Technical Specification Group Services and System Aspects *TSGS#8(00)0452*
Meeting #9, Kapolei, Hawaii, USA, 25-28 September 2000

Source: **TSG SA WG2**
Title: **CRs on 23.127 v.3.1.0**
Agenda Item: **7.2.3**

The following Change Requests (CRs) have been approved by TSG SA WG2 and are requested to be approved by TSG SA plenary #9.
Note: the source of all these CRs is now S2, even if the name of the originating company(ies) is still reflected on the cover page of all the attached CRs.

CRs on 23.127 v.3.1.0

Spec	Rel	CR #	Cat	Title	S2 tdoc #
23.127	R99	012	F	CR on Parlay-OSA alignment: basic service interface	S2-001624
23.127	R99	013	F	CR on Parlay-OSA alignment: initial contact interfaces	S2-001625
23.127	R99	014	F	CR on Parlay-OSA alignment : access SCF	S2-001626
23.127	R99	015	F	CR on Parlay-OSA alignment: load manager SCF	S2-001627
23.127	R99	016	F	CR on Parlay-OSA alignment: fault manager SCF	S2-001628
23.127	R99	017	F	CR on Parlay-OSA alignment: service factory SCF	S2-001629
23.127	R99	018	F	CR on Parlay-OSA alignment: authentication interface	S2-001630

Spec	Rel	CR #	Cat	Title	S2 tdoc #
23.127	R00	011r1	D	Change of TS 23.127 title for version 4.0 and up	S2-001634

3GPP Meeting S2#14
Bristol, UK, 4-8 September 2000

Document S2-001634
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

23.127	**CR**	**11R1**	Current Version:	3.1.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ ↑ *CR number as allocated by MCC support team*

For submission to:	SA#9	for approval	**X**	strategic	□	*(for SMG use only)*
list expected approval meeting # here ↑		for information	□	non-strategic	□	

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from:* ftp://ftp.3gpp.org/Information/CR-Form-v2.doc

Proposed change affects: (U)SIM □ ME □ UTRAN / Radio □ Core Network ☒

(at least one should be marked with an X)

Source:	Ericsson	**Date:**	04.09.2000
Subject:	Change of TS 23.127 title for version 4.0 and up		
Work item:	VHE/OSA		

Category:	F	Correction	□	**Release:**	Phase 2	□
	A	Corresponds to a correction in an earlier release	□		Release 96	□
(only one category	B	Addition of feature	□		Release 97	□
shall be marked	C	Functional modification of feature	□		Release 98	□
with an X)	D	Editorial modification	☒		Release 99	□
					Release 00	☒

Reason for change:

The distribution of tasks for post R99 standardisation makes that a large part of the OSA-related standardisation that was performed by TSG SA WG2 in TS 23.127 will now be addressed by TSG CN WG5 and in other Technical Specifications.

On the other hand, TSG SA WG2 will now concentrate on the Virtual Home Environment (VHE) concept, for which OSA is only a possible toolkit (others being MExE, CAMEL, and USAT).

Consequently, the currently envisaged structuring for TS 23.127 4.x consists of two main sub-clauses: one dedicated to VHE as a whole, and another dedicated to the support of VHE by the above mentioned toolkits.

In order to reflect the new focus of TS 23.127 for post-R99 standardisation, it is therefore proposed to remove “Open Service Architecture” from the title.

Clauses affected:

Other specs affected:	Other 3G core specifications	□	→ List of CRs:
	Other GSM core specifications	□	→ List of CRs:
	MS test specifications	□	→ List of CRs:
	BSS test specifications	□	→ List of CRs:
	O&M specifications	□	→ List of CRs:

Other comments:

help.doc

<---------- double-click here for help and instructions on how to create a CR.

3G TS 23.127 V4.0.0~~3.1.0~~ (2000-09~~6~~)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Virtual Home Environment~~ / Open Service Architecture~~
(Release 2000~~1999~~)

Keywords

VHE, OSA, MExE, CAMEL, USAT

3GPP

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

3GPP Meeting S2#14
Bristol, Great Britain, 4- 8 September 2000

Document S2-001624
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

23.127	**CR**	**12**	Current Version:	3.1.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ ↑ *CR number as allocated by MCC support team*

For submission to:	SA#9	for approval	☒	strategic	□	*(for SMG use only)*
list expected approval meeting # here ↑		for information	□	non-strategic	□	

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from:* ftp://ftp.3gpp.org/Information/CR-Form-v2.doc

Proposed change affects: (U)SIM □ ME □ UTRAN / Radio □ Core Network ☒
(at least one should be marked with an X)

Source: Siemens **Date:** 31.8.2000

Subject: Alignments Parlay <-> OSA

Work item: OSA

Category:	F	Correction	☒	**Release:**	Phase 2	□
	A	Corresponds to a correction in an earlier release	□		Release 96	□
(only one category	B	Addition of feature	□		Release 97	□
shall be marked	C	Functional modification of feature	□		Release 98	□
with an X)	D	Editorial modification	□		Release 99	☒
					Release 00	□

Reason for change: As an ongoing activity, differences between the 23.127 and the Parlay 2.1 specification must be deleted. This CR address differences in the basic service interface.

Clauses affected: 5.4.2

Other specs affected:	Other 3G core specifications	☒	→ List of CRs:	29.198 CR 021R1 (Tdoc N5-00153)
	Other GSM core specifications	□	→ List of CRs:	
	MS test specifications	□	→ List of CRs:	
	BSS test specifications	□	→ List of CRs:	
	O&M specifications	□	→ List of CRs:	

Other comments:

help.doc

<---------- double-click here for help and instructions on how to create a CR.

