however Third Party certificates may be added using a non-(U)SIM approach (e.g. inserted at the time of manufacture, signed package download etc.).

6.13.3.2.1 Administrator of the MExE device is the user

If the administrator is the user, then a check shall be made to determine whether there is a MExE-(U)SIM. If a MExE-(U)SIM is present, then a check shall then be made to determine whether there is a Third Party or an Administrator certificate containing a root public key in the MExE-(U)SIM. The second part of the administrator determination mechanism shall allow the MExE device to determine (via a format) what type of certificate is present:

- certificate present - third party (CP-TP)

  If a Third Party certificate containing a root public key is present in the MExE-(U)SIM then this certificate shall be considered by the MExE device as a Third Party certificate, whilst that valid MExE-(U)SIM application is present in the MExE device. The user shall be queried to allow or disallow the certificate as a Third Party.

- certificate present - administrator (CP-Admin)

  If a temporary Administrator certificate containing a root public key is present in the MExE-(U)SIM, the user shall be queried whether to allow the certificate on the MExE-(U)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 (i.e. the Administrator becomes the User). 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-(U)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 5~~6~~ "Security domains and actions".

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

6.13.3.2.2 Administrator of the MExE device is not the user

If the administrator is not the user, then a check is made to determine if there is a MExE-(U)SIM. If a MExE-(U)SIM is present, then a check is made to see if there is a Third Party or an Administrator certificate containing a root public key in the MExE-(U)SIM.

If an Administrator certificate containing a root public key is present in the MExE-(U)SIM, then a comparison is made of this certificate's root public key with the Administrator root public key on the MExE device for the following cases:

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

If the owner of the public key in the certificate on the MExE-(U)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 5~~6~~ "Security domains and actions".

If the certificate is to be a Third Party certificate containing a root public key, 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.

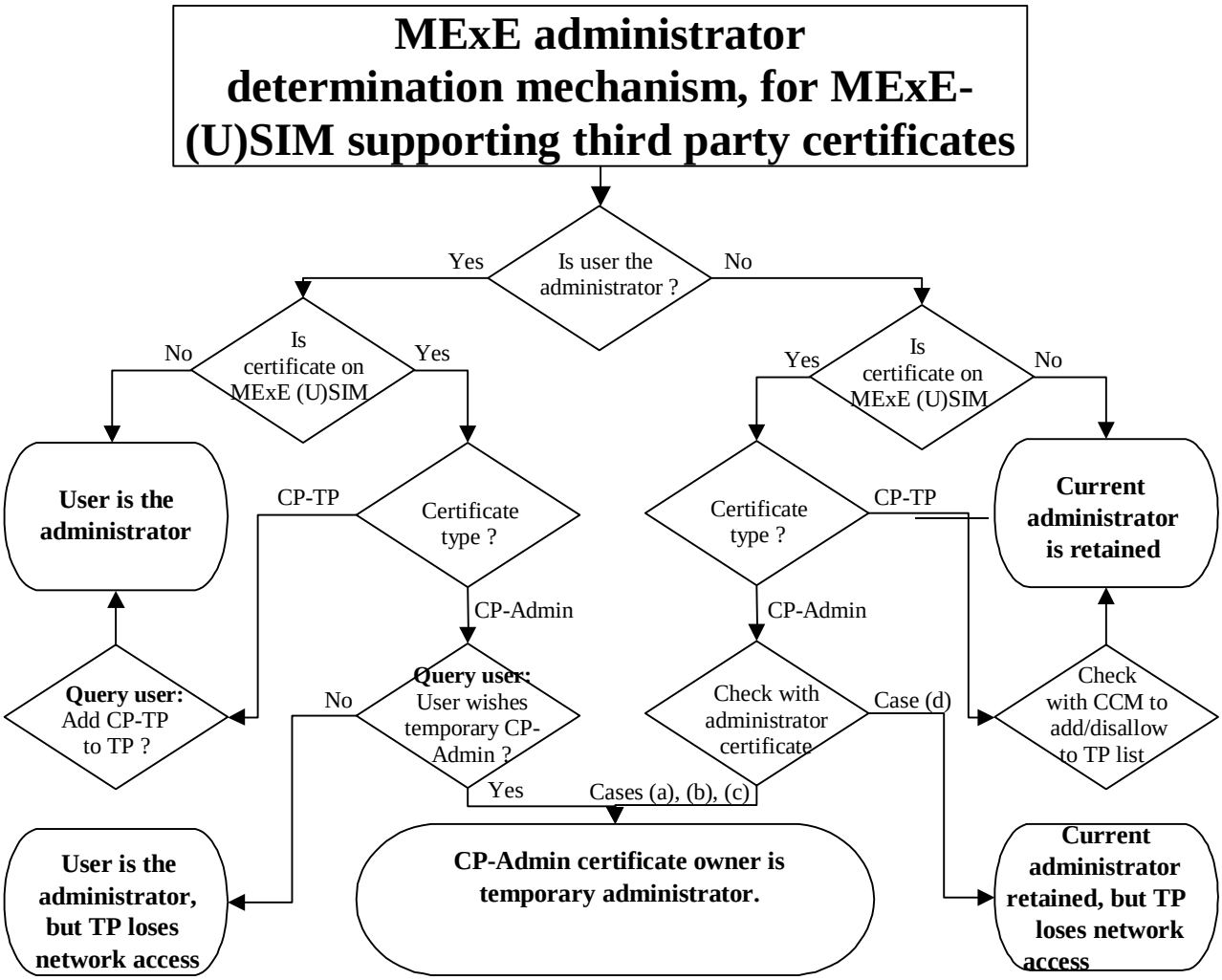


Figure 1214: MExE administrator determination mechanism, for MExE-(U)SIM supporting third party certificates

6.13.4 Administrator root certificate download mechanism

MExE devices supporting (U)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 MExE 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 MExE 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 MExE 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 MExE device unaltered.

7 MExE Classmark 1 (WAP environment)

7.1 Introduction

Classmark 1 MExE devices are based on Wireless Application protocol (WAP).

The Wireless Application Protocol is a standard to present and deliver wireless information and telephony services on mobile phones as well as other wireless terminals. Supporting mandatory features of WAP, WAP enabled devices provide access to the World Wide Web based content for small mobile devices.

7.1.1 WAP MExE devices

Support of WAP in a MExE classmark 1 device as detailed in this clause is mandatory.

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.

7.1.2 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 that a QWERTY keyboard or 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.

7.2 Non Security

7.2.1 WAP components

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.

7.2.2 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

7.2.2.1 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.

7.2.2.2 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 13 "WAP network topology".

