EFI Manage Application Class

Version 1-Nov-2001

Wireless Application Protocol
WAP-267-EFIMAC-20011101-a

Document History	
WAP-267-EFIMAC-20011101-p	Approved
WAP-267-EFIMAC-20011101-p	Proposed
WAP-267-EFIMAC-20011101-d	Draft

Contents

1. Scope

The Wireless Application Protocol (WAP) is a result of continuous work to define an industry-wide specification for developing applications that operate over wireless communication networks. The scope for the WAP ForumTM is to define a set of specifications to be used by service applications. The wireless market is growing very quickly, and reaching new customers and services. To enable operators and manufacturers to meet the challenges in advanced services, differentiation and fast/flexible service creation the WAP Forum defines a set of protocols for the transport, security, transaction, session and application layers. For additional information on the WAP architecture, please refer to "Wireless Application Protocol Architecture Specification" [WAPARCH].

Current trends in telecommunications enable new kinds of functionality in a wireless terminal; either through the integration of new features into the terminal or by allowing new types of devices to be connected to the terminal. Supporting this development in WAP standards will strengthen WAP's position as a platform for advanced wireless data services by providing access to new capabilities.

External Functionality (EF) is a general term for components or entities with embedded applications that execute outside of the Wireless Application Environment (WAE) or other user agent, and conform to the EFI requirements. The External Functionality can be built-in or connected to a WAP terminal. This connection can be permanent or temporary.

An application environment of WAP is the place within the terminal where applications are executed, either in the form of WML pages or in the form of scripts or both. The most convenient way to facilitate the connection between the application and new functionality of the terminal is to specify new standard services that can be accessed by an application that is being executed in WAP application environment. EFI supports the notion of classes, conceptual groups of functions that pertain to the same application areas.

The External Functionality Interface (EFI) specifications in WAP provide methods enabling applications to access External Functionality in a uniform way through the EFI Application Interface (EFI AI). The EFI specifications consists of the Framework, the Process specification and a set of Class Specifications, each one specific to the given application area.

EFI Framework defines the general behaviour of EFI implementation in the WAP terminal while detailed requirements for the class are provided in individual Class Specification documents. The Process specification facilitates the development of Class Specifications by defining steps that should be taken in order to achieve the quality Class Specification.

The EFI Application Interface (EFI AI) is a high level interface that shall suit a number of different applications. Various external functions are grouped in classes that offer common functionality across different makes and versions of terminals and external functionality entities. The EFI Framework provides an extensible set of interfaces that can support services, including the ability to query for the particular service as well as the ability to capture the functionality that is specific to the given device or software installed. However, there is no functionality to dynamically add new services so there is no general service discovery mechanism.

This document provides a class for the management of applications running in application environments outside of the Wireless Application Environment [WAE] and Wireless Telephony Application [WTA]. It is designed to allow WAP applications executing within WAE to start, stop and control external applications and to discover what applications are available to it.

The EFI Manage Application Class is used to manage general applications. It is not designed to replace the access to specific classes of functionality offered through other EFI classes. The class is restricted to managing external applications, that is, applications running outside of WAE or WTA.

This document also provides instructions for implementing the class for the GSM SIM Application toolkit as defined in GSM 11.14 [GSM1114] and 3G TS 31.111 [3G31.111].

2. References

2.1. Normative References

[RFC2119]	“Key words for use in RFCs to Indicate Requirement Levels”. S. Bradner. March 1997. http://www.ietf.org/rfc/rfc2119.txt
[CREQ]	"WAP Conformance Requirement Specification", WAP Forum, WAP-221-CREQ. http://www.wapforum.org/
[3G31.101]	3rd Generation Partnership Project; Technical Specification Group Terminals; UICC-Terminal Interface; Physical and Logical Characteristics (3G TS 31.101 version 3.2.0 Release 1999) http://www.3gpp.org/
[3G31.102]	3rd Generation Partnership Project; Technical Specification Group Terminals; Characteristics of the USIM Application (3G TS 31.102 version 3.2.0 Release 1999) http://www.3gpp.org/
[3G31.111]	3rd Generation Partnership Project; Technical Specification Group Terminals; USIM Application Toolkit (USAT) (3G TS 31.111 version 3.1.0 Release 1999) http://www.3gpp.org/
[GSM11.11]	Digital cellular telecommunications system (Phase 2+) (GSM); Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (GSM 11.11 version 5.3.0 Release 1996) http://www.etsi.org/
[GSM11.14]	Digital cellular telecommunications system (Phase 2+) (GSM); Specification of the SIM application toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (GSM 11.14 version 5.9.0 Release 1996) http://www.etsi.org/
[EFI]	“EFI Framework”, WAP Forum™, WAP-231-EFI. http://www.wapforum.org/