5.4.2 Base Service Interface

This interface provides the base for all interfaces described in the following clauses. It allows an application to set an interface reference to be used by the OSA interfaces for requests and asynchronous responses to the application. For example, when an application wants to be notified upon the receipt of the "called party busy" event, the Service Capability Server must know where to send the notification. This reference can be provided by the application with the setCallBack method across the OSA API.

Name	Base_Service_Interface
Method	**`setCallback()`** This method specifies the reference address of the callback interface that an SCF uses to invoke methods on the application.
Direction	Application to Framework
Parameters	**appInterface** Specifies a reference to the application interface which is used for callbacks.
Returns	
Errors	

Name	Base_Service_Interface
Method	**`setCallbackWithSessionID ()`** This method specifies the reference address of the application’s callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call.
Direction	Application to Framework
Parameters	**appInterface** Specifies a reference to the application interface which is used for callbacks. **sessionID** Specifies the session for which the service can invoke the application’s callback interface.
Returns	
Errors	

3GPP Meeting S2#14
Bristol, Great Britain, 4-8 September 2000

Document S2-001625
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

23.127	**CR**	**13**	Current Version:	3.1.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	SA#9	for approval	X	strategic	□	*(for SMG use only)*
list expected approval meeting # here ↑		for information	□	non-strategic	□	

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from:* ftp://ftp.3gpp.org/Information/CR-Form-v2.doc

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM	□	ME	□	UTRAN / Radio	□	Core Network	X

Source:	Siemens	**Date:**	31.8.2000
Subject:	Alignments Parlay <-> OSA		
Work item:	OSA		

Category:	F	Correction	X	**Release:**	Phase 2	□
	A	Corresponds to a correction in an earlier release	□		Release 96	□
(only one category	B	Addition of feature	□		Release 97	□
shall be marked	C	Functional modification of feature	□		Release 98	□
with an X)	D	Editorial modification	□		Release 99	X
					Release 00	□

Reason for change:	As an ongoing activity, differences between the 23.127 and the Parlay 2.1 specification must be deleted. This CR address differences in the initial contact interfaces.
Clauses affected:	6.1.1

Other specs affected:	Other 3G core specifications	X	→ List of CRs:	29.198 CR 008R1 (Tdoc N5-00135)
	Other GSM core specifications	□	→ List of CRs:	
	MS test specifications	□	→ List of CRs:	
	BSS test specifications	□	→ List of CRs:	
	O&M specifications	□	→ List of CRs:	

Other comments:	

help.doc

<---------- double-click here for help and instructions on how to create a CR.

6.1.1 Initial Contact

The application gains a reference to the Initial Contact SCF for the Home Environment that they wish to access. This may be gained through a URL, a Naming or Trading Service or an equivalent service, a *stringified* object reference, etc. At this stage, the application has no guarantee that this is a reference to the Home Environment.

The application uses this reference to initiate the authentication process with the Home Environment.

Initial Contact supports the initiateAuthentication method to allow the authentication process to take place (using the Authentication SCF defined in subclause 6.1.2). This method must be the first invoked by the application. Invocations of other methods will fail until authentication has been successfully completed.

Once the application has authenticated with the provider, it can gain access to other framework and network service capability features. This is done by invoking the requestAccess method, by which the application requests a certain type of access service capability feature. The OSA Access service capability feature is defined in subclause 6.1.3.

The Initial Contact framework SCF is defined by a unique interface, consisting of the following methods.

Method	**`initiateAuthentication()`** The application uses this method to initiate the authentication process.
Direction	Application to Framework
Parameters	~~**clientAppID**~~ ~~This is an identifier for the application. It is used to identify the application to the framework, (see authenticate() on Authentication). If the clientAppID cannot be found by the framework, an error code is returned by the framework. The value of the parameter fwAuthInterface is NULL in this case.~~ This identifies the application domain to the framework, and provides a reference to the domain's authentication interface. **appDomain** The authInterface parameter is a reference to call the authentication interface of the client application. The type of this interface is defined by the authType parameter. If the interface reference is not of the correct type, the framework returns an error code (`P_INVALID_INTERFACE_TYPE`). **authType** This identifies the type of authentication mechanism requested by theapplication. It provides operators and HE-VASPss with the opportunity to use an alternative to the OSA Authentication interface, e.g. CORBA Security. ~~**appAuthInterface**~~ ~~This provides the reference for the framework to call the authentication interface of the application.~~
Returns	~~**fwAuthInterface**~~**fwDomain** This provides the application domain with a framework identifier, and a reference to call the authentication interface of the framework. ~~This provides the reference for the application to call the authentication SCF of the framework.~~
Errors	

3GPP Meeting S2#14
Bristol, Great Britain, 4- 8 September 2000

Document S2-001626
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

23.127	**CR**	**14**	Current Version:	3.1.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	SA#9	for approval	X	strategic	□	*(for SMG use only)*
list expected approval meeting # here ↑		for information	□	non-strategic	□	

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from:* ftp://ftp.3gpp.org/Information/CR-Form-v2.doc

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM	□	ME	□	UTRAN / Radio	□	Core Network	X

Source:	Siemens	**Date:**	31.8.2000
Subject:	Alignments Parlay <-> OSA		
Work item:	OSA		

Category:	F	Correction	X	**Release:**	Phase 2	□
	A	Corresponds to a correction in an earlier release	□		Release 96	□
(only one category	B	Addition of feature	□		Release 97	□
shall be marked	C	Functional modification of feature	□		Release 98	□
with an X)	D	Editorial modification	□		Release 99	X
					Release 00	□

Reason for change:	As an ongoing activity, differences between the 23.127 and the Parlay 2.1 specification must be deleted. This CR address differences in the access SCF.
Clauses affected:	6.1.3

Other specs	Other 3G core specifications	X	→ List of CRs:	29.198 CR 009R1 (Tdoc N5-00136)
affected:	Other GSM core specifications	□	→ List of CRs:	
	MS test specifications	□	→ List of CRs:	
	BSS test specifications	□	→ List of CRs:	
	O&M specifications	□	→ List of CRs:	

Other comments:	

help.doc

<---------- double-click here for help and instructions on how to create a CR.

