[Open and Secure Digital object Rights Management]

OpenSDoRM API Specification

Identifier:	DoRM-APISPEC
Pages:	109
Version:	2005.10.11
Access:	Private
State:	Work in Progress
Distribution:	Internal to the project

Table of Contents

Purpose

The purpose of this document is to provide a description of the OpenSDoRM DRM system architecture components and the functions which are associated to them. Its main aim is to provide a guideline for someone wishing to DRM-enable some system or application to be able to do so and integrate it with the OpenSDoRM platform.

OpenSDoRM deals with the rights management and not directly with the copy protection tools or mechanisms. It is independent of the protection mechanisms being applied to the content. Therefore it is out of the scope of this document to talk about how the content gets protected or how the content is formatted.

Introduction

The OpenSDoRM platform

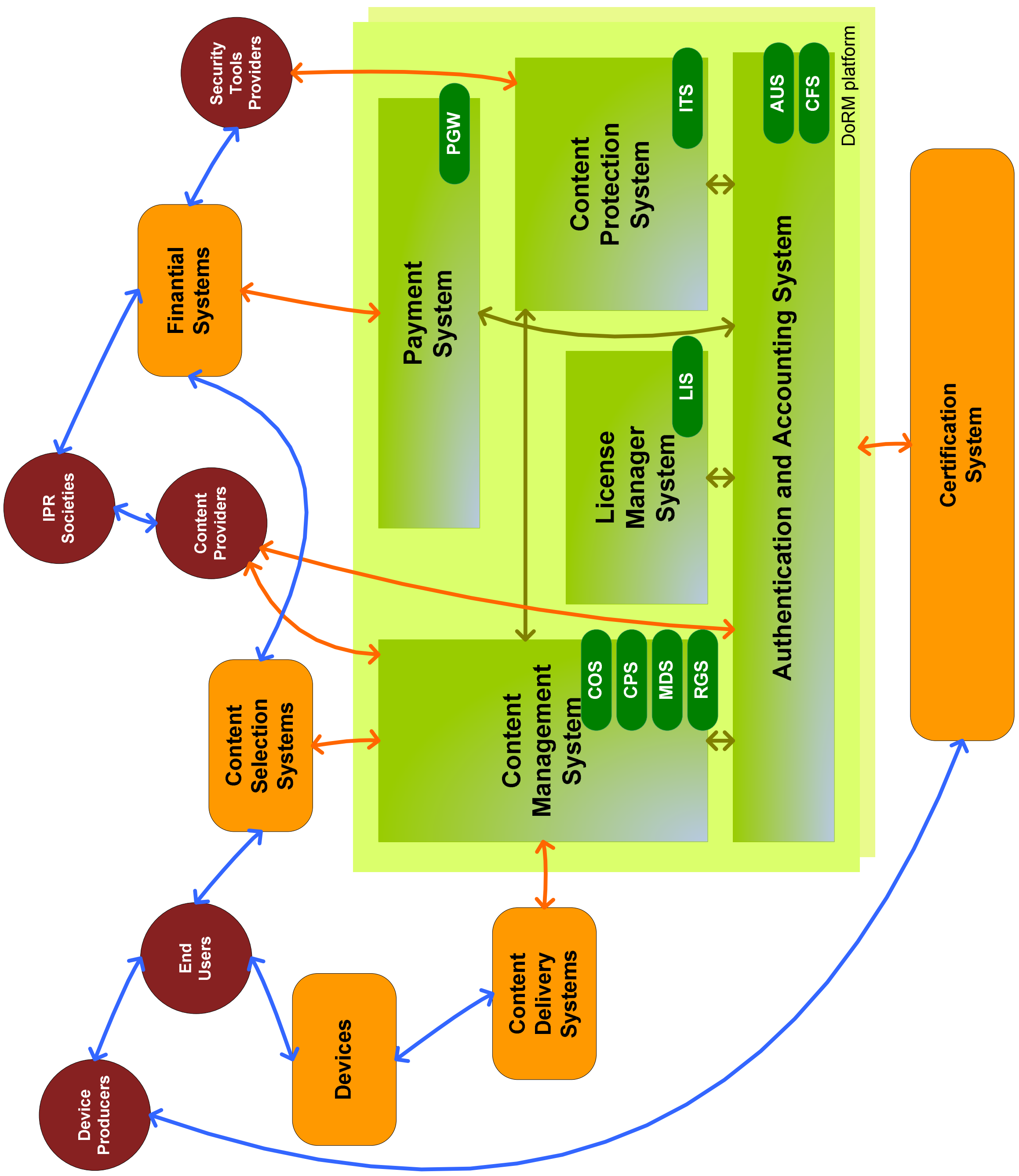
This section of the document details the OpenSDoRM rights management platform, presenting details about its conceptual architecture as well as it provides the description of both the external and internal OpenSDoRM components.

This section is divided in two different parts: the first part describes both the conceptual architecture as well as the different components, while the second provides an in-depth vision of the security of the OpenSDoRM system.

OpenSDoRM components

The OpenSDoRM rights management platform is composed of a set of distributed components that exchange standardized messages over open networks (such as Internet). The OpenSDoRM conceptual architecture (presented on the following picture) defines a scenario capable of handling a multiplicity of different business models for content distribution.

This conceptual architecture lies on three different building blocks: the user (not necessarily the end-users) roles; a set of external entities to the DRM process itself; and the internal DRM entities which provide the DRM functionalities.


Security Tools Providers
Finantial Systems
IPR Societies
Content Providers
Content Selection Systems
End Users
Device Producers
Devices
Content Delivery Systems
Payment System
PGW
Content Protection System
ITS
License Manager System
LIS
Content Management System
COS
CPS
MDS
RGS
Authentication and Accounting System
AUS
CFS
DoRM platform
Certification System

User Roles

These are the roles that are represented by the several entities on the content value-chain. It is important not to mistake these User roles, with end-user roles. The OpenSDoRM framework identifies several of these user roles:

- Author/Owner Societies: these are the societies that are responsible for upholding the copyright on the content and to ensure the revenue of the content author. Examples of such entities are for instance the SPA (Portuguese Author's Society), in Portugal.
- Content Providers: these represent the content authors which produce creative work and that distribute it over some digital medium. Additionally the content authors will be able to define the terms and the conditions under which their content may or may not be used.
- Device Producers: they represent device manufacturers (which may be hardware, software or both) that produce content rendering devices capable of enforcing DRM/CP-related functions on content usage. These device manufacturers can be certified by the DRM platform.
- Security Tools Providers: in order to ensure interoperability among the different content formats and different protection technologies, there must be some technology that may allow a given device to adapt its capabilities to the type of content it is trying to render. These entities provide the technological means for a rendering device to adapt its security features to the type of protection technology enforced upon a given content.
- End Users: these represent the final users that want to select and use the content. Usage of the content at this stage is always controlled by client DRM modules.

External Entities

These entities, represented on the conceptual model, support the non-DRM functions on the architecture. These entities are: Payment Infrastructure, Content Selection Modules, Devices, Content Delivery Servers and the Certification System.

- Payment Infrastructure: this external entity is responsible for handling all the financial transactions, and to assure that the appropriate content value-chain players receive the appropriate incoming for their effort;
- Content Selection Modules: this is either an external entity or module that can be used to find, browse and select content. Examples of such modules are for instance an Electronic Commerce front-store or an Electronic Program Guide;
- Devices: the devices are mechanisms that allow the final users to enjoy content they obtained.
- Content Delivery Servers: this entity represents the servers that will be feeding the content to the final users, or its owned devices.
- Certification System: this is an external (although it can also be internal) entity that issues the appropriate credentials for all components' certification.

DRM Entities

These entities provide on the OpenSDoRM conceptual model the DRM-enabling elements on the system. It is these elements (both server and client-side) that will provide the necessary DRM functionalities to protect the content and to uphold the rights associated to that content and user. The entities are: Content Management System, Payment Gateway Module, License Manager System, Content Protection Tools System and the Authentication and Accounting System. More details about each element provided by the conceptual DRM entities are provided on the technical architecture description.