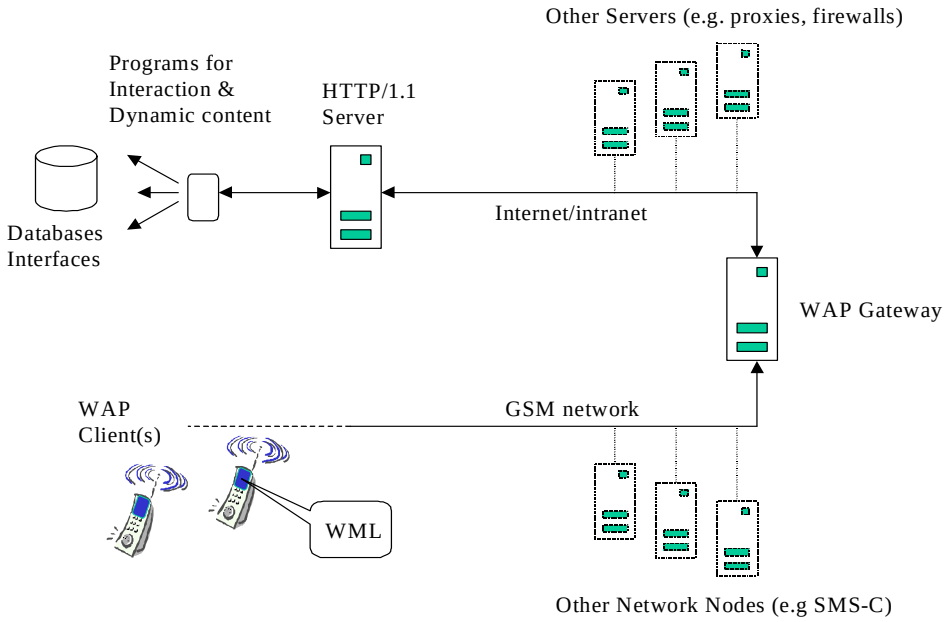


Figure 13: WAP network topology

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.

7.2.2.3 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.

7.2.2.3.1 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.

7.2.3 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.

7.2.4 CC/PP over WSP (Classmark 1)

In Classmark 1, according to the WAP User Agent Profile Specification [17], the CC/PP description is encoded with WBXML [45] after which it is carried over by WSP, [17].

7.3 Security

7.3.1 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.

7.3.2 Local phonebook

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

8 MExE Classmark 2 (PersonalJava environment)

8.1 Introduction

Classmark 2 specifies Personal Java enabled devices with the addition of the JavaPhone API.

The Personal Java[3] application environment is the standard Java environment optimised for consumer electronic devices designed to support World Wide Web content including Java applets. The Personal Java API is a feature level subset of J2SE with some Java packages optional and some API modifications necessary for the needs of small portable devices (for example an optimised version of the Abstract Windowing Toolkit targeted to small displays).

JavaPhone[4] is a vertical extension to the Personal Java platform that defines APIs for telephony control, messaging, address book and calendar information, etc.

8.1.1 Classmark 2 MExE devices

Support of PersonalJava in a MExE classmark 2 device as detailed in this clause is mandatory.

MExE Classmark 2 devices shall be based on the API for Personal Java, which defines the required and optional components of Personal Java /JavaPhone APIs that shall be used to realise a Classmark 2 compliant MExE device.

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

Many aspects of the MExE Classmark 2 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 API shall be supported.

8.1.2 High level architecture

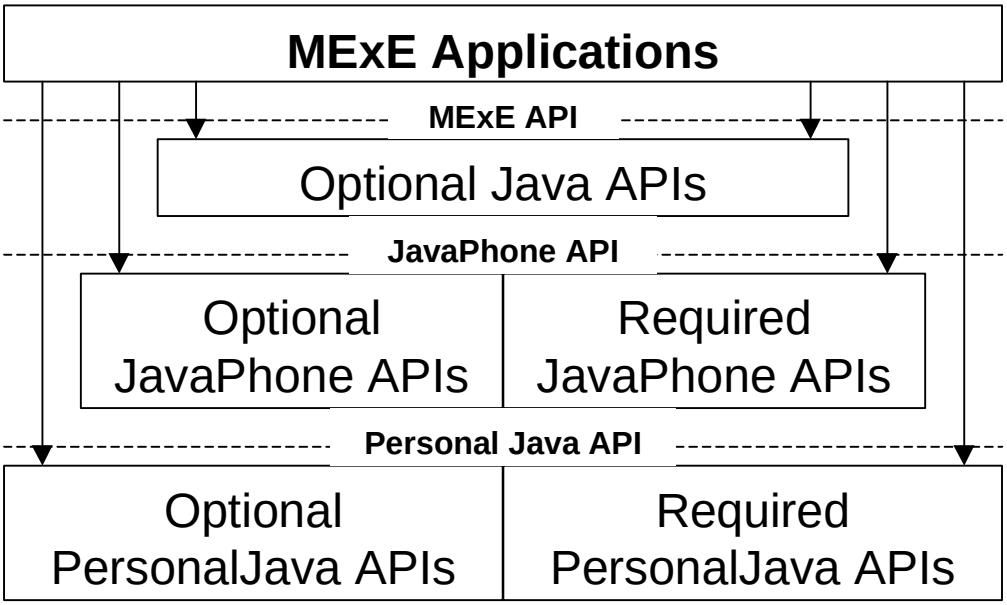


Figure 14: Basic functional architecture of a PersonalJava MExE device

The functional architecture of a Java MExE classmark 2 device is shown in figure 14 "Basic functional architecture of a PersonalJava MExE device". Java applets, applications, and services access functionality via the MExE PersonalJava API. The MExE PersonalJava API is based on a combination of optional Java APIs and the Wireless Profile of the JavaPhone API [4]. The JavaPhone API is based on the PersonalJava API [3].

8.2 Non Security

8.2.1 High level functions

8.2.1.1 Optional Java packages

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. Java packages are containers for the highest level of functionality in the Java language. In some cases, optional Java packages are specified in terms of Java classes and interfaces. Classes and interfaces are elements contained inside packages.

The following table 94 "Optional Java packages of the Wireless Profile of the JavaPhone APIs" specifies the defined optional Java packages of the Wireless Profile of the JavaPhone APIs [4]. 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 94: Optional Java packages of the Wireless Profile of the JavaPhone APIs

JavaPhone API	Java package	Optional/Mandatory
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

8.2.1.2 Required and optional PersonalJava APIs

MExE classmark 2 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.

8.2.1.3 Required and optional JavaPhone APIs

The JavaPhone APIs extend the PersonalJava APIs to provide functionality unique to telephony devices. MExE classmark 2 devices shall support the Wireless Profile of the JavaPhone API specification [4]. MExE classmark 2 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 MExE classmark 2 devices.

8.2.1.3.1 Application installation

MExE classmark 2 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

  the use of icons to launch applications is optional, however 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.

8.2.1.3.2 Power

MExE classmark 2 devices shall support the Power Monitor package (javax.power.monitor) as specified by the JavaPhone API to access the power level of the MExE 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. MExE classmark 2 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.

8.2.1.3.3 Datagram recipient addressing

The syntax described in Concrete Addressing [4] specifies the format to be used for raw text-only GSM SMS messages, UDP datagram via IP, and WAP datagram via GSM SMS message(s).

As a minimum, the formats above shall be supported if the MExE device supports the relevant bearer/transport combination.

NOTE: For the purposes of this clause, "GSM SMS" means SMS as defined by the 3GPP specifications including 3GPP TS 23.040.

8.2.1.4 Required and optional MExE PersonalJava APIs

MExE classmark 2 devices shall not be required to support any other Java APIs.