6.1.3 OSA Access

During an authenticated session accessing the Framework, the application will be able to select and access an instance of a framework or network service capability feature.

Access to framework SCFs is gained by invoking the obtainInterface, or obtainInterfaceWithCallback methods. The latter is used when a callback reference is supplied to the framework. For example, a network SCF discovery interface reference is returned when invoking obtainInterface with "discovery" as the SCF name.

In order to use network SCFs, the application must first be authorised to do so by establishing a service agreement with the Home Environment. The application uses the discovery SCF to retrieve the ID of the network SCF they wish to use.They may then use the accessCheck method to check that they are authorised to use the network SCF. The selectService method is used to tell the Home Environment that the application wishes to use the network SCF. The signServiceAgreement method is used to digitally sign the agreement, and provide non-repudiation for both parties in agreeing that the SCF would be available for use.

Establishing a service agreement is a business level transaction, which requires the HE-VASP that owns the application to agree terms for the use of an SCF with the Home Environment. Service agreements can be reached using either off-line or on-line mechanisms. Off-line agreements will be reached outside of the scope of OSA interactions, and so are not described here. However, applications can make use of service agreements that are made off-line. Some Home Environments may only offer off-line mechanisms to reach service agreements.

After a service agreement has been established between the application and the Home Environment domains, the application will be able to make use of this agreement to access the network SCF.

The accessCheck method allows the application to check whether it has permission to access (read, write, etc) to a specified SCF, and specific SCF features. The application defines the security domain and context of access to the SCF. The access control policy is based on a number of conditions, events and permissions that determine whether the application is authorised to access the SCF/feature.

The accessCheck method is optional, in that can be called by the application to check that it has permission to use specific SCF features, before starting an SCF instance. It is not compulsory for the application to make this check before selecting a network SCF and signing a service agreement to use an instance of the SCF. If the accessCheck method confirms that the application has permission to use a specific SCF feature, then this feature should be available to the application when using the SCF instance. The Home Environment may include the results of the accessCheck as part of the service agreement, that is signed before using an SCF instance, thereby assuring the application that the SCF features will be available.

The selectService method is used to identify the SCF that the application wishes to use. A list of service properties initialises the SCF, and an SCF token is returned. The application and Home Environment must sign a copy of the service agreement to confirm the use of the SCF. The framework invokes signServiceAgreement method on the applications’s Access callback interface with the service agreement text to be signed. The application uses its digital signature key to sign the agreement text, and return the signed text to the framework. The application then calls the signServiceAgreement method on the OSA Access SCF. The framework signs the agreement text, retrieves a reference to a network manager interface for the selected SCF (using the getServiceManager method defined in clause 8), and returns this reference to the client application. In addition, the OSA Access interface may be invoked by SCSs in the context of SCF registration, see subclause 8.1.

The OSA Access framework SCF is defined by a single interface, which consists of the following methods.

Method	**`obtainInterface ()`** The application uses this method to obtain interface references to other framework SCFs (e.g. discovery, load manager). (The obtainInterfacesWithCallback method should be used if the application is required to supply a callback interface to the framework.)
Direction	Application to network
Parameters	**interfaceName**

	The name of the framework SCF to which a reference to the interface is requested.
Returns	**fwInterface** This is the reference to the SCF interface requested.
Errors	**INVALID_INTERFACE_NAME** Returned if the interfaceName is invalid.

Method	**obtainInterfaceWithCallback ()** The application uses this method to obtain interface references to other framework SCFs (e.g. discovery, load manager), when they are required to supply a callback interface to the framework. (The obtainInterface method should be used when no callback interface needs to be supplied.)
Direction	Application to network
Parameters	**interfaceName** The name of the framework SCF to which a reference to the interface is requested. **appInterface** This is the reference to the application interface which is used for callbacks. If an application interface is not needed, then this method should not be used. (The obtainInterface method should be used when no callback interface needs to be supplied.)
Returns	**fwInterface** This is the reference to the SCF requested.
Errors	**INVALID_INTERFACE_NAME** Returned if the interfaceName is invalid.

Method	**accessCheck()** This method may be used by the application to check whether it has been granted permission to access the specified SCF. The response is used to indicate whether the request for access has been granted or denied and if granted the level of trust that will be applied.
Direction	Application to network
Parameters	**serviceToken** The serviceToken identifies the specific SCF that the client application wishes to access. The service Token identifies the service type and service properties selected by the client application when it invoked selectService(). **securityContext** A context is a group of security relevant attributes that may have an influence on the result of the accessCheck request. **securityDomain** The security domain in which the application is operating may influence the access control decisions and the specific set of features that the requestor is entitled to use. **group** A group can be used to define the access rights associated with all applications that belong to that

	group. This simplifies the administration of access rights. **serviceAccessTypes** These are defined by the specific Security Model in use but are expected to include: Create, Read, Update, Delete as well as those specific to SCFs.
Returns	**serviceAccessControl** This is a structure containing: • policy: indicates whether access has been granted or denied. If granted then the parameter trustLevel must also have a value. • trustLevel: The trustLevel parameter indicates the trust level that the Home Environment has assigned to the application.
Errors	

Method	**`selectService ()`** This method is used by the application to identify the network SCF that the application wishes to use.
Direction	Application to network
Parameters	**serviceID** This identifies the SCF required. **serviceProperties** This is a list of the properties that the SCF should support. These properties (names and values) are used to initialise the SCF instance for use by the application.
Returns	**serviceToken** This is a free format text token returned by the framework, which can be signed as part of a service agreement. This will contain operator specific information relating to the service level agreement. The serviceToken has a limited lifetime. If the lifetime of the serviceToken expires, a method accepting the serviceToken will return an error code (`INVALID_Service_TOKEN`). Service Tokens will automatically expire if the application or framework invokes the endAccess method on the other's corresponding access interface.