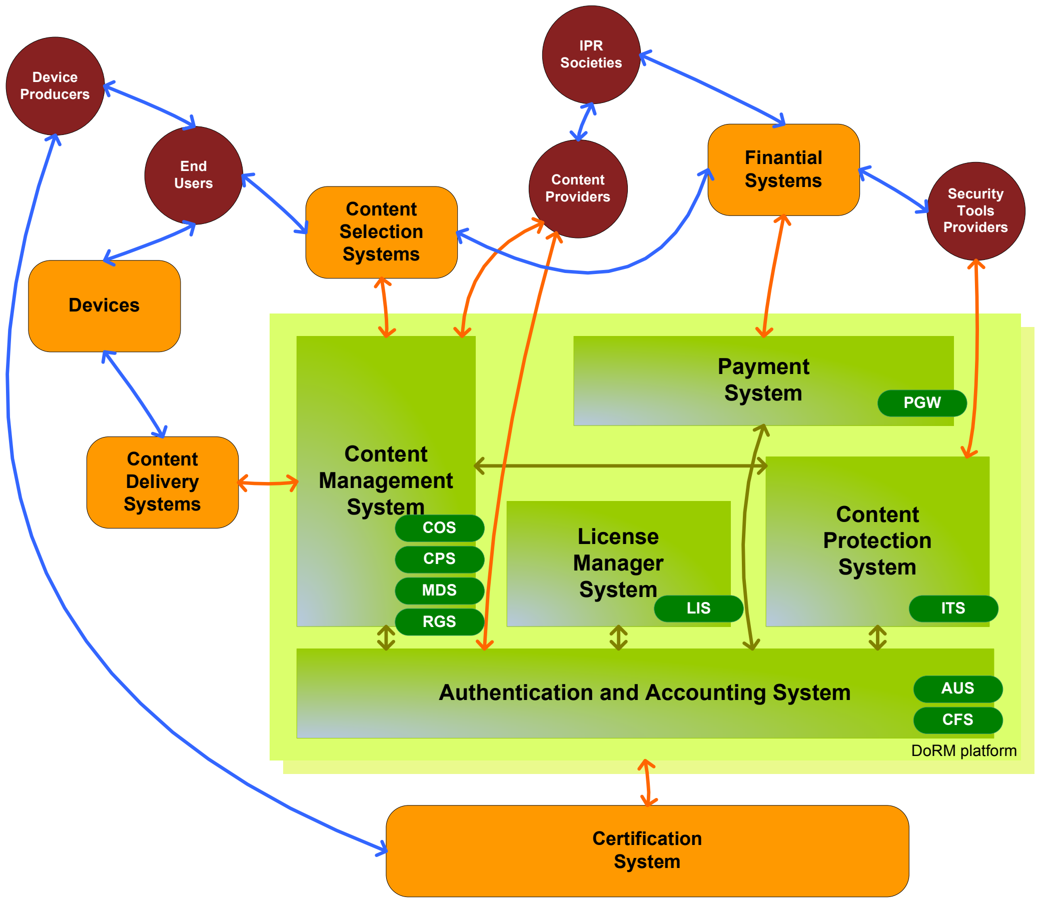


Figure 1 - OpenSDoRM main elements

Technical Architecture Description

OpenSDoRM is, overall, a distributed architecture. Has such, all its elements communicate with each other through the use of a set of well-known interfaces, published here in the form of WSDL. Special security considerations are taken into account assuring the secure communication of every component on the platform.

The whole infrastructure was designed with the concern to be adaptable and applicable to all types of content and business models (download, super-distribution, streaming or even broadcasting).

The diagram below shows the complexity of intercommunication between the elements in the platform. Process and data flows represented in orange are interactions with external entities, while process and data flows represented in green are internal interactions. Each of these interactions is well documented below, in the Services Description section.

External Components & Interfaces

This part will present in detail the components and actors that may interact externally with the OpenSDoRM architecture, namely, the User, the IPMP Tools Provider, the Content Provider, the Payment Infrastructure and the Certification Authority.

- The User represents a person who wishes to consume a piece of content. This content may or may not be protected. However the way to access and display such content may require the use of protected devices, software and licenses. The user will make requests to Open SDRM in order to: identify him, download licenses and play multimedia using a Media Player, embedded or not on a web browser. In a final analysis, the User interaction with OpenSDoRM will always result in one of two things: either the user can play/render the content and enjoy it or he/she can't; being then informed of the reason for this prevention.
- The IPMP Tools Provider is any organization that produces tools and technologies for encryption, scrambling, watermarking and others that can be applied to content protection. These tools will be made available to OpenSDoRM for use in content rights protection. These tools will need to comply with some guidelines. These guidelines and a subscription, are translated into a business relation that must exist between a given Content Provider and the IPMP Tools Provider, since mostly, a given producer and/or distributor of content, may want to choose which type of protection the content will have and, respectively, which tools can be applied to the content and from which supplier.
- The Content Provider is any multimedia content supplier that feeds OpenSDoRM with content and optional metadata. The content can be complex multimedia content that is ready for distribution, or simple content, for example JPEG images, that can be edited and combined with other content. As the mentioned, the MOSES project has addressed MPEG-4 content.
- The Payment Infrastructure facilitates OpenSDoRM e-commerce features by providing services for handling electronic payments. The interface between OpenSDoRM and the Payment Infrastructure is generic and independent of the payment method, allowing therefore a multiplicity of payment systems.
- The Certification Authority is responsible for receiving requests for and issuing credentials to entities. These credentials will be used by entities to authenticate themselves to each other, allowing the establishment of secure and authenticated communication channels between them. All the components in the OpenSDoRM architecture communicate using the channel security provided by the SSL/TLS protocol. This Certification Authority may be internal to OpenSDoRM, and therefore entirely managed by some entity, or it may be an external commercial Entity.

Internal Components & Interfaces

In this part, the internal components of the OpenSDoRM platform and the corresponding interfaces are presented. These components include: Media Application, Media Delivery Server, Commerce Server, Authentication Server, License Server, IPMP Tools Server, Registration Server, Content Preparation Server and the Payment Gateway.

- Content Preparation server (CPS): this server component is responsible for the content preparation. It receives raw content from a specified source or sources and encodes it on a specified format, adds metadata and protects it. Under the MOSES project, content has been encoded in MPEG-4 format, according to some pre-established templates. These templates will allow the creation of MPEG-4 files containing music files in MP3 or AAC format together with some JPEG images about the album and artist.
- Payment Gateway (PGW): is a server component responsible for verifying and validating the payment methods provided by the User for a Commerce Server;

- Commerce server (COS): is a server component responsible for trading the content with the users. Normally, content is chosen via web browser, some very generic metadata might be consulted, information about the price is also available, and especially the content usage conditions might be established.
- Media Delivery server (MDS): is a server component responsible for exchanging pieces of content with the client. This Media Delivery server will implement a specific protocol (download: FTP, HTTP or other; streaming: RTSP or other; broadcast) to exchange protected content with the client Media Application.
- Registration server (RGS): is a server component whose role is to assign unique identifiers to content and to register metadata information for that specific content. This architecture was designed to be as close as possible to ISO standards and therefore, for this unique ID, OpenSDoRM follows the MPEG-21 directives about Digital Item Identification (DII), using a reduced version of the MPEG-21 DII Digital Object Identifiers (DOI).
- Authentication server (AUS): is responsible for authenticating all the internal and external entities to the DRM system. It validates the access rights of all the entities and components in the system working as a SSO point, registering and managing components and users on the system. It uses cryptographic XML credentials to authenticate both components and users in order to authenticate the transactions exchanged between them (XML Encryption and XML Signatures).
- License server (LIS): is a server component responsible for house-keeping the rules associating a user, the content and his/her corresponding access rights. This component will accept connections from authenticated Media Players clients for downloading of licenses, which will be applied to the protected content through an appropriate IPMP tool. The licenses are XML formatted using Open Digital Rights Language (ODRL/OMA profile) or the Rights Expression Language (REL), developed by MPEG-21.
- IPMP tools server (ITS): is the server component responsible for registering new IPMP tools and for receiving authenticated client Media Application requests for the downloading of a specific IPMP tool. It is also responsible for making IPMP tools available to the Content Preparation Server to allow the protection of content.
- Media Application (MPL) This component represents the software that will be used to render the content. This is a generic component with the particularity of being able to display/playback the appropriate content for which the necessary audio/video codec should be available (if this codec is not available it must be downloaded from a remote secure server). This Media Application may work with one or several IPMP tools in order to control how the content is accessed by a particular user. This component works on the client side of the general architecture; however it plays an important role in the DRM functions. The Media Application design is fully compatible with the IPMPX design and took in consideration that content can be exchanged on-line and off-line as well, since it supports the "Tools in Content" functionality where Content, Tools and Licenses can be packaged and distributed i.e. via Bluetooth together to a certain device.