2.2. Informative References

[WAE]	"Wireless Application Environment Specification", WAP Forum™, WAP-236-WAESpec. http://www.wapforum.org/
[WAPARCH]	“WAP Architecture”, WAP Forum™, WAP-210-WAPArch. http://www.wapforum.org/
[WTA]	"WAP Wireless Telephony Application", WAP Forum™, WAP-266-WTA. http://www.wapforum.org/

3. Terminology and Conventions

3.1. Conventions

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

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

3.2. Definitions

Broker	The conceptual entity that exists between the EF Units, EF Class Agents and the EFI AI. The EF Broker maintains the list of available functionality and routes requests to the correct EF Unit or EF Class Agent or handles them itself.
Class	The collection of all EF Units and EF Class Agents that share the same functionality according to the same Class Specification.
Class Agent	The conceptual active element that provides added functionality on the basis of EF Units of the same EF Class Realisation.
Class Realisation	The collection of EF Units and optionally the EF Class Agent that belong to the same EF Class and are available to a particular Terminal.
Class Specification	The definition of services that are provided by every EF Unit that belongs to the given class and services provided by the EF Class Agent.
Entity	The conceptual component that expresses the EFI view on a software or hardware component of the mobile terminal that exposes some of its function for the purpose of EFI.
Implementation	The software and hardware that is used in the particular terminal to implement the functionality.
Mobile Equipment	see ‘WAP Terminal’
Native Application	An application running in an application environment outside of the Wireless Application Environment [WAE] and Wireless Telephony Application [WTA].
Origin Server	The server on which a given resource resides or is to be created. Often referred to as a web server or an HTTP server.
Registry	The conceptual place where information about available EF Units and EF Class Agents is stored and then made accessible by the EF Broker.
SAT application	A native application running in the SAT application environment.
Server	Any of the components of the EFI conceptual architecture that can be addressed to provide the service for an application; a collective name for the EF Broker, EF Units and EF Class Agents.
Service	The specified functionality provided by one of the servers: EF Broker, EF Class Agent or EF Unit.
Terminal	see 'WAP Terminal'
Unit	The conceptual component that resides in or outside the WAP terminal and provides access to the EF Services on the EF Entities.
WAP Application	The executable or interpretable code that is running within the wireless application environment [WAE]; a WAP application may use various APIs to access EFI services.
WAP Terminal	The physical unit where the WAE executes.

3.3. Abbreviations

AI	Application Interface.
API	Application Programming Interface
EF	External Functionality
EFI	External Functionality Interface
EFE	External Functionality Entity
ME	Mobile Equipment, the WAP Terminal
PDA	Personal Digital Assistant
PC	Personal Computer
SAT	SIM Application Toolkit
SIM	Subscriber Identity Module
USAT	USIM Application Toolkit
USIM	Universal Subscriber Identity Module
WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WINA	WAP Interim Naming Authority
WML	Wireless Markup Language
WTA	Wireless Telephony Application

4. Introduction

There may be multiple execution environments associated with a WAP terminal in which WAP applications may run. The WAP specifications define the Wireless Application Environment [WAE]. Others include native applications resident on the WAP terminal, for example a PDA, native applications running on a SIM card [GSM1111, 3G31.101, 3G31.102], or native applications running on a device external to the WAP terminal such as a PC that is attached to the WAP terminal. This class provides services that may be used by a WAP application running in the WAE to interact with native applications running in other execution environments.

The class is designed to allow a WAP application to query an execution environment to determine the native applications available to it to manage. The WAP application may then start, stop, suspend, resume and exchange data with the native application it is managing. Note that the full extent of management capabilities available to the WAP application depend on the particular native application and the application environment in which it is running. For exa mple, not all native applications will offer the ability to suspend and resume.

4.1. Relation to EFI

This class is designed to treat each of the application environments as a unit. The native applications available within these application environments are managed using the services offered by each of the units.

The name of the application environment to which a unit corresponds may be determined from the attributes of the unit, see section 7.1 below.