MExE classmark 2 devices may optionally support any other Java APIs which comply with the MExE security requirements in table 5~~6~~ "Security domains and actions", such as:

- OCF SmartCard API OpenCard, available from [21]. If the MExE device supports smartcards other than the (U)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.

8.2.1.5 Mandated services and applications

8.2.1.5.1 Network protocol support

Support for network protocols in MExE classmark 2 devices is specified in the following table 105 "Support for network protocols":.

Table 105: Support for network protocols

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

8.2.2 CC/PP over HTTP (Classmark 2)

In Classmark 2 the CC/PP is carried over by using CC/PP over HTTP, [15] and optionally CC/PP over WSP, [17].

8.2.3 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.

8.3 Security

If the 3 MExE security domains defined in clause 6.1 "Generic security" are not supported, then the Java security described in this clause is optional.

8.3.1 PersonalJava 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 security domain shall be implemented as running in a sandbox, configured with different privileges corresponding to those of the domain. If the security domains are not supported then the Java sandbox security model shall be supported and it shall be configured for untrusted MExE executables support only, as defined in clause 6.3 “MExE executable permissions”.Using the fine grain Java security system [19], each MExE security domain will be a set of constraints within which a Java fine grain security domain can be configured.

8.3.1.1 Java applet certification in PersonalJava

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".)

8.3.1.2 Java application signature verification in PersonalJava

The verification of the certification of the application or applet shall be performed as described in clause 6.6 ”Root Public keys” and clause 6.13.2 “Provisioned mechanism for designating administrative responsibilities and adding third parties in a MExE MS”.

8.3.1.3 Java loading native libraries in PersonalJava

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

8.3.2 Installing MExE native libraries

A signed native library whose signature verifies as describe in clause 6.6.2 "Manufacturer root public key" 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.

9 MExE Classmark 3 (J2ME CLDC environment)

9.1 Introduction

Classmark 3 MExE devices are based on the Connected Limited Device Configuration (CLDC) with the Mobile Information Device Profile (MIDP).

Java 2 Micro Edition (J2ME) is a version of the Java 2 platform targeted at consumer electronics and embedded devices. CLDC consists of a virtual machine and a set of APIs suitable for providing tailored runtime environments. The J2ME CLDC is targeted at resource constrained connected devices (e.g. memory size, processor speed etc.).

9.1.1 Classmark 3 MExE devices

Support of CLDC/MIDP in a MExE classmark 3 device as detailed in this clause is mandatory.

MExE Classmark 3 devices are based on the J2ME Connected Limited Device Configuration (CLDC) with the Mobile Information Device Profile (MIDP).

All APIs defined by CLDC and MIDP shall be supported by a MExE classmark 3 device.

9.1.2 High level architecture

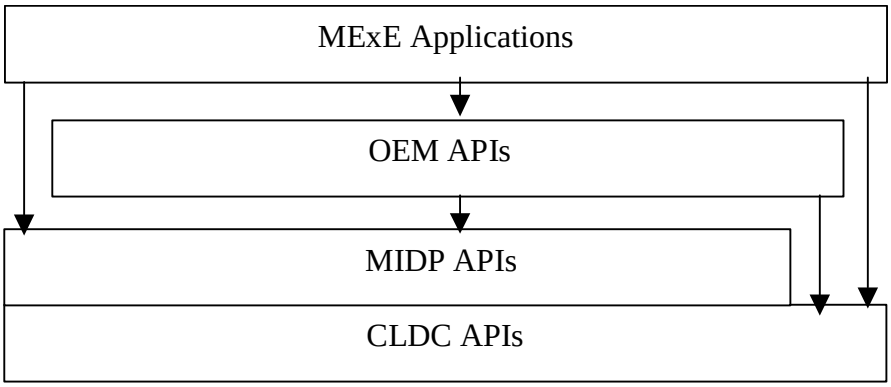


Figure 15: Functional architecture of a Classmark 3 MExE device

The functional architecture of a Classmark 3 MExE device is shown in figure 15 "Functional architecture of a Classmark 3 MExE device". The MExE API is based on the combination of CLDC APIs and MIDP APIs. OEM specific APIs are outside the scope of MExE specification. CLDC and MIDP APIs are defined in J2ME [34] and [35].

9.1.3 High level functionality

J2ME CLDC and MIDP addresses a large market of resource-constrained devices and is aimed to provide complete end-to-end solution for creating dynamically extensible networked products and applications. It allows the use of Java programming language as the standard platform for secure delivery of dynamic content for the extensible next-generation devices.

In order to fit into various types of the devices and support extensibility, J2ME defines in *Configuration* a minimum platform with a virtual machine features and minimum libraries which are available on all devices of similar class. In a *Profile* J2ME addresses the specific demand of a certain category of the devices allowing additional APIs. Profile is implemented on top of configuration (see figure 15 "Functional architecture of a Classmark 3 MExE device"). Classmark 3 MExE device shall be based on the following types of configuration and profile: Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP).

9.2 Non Security

9.2.1 Connected Limited Device Configuration (CLDC)

Classmark 3 devices shall support CLDC specification [34].

CLDC provides only high level libraries without focus on any specific device categories. Defining "the lowest common denominator" of Java technology all features included in CLDC must be generally applicable to a wide variety of the devices. CLDC does not address to a certain device category. Such features are specified in a profile. CLDC does not define any optional features.

The classes provided by CLDC are either subset of J2SE (Standard Edition) classes or CLDC specific classes which can be mapped onto J2SE. Classes belonging to the packages: Java.io, Java.lang, Java.util are a subset of corresponding Java2SE libraries, while classes specified in Javax.microedition.io are specific CLDC classes, which, however, can be mapped onto Java2SE.

Javax.microedition.io provides generic connection framework for supporting input/output and networking in a generalized and extensible manner. The framework is a functional subset of Java2SE classes which can be mapped to common low-level hardware or to any Java2SE implementation. It does not provide a set of different kinds of abstractions for different forms of communications, but rather a set of related abstractions are used at the application programming level.

The framework uses a hierarchy of Connection interfaces that group together classes of protocols with the same semantics. The actual supported protocols or implementation of the specific protocols is outside the scope of CLDC Generic Connection Framework and is maintained at the profile level.

The basic set of available Connection interfaces is the following:

- Connection;
- ContentConnection;
- Datagram;
- DatagramConnection;
- InputConnection;
- OutputConnection;
- StreamConnection;
- StreamConnectionNotifier

9.2.2 Mobile Information Device Profile (MIDP)

MExE classmark 3 devices shall support MIDP specification [35]. MIDP is based on CLDC. Some of the features of CLDC are modified or extended by MIDP [35].

9.2.2.1 Networking

While CLDC specifies only a generic Connector used for all types of connections, MIDP extends connectivity support by providing support of the subset of the HTTP protocol. HttpConnection API provides the additional functionality to set request header, parse response headers and perform HTTP specific functions. The API must support RFC 2396 [40] and RFC 2616 [41].

The MIDP does not provide support for datagrams. If a Datagram API is to be implemented, the DatagramConnection interface defined in CLDC shall be used.

