3GPP TSG-CN Meeting #24 **NP-040253**
02 – 04 June 2004, Seoul, KOREA

Source: **CN5 (OSA)**

Title: **4 Rel-4 CRs 29.198-03 OSA API Part 3: Framework**

Agenda item: **7.10 (OSA Enhancements [OSA1])**

Document for: **APPROVAL**

Doc-1st-	Spec	CR	Rev	Phase	Subject	Cat	Version	Doc-2nd-	Workite
NP-040253	29.198-03	109	-	Rel-4	Correct alignment between ETSI/Parlay version of OSA and the 3GPP OSA, by clarifying erroneous field in TpServiceProfileDescription	F	4.8.0	N5-040066	OSA1
NP-040253	29.198-03	105	-	Rel-5	Correct alignment between ETSI/Parlay version of OSA and the 3GPP OSA, by clarifying erroneous field in TpServiceProfileDescription	A	5.6.0	N5-040058	OSA1
NP-040253	29.198-03	106	-	Rel-6	Correct alignment between ETSI/Parlay version of OSA and the 3GPP OSA, by clarifying erroneous field in TpServiceProfileDescription	A	6.0.1	N5-040059	OSA1
NP-040253	29.198-03	122	-	Rel-4	Correction of Digital Signature with NO signing algorithm	F	4.8.0	N5-040078	OSA1

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-03 CR 105 ⌘rev - ⌘ Current version: 5.6.0 ⌘

*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 X

Title:	⌘	Correct alignment between ETSI/Parlay version of OSA and the 3GPP OSA, by clarifying erroneous field in TpServiceProfileDescription
Source:	⌘	CN5 Parlay Gareth Carroll (Open API Solutions)
Work item code:	⌘	OSA1
Date:	⌘	09/02/2004
Category:	⌘	A
Release:	⌘	REL-5

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: ⌘ There is an error in the definition of the Service Profile (TpServiceProfileDescription). This structure contains a field called ServiceTypeName. As a profile will never exist independently of a service contract then this field MUST NOT contradict the value in the corresponding field in the service contract. It is surely impossible to have a service profile (which is a restriction of a service contract) for a different service type than the corresponding service contract. The presence of this field calls for the Framework to validate or ignore it when it is passed in to createServiceProfile. A note should be added to this field to explain its actual role, as the field cannot be removed due to backwards compatibility reasons.

Since the intention is to keep the data types common between the ETSI/Parlay version of OSA and the 3GPP OSA specification, any change to the datatypes in the ETSI/Parlay specification must also be reflected in the 3GPP specification.

Summary of change: ⌘ We propose to add a note to the ServiceTypeName field stating the correct behaviour with regards to the value:

- When the FW gives a TpServiceProfileDescription to the enterprise operator, it should set this field to the same value as the corresponding field of the service contract;
- When the enterprise operator passes a TpServiceProfileDescription to the FW, the FW should ignore the value sent in this field to ensure interoperability;
- The enterprise operator should be required to set the field to the correct value when passing a TpServiceProfileDescription to the FW (this will ensure that there are no problems for Framework implementations that do not ignore the value)

Consequences if not approved: ⌘ This contribution was accepted for the ETSI/Parlay specifications. If it is not approved for the 3GPP specification, then a misalignment will occur.
Any misalignment between the ETSI/Parlay and 3GPP specifications will lead to incompatibilities between equipment developed from the ETSI/Parlay

	specifications and equipment developed from the 3GPP specifications. It will force vendors to develop different versions of their OSA products, one for operators requiring support of 3GPP specifications, and one for all other operators. This will increase development costs unnecessarily, increase the cost of deploying OSA, and fragment the single developer community which has formed around OSA.

Clauses affected:	⌘	10.5.32			
		Y	**N**		
Other specs affected:	⌘		**X**	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘	This is the Rel-5 Mirror CR to Rel-4 CR to 29.198-03 in N5-040066.			