Errors	**`INVALID_SERVICE_ID`**
	Returned if the serviceID is not recognised by the framework
	`INVALID_SERVICE_PROPERTY`
	Returned if a property is not recognised by the framework

Method	**`signServiceAgreement()`***`(application to network)`* This method is used by the application to request that the framework sign an agreement on the SCF, which allows the application to use the SCF. If the framework agrees, both parties sign the service agreement, and a reference to the manager interface of the SCF is returned to the application.
Direction	Application to network
Parameters	**serviceToken** This is the token returned by the framework in a call to the `selectService()` method. This token is used to identify the SCF instance requested by the application. **agreementText** This is the agreement text that is to be signed by the framework using the private key of the framework. **signingAlgorithm** This is the algorithm used to compute the digital signature.
Returns	**signatureAndServiceMgr** This is a reference to a structure containing the digital signature of the framework for the service agreement, and a reference to the manager interface of the SCF: • The digitalSignature is the signed version of a hash of the service token and agreement text given by the application. • The serviceMgrInterface is a reference to the manager interface for the selected SCF.
Errors	**`INVALID_SERVICE_TOKEN`** Returned if the serviceToken is not recognised by the framework

Method	**`signServiceAgreement()`***`(network to application)`* This method is used by the framework to request that the application sign an agreement on the SCF. It is called in response to the application calling the selectService() method on the Access SCF of the framework. The framework provides the service agreement text for the application to sign. If the application agrees, it signs the service agreement, returning its digital signature to the framework.
Direction	Network to application
Parameters	**serviceToken** This is the token returned by the framework in a call to the `selectService()` method. This token is used to identify the SCF instance to which this service agreement corresponds. (If the application selects many SCFs, it can determine which selected SCF corresponds to the service agreement by matching the service token.) **agreementText** This is the agreement text that is to be signed by the application using the private key of the application. **signingAlgorithm** This is the algorithm used to compute the digital signature.
Returns	**digitalSignature** The digitalSignature is the signed version of a hash of the service token and agreement text given by the framework.
Errors	

Method	**`terminateServiceAgreement()`***`(application to network)`* This method is used by the application to terminate a service agreement for the SCF.
Direction	Application To Network
Parameters	**serviceToken** This is the token passed back from the framework in a previous `selectService()` method call. This token is used to identify the service agreement to be terminated. **terminationText** This is the termination text describes the reason for the termination of the service agreement. **digitalSignature** This is a signed version of a hash of the service token and the termination text. The signing algorithm used is the same as the signing algorithm given when the service agreement was signed using `signServiceAgreement()`.The framework uses this to check that the `terminationTex`t has been signed by the application. If a match is made, the service agreement is terminated, otherwise an error is returned.
Returns	
Errors	

Method	**terminateServiceAgreement()** (network to application) This method is used by the framework to terminate a service agreement for the SCF.
Direction	Network to application
Parameters	**serviceToken** This is the token passed back from the framework in a previous selectService() method call. This token is used to identify the service agreement to be terminated. **terminationText** This is the termination text describes the reason for the termination of the service agreement. **digitalSignature** This is a signed version of a hash of the service token and the termination text. The signing algorithm used is the same as the signing algorithm given when the service agreement was signed using signServiceAgreement(). The framework uses this to confirm its identity to the application. The application can check that the terminationText has been signed by the framework.
Returns	
Errors	

Method	**endAccess()** The endAccess method is used to end the application's access session with the framework. The application requests that its access session be ended. After it is invoked, the application will not longer be authenticated with the framework. The application will not be able to use the references to any of the framework SCFs gained during the access session. Any calls to these SCF interfaces will fail.
Direction	Application To Network
Parameters	**endAccessProperties** This is a list of properties that can be used to tell the framework the actions to perform when ending the access session (e.g. existing service sessions may be stopped, or left running). If a property is not recognised by the framework, an error code (P_INVALID_PROPERTY) is returned.
Returns	
Errors	

Method	**`terminateAccess ()`** The terminateAccess method is used to end the application's access session with the framework (e.g. this may be done if the framework believes the application is masquerading as someone else. Using this method will force the application to re-authenticate if it wishes to continue using the framework SCFs.) After terminateAccess() is invoked, the application will not longer be authenticated with the framework. The application will not be able to use the references to any of the framework SCFs gained during the access session. Any calls to these interfaces will fail.
Direction	Network to application
Parameters	**terminationText** This is the termination text describes the reason for the termination of the access session. **signingAlgorithm** This is the algorithm used to compute the digital signature. **digitalSignature** This is a signed version of a hash of the termination text. The framework uses this to confirm its identity to the application. The application can check that the `terminationText` has been signed by the framework.
Returns	
Errors	

3GPP Meeting S2#14
Bristol, Great Britain, 4 - 8 September 2000

Document S2-001627
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

23.127 CR 15 Current Version: 3.1.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	SA#9	for approval	☒	strategic	□	*(for SMG use only)*
list expected approval meeting # here ↑		for information	□	non-strategic	□	

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from:* ftp://ftp.3gpp.org/Information/CR-Form-v2.doc

Proposed change affects: (U)SIM □ ME □ UTRAN / Radio □ Core Network ☒
(at least one should be marked with an X)

Source: Siemens **Date:** 31.8.2000

Subject: Alignments Parlay <-> OSA

Work item: OSA

Category:	F Correction	☒	**Release:**	Phase 2	□
	A Corresponds to a correction in an earlier release	□		Release 96	□
(only one category	B Addition of feature	□		Release 97	□
shall be marked	C Functional modification of feature	□		Release 98	□
with an X)	D Editorial modification	□		Release 99	☒
				Release 00	□

Reason for change: As an ongoing activity, differences between the 23.127 and the Parlay 2.1 specification must be deleted. This CR address differences in the load manager SCF.