9.2.2.2 MID Applications (MIDlet)

A MIDP application (or MIDlet) uses the APIs defined by the MIDP and CLDC specifications. One or more MIDlets may be packed in one JAR file. Sharing of data between MIDlets is controlled by the individual APIs (e.g. Record Management System API).

Application Management Software provides an environment in which a MIDlet is installed, started, stopped and uninstalled. Each JAR file can be accompanied by an Application Descriptor (a text file consisting of name/value pairs), which is used to manage MIDlet and is used by MIDlet for configuration specific attributes. With the help of descriptor file, verification prior to software download is done to ensure that the MIDlet is suited to the device: Java Application Manager checks if the application already exists on the device, verifies the version number (whether an update is needed or not) and reading the JAR-file-size information ensures that there is sufficient amount of memory on the device to save the file. The minimum attributes which the Application Descriptor must contain are the following:

- MIDlet-Name;
- MIDlet-Version;
- MIDlet-Vendor;
- MIDlet-Jar-URL;
- MIDlet-Jar-Size.

Mandatory and optional attributes are defined in [35]. If the mandatory attributes are not identical in the descriptor file and in the manifest file, the JAR file shall not be installed.

9.2.2.3 MIDlet Suites

MIDlets may be packaged together in a single JAR file, forming a MIDlet suite. MIDlets in a MIDlet suite share the classes in the JAR file and the persistent storage is the MIDP Record Management System.

MIDlets in a MIDlet suite may be discovered, transferred, installed and deleted together as a packaged set of MIDlets. The deletion of a MIDlet in a MIDlet suite may result in the deletion of the entire MIDlet suite, in which case the user shall be notified of the deletion of the MIDlet suite.

9.2.2.4 Record Storage

The MIDP provides a mechanism for MIDlets to persistently store data and later retrieve it. The persistent storage mechanism is called Record Management System. Record stores are created in platform-dependent locations and are not exposed to MIDlets. The record store maintains a version number, which is incremented each time the content of the record store is modified. A record store is shared between all MIDlets in a MIDlet suite.

9.2.3 Required and optional MExE APIs

Support of any other Java APIs besides CLDC and MIDP is not mandated in a Classmark 3 MExE device. A Classmark 3 MExE device may optionally support any other Java APIs which comply with the MExE security requirements.

9.2.4 Service discovery and management

A browser installed on a MExE device should support MIME type text/vnd.sun.j2me.app-descriptor. This support allows the user to browse and discover a Java application which can then be downloaded. Capability negotiation information in the request header can determine which application to present. MIDlets and MIDlet suites should be indicated to the user, and if the MExE device has a display, may be presented as an icon and a tag or as a textual tag only.

A JAD file can be downloaded and used to determine if the MIDlet is deemed suitable for download and installation. If it is suitable, the JAR file can be downloaded and installed. If not, the MExE device should be able to prompt the user so that the user might choose to take such actions such as deletion of some existing applications if there is not enough space to install the new application. If the application chosen to be installed already exists on the device, the user should be notified so that he could take further actions either to download the chosen version or to retain the existing one.

The user should be able either to launch the MIDlet immediately or later.

9.2.5 Transfer of capability negotiation information in Classmark 3

In Classmark 3 the CC/PP is carried over by using CC/PP over HTTP, [15] and optionally CC/PP over WSP, [17].

Also MIDP itself provides a simple mechanism for applications to indicate the capabilities they require. The Java Application Descriptor File (JAD), which is a file that can be stored and downloaded separately to the application itself, contains information such as application name, version number, JAR file size, data storage requirements etc. The Application Descriptor can accompany the JAR file and can be used to ensure prior to the actual application download that the application suits the MExE device. The JAD file is described in more details in the clause 9.2.2.2 " MID Applications (MIDlet)".

9.3 Security

If the 3 MExE security domains defined in clause 6.1 "Generic security" are not supported, then the Java security described in this clause is optional.

9.3.1 CLDC security

A Java execution environment running on a Classmark 3 MExE device shall comply with the security requirements defined in the CLDC specification [34]. That is, it shall comply with both the low-level virtual machine security requirements and the application-level security requirements.

The application-level CLDC security requirements define a sandbox security model where Java classfiles are verified. Java APIs available to the application are limited to those APIs which have been defined by the configuration and profiles supported by the MExE device. Downloading and management of the Java applications on the MExE device takes place at the native level, no user-definable Java class loaders are provided and the set of APIs available to a MIDlet is closed.

The low-level CLDC virtual machine security requirements define a Java classfile pre-verification mechanism which takes place off- MExE device (e.g. on the server prior to downloading) and inserts a special attribute called a "stack map" into class files to facilitate runtime verification of the same classfiles.

10 MExE classmark 4 (CLI Compact environment)

10.1 Introduction

Classmark 4 specifies CLI Compact Profile enabled devices.

The CLI [50] environment is a programming language neutral, OS and CPU portable environment. The CLI can support applications and services written in a wide range of programming languages, for example Visual Basic, ECMAScript and C#. The CLI Compact Profile specifies a minimal set of class libraries, supporting common runtime library features as well as web services infrastructure, including HTTP[9], TCP/IP, XML[36] & SOAP[51]. Such devices not only may have limited memory and CPU capability, but also limited (or no) display.

10.1.1 High level architecture

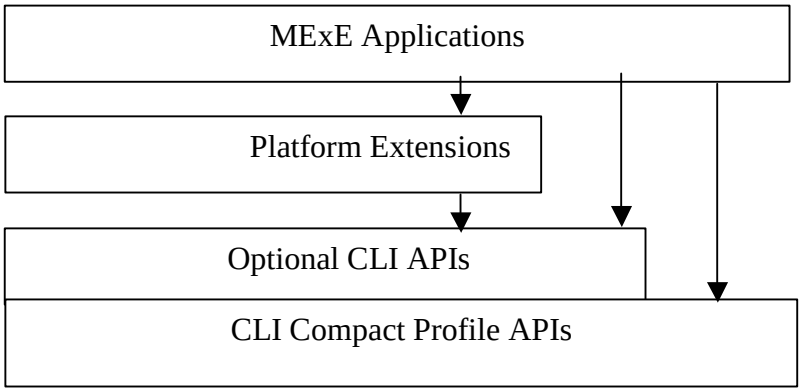


Figure 16: Basic functional architecture of a CLI MExE device

The functional architecture of a CLI MExE classmark 4 device is shown in Figure 16 "Basic functional architecture of a CLI MExE device". CLI based applications and services access functionality via the MExE CLI Compact Profile API. Additional CLI APIs and OEM specific APIs are outside the scope of the MExE specification. The CLI Compact Profile APIs are defined in CLI specified by ECMA [50].

10.2 Non security

Support of CLI Compact Profile in a MExE classmark 4 UE as detailed in this subclause is mandatory.

MExE Classmark 4 devices shall be based CLI Compact Profile specifications [50]. The specifications define the runtime environment and APIs available to a CLI based MExE device such that services (specified in the form of language independent classes and interfaces) can control such a device in a standardised way.

All mandatory components of the CLI compact profile shall be included. Additional CLI APIs or OEM APIs may be available. It shall be possible for services and applications to determine the presence of additional parts of the functionality. When an additional optional CLI component is implemented, the component shall be fully implemented.