OpenSDoRM Security

The distributed nature of OpenSDoRM was already mentioned. As such, and inherently to present day communications, all communication between each component of the platform will usually take place within insecure networks. Furthermore the components communicate with a text-based protocol. This introduces special needs regarding the security of this communication.

An underlying concept behind the OpenSDoRM platform is the existence of two security layers as displayed below. A first security layer is established at the communication level, which will provide the necessary secure and authenticated communication medium to components to communicate with each other. A second layer is established at the application level, ensuring the security, integrity, authentication and non-repudiation mechanisms needed by the different components.

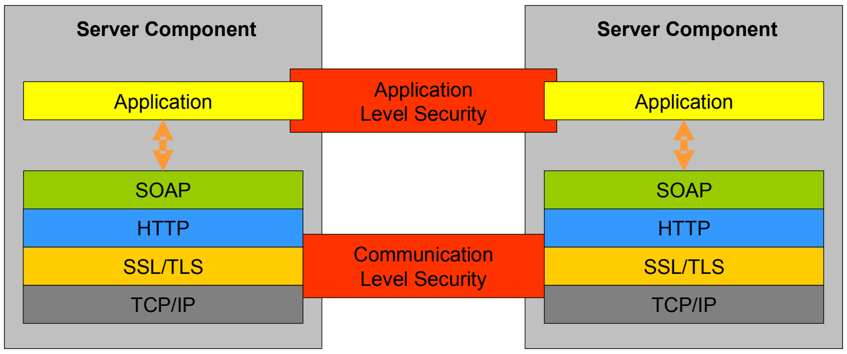


Figure 2 - OpenSDoRM Security Layers

Each of the messages exchanged between the different components share a common structure. The format as shown below is composed of:

- **Component ID**: a 128 bits identifier (generated by an MD5 hashing algorithm) that identifies uniquely this component. This identifier was issued by the AUS;
- **Message Contents**: this is a set of different fields of the message, and typically it is different from component to component;
- **Digital Signature**: the digital signature of the message, to avoid the message contents tampering.

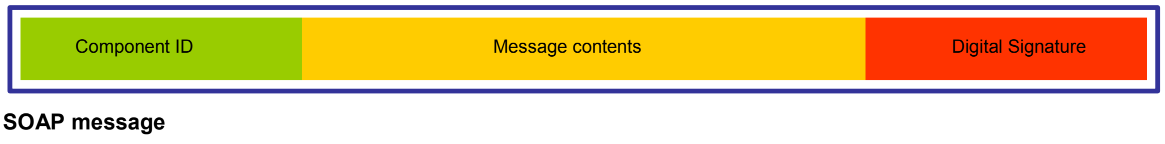


Figure 3 - SOAP Message Structure

Device Identification and Authentication

A device is a piece of hardware and/or software with the capability to play audio-visual content. The device can be used by used only by one user (if we consider for instance a personal mobile phone or a portable MP3 player) or by several users (a set-top box in the case of Interactive TV). Therefore in our platform, there must be two levels of authentication - the device must be authenticated, and each of the users that use the device must also authenticate.

In what concerns to the device identification and authentication, the Device Manufacturer places a digital credential inside the device. This credential is an X509 certificate obtained from a trustworthy Certification Authority (CA). So every Device Manufacturers (DMan) put on their devices, their own certificates. Off course, a first step must occur, that is that each of the DMan must be certified by a ROOT CA (rCA).

So, each DMan have their own device manufacturer certificates:

$Cert_{rCA}^{DMan[1]}$, $Cert_{rCA}^{DMan[2]}$, ..., $Cert_{rCA}^{DMan[n]}$

And each Devices (Dev) also have their own Certificates:

$Cert_{DMan[1]}^{Dev[1]}$, $Cert_{DMan[1]}^{Dev[2]}$, ..., $Cert_{DMan[1]}^{Dev[n]}$

$Cert_{DMan[2]}^{Dev[1]}$, $Cert_{DMan[2]}^{Dev[2]}$, ..., $Cert_{DMan[2]}^{Dev[n]}$

...

$Cert_{DMan[n]}^{Dev[1]}$, $Cert_{DMan[n]}^{Dev[2]}$, ..., $Cert_{DMan[n]}^{Dev[n]}$

In that way, when two devices wish to communicate, or even a device wishes to communicate with any other component, they can trust each other just by presenting their credentials/certificates to each other (performing a mutual authentication). The mutual authentication will succeed if they all share the same rCA.

Therefore for any $Cert_{DMan[n]}^{Dev[n]}$ the authentication will succeed if the issuer of the certificate (Device Manufacturer) have on their certification path/chain a common certificate issued by the same rCA.

Basically what we have is two roles:

1. One major entity (ROOT CA) delegates its trust on the Device Manufacturers, saying to the rest of the world that the Device Manufacturer is trustworthy;
2. The Device Manufacturer puts its trust on the Device supplied to the User.

This is a normal certification mechanism that currently exists and that is widely deployed, especially on the Internet (for instance in SSL/TLS protocol) and using PKI mechanisms.

The mutual authentication mechanisms works in the following manner:

1. Device A (Dev[a]) sends its certificate ($Cert_{DMan[x]}^{Dev[a]}$) to Device B (Dev[b]);
2. Dev[b] sends its certificate to Dev[a] ($Cert_{DMan[y]}^{Dev[b]}$);
3. Dev[b] verifies the following:
   - If there is a common rCA certificate in the Dev[a] certificate chain;
   - If yes, validate the Dev[a] digital signature;
   - The certificate validity date;
4. Dev[a] performs the same validations on Dev[b] certificate;
5. Optionally Dev[a] and Dev[b] can challenge each other, like this:
   - Dev[a] selects a random number, and ciphers it with the Dev[b] public-key;
   - Dev[b] receives it and deciphers it using their private-key. Dev[b] ciphers the result of the previous operation with the public-key of Dev[a];
   - Dev[a] receives the data and deciphers it with the corresponding private-key. If the result is equal to the first selected number, then the mutual authentication succeeds.
6. Conclusion of the mutual authentication.

Therefore, the identification of the device is provided by the certificate that is placed on the device by the Device Manufacturer. This certificate contains also information about a common rCA that enables the trust between different devices from different Device Manufacturers. At the same time, this device certificates are used for device authentication as well.

Server Components Certification

In order to establish the secure transport layer, the software components of the OpenSDoRM architecture, use the SSL/TLS protocol to ensure such functionality. Each of the servers, on which the software components are installed, need to have a X.509 certificate issued by a Certification Authority (CAU). If more than one of the OpenSDoRM components are installed on the same server then such one of this certificates are necessary. The CAU can be operated internally by OpenSDoRM itself or can be an external and commercial one.

In this way, OpenSDoRM can establish an underlying secure and authenticated transport channel that will allow the messages to flow from component to component securely.

- Each component computes a key pair (public and private) , K_{pub}^{Server}, K_{priv}^{Server}, using the RSA algorithm and create Certificate Signing Request (CSR) using its public key and some additional information sending it after to the CAU
- The CAU verifies the CSR validity and issues the X.509 SSL certificate to the appropriate component, $Cert_{X.509}^{Server}$;
- The X.509 SSL certificate is installed and the components can use SSL/TLS to communicate, establishing therefore the secure transport layer.