Clauses affected: 6.3.1

Other specs	Other 3G core specifications	☒	→ List of CRs:	29.198 CR 011R1 (Tdoc N5-00138)
affected:	Other GSM core specifications	□	→ List of CRs:	
	MS test specifications	□	→ List of CRs:	
	BSS test specifications	□	→ List of CRs:	
	O&M specifications	□	→ List of CRs:	

Other comments:

help.doc

<---------- double-click here for help and instructions on how to create a CR.

6.3 Integrity Management SCFs

6.3.1 Load Manager

The Load Manager SCF permits to manage the load on both the application and network sides.

The framework API should allow the load to be distributed across multiple machines and across multiple component processes, according to a load balancing policy. The separation of the load balancing mechanism and load balancing policy ensures the flexibility of the load balancing functionality. The load balancing policy identifies what load balancing rules the framework should follow for the specific application. It might specify what action the framework should take as the congestion level changes. For example, some real-time critical applications will want to make sure continuous service is maintained, below a given congestion level, at all costs, whereas other applications will be satisfied with disconnecting and trying again later if the congestion level rises. Clearly, the load balancing policy is related to the QoS level to which the application is subscribed.

The Load Manager SCF consists of a single interface. Most methods are asynchronous, in that they are one-way invocations. Consequently, they do not lock a thread into waiting whilst a transaction performs. In this way, the application server can handle many more calls, than one that uses synchronous message calls.

The load management methods do not exchange callback interfaces as it is assumed that the application has supplied its Load Management callback interface at the time it obtains the Framework's Load Manager SCF, by use of the `obtainInterfaceWithCallback` method on the OSA Access SCF.

Method	**`reportLoad()`** The application notifies the framework about its current load level (0,1, or 2) when the load level on the application has changed. At ***level 0*** load, the application is performing within its load specifications (i.e. it is not congested or overloaded). At ***level 1*** load, the application is overloaded. At ***level 2*** load, the application is severly overloaded.
Direction	Application to network
Parameters	~~**requester**~~ ~~Specifies the application interface for callbacks.~~ **loadLevel** Specifies the load level for which the application reported.
Returns	
Errors	

Method	**`enableLoadControl()`** Upon detecting load condition change, (i.e. load level changing from 0 to 1, 0 to 2, 1 to 2 or 2 to 1, for the SCFs or framework which has been registered for load control), the framework enables load management activity at the application based on the policy.
Direction	Network to application
Parameters	**loadStatistics** Specifies the new load statistics
Returns	

Errors	

Method	`disableLoadControl()` After load level of the framework or SCF which has been registered for load control moves back to normal, framework disables load control activity at the application based on policy.
Direction	Network to application
Parameters	**serviceIDs** Specifies the framework and SCFs for which the load has changed to normal. The serviceIDs is null to specify the framework only.
Returns	
Errors	

Method	`resumeNotification()` Resume the notification from an application for its load status after the detection of load level change at the framework and the evaluation of the load balancing policy.
Direction	Network to application
Parameters	
Returns	
Errors	

Method	`suspendNotification()` Suspend the notification from an application for its load status after the detection of load level change at the framework and the evaluation of the load balancing policy.
Direction	Network to application
Parameters	
Returns	
Errors	

Method	**queryLoadReq ()** The application requests load statistic records for the framework and specified SCFs.
Direction	Application to Network
Parameters	~~**requester**~~ ~~Specifies the application interface for callbacks.~~ **serviceIDs** Specifies the framework, SCFs or applications for which the load statistics shall be reported. The serviceIDs is `null` for framework load statistics only. **timeInterval** Specifies the time interval within which the load statistics are generated.
Returns	
Errors	

Method	**queryLoadRes()** Returns load statistics to the application which requested the information.
Direction	Network to application
Parameters	**loadStatistics** Specifies the framework-supplied load statistics.
Returns	
Errors	

Method	**queryLoadErr()** Returns an error code to the application that requested load statistics.
Direction	Network to application
Parameters	**loadStatisticsError** Specifies the framework-supplied error code.
Returns	
Errors	

Method	`queryAppLoadReq()` The framework requests for load statistic records produced by a specified application.
Direction	Network to application
Parameters	**serviceIDs** Specifies the SCFs or applications for which the load statistics shall be reported. **timeInterval** Specifies the time interval within which the load statistics are generated.
Returns	
Errors	

Method	`queryAppLoadRes ()` Report load statistics back to the framework that requested the information.
Direction	Application to network
Parameters	**loadStatistics** Specifies the load statistics in the application.
Returns	
Errors	

Method	`queryAppLoadErr()` Return an error response to the framework that requested the application’s load statistics information.
Direction	Application to network
Parameters	**loadStatisticsError** Specifies the error code associated with the failed attempt to retrieve the application’s load statistics.
Returns	
Errors	

Method	**`registerLoadController ()`** Register the application for load management under various load conditions.
Direction	Application to network
Parameters	~~**requester**~~ ~~Specifies the application interface for callbacks.~~ **serviceIDs** Specifies the framework and SCFs to be registered for load control. To register for framework load control only, the `serviceIDs` is null.
Returns	
Errors	