10.2.1 High level functionality

CLI provides a language-neutral, CPU and OS portable, secure infrastructure for executing applications and services that interoperate seamlessly with highly available web services. The CLI Compact Profile provides a mobile client-focussed subset of these services on a broad market of connected devices. Using multiple programming languages for application and service creation allows adoption of a large pool of programming talent, as well as interoperability between existing service components.

Functionality is exposed to applications and services in the form of classes and interfaces. Classes can be written in any supported language. APIs can be exposed to the developer using the language syntax of choice. Classes and interfaces are collected into namespaces, which aim to represent a coherent, mutually-dependant set of functionality.

The following Table 11 "CLI Compact Profile Namespaces" specifies the defined mandatory namespaces, i.e. the namespaces defined in the CLI Compact Profile containing classes which are required for CLI Compact Profile conformance.

Table 11: CLI Compact Profile Namespaces

CLI Compact Profile Namespaces
System
System.Collections
System.Globalization
System.IO
System.Text
System.Threading
System.Runtime.CompilerServices
System.Reflection
System.Net
System.XML

The namespaces outlined in Table 11 define the core of the CLI programming model. The System namespace defines data types, including simple data types such as integers, collections such as arrays, and string data types with methods for textual manipulation. System.Globalization enables applications to adapt at runtime to user and cultural UI preferences by modifying list sorting order, currency symbol selection, date and calendar formats, input methods, and language presentation within text strings. The programming model supports threads, with thread manipulation primitives defined in System.Threading. System.Reflection enables progammatic inspection of application metadata such as class structure, properties, and data types on method parameters. System.Net includes support for transport-independent sockets, HTTP connections, and infrastructure for consuming web services. System.XML enables simple parsing and construction of XML objects.

10.2.2 Network protocol support

Support for network protocols in MExE classmark 4 devices is specified in the following Table 12 "Support for network protocols in Classmark 4 devices":

Table 12: Support for network protocols in Classmark 4 devices

Protocol	Optionality
HTTP/1.1 [9]	Mandatory
HTTPS	Mandatory
SOAP [51]	Mandatory
Gopher	Optional
ftp	Optional
mailto [25]	Optional
File	Optional

10.2.3 Power Management

MExE Classmark 4 devices have no application or service accessible APIs to detect the power level of the device. Classmark 4 applications or services may be paused by the MExE device if power passes below a certain threshold. Such an activity is implementation dependant.

10.2.4 CLI charging support

MExE CLI devices do not require any additional specific charging (e.g. charging records) or subscription management. Use of network resources from a CLI application or service is bearer independent and runs as applications on top of the bearer network.

10.2.5 CC/PP over HTTP or WSP (Classmark 4)

In Classmark 4 the CC/PP is carried over by using CC/PP over HTTP [15] and optionally CC/PP over WSP, [17].

10.3 Security

10.3.1 CLI Security

Support of CLI security in a MExE classmark 4 device, as detailed in this clause, is mandatory.

The CLI specification includes a detailed algorithm to verify the safety and integrity of CLI application code, known as "application verification" [50]. An application is either verified as "safe" or "unsafe". The result of the application verification process is presented to the rest of the CLI runtime environment before program execution. This is distinct from the certificate and signature verification that MExE performs before executing an application inside the MExE environment.

Applications that are verified as unsafe may contain code that could potentially access the environment in an unauthorized manner. For example, code that uses pointer arithmetic or accesses arrays using out-of-bounds indices would be deemed unsafe. An unsafe application shall not be permitted to execute in the MExE environment. Once an application is verified as safe, this information is passed on to the rest of the CLI runtime environment, which will then apply the MExE security policy, as specified under clause 6.1, "Generic Security".

The CLI runtime environment may be able to load libraries that are intrinsically part of the MExE device implementation. Such intrinsic libraries may contain unsafe code, however other libraries containing unsafe code are prohibited.

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

A.1 PKCS#15 certificate object attributes presentation

Details from PKCS#15[32] in this Annex A.1 “PKCS#15 certificate object attributes presentation” 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	a cross-reference back to the authentication object, which describes the properties 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-1hash, 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 [32].

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 MExE device.

`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 [43]). 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 PKCS#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 MExE device shall recognise all optional present fields above. The MExE device shall accept and ignore all unused fields or new field extensions.

A.3 Coding and storage in MExE-(U)SIM

See detail of file hierarchy and file properties in (U)SIM document [27] and [39].

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] clause 6.6 "Certificates" , 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] clause 6.6 "Certificates", 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): MExE executable life cycle

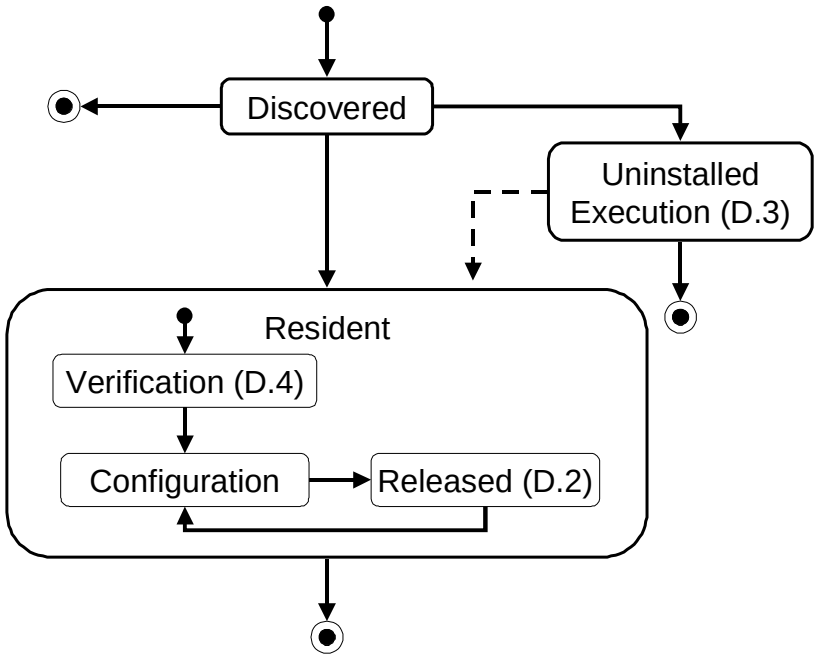
This is a conceptual description of the life cycle of a MExE executable. There may be small deviations in a specific classmark.

The Unified Modelling Language (UML) [38] is used in the description. (This is a brief description of the symbols. A rounded rectangle is a state. An arrow is a transition between states. A dot is an initial state indicating the starting state when the enclosing state is entered. A circle with a dot is a final state. When a final state is activated the enclosing state ends.)

Figures in parenthesis are references to clauses in the specification.