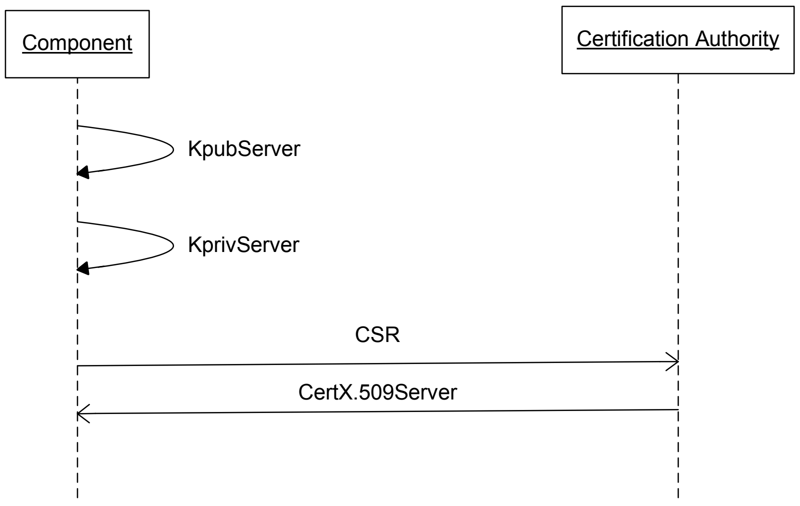


Figure 4 - Components certification process

Registration of Components on OpenSDoRM

The architecture requires that both components and Users on the OpenSDoRM architecture to be registered, in order to establish the Application/Transaction level security.

Concerning components (COS, PGW, RGS, LIS, ITS, CFS, MDS, CPS) those are registered on OpenSDoRM AUS. In order to complete this process the following steps are necessary, during the installation of each of the components:

- Each component computes a key-pair (currently OpenSDoRM uses a 1024 bit length RSA keys, but higher key lengths are also possible): $Kpub^{Component}$, $Kpriv^{Component}$ (respectively the public and private keys);
- The component administrator selects a login and a password, and ciphers the $Kpriv^{Component}$, using AES, with the key (K_{AES}) deduced from the hash of the concatenation of the login and password selected: K_{AES} := MD5(login+password). The ciphered component private key gets then protected from unauthorized usage: $K_{AES}[Kpriv^{Component}]$.
- The component then connects to the AUS and sends some registration information together with the $Kpub^{Component}$.
- AUS verifies the information sent by the component, validates and registers it, and issues a certificate for the component: $Cert_{AUS}^{Component}$. This certificate contains, among other information, a unique identifier of the component and the public key. This certificate is return to the component.

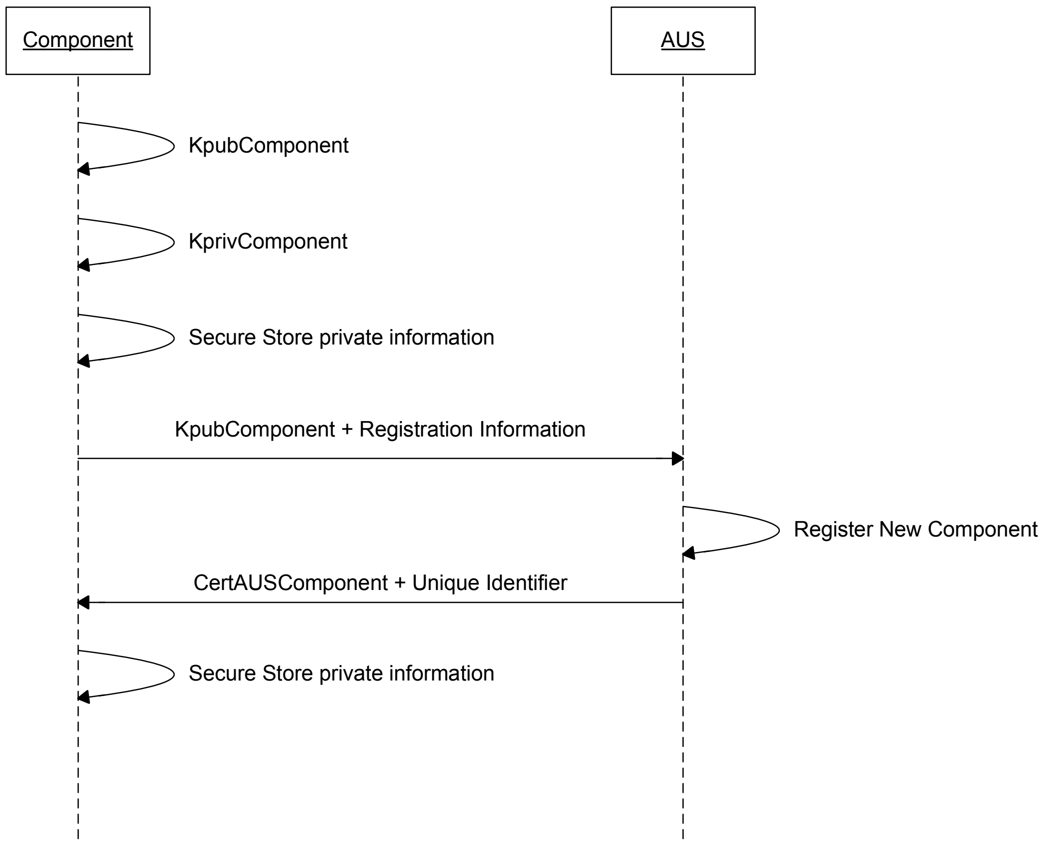

Component
AUS
KpubComponent
KprivComponent
Secure Store private information
KpubComponent + Registration Information
Register New Component
CertAUSComponent + Unique Identifier
Secure Store private information

Figure 5 - Components registration process

With these component certificates, each of the components will be able to establish trust relationships among them and sign and authenticate all the transactions – this establishes then the Application Level security.

User's registration on the OpenSDoRM platform

In OpenSDoRM three components interact directly with external users/entities – MPL, CPS and ITS. These users, respectively Content Users, Content Providers and IPMP Tools Providers are registered on the platform, through the AUS.

Content Providers and IPMP Tools Providers, subscribe respectively on the CPS and ITS, relying on the registration and authentication functionalities of the AUS. Therefore, when a new user subscribes, it provides some personal information, a login and password and requests the registration. The following processes can be described like this:

- The components (ITS and CPS) gather the new registrant information (Info) and request the registration of a new user on the AUS;
- The components build a new message: $SignKpriv^{Component}\{Component_{ID}, Info\}$. This message is send to AUS;
- AUS verifies and validates the message, registering the new User and returning a unique $User_{ID}$ to the component.

Registering a Content User is a more complex process. This is due to the fact that while both Content Providers and IPMP Tool Providers have their information stored on remote servers, Content Users rely on their own platforms to store their data. In order to provide some additional degree of security, OpenSDoRM provides a digital wallet, capable of storing sensitive information such as cryptographic data and licenses in a secure way. The process to register new Content Users can be described in the following steps:

- When the user runs the wallet for the first time, it creates the User a RSA key pair ($Kpriv^{User}$, $Kpub^{User}$) and asks the user to enter a login and a password;
- Using the entered login and password, it creates the secure repository master key: K_{AES} = MD5(login+password), and stores sensitive information (Info) on it: $K_{AES}[Info]$;
- The wallet asks the user to enter some personal data ($Person_{Data}$) and also some payment data (Pay_{Data}) used to charge the user for any commercial content usage;
- The wallet requests the AUS to register a new User, sending all the information ciphered with the AUS $Kpub^{AUS}$: $Kpub^{AUS}[Person_{Data}, Pay_{Data}, KPriv^{User}, Kpub^{User}]$;
- AUS receives the data, deciphers it and registers the User. AUS responds to the Wallet with a new certificate generated for the User: $Cert_{AUS}^{User}$, containing among other information the unique identifier of the User, its public key, the identification of the AUS its signature;
- The wallet stores all the relevant information on the secure repository: $K_{AES}[Cert_{AUS}^{User}]$.