10.5 Service Subscription Data Definitions

10.5.32 TpServiceProfileDescription

This data type is a `Sequence of Data Elements` which describes a Service Profile. A service contract contains one or more Service Profiles, one for each SAG in the enterprise operator domain. A service profile is a restriction of the service contract in order to provide restricted service features to a SAG. It is a structured data type which consists of:

Sequence Element Name	Sequence Element Type
`ServiceContractID`	TpServiceContractID
`ServiceStartDate`	TpServiceStartDate
`ServiceEndDate`	TpServiceEndDate
`ServiceTypeName`	TpServiceTypeName (See Note 1)
`ServiceSubscriptionProperties`	TpServiceSubscriptionProperties

Note 1: When the Framework returns a TpServiceProfileDescription to the enterprise operator, it should set this field to the same value as the corresponding field of the service contract; When the enterprise operator passes a TpServiceProfileDescription to the Framework, the Framework should ignore the value sent in this field to ensure interoperability; The enterprise operator should be required to set the field to the correct value when passing a TpServiceProfileDescription to the Framework.

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03** **CR** **106** ⌘ **rev** **-** ⌘ Current version: **6.0.1** ⌘

*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 **X**

Title:	⌘	Correct alignment between ETSI/Parlay version of OSA and the 3GPP OSA, by clarifying erroneous field in TpServiceProfileDescription
Source:	⌘	CN5 Parlay Gareth Carroll (Open API Solutions)
Work item code:	⌘	OSA1
Date:	⌘	09/02/2004
Category:	⌘	**A**
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:	⌘	There is an error in the definition of the Service Profile (TpServiceProfileDescription). This structure contains a field called ServiceTypeName. As a profile will never exist independently of a service contract then this field MUST NOT contradict the value in the corresponding field in the service contract. It is surely impossible to have a service profile (which is a restriction of a service contract) for a different service type than the corresponding service contract. The presence of this field calls for the Framework to validate or ignore it when it is passed in to createServiceProfile. A note should be added to this field to explain its actual role, as the field cannot be removed due to backwards compatibility reasons. Since the intention is to keep the data types common between the ETSI/Parlay version of OSA and the 3GPP OSA specification, any change to the datatypes in the ETSI/Parlay specification must also be reflected in the 3GPP specification.
Summary of change:	⌘	We propose to add a note to the ServiceTypeName field stating the correct behaviour with regards to the value: - When the FW gives a TpServiceProfileDescription to the enterprise operator, it should set this field to the same value as the corresponding field of the service contract; - When the enterprise operator passes a TpServiceProfileDescription to the FW, the FW should ignore the value sent in this field to ensure interoperability; - The enterprise operator should be required to set the field to the correct value when passing a TpServiceProfileDescription to the FW (this will ensure that there are no problems for Framework implementations that do not ignore the value)
Consequences if not approved:	⌘	This contribution was accepted for the ETSI/Parlay specifications. If it is not approved for the 3GPP specification, then a misalignment will occur. Any misalignment between the ETSI/Parlay and 3GPP specifications will lead to incompatibilities between equipment developed from the ETSI/Parlay

	specifications and equipment developed from the 3GPP specifications. It will force vendors to develop different versions of their OSA products, one for operators requiring support of 3GPP specifications, and one for all other operators. This will increase development costs unnecessarily, increase the cost of deploying OSA, and fragment the single developer community which has formed around OSA.

Clauses affected:	⌘	10.5.32			
		Y	**N**		
Other specs affected:	⌘		**X**	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘	This is the Rel-6 Mirror CR to Rel-4 CR to 29.198-03 in N5-040066.			

10.5 Service Subscription Data Definitions

10.5.32 TpServiceProfileDescription

This data type is a Sequence of Data Elements which describes a Service Profile. A service contract contains one or more Service Profiles, one for each SAG in the enterprise operator domain. A service profile is a restriction of the service contract in order to provide restricted service features to a SAG. It is a structured data type which consists of:

Sequence Element Name	Sequence Element Type
ServiceContractID	TpServiceContractID
ServiceStartDate	TpServiceStartDate
ServiceEndDate	TpServiceEndDate
ServiceTypeName	TpServiceTypeName (See Note 1)
ServiceSubscriptionProperties	TpServiceSubscriptionProperties

Note 1: When the Framework returns a TpServiceProfileDescription to the enterprise operator, it should set this field to the same value as the corresponding field of the service contract; When the enterprise operator passes a TpServiceProfileDescription to the Framework, the Framework should ignore the value sent in this field to ensure interoperability; The enterprise operator should be required to set the field to the correct value when passing a TpServiceProfileDescription to the Framework.