D.1 State of a MExE executable

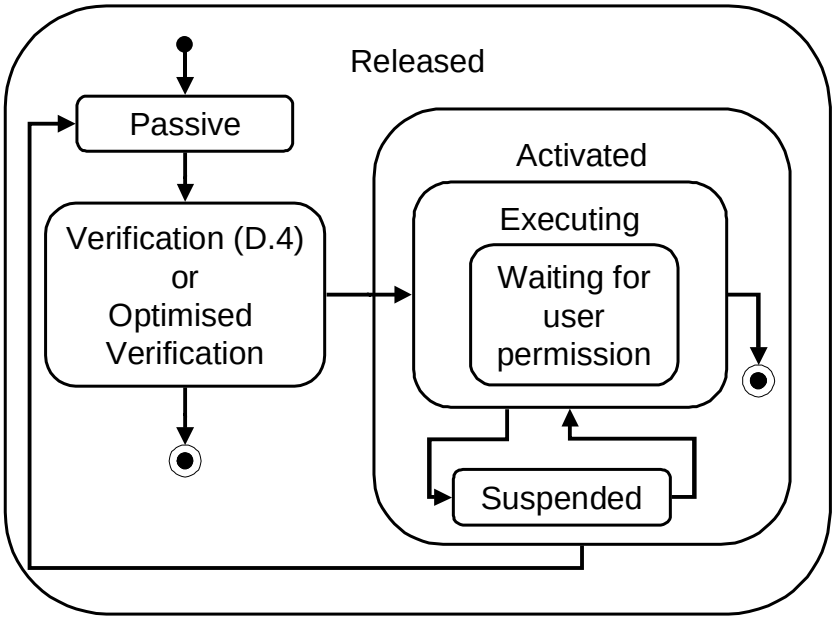
The life cycle of MExE executables (clause 5.3 "Provisioning and management of services") is described using a state machine. In a MExE device a MExE executable can have the following states and transitions between states.



State or Transition (=>)	Description
Initial => Discovered	The MExE executable is discovered (clause 5.3.1 "Service discovery").
Discovered	The MExE executable is discovered and can be installed or executed without installation. (Only executables useable on the MExE device should enter this state.)
Discovered => Resident	The discovered executable is selected to be installed and the executable is transferred (clause 5.3.2 "Service transfer") to the MExE device for installation.
Discovered => Uninstalled Execution	The discovered executable is selected to be executed without installation.
Discovered => final state	The executable is undiscovered.
Resident	The executable is stored in the MExE device. It has been transferred or is pre-loaded.
Verification	This is the initial sub-state of the Resident state. This is a composite state. There is a description of the Verification state in D.4.
Verification => Configuration	The result of the verification indicates that the executable can be installed in one of the Domains.
Configuration	This is a sub-state of the Resident state. The executable can be configured, manually or automatically (clause 5.3.3 "Service installation and configuration").
Configuration => Released	The service is released for execution.
Released	This is a sub-state of the Resident state. The executable is resident, configured and released for execution. This is a composite state and there is a description of it in D.2.
Released => Configuration	The executable is blocked for execution or an executable has changes security domain (The user shall have the possibility to review the configuration before the executable is released for execution with different privileges.).
Resident => final state	The Resident state is left when the service is deleted (clause 5.3.6 "Service deletion"). From the MExE device point of view the executable does not exist any more. (The Integrity and Certification Validation (clause 6.10 "Certificate management") can also force a deletion)
Uninstalled Execution	The executable is executed without installation. This is a composite state. There is a description of the Uninstalled Execution state in D.3.
Uninstalled Execution => final state	The Uninstalled Execution state is left when the executable terminates by itself or when the user terminates the executable (clause 5.3.5 “Service termination”). From the MExE device point of view the executable does not exist any more.
Uninstalled Execution => Resident	This is a possible but unusual transition. A MExE executable that has been used for uninstalled execution is installed without retransferring.

D.2 Released state

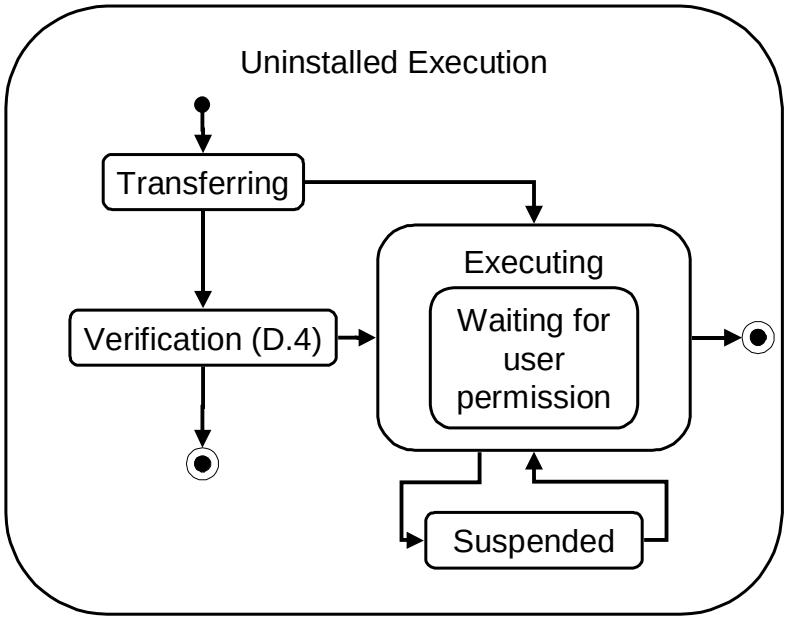
A MExE executable in the Released state is resident, configured and released for execution.



State or Transition (=>)	Description
Passive	This is the initial state. The executable can be invoked.
Passive => Verification	The MExE executable is invoked.
Verification (or Optimised Verification)	The verification can either be done according to the Verification state described in D.4 or as an Optimised Verification described in (clause 6.2.2 "Optimised application signature verification").
Verification => Executing	The result of the verification indicates that the executable may be executed. The Activated state and its sub-state Executing are entered.
Verification => final state	The MExE executable has changed its security state and it must not be executed.
Activated	The MExE executable is activated.
Executing	This is a sub-state of Activated. The executable executes (if it is not waiting for user permission).
Executing => final state	The Executing state is left when the executable terminates by itself. The Activated state is left and the Passive state is entered.
Waiting for user permission	This is a sub-state to Executing. (If there is support for multi-threaded applications, this state can be a state concurrent to the Executing state.) The MExE executable is waiting for permission to perform some action (clause 6.3.1 "MExE executable permissions for operator, manufacturer and third party security domains").
Suspended	This is a sub-state of Activated. The execution is suspended (clause 5.3 "Provisioning and management of services").
Activated => Passive	The Activated state is left when the executable terminates by itself or when the user terminates the executable (clause 5.3.5 "Service termination").