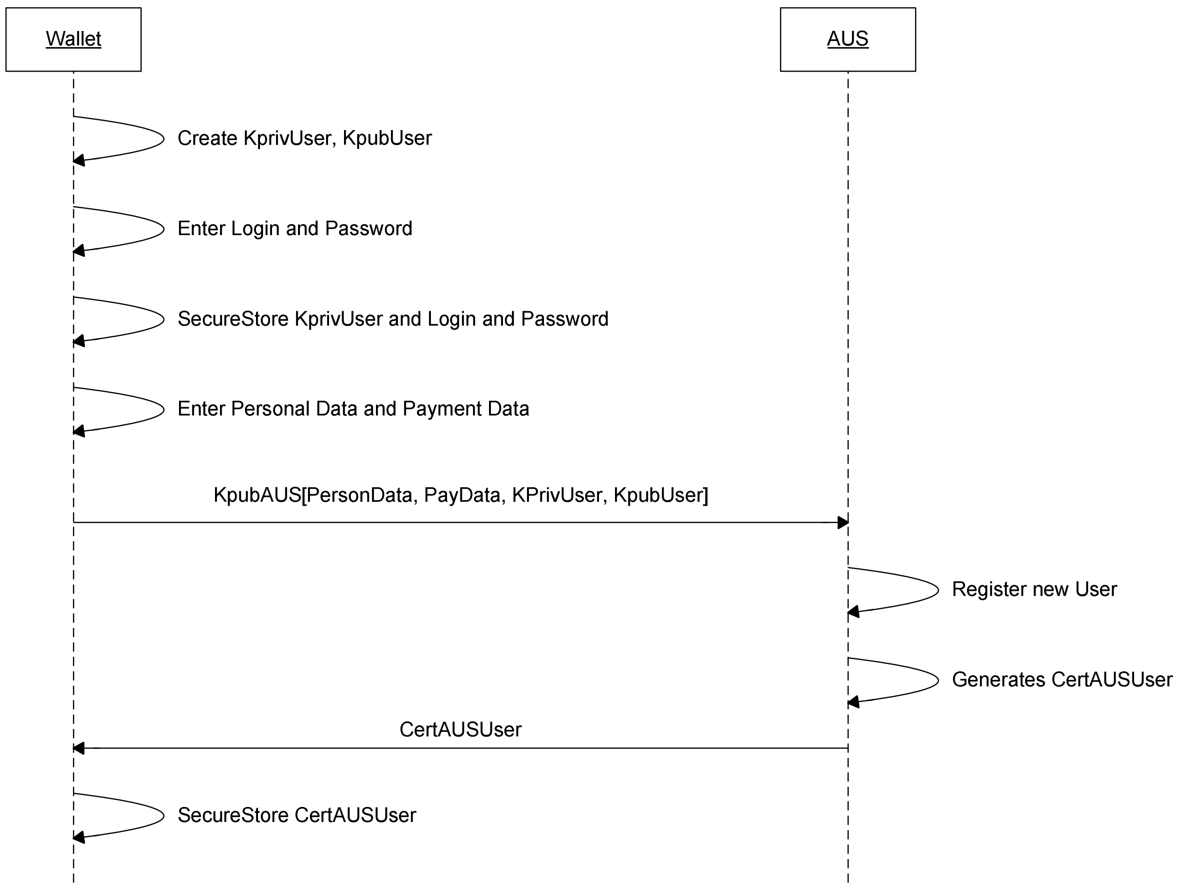

Wallet
AUS
Create KprivUser, KpubUser
Enter Login and Password
SecureStore KprivUser and Login and Password
Enter Personal Data and Payment Data
KpubAUS[PersonData, PayData, KPrivUser, KpubUser]
Register new User
Generates CertAUSUser
CertAUSUser
SecureStore CertAUSUser

Figure 6 - Users registration process

Components message exchange

The process for the components to exchange messages and to verify the authenticity and validity of such messages is composed of the following steps:

- The sender component (CSender) composes a message using the following syntax: $SignKpriv^{CSender}\{CSender_{ID}, Payload, Cert_{AUS}^{CSender}\}$;
- The receiver component (CReceiver) receives the message and verifies the trust on the message. This trustability is assured in the following way:
- CReceiver gets $Cert_{AUS}^{CSender}$ and checks if it was issued by a AUS in which CReceiver trusts.
- This verification can be conducted if CReceiver has also a certificate issued by AUS: $Cert_{AUS}^{CReceiver}$.
- After the trust is established, the message signature can be verified and validated and CReceiver can trust its contents, and also in the component who has sent this message;
- CReceiver can then process the message payload and return its results for the CSender;

- CReceiver returns the following message to CSender: SignKpriv^{CReceiver}{CReceiver$_{ID}$, Results, Cert$_{AUS}$^{CReceiver}}.

This processing flow is displayed in the next picture.

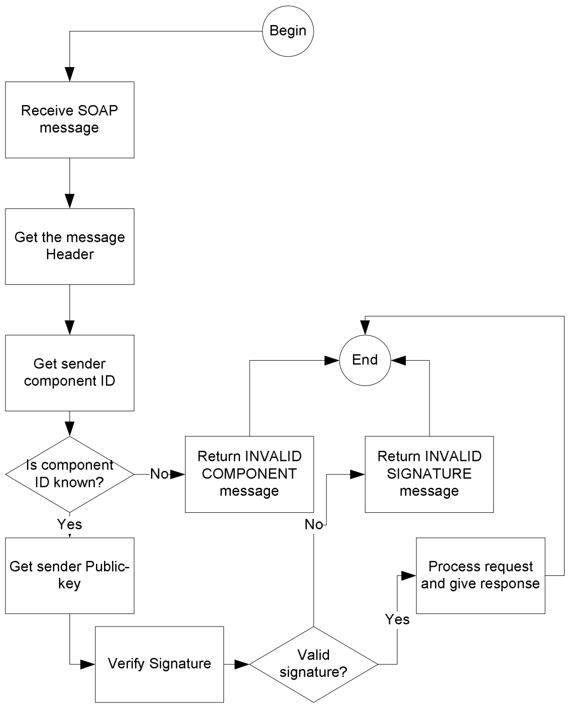


Figure 7 - Message security and integrity verification

Payment information

Payment of content usage is one of the questions that OpenSDoRM also deals and incorporates mechanisms for payment, although the payment method is outside the scope of the OpenSDoRM itself.

To provide this functionality a direct trust relationship must be established between the COS and the PGW. Therefore the COS (that relies on the payment functionalities) needs to subscribe a PGW. The process to subscribe a PGW can be described as the following:

- The COS connects to the AUS and asks the AUS which are the PGW available on the system. COS sends SignKpriv^{COS}{COS$_{ID}$, RequestAvailablePGWs} to AUS;
- AUS verifies the message, and returns an answer to the COS: SignKpriv^{AUS}{<ListOfAvailablePGWs, Cert$_{AUS}$^{PGW}>};
- The COS selects one available PGW and sends to it a subscription request: SignKpriv^{COS}{AUS$_{ID}$, SubscribePGW, Cert$_{AUS}$^{COS}};

PGW receives the request from the COS, validates its request and subscribes the COS. Therefore, this PGW will be used to validate and process payments used by a given User.