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03** **CR** **109** ⌘ **rev** **-** ⌘ Current version: **4.8.0** ⌘

*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 **X**

Title:	⌘	Correct alignment between ETSI/Parlay version of OSA and the 3GPP OSA, by clarifying erroneous field in TpServiceProfileDescription
Source:	⌘	CN5 Parlay Gareth Carroll (Open API Solutions)
Work item code:	⌘	OSA1
Date:	⌘	09/02/2004
Category:	⌘	**F**
Release:	⌘	*REL-4*

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: ⌘ There is an error in the definition of the Service Profile (TpServiceProfileDescription). This structure contains a field called ServiceTypeName. As a profile will never exist independently of a service contract then this field MUST NOT contradict the value in the corresponding field in the service contract. It is surely impossible to have a service profile (which is a restriction of a service contract) for a different service type than the corresponding service contract. The presence of this field calls for the Framework to validate or ignore it when it is passed in to createServiceProfile. A note should be added to this field to explain its actual role, as the field cannot be removed due to backwards compatibility reasons.

Since the intention is to keep the data types common between the ETSI/Parlay version of OSA and the 3GPP OSA specification, any change to the datatypes in the ETSI/Parlay specification must also be reflected in the 3GPP specification.

Summary of change: ⌘ We propose to add a note to the ServiceTypeName field stating the correct behaviour with regards to the value:

- When the FW gives a TpServiceProfileDescription to the enterprise operator, it should set this field to the same value as the corresponding field of the service contract;
- When the enterprise operator passes a TpServiceProfileDescription to the FW, the FW should ignore the value sent in this field to ensure interoperability;
- The enterprise operator should be required to set the field to the correct value when passing a TpServiceProfileDescription to the FW (this will ensure that there are no problems for Framework implementations that do not ignore the value)

Consequences if not approved: ⌘ This contribution was accepted for the ETSI/Parlay specifications. If it is not approved for the 3GPP specification, then a misalignment will occur. Any misalignment between the ETSI/Parlay and 3GPP specifications will lead to incompatibilities between equipment developed from the ETSI/Parlay

	specifications and equipment developed from the 3GPP specifications. It will force vendors to develop different versions of their OSA products, one for operators requiring support of 3GPP specifications, and one for all other operators. This will increase development costs unnecessarily, increase the cost of deploying OSA, and fragment the single developer community which has formed around OSA.

Clauses affected:	⌘	10.5.32			
		Y	**N**		
Other specs affected:	⌘		**X**	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘	This CR has Rel-5 and Rel-6 Mirror CRs in N5-040058 and N5-030059.			

10.5 Service Subscription Data Definitions

10.5.32 TpServiceProfileDescription

This data type is a `Sequence of Data Elements` which describes a Service Profile. A service contract contains one or more Service Profiles, one for each SAG in the enterprise operator domain. A service profile is a restriction of the service contract in order to provide restricted service features to a SAG. It is a structured data type which consists of:

Sequence Element Name	Sequence Element Type
`ServiceContractID`	TpServiceContractID
`ServiceStartDate`	TpServiceStartDate
`ServiceEndDate`	TpServiceEndDate
`ServiceTypeName`	TpServiceTypeName (See Note 1)
`ServiceSubscriptionProperties`	TpServiceSubscriptionProperties

Note 1: When the Framework returns a TpServiceProfileDescription to the enterprise operator, it should set this field to the same value as the corresponding field of the service contract; When the enterprise operator passes a TpServiceProfileDescription to the Framework, the Framework should ignore the value sent in this field to ensure interoperability; The enterprise operator should be required to set the field to the correct value when passing a TpServiceProfileDescription to the Framework.

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040078**
Meeting #26, Atlanta, GA, USA, 16-20 February 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03** **CR** **122** ⌘ **rev** **-** ⌘ Current version: **4.8.0** ⌘

*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 **X**

Title:	⌘	Correction of Digital Signature with NO signing algorithm
Source:	⌘	CN5 Fujitsu Limited. (Yumi Suzuki), NTT(Atsushi Iwasaki)
Work item code:	⌘	OSA1
Date:	⌘	20/02/2004
Category:	⌘	**F**
Release:	⌘	*REL-4*

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:	⌘	This contribution discusses about the digital signature if the signing algorithm is not used. At the 2nd Parlay/OSA Plug Test, we encountered the problem that the making mechanism of the digital signature and the validity logic of the digital signature differ with each vendor in case of no signing algorithm.
Summary of change:	⌘	The current specification does not mention about which string should be passed as the digital signature and how it should be checked when signing algorithm is not required. The Framework and the client can just ignore the digital signature, but also can verify it regardless of the signing algorithm. From the point of security, we propose to add some description in order to create the digital signature which contains the service token and agreement text whether the signing algorithm exists or not, and to confirm it each other.
Consequences if not approved:	⌘	Interoperability problems

Clauses affected:	⌘	6.3.1.2, 7.3.2.1, 7.3.2.2, 10.3.10

Other specs affected: ⌘	Y	N	
		X	Other core specifications
		X	Test specifications
		X	O&M Specifications

⌘

Other comments:	⌘	7

6.3.1.2 Interface Class IpClientAccess

Method

terminateAccess()

The terminateAccess operation is used by the framework to end the client's access session.