Method	**`unregisterLoadController ()`** Unregister the application for load management.
Direction	Application to network
Parameters	~~**requester**~~ ~~Specifies the application interface for callbacks.~~ **serviceIDs** Specifies the framework or SCFs to be unregistered for load control.
Returns	
Errors	

Method	**`resumeNotification ()`** Resume load management notifications to the application for the framework and specified SCFs after their load condition changes.
Direction	Application to network
Parameters	**serviceIDs** Specifies the framework and SCFs for which notifications are to be resumed. The serviceIDs is null to resume notifications for the framework only.
Returns	
Errors	

Method	**`suspendNotification()`** Suspend load management notifications to the application for the framework and specified SCFs, while the application handles a temporary load condition.
Direction	Application to network
Parameters	**serviceIDs** Specifies the framework and SCFs for which notifications are to be suspended. The serviceIDs is null to suspend notifications for the framework only.
Returns	
Errors	

3GPP Meeting S2#14
Bristol, Great Britain, 4 - 8 September 2000

Document S2-001629
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

23.127	**CR**	**17**	Current Version: 3.1.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	SA#9	for approval	X	strategic	(for SMG use only)
list expected approval meeting # here ↑		for information		non-strategic	

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from:* ftp://ftp.3gpp.org/Information/CR-Form-v2.doc

Proposed change affects: (U)SIM ☐ ME ☐ UTRAN / Radio ☐ Core Network ☒
(at least one should be marked with an X)

Source: Siemens **Date:** 31.8.2000

Subject: Alignments Parlay <-> OSA

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	X	**Release:**	Phase 2	
A	Corresponds to a correction in an earlier release			Release 96	
B	Addition of feature			Release 97	
C	Functional modification of feature			Release 98	
D	Editorial modification			Release 99	X
				Release 00	

Reason for change: As an ongoing activity, differences between the 23.127 and the Parlay 2.1 specification must be deleted. This CR address differences in the service factory SCF.

Clauses affected: 8.2.2

Other specs affected:

Other 3G core specifications	X	→ List of CRs:	29.198 CR 014R1 (Tdoc N5-00141)
Other GSM core specifications		→ List of CRs:	
MS test specifications		→ List of CRs:	
BSS test specifications		→ List of CRs:	
O&M specifications		→ List of CRs:	

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

8.2.2 Service Factory

The Service Factory interface allows the framework to get access to a manager interface of a network SCF. It is used during the signServiceAgreement, in order to return an SCF manager interface reference to the application. Each SCF has a manager interface that is the initial point of contact for the network SCF. E.g., the call control SCF uses the Call Manager interface.

Method	**`getServiceManager()`** This method returns an SCF manager interface reference for the specified application. Usually, but not necessarily, this involves the instantiation of a new SCF manager interface.
Direction	Network to network (framework to service capability server)
Parameters	**application** Specifies the application for which the SCF manager interface is requested. **serviceProperties** Specifies the actual service property {name,value} pairs selected by the enterprise operator/client application when it invoked the *IpAccess.selectService* method.
Returns	**serviceManager** Specifies the SCF manager interface reference for the specified application.
Errors	-

3GPP Meeting S2#14
Bristol, Great Britain, 4 - 8 September 2000

Document S2-001628
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

23.127	**CR**	**16**	Current Version:	3.1.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	SA#9	for approval	X	strategic		(for SMG
list expected approval meeting # here ↑		for information		non-strategic		use only)

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from:* ftp://ftp.3gpp.org/Information/CR-Form-v2.doc

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM		ME		UTRAN / Radio		Core Network	X

Source: Siemens **Date:** 31.8.2000

Subject: Alignments Parlay <-> OSA

Work item: OSA

Category:	F	Correction	X	**Release:**	Phase 2	
	A	Corresponds to a correction in an earlier release			Release 96	
(only one category	B	Addition of feature			Release 97	
shall be marked	C	Functional modification of feature			Release 98	
with an X)	D	Editorial modification			Release 99	X
					Release 00	

Reason for change: As an ongoing activity, differences between the 23.127 and the Parlay 2.1 specification must be deleted. This CR address differences in the fault manager SCF.

Clauses affected: 6.3.2

Other specs affected:	Other 3G core specifications	X	→ List of CRs:	29.198 CR 010R1 (Tdoc N5-00137), 29.198 CR 011R1 (Tdoc N5-00138), 29.198 CR 013R1 (Tdoc N5-00140)
	Other GSM core specifications		→ List of CRs:	
	MS test specifications		→ List of CRs:	
	BSS test specifications		→ List of CRs:	
	O&M specifications		→ List of CRs:	

Other comments:

help.doc

<---------- double-click here for help and instructions on how to create a CR.

6.3.2 Fault Manager

This SCF is used by the application to inform the framework of events which affect the integrity of the framework and SCFs, and to request information about the integrity of the system.

It consists of a single interface, with the following methods.