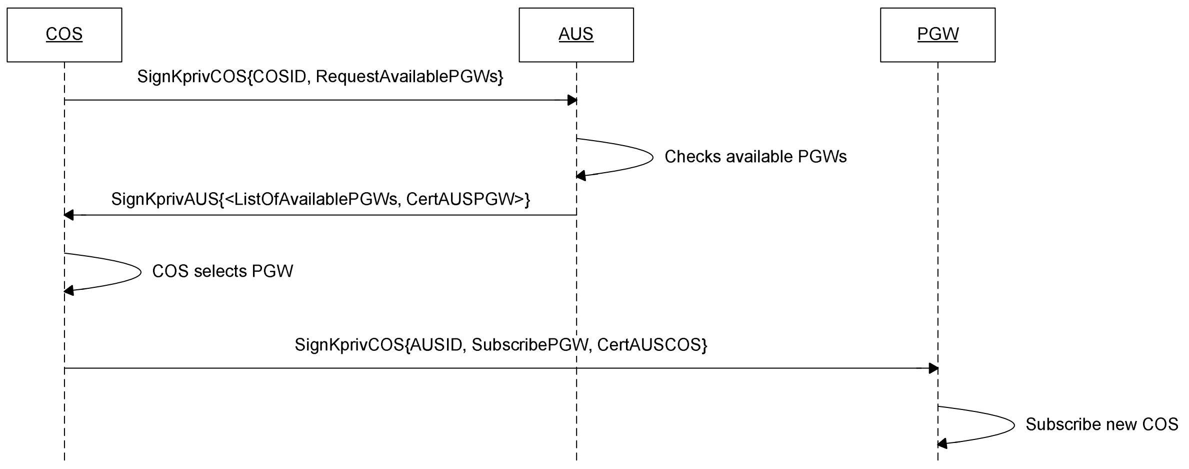

COS
AUS
PGW
SignKprivCOS{COSID, RequestAvailablePGWs}
Checks available PGWs
SignKprivAUS{<ListOfAvailablePGWs, CertAUSPGW>}
COS selects PGW
SignKprivCOS{AUSID, SubscribePGW, CertAUSCOS}
Subscribe new COS

Figure 8 - Payment gateway subscription process

Paying services with OpenSDoRM

Using the payment service provided in OpenSDoRM involves two steps: validating the payment instrument and capturing the payment.

Validating the payment instrument is an important step on the system, since it will allow the COS to be sure that the payment method supplied by the user is authentic and valid, and that the transaction can be conducted without problems. Validating the payment involves the following steps:

- The COS sends information about the payment details, namely information about the User order and the price to pay for it, to AUS: $\text{SignKpriv}^{\text{COS}}\{\text{COS}_{\text{ID}}, \text{U}_{\text{ID}}, \text{PGW}_{\text{ID}}, \text{PayData}\}$;
- AUS verifies and validates the COS request and checks the UID in order to retrieve the appropriate payment method choose by the User upon registration on the AUS. This data is ciphered with the public key of the PGW: $\text{Kpub}_{\text{PGW}}[\text{PaymentClearence}_{\text{U}}]$;
- The AUS returns this information for the COS, signing it: $\text{SignKpriv}_{\text{AUS}}\{\text{Kpub}_{\text{PGW}}[\text{PaymentClearence}_{\text{U}}]\}$;
- This information is then passed by the COS to the PGW, requesting it to validate the payment transaction: $\text{SignKpriv}_{\text{COS}}\{\text{COS}_{\text{ID}}, \text{Kpub}_{\text{PGW}}[\text{PaymentClearence}_{\text{U}}]\}$;
- PGW validates the message and deciphers the User payment clearance, using this information to communicate to the corresponding Payment Infratructure, validating it. After, the PGW returns the result of the payment validation to the COS: $\text{SignKpriv}_{\text{PGW}}\{\text{PGW}_{\text{ID}}, \text{Transaction}_{\text{ID}}\}$;

This concludes the payment method validation on the PGW. This process assures the COS that the services he is supplying to the User will be in fact charged.

The second step in the payment procedure involves the payment capture. This process requires that first a payment capture has occurred and second that the COS possess a valid $\text{Transaction}_{\text{ID}}$. The capture process can be described in the following:

- COS sends a message to PGW: $\text{SignKPriv}_{\text{COS}}\{\text{COS}_{\text{ID}}, \text{Transaction}_{\text{ID}}\}$;

- PGW validates the message and verifies the $Transaction_{ID}$, in order to evaluate if that transaction is in fact pending, and processes the payment;
- PGW returns and a result status to the COS: $SignKPriv_{PGW}\{PGW_{ID}, Transaction_{ID}, Result\}$.

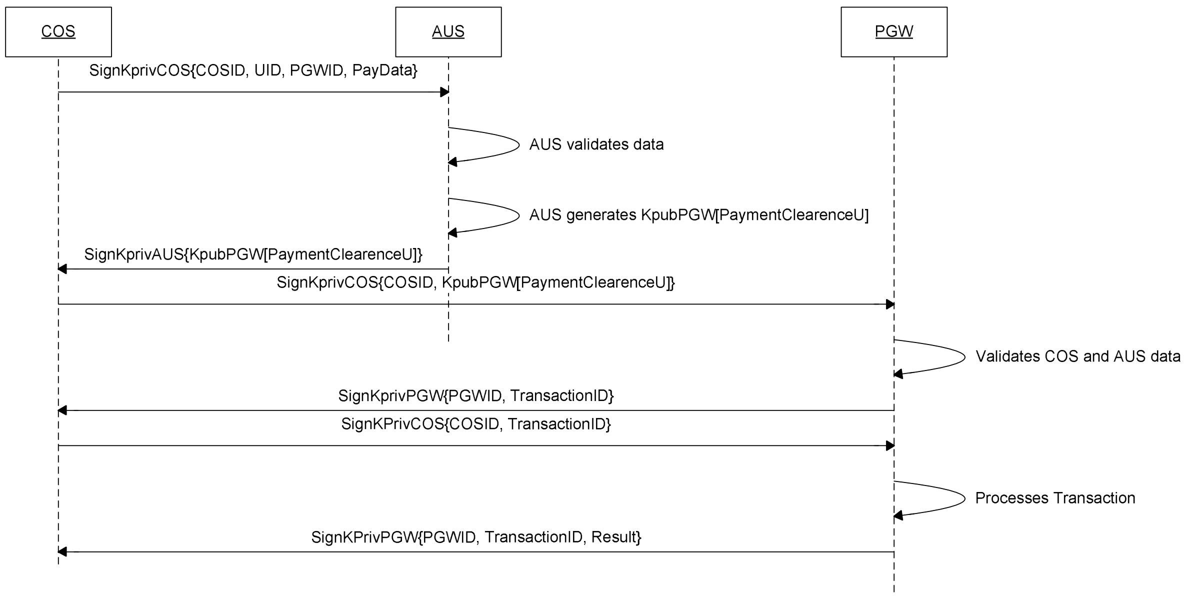

COS
AUS
PGW
SignKprivCOS{COSID, UID, PGWID, PayData}
AUS validates data
AUS generates KpubPGW[PaymentClearenceU]
SignKprivAUS{KpubPGW[PaymentClearenceU]}
SignKprivCOS{COSID, KpubPGW[PaymentClearenceU]}
Validates COS and AUS data
SignKprivPGW{PGWID, TransactionID}
SignKPrivCOS{COSID, TransactionID}
Processes Transaction
SignKPrivPGW{PGWID, TransactionID, Result}

Figure 9 - Payment process

License Production

One of the major functionalities of the OpenSDoRM platform resides on the fact that it can control the way the Users access and use the content protected by the platform. This process is ensured by the production of licenses. These are later applied on the content of the user on the client player by the appropriate set of IPMP tools. These licenses are produced and stored securely by the LIS, according to the choices made by the User and after the payment has been performed. The process can be described in the following steps:

- The User selects a set of available conditions, that allow him to define the usage conditions (rights) of the content the User wants to access;
- COS sends a message to the LIS, requesting the production of a new license, for a specific content, and for a given User: $SignKpriv_{COS}\{U_{ID}, Content_{ID}, LicenseConditions, Cert_{AUS}^{COS}\}$;
- LIS receives the request, verifies it and validates it. LIS generated the license using the appropriate language and parameters, contacting after the AUS for ciphering the license data for the User: $SignKpriv_{LIS}\{License\}$;
- AUS receives the data, retrieves the $Kpub_{User}$ and ciphers the received data: $Kpub_{U}[License]$, returning it afterwards to the LIS: $SignKpriv_{AUS}\{Kpub_{User}[License]\}$;
- LIS stores $Kpub_{User}[License]$;

License Download

When the User tries to access the content on the client side the player verifies that a license is needed to access the content. The player contacts the wallet to try to obtain the required licenses and corresponding keys to access the content. This process can be described in the following steps:

- The player contacts the wallet to obtain the license for the ContentID and UserID;
- The wallet checks on its secure repository if a license for that specific ContentID is already there. If that is true than this license is returned for the player in order for the content to be deciphered and accessed, controlled by a set of IPMP tools. If the wallet doesn't contain the license, it will request it from the LIS: $\text{SignKpriv}_U\{\text{Cert}_{AUS}{}^{User}, \text{Content}_{ID}\}$;
- LIS receives the data, validates it and retrieves the license from the database, passing it to the wallet: $\text{SignKpriv}_{LIS}\{\text{Kpub}_{User}[\text{License}], \text{Cert}_{AUS}{}^{LIS}\}$;
- The wallet receives the data from the LIS, validates the message and deciphers the license that is passed to the player. Also the license is stored on the wallet secure repository for future accesses.