There are no specified class agent services for this class.

5. Security

When external applications are invoked, the class realization MUST make sure that the user has authorized execution of the native application. Class realizations MAY rely on the security mechanisms provided by the external application environment to accomplish this. If so, these security mechanisms must be clearly stated in section 9, “Application Environments” for said application environment.

6. Versions

The general versioning scheme for EFI class definitions is described in the EFI Framework specification [EFI]. This class specification defines the version number that MUST be reported by all EF Units that implement this class specification. The following table maps the specification identifier to the version number that is reported by all EF Units that are conformant to that version of the specification.

Specification Identifier	Version to be Reported
WAP-267-EFIMAC-20011101-a	1.0

7. Naming

7.1. Class Name

The name of this class is `ManageApp`. Class realizations conforming to the manage application class MUST use this class name. This is the name that is used by application developers to query the available units that represent the external application environments.

7.2. Environment Name Attribute

The attribute `envName` MUST be defined for units belonging to the manage application class. This attribute provides the standard name for an application environment, where such a standard name exists. Standard names MUST start with underscore.

The standard name for an application environment is defined in the implementation specific to that environment. For example, for the SIM application toolkit environment, the attribute `envName` is “ `_SAT`”.

If a standard environment name for the class has not been specified, then the `envName` SHOULD NOT start with an underscore to distinguish it from standard environment names.

7.3. Other

The implementation notes for an environment may define further attributes for a particular environment.

8. Description of Services

8.1. Application Discovery

SERVICE NAME:	*Outcontainer = getInfo*
STATUS:	MANDATORY
DESCRIPTION:	This service is used to determine what native applications are available for control by the WAP device in a particular environment.
PARAMETERS:	None: The service is addressed to the specified unit.
RETURN VALUE:	*outcontainer consists of name/value pairs* *<app_name>=<app_desc>*

EXAMPLE:

```
units = EFI.getUnits( “ManageApp” );
firstUnit = String.getElementAt( units, 0, “&” );
apps = EFI.call(firstUnit+”/getInfo”);
// use Container functions to iterate through ‘apps’
// and display the app name and description to the user
```

For each native application known in the specified environment the service returns an application name (used to identify the application in other services) assigned by the class, and an application description. The application name MUST be unique within an environment, but need not be unique across all environments. The application name MUST also be a valid service name as defined in [EFI]. How the application name is assigned is implementation specific. The <app_desc> is a string that may contain a description of the native application, and is an optional (that is, it may be an empty string).

8.2. Start Application

Each native application that is returned for the environment is treated as a service of the unit representing the environment. The service name is identical to the <app_name>. Parameters may be passed to the native application in the invocation. Result values are returned in the output container. The names of the parameters and return values are specific to the native application being run.

Note that native applications may be started with either the *efi.invoke* function, or the *efi.call* function.

8.3. Control Application

Native applications are controlled by a WAP application using the *efi.control* function. In addition to the standard actions defined in [EFI], the following actions may be used to control native applications.

8.3.1. Suspend Application

The suspend control is used to place a native application in a suspended state. This application could later be resumed. Not all native applications may make the suspend operation available.

A WAP application MAY suspend a running native application by using the *EFI.control* function with the action *suspend*.

The action *suspend* is a non-standard action, and has the value 101.

If the native application returns data upon being suspended, this is returned in the out container from *EFI.control*, otherwise the empty container is returned.

If the native application cannot be suspended for any reason, the function MUST return *Invalid*.

8.3.2. Resume Application

The resume control is used to resume a native application that has previously been suspended.

The native application is identified by the instance returned when the application was started. A native application is resumed using the *EFI.control* function with the action *resume*.

The action *resume* is a non-standard action, and has the value 102.

Data may optionally be passed to the native application on resumption in the *in* container.

If the native application returns data on being resumed, this data is returned in the *out* container. The format of this data is application dependent.

If the native application cannot be resumed for any reason, the function MUST return *Invalid*.

8.3.3. Additional Control Types

Particular application environments may offer additional control modes to WAE in order to control a native application. These MUST be defined in the implementation notes of an environment. If an environment is passed a control action code that it does not understand, it MUST return *Invalid*.