Method	**activityTestReq()** This method may be used by the application to test that the framework or an SCF is methodal. On receipt of this request, the framework must carry out a test on the specified SCF or the framework itself to check that it is operating correctly and report the test result.
Direction	Application to network
Parameters	**activityTestID** The identifier provided by the application to correlate the response (when it arrives) with this request. **svcID** This parameter identifies which SCF the application is requesting the activity test to be done for. A null value denotes that the activity test is being requested for the framework. ~~**appID**~~ ~~This parameter identifies which application is requesting the activity test, and therefore which application to send the result to.~~
Returns	
Errors	

Method	**activityTestRes()** The framework returns the result of the activity test in this method, along with a test identifier to allow correlation of result to request within the application.
Direction	Network to application
Parameters	**activityTestID** The identifier provided by the application (in the request), to correlate this response with the original request. **activityTestResult** The result of the activity test.
Returns	
Errors	

Method	**appActivityTestReq ()** This method is invoked by the framework to request that the application carries out an activity test to check that is it operating correctly.
Direction	Network to application

Parameters	**activityTestID** The identifier provided by the application (in the request), to correlate this response with the original request.
Returns	
Errors	

Method	**`appActivityTestRes ()`** This method is used by the application to return the result of a previously requested activity test.
Direction	Application to network
Parameters	**ActivityTestID** The identifier is used by the framework to correlate this response (when it arrives) with the original request. **ActivityTestResult** The result of the activity test.
Returns	
Errors	

Method	**`fwFaultReportInd ()`** This method is invoked by the framework to notify the application of a failure within the framework. The application must not continue to use the framework until it has recovered (as indicated by a `fwFaultRecoveryInd`).
Direction	Network to application
Parameters	**fault** Specifies the fault that has been detected.
Returns	
Errors	

Method	**`fwFaultRecoveryInd ()`** This method is invoked by the framework to notify the application that a previously reported fault has been rectified.
Direction	Network to application
Parameters	**fault** Specifies the fault from which the framework has recovered.
Returns	
Errors	

Method	**`svcUnavailableInd ()`** This method is used by the application to inform the framework that it can no longer use the indicated SCF (either due to a failure in the application or in the SCF). On receipt of this request, the framework should take the appropriate corrective action. The framework assumes that the session between this application and instance SCF is to be closed and updates its own records appropriately as well as attempting to inform the SCF instance and/or its administrator. If the application then tries to continue the use of this session it should be returned an error.
Direction	Application to network
Parameters	**serviceID** The identity of the SCF which can no longer be used. ~~**appID**~~ ~~The identity of the application sending the indication.~~
Returns	
Errors	

Method	**`svcUnavailableInd ()`** This method is used by the framework to inform the application that it can no longer use the indicated SCF due to a failure in the SCF. On receipt of this request, the application must act to reset its use of the specified SCF (using the normal mechanisms such as the discovery and authentication interfaces to stop use of this SCF instance and begin use of a different SCF instance).
Direction	Network to application
Parameters	**serviceID** The identity of the SCF which can no longer be used. **reason** The reason why the SCF is no longer available.
Returns	
Errors	

Method	**`svcUnavailableInd ()`** This method is used by the client application to inform the framework that it can no longer use the indicated SCF (either due to a failure in the application or in the SCF). On receipt of this request, the framework should take the appropriate corrective action. The framework assumes that the session between this application and SCF instance is to be closed and updates its own records appropriately as well as attempting to inform the SCF instance and/or its administrator. Attempts by the application to continue using this session should be rejected.
Direction	Application to network
Parameters	**serviceID** The identity of the SCF which can no longer be used.
Returns	
Errors	

Method	**fwUnavailableInd ()** The framework invokes this method to inform the client application that it is no longer available.
Direction	Network to application
Parameters	**reason** Identifies the reason why the framework is no longer available
Returns	
Errors	

Method	**genFaultStatsRecordReq ()** This method is used by the application to solicit fault statistics from the framework. On receipt of this request, the framework must produce a fault statistics record, which is returned to the application. The fault statistics record must contain information about faults relating to the SCFs specified in the serviceIDList parameter, during the specified period.
Direction	Application to Network
Parameters	**timePeriod** The period over which the fault statistics are to be generated. A null value leaves this to the discretion of the framework. **serviceIDList** This parameter lists the SCFs that the application would like to have included in the general fault statistics record. If the application would like the framework fault statistics to be included it should include the NULL serviceID. ~~**appID**~~ ~~This parameter identifies which application is requesting the statistics record, and therefore which application to send the record to.~~
Returns	
Errors	

Method	**`genFaultStatsRecordRes ()`** This method is used by the framework to provide fault statistics to an application in response to a `genFaultStatsRecordReq`.
Direction	Network to application
Parameters	**faultStatistics** The fault statistics record. **serviceIDs** This parameter lists the SCFs that have been included in the general fault statistics record. The framework is denoted by the NULL service`ID`.

3GPP Meeting S2#14
Bristol, Great Britain, 4 - 8 September 2000

Document S2-01630
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

23.127	**CR**	**18**	Current Version:	3.1.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	SA#9	for approval	X	strategic	□	*(for SMG use only)*
list expected approval meeting # here ↑		for information	□	non-strategic	□	

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from:* ftp://ftp.3gpp.org/Information/CR-Form-v2.doc

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM	□	ME	□	UTRAN / Radio	□	Core Network	X

Source:	Siemens	**Date:**	31.8.2000
Subject:	Alignments Parlay <-> OSA		
Work item:	OSA		

Category:	F	Correction	X	**Release:**	Phase 2	□
	A	Corresponds to a correction in an earlier release	□		Release 96	□
(only one category	B	Addition of feature	□		Release 97	□
shall be marked	C	Functional modification of feature	□		Release 98	□
with an X)	D	Editorial modification	□		Release 99	X
					Release 00	□

Reason for change:	As an ongoing activity, differences between the 23.127 and the Parlay 2.1 specification must be deleted. This CR address differences in the authentication interface.
Clauses affected:	6.1.2

Other specs	Other 3G core specifications	X	→ List of CRs:	29.198 CR 010R1 (Tdoc N5-00137)
affected:	Other GSM core specifications	□	→ List of CRs:	
	MS test specifications	□	→ List of CRs:	
	BSS test specifications	□	→ List of CRs:	
	O&M specifications	□	→ List of CRs:	

Other comments:	

help.doc

<---------- double-click here for help and instructions on how to create a CR.

6.1.2 Authentication

Once the application has made initial contact with the Home Environment, authentication of the application and Home Environment may be required.