The downloaded license is kept in the LIS for later crash recovery in an event of failure and later expiration checks.

License Expiry

Depending on the rights specified, a license will eventually expire. Rights such as a play count or a validity period may restrict the access to content to a certain number of times or to a certain time frame. The state of the license is maintained within the digital wallet. Upon expiration, for example when a play count reaches zero, the wallet automatically checks at the LIS for a new license for that particular content. If there is no license available and the user wants to continue with the consumption of the content he has purchase a new License as described before. The LIS also applies an internal checking algorithm to manage the state of its licenses. Licenses that expired will be removed from the LIS.

Services Description

The following section specifies to a greater extent all the services made available by each component on the platform.

AUS – Authentication Service

Description

The Authentication Service (AUS) is responsible in the OpenSDoRM system for two major authentication operations:

1. Authenticate the different OpenSDoRM components;
2. Authenticate the users that use the different components.

Function List Overview

The following table resumes the list of functionalities that are provided by the AUS. Annexed to the final of this document there is a WSDL description of the service.

Function Name	*Description*
AUSrequestModifyUserSubscription	This function is used to modify the data used by a User to subscribe to the Autentication Service.
AUSrequestUserSubscription	This function is used to subscribe a new User on the Authentication Service.
AUSrequestAuthentication	This function is used to validate and authenticate a user.
AUSrequestDeleteUserSubscription	This function is used to un-subscribe/remove a user from the system.
AUSrequestComponentSubscription	This function is used to register a new component on the system. This action will make a component valid in the OpenSDoRM context.
AUSrequestUserInfo	This function is used to request public user information from the Authentication Service component.
AUSrequestUserPaymentInfo	This function is used to request payment information for a given user from the AUS.
AUSrequestListOfPGW	This function is used to get a list of valid OpenSDoRM billing components from AUS.
AUSrequestMutualValidation	This function is an ah-hoc one, specific to the implementation of OpenSDoRM in music-4you (MOSES IST project).
AUSrequestWalletVerification	This function is used to validate if a given user has or not a Wallet.

Function List Detailed

AUSrequestModifyUserSubscription		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
name	String	The name of the user.
address	String	The address of the user.

email_address	String	The email address of the user that is being registered.
authentication_type	String	This field identifies the type of authentication that is going to be used by the user. It can be: LOGINPASSWORD or PUBLICKEY.
uid	String	This is the unique identifier of the user.
username	String	This is the username choosen by the user.
password	String	This is the password choosen by the user.
arbitrary_data	String	This is some additional arbitrary data that can be registered with the user.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

AUSrequestUserSubscription		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
name	String	The name of the user that is being registered.
address	String	The address of the user that is being registered.
email_address	String	The email address of the user that is being registered.
authentication_type	String	This field identifies the type of authentication that is going to be used by the user. It can be: LOGINPASSWORD or PUBLICKEY.
uid	String	This is the unique identifier of the user.
username	String	This is a username choosen by the user that is registering.
password	String	This is a password choosen by the user that is registering.

arbitrary_data	String	This is some additional arbitrary data that can be registered with the user.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
user_id	String	The user_id that the system has generated for this user

AUSrequestAuthentication		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
authentication_type	String	This field identifies the type of authentication that is going to be used by the user. It can be: LOGINPASSWORD or PUBLICKEY.
username	String	This is the username choosen by the user.
password	String	This is the password choosen by the user.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
user_id	String	The user_id that the system has associated with this user

AUSrequestDeleteUserSubscription		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.

signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
uid	String	This is the unique identifier of the user.
username	String	This is a username choosen by the user.
password	String	This is a password choosen by the user.
arbitrary_data	String	This is some additional arbitrary data.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

AUSrequestComponentSubscription		
Input Parameters		
public_key	String	This is the public key of the component, needed to register the component in the system.
password	String	This is a password choosen by the administrator that is registering the new component.
arbitrary_data	String	This is some additional arbitrary data that can be registered with the component.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
certificate	String	Certificate of the component freshly generated by the Authentication Service.

AUSrequestUserInfo		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
authentication_type	String	This field identifies the type of

		authentication that is going to be used by the user. It can be: LOGINPASSWORD or PUBLICKEY.
uid	String	This is the unique identifier of the user.
username	String	This is a username choosen by the user.
password	String	This is a password choosen by the user.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
name	String	Information about the user.
address	String	Information about the user.
email_address	String	Information about the user.
uid	String	This is the unique identifier of the user.
other_data	String	This is some additional arbitrary data that could have been registered with the user.
certificate	String	Certificate of the user, generated by the system at registration time.

AUSrequestUserPaymentInfo		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
pgw_identification	String	Identification of the PGW that will process the payment, in order to allow only this PGW to access the payment information (public key encryption).
user_identification	String	The user that started the transaction.
pvalue	String	The value of the transaction.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function

		succeeded, if it starts by –ERR something wrong has occurred.
payclearer	String	Data to be sent to the PGW.

AUSrequestListOfPGW		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
authentication_type	String	This field identifies the type of authentication that is going to be used by the user. It can be: LOGINPASSWORD or PUBLICKEY.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
list_of_pgw	String	A list of PGW services available on the system.

AUSrequestMutualValidation		
Input Parameters		
uid	String	This is the unique system identifier of the user.
login	String	This is the user's human readable login.
hash	String	HASH of the user's username and password.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

AUSrequestWalletVerification
Input Parameters

uid	String	This is the unique identifier of the user.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

Examples

CFS – Configuration Service

Description

The CFS is responsible for keeping centralized configuration and setup data, essential to the correct functioning of the system as a whole.

Function List Overview

The following table resumes the list of functionalities that are provided by the CFS. Annexed to the final of this document there is a WSDL description of the service.

Function Name	***Description***
CFSrequestServerLocation	This function is used for a component to know the location of any other component that is part of the OpenSDoRM system.
CFSrequestLocationStorage	This function is used to request a new server storage location. This is used to add a new OpenSDoRM server on a central point where all the other services can contact.
CFSrequestLocationDelete	This function is used to remove a component location from the service.

Function List Detailed

CFSrequestServerLocation		
Input Parameters		
id	String	This is the unique identifier of the component that is being searched.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

location	String	The URL of the component.

CFSrequestLocationStorage		
Input Parameters		
id	String	This is the unique identifier of the component that is being added to the system.
location	String	The URL of the component.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

CFSrequestLocationDelete		
Input Parameters		
id	String	This is the unique identifier of the component that is being deleted.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

Examples

ITS – Protection Tools Service

Description

The ITS is responsible for managing the usage of protection tools throughout the system, from content producer to the end user.

Function List Overview

The following table resumes the list of functionalities that are provided by the ITS. Annexed to the final of this document there is a WSDL description of the service.

Function Name	***Description***
ITSrequestIPMPToolsList	This function is used to request a list of protection tools from the ITS.

Function Name	***Description***
ITSrequestIPMPToolDownload	This function is used to get the download location of a given protection tool.
ITSaddNewIPMPTool	This function is used to add a new protection tool to the ITS.
ITSrequestIPMPToolDetails	This function is used to get the details about a specific protection tool.

Function List Detailed

ITSrequestIPMPToolsList		
Input Parameters		
-	-	-
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
ipmp_tools_list	String	A list containg the IPMP tools available.

ITSrequestIPMPToolDownload		
Input Parameters		
ipmp_tool_id	String	The unique identifier of the tool.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
ipmp_tool_url	String	The location URL of the tool, for download.