9. Application Environments

9.1. SIM Application Toolkit

This section describes how the SIM Application Toolkit (SAT) unit behaves when managing SIM application toolkit (SAT) applications. Note that SAT is specific to GSM WAP devices. The SAT is described in [GSM11.14] and the SIM is defined in [GSM11.11]. The equivalent specifications for 3G are [3G31.101] and [3G31.102] for the USIM, and [3G31.111] for the USAT.

The class will be capable of managing the SAT applications declared to the ME by the SIM in the `SET UP MENU` command.

9.1.1. Security

The class will ensure that a SAT application is not started without the user's confirmation, as defined in section 5. The SAT application environment does not provide additional security mechanisms.

9.1.2. Environment Name

The SIM Application Toolkit environment is named "`_SAT`".

9.1.3. Application Discovery

The response to a *getInfo* MUST contain the information provided to the Mobile Equipment (ME) by the SIM in the `SET UP MENU` command. For each of the available SAT applications, the command provides an `Identifier of item` field and a `Text string of item` field. The application name, <app_name>, assigned by the SAT unit MUST be the `Identifier of item` from the `SET UP MENU` command and the associated description, <app_desc>, MUST be the respective `Text string of item`

9.1.4. Start Application

When starting a service, the SAT unit MUST initiate a SAT application by issuing an `ENVELOPE` command to the SIM containing a `MENU SELECTION`. Before issuing the `ENVELOPE` command, the ME MUST first display the associated `Text string of item` field to the user and request confirmation to start the SAT application.

The `Item identifier` field of the `ENVELOPE (MENU SELECTION)` MUST be set to `Identifier of Item` field associated with the <app_name> of the service.

If the SIM indicates that `Identifier of Item` is not recognised then the error code –404 MUST be returned. This is indicated by the SIM with the status word values of 9E XX or 6F XX

If EFI.invoke is used to start the service, the service MUST complete after the SIM sends the status bytes in response to the `ENVELOPE` command.

If EFI.call is used to start the service, the service MUST complete at the first instance after sending the `ENVELOPE` command in which the status bytes returned by the SIM are not equal to 91 XX, that is, when there are no proactive commands waiting.

If any application parameters are specified, these MUST be ignored.

If the response to the ENVELOPE command is anything other than 90 00, 91 XX, 9E XX or 6F XX then the error code –500 MUST be returned.

9.1.5. Control Application

9.1.5.1. Stop

Stopping the application is not available for a SAT application, and *Invalid* MUST be returned.

9.1.5.2. Suspend

Suspending an application is not available for a SAT application, and *Invalid* MUST be returned.

9.1.5.3. Resume

As suspending an application is not available for a SAT application, neither is resumption. If an attempt is made to resume a SAT application, then *Invalid* MUST be returned.

Appendix A. Static Conformance Requirements

The notation used in this appendix is specified in [CREQ]. The acronym “MAC” is used to refer to the Manage Application Class in the static conformance requirements.

A.1. Services

Item	Function	Reference	Page	Status	Requirements
MACLIB-C-1	getInfo()	8.1	10	M	
MACLIB-C-2	Use EFI.control to suspend native applications (value 101). If the application cannot be suspended for any reason, the function returns *Invalid*.	8.3.1	10	M	
MACLIB-C-3	Use EFI.control to resume native applications (value 102) If the application cannot be resumed for any reason, the function returns *Invalid*.	8.3.2	11	M	
MACLIB-C-4	Return *Invalid* for unsupported control values	8.3.3	11	M	
MACLIB-C-5	Class realization makes sure that the user has authorized execution of the native application.	5	9	M	

A.2. Naming

Item	Function	Reference	Page	Status	Requirements
MACNAME-C-1	Use ManageApp class name	7.1	9	M	
MACNAME-C-2	Use `envName` attribute	7.1	9	M	

A.3. SIM Application Environment

Item	Function	Reference	Page	Status	Requirements
MACSIM-C-1	SIM Application Toolkit	7.1	9	O	MACSIM-C-2 AND MACSIM-C-3 AND MACSIM-C-4 AND MACSIM-C-5
MACSIM-C-2	Use `SAT` environment name	9.1.2	12	O	
MACSIM-C-3	Get list of SAT applications	9.1.3	12	O	
MACSIM-C-4	Start SAT applications	9.1.4	12	O	
MACSIM-C-5	Request user confirmation to start SAT application	9.1.4	12	O	

Appendix B. Change History (Informative)

Type of Change	Date	Section	Description
Class 0	11-September 2001		The initial version of this document.