D.3 Uninstalled Execution state

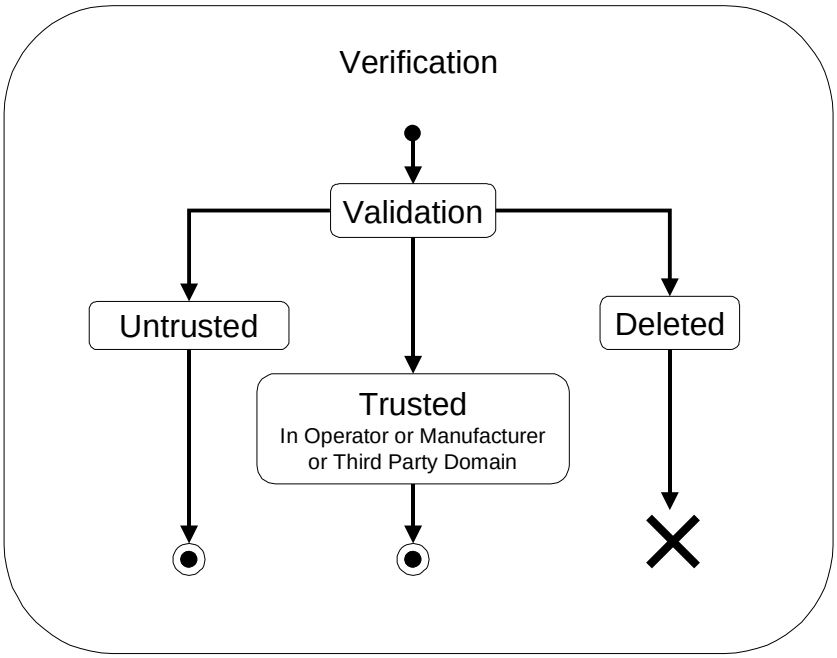
In the Uninstalled Execution state a MExE executable is executed without installation.



State or Transition (=>)	Description
Transferring	This is the initial state of Uninstalled Execution. The MExE executable is transferred to the MExE device.
Transferring => Verification	If the executable is signed the Verification state is entered after the transferring is finished.
Transferring => Executing	If the executable is not signed the Executing state is entered. (To allow streaming, this can be done before the transfer is finished.)
Verification	This is a composite state. There is a description of the Verification state in D.4.
Verification => Executing	The result of verification indicates that the executable may be executed.
Verification => final state	The result of verification indicates that the executable must not be executed and the Uninstalled Execution state is ended.
Executing	The MExE executable is executing.
Executing => final state	The Executing state is left when the execution terminates by itself (The Uninstalled Execution state is left.)
Waiting for user permission	This is a sub-state to Executing. (If there is support for multi-threaded applications, this state can be a state concurrent to the Executing state.) The MExE executable is waiting for permission to perform some action (clause 6.3.1 "MExE executable permissions for operator, manufacturer and third party security domains").
Suspended	The execution is suspended (clause 5.3 "Provisioning and management of services")

D.4 Verification

The integrity and certification validation (clause 6.10 "Certificate management") is done in the Verification state. The result of validation determines the change of state.



State	Description
Validation	This is the initial state. The integrity and certification validation (clause 6.10 "Certificate management") is done.
Untrusted	The executable is untrusted (clause 6.1 "Generic security")
Trusted in Operator Domain	The executable is verified to belong to the Operator Domain (clause 6.1 "Generic security").
Trusted in Manufacturer Domain	The executable is verified to belong to the Manufacturer Domain (clause 6.1 "Generic security").
Trusted in Third Party Domain	The executable is verified to belong to the Third Party Domain (clause 6.1 "Generic security").
Deleted	The executable is not permitted in any Domain and it is deleted.

Annex E (informative): MExE conformance requirements

The table of Conformance Requirements define the minimum set of features that a conformant MExE device must implement.

Legend:-

M - Mandatory feature/requirement

O - Optional feature

N/A - Feature is not applicable: the MExE specification does not prevent this feature from being implemented; however, support of this feature is not required for a MExE device to be regarded as being compliant with a specific MExE Classmark device. Therefore the specification does not indicate whether the feature is optional or mandatory.

M/O – Support as such is required. Mandatory and Optional features are gathered into a table

ID	Requirement	Reference	CM1	CM2	CM3
4-1	Support of at least one MExE classmark on a MExE device	4	M	M	M
4-2	Support of multiple combinations of MExE classmarks	10.1	O	O	O
4-3	Support of WAP	4.2.5	M	O	O
4-4	If Classmark 1 services are supported by non-Classmark 1 devices, Classmark 1 services shall execute in the same manner as they execute in a MExE Classmark 1 device	4.5.1	N/A	M	M
4-5	Support of PersonalJava	8.1, 8.1.1	O	M	O
4-6	If Classmark 2 services are supported by non-Classmark 2 devices, Classmark 2 services shall execute in the same manner as they execute in a MExE Classmark 2 device.	4.5.2	M	N/A	M
4-7	Support of CLDC and MIDP	9.1, 9.1.1	O	O	M
4-8	If Classmark 3 services are supported by non-Classmark 3 devices, Classmark 3 services shall execute in the same manner as they execute in a MExE Classmark 3 device.	4.5.3	M	M	N/A
4-9	Support of capability negotiation process	4.2	M	M	M
4-10	Support for interaction between the MExE device and the MSE by the use of HTTP/1.1or HTTP/1.1 derived protocol (e.g. WSP)	4.2	M	M	M
4-11	Support of the properties in the UAProf schema for capability negotiation	5.2.1	M/O	M/O	M/O
4-12	Support of content negotiation	4.2	O	O	O
4-13	Support of user profiles	5.2	O	O	O
4-14	Support of more than one user profile (if user profiles supported)	5.2.2	O	O	O
4-15	Ability to retain the user profile in the network (if user profiles supported)	5.2.2	O	O	O
4-16	User permission for retaining the user profile in the network (if user profiles supported)	5.2.2	M	M	M
4-17	Support of direct and indirect referencing mechanisms in retrieval of MExE preferences (if user profiles supported)	8.2.2	O	O	O
4-18	Support of the properties in the UAProf schema for user preference information (if user profiles supported)	8.2.2	M	M	M
4-19	Support of user interface personalisation	5.1	O	O	O
4-20	Support of direct and indirect referencing mechanisms in retrieval of user interface personalisation preferences	5.1.1	O	O	O
4-21	Ability to support VHE	9.2.5	O	O	O