The API supports multiple authentication techniques. The procedure used to select an appropriate technique for a given situation is described below. The authentication mechanisms may be supported by cryptographic processes to provide confidentiality, and by digital signatures to ensure integrity. The inclusion of cryptographic processes and digital signatures in the authentication procedure depends on the type of authentication technique selected. In some cases strong authentication may need to be enforced by the Home Environment to prevent misuse of resources. In addition it may be necessary to define the minimum encryption key length that can be used to ensure a high degree of confidentiality.

The application must authenticate with the framework before it is able to use any of the other interfaces supported by the framework. Invocations on other interfaces will fail until authentication has been successfully completed.

1) The application calls initiateAuthentication on the Home Environment's framework Initial interface. This allows the application to specify the type of authentication process. This authentication process may be specific to the Home Environment, or the implementation technology used. The initiateAuthentication method can be used to specify the specific process, (e.g. CORBA security). OSA defines a generic authentication service capability feature (Authentication), which can be used to perform the authentication process. The initiateAuthentication method allows the application to pass a reference to its own authentication interface to the Framework, and receive a reference to the Authentication interface supported by the framework, in return.

2) The application invokes the selectAuthMethod on the framework's Authentication SCF. This includes the authentication capabilities of the application. The framework then chooses an authentication method based on the authentication capabilities of the application and the framework. If the application is capable of handling more than one authentication method, then the framework chooses one option, defined in the prescribedMethod parameter. In some instances, the authentication capability of the application may not fulfil the demands of the framework, in which case, the authentication will fail.

3) The application and framework interact to authenticate each other. Depending on the method prescribed, this procedure may consist of a number of messages e.g. a challenge/ response protocol. This authentication protocol is performed using the authenticate method on the Authentication interface. Depending on the authentication method selected, the protocol may require invocations on the Authentication SCF supported by the framework; or on the application counterpart; or on both.

The Authentication framework SCF is defined by a single interface, consisting of the following methods.

Method	**`selectAuthMethod ()`** The application uses this method to initiate the authentication process. The mechanism returned by the framework is the mechanism it prefers. This should be within capability of the application. If a mechanism that is acceptable to the framework within the capability of the application cannot be found, the framework returns an error code (INVALID_AUTH_CAPABILITY).
Direction	Application to network
Parameters	**authCaps~~ability~~** This is the means by which the authentication mechanisms supported by the application are conveyed to the framework.
Returns	**prescribedMethod** This is returned by the framework to indicate the mechanism it prefers for the authentication process. If the value of the prescribedMethod returned by the framework is not understood by the application, it is considered a fatal error and the application must abort.
Errors	**INVALID_AUTH_CAPABILITY** No acceptable authentication mechanism could be found by the framework.

Method	**`authenticate ()`** *`(application to network)`* This method is used by the application to authenticate the framework using the mechanism indicated in prescribed Method. The framework must respond with the correct responses to the challenges presented by the application. The clientAppID received in the `initiateAuthentication()`

	can be used by the framework to reference the correct public key for the application (the key management system is currently outside of the scope of the OSA specification). The number of interactions and the order of the interactions is dependent on the prescribedMethod.
Direction	Application to network
Parameters	**prescribedMethod** This parameter contains the method that the framework has specified as acceptable for authentication (see selectAuthMethod). **challenge** The challenge presented by the application to be responded to by the framework. The challenge mechanism used will be in accordance with the IETF *PPP Authentication Protocols - Challenge Handshake Authentication Protocol* [RFC 1994, August1996]. The challenge will be encrypted with the mechanism prescribed by `selectAuthMethod()`.
Returns	**response** This is the response of the framework to the challenge of the application in the current sequence. The response will be based on the challenge data, decrypted with the mechanism prescribed by `selectAuthMethod()`.
Errors	

Method	**`authenticate()`** *`(network to application)`* This method is used by the framework to authenticate the application using the mechanism indicated in prescibedMechanism. The application must respond with the correct responses to the challenges presented by the framework. The number of interactions and the order of the interactions is dependant on the prescribedMethod. (These may be interleaved with authenticate() calls by the application on the Authentication interface. This is defined by the prescribedMethod.)
Direction	Network to application
Parameters	**prescribedMethod** This parameter contains the agreed method for authentication (see selectAuthMethod on the Authentication interface.) **challenge** The challenge presented by the framework to be responded to by the application. The challenge mechanism used will be in accordance with the IETF *PPP Authentication Protocols - Challenge Handshake Authentication Protocol* [RFC 1994, August1996]. The challenge will be encrypted with the mechanism prescribed by `selectAuthMethod()`.
Returns	**response** This is the response of the application to the challenge of the framework in the current sequence. The response will be based on the challenge data, decrypted with the mechanism prescribed by `selectAuthMethod()`.
Errors	**`INVALID_AUTHENTICATION`** The application could not be authenticated.

Method	**`abortAuthentication()`***`(application to network)`* The application uses this method to abort the authentication process. This method is invoked if the application no longer wishes to continue the authentication process, (e.g. if the framework responds incorrectly to a challenge.) If this method has been invoked, calls to the requestAccess method on

	Initial Contact will return an error code (INVALID_AUTHENTICATION) until the application has been properly authenticated.
Direction	Application to network
Parameters	
Returns	
Errors	

Method	**abortAuthentication()***(network to application)* The framework uses this method to abort the authentication process. This method is invoked if the framework wishes to abort the authentication process, (e.g. if the application responds incorrectly to a challenge.) If this method has been invoked, calls to the requestAccess method on Initial will return an error code (INVALID_AUTHENTICATION), until the application has been properly authenticated.
Direction	Network to application
Parameters	
Returns	
Errors	