After terminateAccess() is invoked, the client will no longer be authenticated with the framework. The client will not be able to use the references to any of the framework interfaces gained during the access session. Any calls to these interfaces will fail. If at any point the framework's level of confidence in the identity of the client becomes too low, perhaps due to re-authentication failing, the framework should terminate all outstanding service agreements for that client, and should take steps to terminate the client's access session WITHOUT invoking terminateAccess() on the client. This follows a generally accepted security model where the framework has decided that it can no longer trust the client and will therefore sever ALL contact with it.

Parameters

terminationText : in TpString

This is the termination text describes the reason for the termination of the access session.

signingAlgorithm : in TpSigningAlgorithm

This is the algorithm used to compute the digital signature. If the signingAlgorithm is invalid, or unknown to the client, the P_INVALID_SIGNING_ALGORITHM exception will be thrown.

digitalSignature : in TpOctetSet

This is a signed version of a hash of the termination text. If No signing algorithm is used, the digitalSignature is the octet sequence of the termination text itself. The framework uses this to confirm its identity to the client. The client can check that the terminationText has been signed by the framework. If a match is made, the access session is terminated, otherwise the P_INVALID_SIGNATURE exception will be thrown.

Raises

TpCommonExceptions, P_INVALID_SIGNING_ALGORITHM, P_INVALID_SIGNATURE

7.3.2.1 Interface Class IpAppServiceAgreementManagement

Method

signServiceAgreement()

Upon receipt of the initiateSignServiceAgrement() method from the client application, this method is used by the framework to request that the client application sign an agreement on the service. The framework provides the service agreement text for the client application to sign. The service manager returned will be configured as per the service level agreement. If the framework uses service subscription, the service level agreement will be encapsulated in the subscription properties contained in the contract/profile for the client application, which will be a restriction of the registered properties. If the client application agrees, it signs the service agreement, returning its digital signature to the framework.

Returns <digitalSignature> : The digitalSignature is the signed version of a hash of the service token and agreement text given by the framework. If No signing algorithm is used, the digitalSignature is the octet sequence of the service token and the agreement text. If the signature is incorrect the serviceToken will be expired immediately.

Parameters

serviceToken : in TpServiceToken

This is the token returned by the framework in a call to the selectService() method. This token is used to identify the service instance to which this service agreement corresponds. (If the client application selects many services, it can

determine which selected service corresponds to the service agreement by matching the service token.) If the serviceToken is invalid, or not known by the client application, then the P_INVALID_SERVICE_TOKEN exception is thrown.

agreementText : in TpString

This is the agreement text that is to be signed by the client application using the private key of the client application. If the agreementText is invalid, then the P_INVALID_AGREEMENT_TEXT exception is thrown.

signingAlgorithm : in TpSigningAlgorithm

This is the algorithm used to compute the digital signature. If the signingAlgorithm is invalid, or unknown to the client application, the P_INVALID_SIGNING_ALGORITHM exception is thrown.

Returns

TpOctetSet

Raises

TpCommonExceptions, P_INVALID_AGREEMENT_TEXT, P_INVALID_SERVICE_TOKEN, P_INVALID_SIGNING_ALGORITHM

Method

terminateServiceAgreement()

This method is used by the framework to terminate an agreement for the service.

Parameters

serviceToken : in TpServiceToken

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. If the serviceToken is invalid, or unknown to the client application, the P_INVALID_SERVICE_TOKEN exception will be thrown.

terminationText : in TpString

This is the termination text that describes the reason for the termination of the service agreement.

digitalSignature : in TpOctetSet

This is a signed version of a hash of the service token and the termination text. If No signing algorithm is used, the digitalSignature is the octet sequence of the termination text itself. 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 client application. The client application can check that the terminationText has been signed by the framework. If a match is made, the service agreement is terminated, otherwise the P_INVALID_SIGNATURE exception will be thrown.