ID	Requirement	Reference	CM1	CM2	CM3
4-22	Storage of the VHE characteristics as a part of the user profile (if VHE and user profile is supported)	9.2.5	M	M	M
4-23	Capability to discover new services	5.1.3.1	M	M	M
4-24	Support for a browser for service discovery	5.1.3.1	O	O	O
4-25	Ability to control service installation and configuration	4.10.3	N/A	M	M
4-26	Ability to determine which services are transferred to, resident, configured or executing on the MExE device (provide the name and, if available, version number)	4.10.4	M	M	M
4-27	Service termination capability	4.10.5	M	M	M
4-28	Capability to delete a service	4.10.6	M	M	M
4-29	User's ability to terminate or suspend any active connection associated with any MExE executable	5.3	M	M	M
4-30	User's ability to obtain information on all connections associated with any MExE executable on the MExE device	5.3	M	M	M
4-31	Support of journaling of network events by MExE executables	5.4	M	M	M
4-32	Management of the journal by the MExE device, with no access to it by MExE executables	5.4	M	M	M
4-33	Indicate whenever network activity is in progress	5.4.1	O	O	O
4-34	Support of QoS management by MExE	5.4.2, 9	O	O	O
4-35	Support of core software download functionality	5.5	O	O	O
4-36	Core software download (if supported) only under control of the MExE device manufacturer	5.5	M	M	M
5-1	Call control using WTA scripts	7.3.1	M	O	O
6-1	Support of the Wireless Profile of the JavaPhone API specification (Optional Java packages of Wireless Profile of the JavaPhone APIs as presented in Table 94 "Optional Java packages of the Wireless Profile of the JavaPhone APIs")	8.2.1.1, 8.2.1.3	O	M/O	O
6-2	Support of the JAR file manifest entries as per JavaPhone specification	8.2.1.3.1	O	M	O
6-3	The use of icons to launch applications	8.2.1.3.1	O	O	O
6-4	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	8.2.1.3.1	O	M	O
6-5	Implementation of "BatteryCritical", "BatteryNormal" event generation	8.2.1.3.2	O	M	O
6-6	Support for the following formats in Datagram recipient addressing: raw text-only GSM SMS message, UDP datagram via IP, and WAP datagram via GSM SMS message(s)	8.2.1.3.3	O	M	O
6-7	Support any other Java APIs which comply with the MExE security requirements in Table 56 "Security domains and actions"	8.2.1.4	O	O	O
6-8	Support for network protocols as per table 105 "Support for network protocols"	8.2.1.5.1	O	M/O	O
6-9	Support of MIDlet discovery and management via a browser using MIME type text/vnd.sun.j2me.app-descriptor	9.2.4	O	O	O
6-10	Indication of MIDlets and MIDlet suites to the user (with a tag or icon and tag)	9.2.4	O	O	O
7-1	Support of charging regimes of MExE services (charging mechanisms are outside the scope of MExE specification).	5.6.1	O	O	O
8-1	Support of the untrusted area	6.1	M	M	M
8-2	Support of all three security domains together (i.e. operator, manufacturer and third party), or no security domains at all	6.1	M	M	M
8-3	Security restrictions shall apply to MExE executables when API functionality is directly or indirectly called by MExE executables	6.3	M	M	M
8-4	Support for permissions of operator, manufacturer and third party security domains in the order of restriction (as defined in table 56 of MExE specification).	6.3.1	M	M	M
8-5	Access by MExE untrusted executables limited to the functionality specified in the table 67 of MExE specification	6.3.2	M	M	M
8-6	Separation of the user interface input and output streams between different MExE executables (except for the MIDlets in the same MIDlet suite)	6.11	M	M	M
8-7	Support of single action permission with a prompt for the user	6.5	M	M	M
8-8	Support of session permission and blanket permission with a prompt for the user	6.5	O	O	O
8-9	Indication to the user whenever user permission is sought by an untrusted MExE executable	6.5	M	M	M
8-10	Ability of the user to request to be informed of the "subject" field of the	6.5	M	M	M

ID	Requirement	Reference	CM1	CM2	CM3
	certificate of the signer (if secure domains supported)				
8-11	Support for public key based solution of content authentication (if secure domains supported)	6.7	M	M	M
8-12	Support of certificate chains (if secure domains supported)	6.7	M	M	M
8-13	Support at least one level of certificate under operator, manufacturer or Third Party root public keys (if secure domains supported)	6.7	M	M	M
8-14	Secure installation of root public keys in the MExE device (if secure domains supported)	6.7.1	M	M	M
8-15	Prohibition to share public keys between domains (if secure domains supported)	6.7.1	M	M	M
8-16	Support the use and management of an operator root public key on the (U)SIM (if secure domains supported)	6.6.1	M	M	M
8-17	Prohibition of the user to add or delete any type of operator public keys (if secure domains supported)	6.6.1	M	M	M
8-18	Support of operator and manufacturer disaster recovery root public keys (if secure domains supported)	8.5.	O	O	O
8-19	Support of the use and management of the operator root public key (if secure domains supported)	8.5.1.	M	M	M
8-20	Support of the use and management of the manufacturer root public key (if secure domains supported)	6.6.2	M	M	M
8-21	Support of the use and management of the third party root public keys (if secure domains supported)	6.6.3	M	M	M
8-22	Support of the use and management of the administrator root public key (if secure domains supported)	6.13.1	M	M	M
8-23	Support of the administrator designation mechanism (if secure domains supported)	6.13.1	M	M	M
8-24	Support of the certificate configuration management (if secure domains supported)	6.10	M	M	M
8-25	Use of the CCM by MExE device to determine the third party certificates that are trusted for the use on the MExE device (if secure domains supported)	6.10.1	M	M	M
8-26	Additional support of other means to enable/disable root certificates (if secure domains supported)	6.10.1	O	O	O
8-27	Support of authorised CCM download mechanisms (if secure domains supported)	8.7.	M	M	M
8-28	When the administrator is changed, then the CCM shall also be changed. (if secure domains supported)	8.7.	M	M	M
8-29	Support of provisioned mechanism for designating administrative responsibilities and adding third parties in a MExE device (if secure domains supported)	6.13.2	M	M	M
8-30	Support of the cases: the user is the owner, the user is at remote location, the owner of the MExE-(U)SIM wants to be a temporary administrator (if secure domains supported)	6.13.2	M	M	M
8-31	Support for determining the administrator of the MExE device (if secure domains supported)	6.13.3	M	M	M
8-32	Either sandbox or fine grain Java security shall be supported	8.3.1	N/A	M	N/A
8-33	Support for trusted applets (if secure domains supported)	8.3.1	N/A	O	O
8-34	Verification of the certification of the application or applet (if secure domains supported)	8.3.1.2	M	M	M
8-35	Java loading native libraries that are intrinsically part of the MExE device implementation, and MExE native libraries	8.3.1.3	O	O	O
8-36	No loading of other native libraries	8.3.1.3	N/A	M	N/A
8-37	Support of the JAR file format devices for securely packaging objects that are to be downloaded and installed on the MExE device	10.3.1	N/A	M	M
8-38	Support for other proprietary means of downloading and installing objects	10.3.1	O	O	O
8-39	Support of MExE native library signed package installation	8.3.2	N/A	O	O
8-40	Support for the case 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 (if secure domains supported).	6.8.2	N/A	M	M
8-41	Support of installation of other signed data (e.g. proprietary binaries or Java classes such as native DSP code, provisioned functionality upgrades and patches) (if secure domains supported).	6.8.3	O	O	O
8-42	Support for administrator root certificate mechanism (if secure domains supported).	6.13.4	M	M	M

ID	Requirement	Reference	CM1	CM2	CM3
8-43	Support of alternative methods to download an administrator root certificate (if secure domains supported).	6.13.4	O	O	O
8-44	Support of optimised signature verification of applications (if secure domains supported).	6.8.1	O	O	O
9-1	Support of QoS API by MExE device	9	O	O	O
9-2	Support of a basic QoS operations	5.5.2	O	O	O
9-3	Support of MExE QoS API by MExE QoS Manager	5.5.3	O	O	O
9-4	Provision of the MExE QoS Manager functions	5.5.3	O	O	O
9-5	Ability to manage QoS through the MExE device's MMI	5.5.3	O	O	O
9-6	QoS control by MExE QoS Manager, if it is not provided in the network control	5.5.4	O	O	O
9-7	Provision of a standard set of parameters by a QoS API to MExE executable	5.5.5	O	O	O
9-8	Ability of MExE QoS Manager to deal independently with each of the several simultaneous QoS streams	5.5.7	O	O	O