Raises

TpCommonExceptions, P_INVALID_SERVICE_TOKEN, P_INVALID_SIGNATURE

7.3.2.2 Interface Class IpServiceAgreementManagement

Method

signServiceAgreement()

This method is used by the client application to request that the framework sign an agreement on the service, which allows the client application to use the service. If the framework agrees, both parties sign the service agreement, and a reference to the service manager interface of the service is returned to the client application. The service manager returned will be configured as per the service level agreement. If the framework uses service subscription, the service level agreement will be encapsulated in the subscription properties contained in the contract/profile for the client application, which will be a restriction of the registered properties. If the client application is not allowed to access the service, then an error code (P_SERVICE_ACCESS_DENIED) is returned.

Returns <signatureAndServiceMgr> : This contains the digital signature of the framework for the service agreement, and a reference to the service manager interface of the service.

```
structure TpSignatureAndServiceMgr {
        digitalSignature: TpOctetSet;
            serviceMgrInterface: IpServiceRef;
        };
```

The digitalSignature is the signed version of a hash of the service token and agreement text given by the client application. If No signing algorithm is used, the digitalSignature is the octet sequence of the service token and the agreement text given by the client application.
The serviceMgrInterface is a reference to the service manager interface for the selected service.

Parameters

serviceToken : in TpServiceToken

This is the token returned by the framework in a call to the selectService() method. This token is used to identify the service instance requested by the client application. If the serviceToken is invalid, or has expired, an error code (P_INVALID_SERVICE_TOKEN) is returned.

agreementText : in TpString

This is the agreement text that is to be signed by the framework using the private key of the framework. If the agreementText is invalid, then an error code (P_INVALID_AGREEMENT_TEXT) is returned.

signingAlgorithm : in TpSigningAlgorithm

This is the algorithm used to compute the digital signature. If the signingAlgorithm is invalid, or unknown to the framework, an error code (P_INVALID_SIGNING_ALGORITHM) is returned.

Returns

TpSignatureAndServiceMgr

Raises

TpCommonExceptions,P_ACCESS_DENIED,P_INVALID_AGREEMENT_TEXT,P_INVALID_SERVICE_TOKEN,P_INVALID_SIGNING_ALGORITHM,P_SERVICE_ACCESS_DENIED

Method

terminateServiceAgreement()

This method is used by the client application to terminate an agreement for the service.

Parameters

`serviceToken : in TpServiceToken`

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. If the serviceToken is invalid, or has expired, an error code (P_INVALID_SERVICE_TOKEN) is returned.

`terminationText : in TpString`

This is the termination text that describes the reason for the termination of the service agreement.

`digitalSignature : in TpOctetSet`

This is a signed version of a hash of the service token and the termination text. If No signing algorithm is used, the digitalSignature is the octet sequence of the termination text itself. 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 terminationText has been signed by the client application. If a match is made, the service agreement is terminated, otherwise an error code (P_INVALID_SIGNATURE) is returned.

Raises

`TpCommonExceptions, P_ACCESS_DENIED, P_INVALID_SERVICE_TOKEN, P_INVALID_SIGNATURE`

10.3.10 TpSignatureAndServiceMgr

This is a Sequence of Data Elements containing the digital signature of the Framework for the service agreement, and a reference to the SCF manager interface of the SCF.

Sequence Element Name	Sequence Element Type
`DigitalSignature`	`TpOctetSet`
`ServiceMgrInterface`	`IpServiceRef`

The digitalSignature is the signed version of a hash of the service token and agreement text given by the client application. If No signing algorithm is used, the digitalSignature is the octet sequence of the service token and agreement text given by the client application.

The ServiceMgrInterface is a reference to the SCF manager interface for the selected SCF.

10.3.11 SigningAlgorithm

This data type is identical to a TpString, and is defined as a string of characters that identify the signing algorithm that shall be used. Other Network operator specific capabilities may also be used, but should be preceded by the string "SP_". The following values are defined.

String Value	Description
NULL	An empty (NULL) string indicates no signing algorithm is required
P_MD5_RSA_512	MD5 takes an input message of arbitrary length and produces as output a 128-bit message digest of the input. This is then encrypted with the private key under the RSA public-key cryptography system using a 512-bit key.
P_MD5_RSA_1024	MD5 takes an input message of arbitrary length and produces as output a 128-bit message digest of the input. This is then encrypted with the private key under the RSA public- key cryptography system using a 1024-bit key