ITSaddNewIPMPTool		
Input Parameters		
ipmptoolid	String	This is the unique identifier of the IPMP tool.
ipmptoolurl	String	The location of the tool.
ipmptooldesc	String	A textual description of the tool.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something

		wrong has occurred.

ITSrequestIPMPToolDetails		
Input Parameters		
ipmptoolid	String	This is the unique identifier of the tool.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
ipmptoolid	String	This is the unique identifier of the tool.
ipmptoolurl	String	The location of the tool.
ipmptooldesc	String	A textual description of the tool.

Examples

LIS – License Service

Description

This service is responsible for managing all the information related with the licenses on the OpenSDoRM system. This service is capable of handling the following functions:

- Storing the cryptographic keys which are associated with the content which is protected;
- Producing the licenses according to a set of conditions established by the content provider and the final user, for a specific content and specific user;
- Support the license download for a given user and content identifier.

The service is independent of the type of Rights Expression Language (REL) that is used to express the rights on the license. OpenSDoRM uses a template-like system that supports several types of licenses.

Function List Overview

The following table resumes the list of functionalities that are provided by the LIS. Annexed to the final of this document there is a WSDL description of the service.

Function Name	***Description***
LISrequestContentKeyStore	This function is used to store content keys on the LIS. The server can store one or more keys on the service for a given content.
LISrequestLicenseList	This function is used to list licenses from the LIS, that is according to a given criteria.

Function Name	***Description***
LISrequestLicenseDownload	This function is used to request the download of a license that was created previously on the system.
LISrequestLicenseDownloadSpecial	This function is used to request a license from the server.
LISrequestLicenseDelete	This function is requested to delete licenses from the license system.
LISrequestLicenseCreation	This function is used to request the creation of a license on the system.
LISrequestLicenseUpdate	This function is used to update the license parameters of a license that is already present on the system.
LISrequestLicensePassing	This function is used to pass licenses between users.

Function List Detailed

The following tables detail each of the functions which were identified previously on the previous table.

LISrequestContentKeyStore		
Input Parameters		
key	String	This is the content key, or list of keys that were used to protect the content
cid	String	This is the unique identifier of the content
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

LISrequestLicenseList		
Input Parameters		
uid	String	This is the unique identifier of the user that is requesting to list the licenses.
cid	String	This is the unique identifier of the content
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function

		succeeded, if it starts by –ERR something wrong has occurred.
list_of_licenses	String	This is a XML formatted message that contains all the licenses that a specific user (uid) has for a given content (cid).

LISrequestLicenseDownload		
Input Parameters		
uid	String	This is the unique identifier of the user that is requesting the operation.
cid	String	This is the unique identifier of the content
authData	String	This field contains an XML formatted message containing data that is used to authenticate the user.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
license	String	This is a XML formatted message that contains the requested license.

LISrequestLicenseDownloadSpecial		
Input Parameters		
uid	String	This is the unique identifier of the user that is requesting the operation.
cid	String	This is the unique identifier of the content.
authData	String	This field contains an XML formatted message containing data that is used to authenticate the user.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
license	String	This is a XML formatted message that contains the requested license.

LISrequestLicenseDelete		
Input Parameters		
uid	String	This is the unique identifier of the user that is requesting the operation.

cid	String	This is the unique identifier of the content.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

LISrequestLicenseCreation		
Input Parameters		
uid	String	This is the unique identifier of the user that is requesting the operation.
cid	String	This is the unique identifier of the content.
licdata	String	This is a formatted message containing data that composes the license
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

LISrequestLicenseUpdate		
Input Parameters		
uid	String	This is the unique identifier of the user that is requesting the operation.
cid	String	This is the unique identifier of the content.
licdata	String	This field a formatted message containing data that composes the license
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

LISrequestLicensePassing		
Input Parameters		
uid_source	String	The unique identifier of the user that wants to pass the license
uid_target	String	The unique identifier of the user that will receive the license
cid	String	The unique identifier of the content
rights	String	A formatted message containing the rights

		associated with the license
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

Examples

MDS - Media Delivery Service

Description

This service is responsible for "knowing" the whereabouts of any given content protected by the platform.

Function List Overview

The following table resumes the list of functionalities that are provided by the MDS. Annexed to the final of this document there is a WSDL description of the service.

Function Name	*Description*
MDSrequestContentStorage	This function is used to store a given content location (not the content itself) on the media delivery system.
MDSrequestContentDelivery	This function is used to notify the system that a specific content was requested by a given user.
MDSrequestContentDownload	This function is used to notify the system that a specific content was downloaded from the system.
MDSrequestContentDelete	This function is used to notify the system that a given content was deleted from the system.
MDSrequestDirectoryList	This function is used to list the available content.
MDSrequestUpdate	This function is used to update the details of the content on the system.

Function List Detailed

MDSrequestContentStorage		
Input Parameters		
cid	String	The unique identifier of the content.
filetype	String	The filetype of the content.
protocol	String	The protocol used to access the content's

		location.
location	String	The content location URL.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

MDSrequestContentDelivery		
Input Parameters		
user_id	String	The unique identifier of the user that is requesting the operation.
content_id	String	This is the unique identifier of the content
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

MDSrequestContentDownload		
Input Parameters		
user_id	String	The unique identifier of the user that is requesting the operation.
content_id	String	This is the unique identifier of the content
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
url	String	URL of the content ready for downloading.

MDSrequestContentDelete		
Input Parameters		
content_id	String	This is the unique identifier of the content.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

MDSrequestDirectoryList

Input Parameters		
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
clist	String	

MDSrequestUpdate		
Input Parameters		
cid	String	The unique identifier of the content.
filetype	String	The filetype of the content.
protocol	String	The protocol used to access the content's location.
location		The content location URL.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

Examples

PGW - Billing Service

Description

This service handles all payment necessities of the system.

Function List Overview

The following table resumes the list of functionalities that are provided by the PGW. Annexed to the final of this document there is a WSDL description of the service.

Function Name	***Description***
PGWrequestPaymentClearence	This function is used to request a payment clearance from the billing system, clearing the payment instrument that was used by the user.
PGWrequestPaymentCapture	This function is used to capture the payment from a previously cleared transaction.

Function Name	*Description*
PGWrequestCOSSubscribe	This function is used for a new content store (or any other content selection module) to subscribe a billing system.

Function List Detailed

PGWrequestPaymentClearence		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
data	String	Data relevant to the payment operation.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
transaction_number	String	The transaction number.

PGWrequestPaymentCapture		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
tid	String	The transaction's unique identifier
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

transaction_number	String	The transaction number.

PGWrequestCOSSubscribe		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
certificate	String	The certificate of the COS.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
result_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
cert	String	

Examples

RGS – Registration Service

Description

The Registration Service is responsible for assigning unique content identifiers and for registering specific metadata associated with a given content.

Function List Overview

The following table resumes the list of functionalities that are provided by the RGS. Annexed to the final of this document there is a WSDL description of the service.

Function Name	***Description***
RGSrequestContentRegistration	This function is used to register a new content on the system.
RGSrequestMetadataRegistration	This function is used to register metadata that is associated with a specific content.
RGSrequestListAvailableContent	This function is used to list the available content on the system that matches with specific criteria.

Function Name	***Description***
RGSrequestListMetadata	This function is used to list the available metadata associated with content on the system that matches with specific criteria.

Function List Detailed

RGSrequestContentRegistration		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
hash	String	The HASH value of the original content.
file	String	A preview file of the content.
additional_data	String	This is some additional data that can be registered with the user.
aus_cert	String	The AUS certificate of the requesting entity.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
status_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
content_id	String	The content’s unique identifier assigned by the RGS.

RGSrequestMetadataRegistration		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
content_id	String	The unique identifier of the content.
metadata	String	Metadata associated with the content
aus_cert	String	The AUS certificate of the requesting entity.

signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
status_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.

RGSrequestListAvailableContent		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
criteria	String	SQL formatted query to access available content.
aus_cert	String	The AUS certificate of the requesting entity.
signature	String	This is the signature of the message, generated by the component requesting this message.
Output Parameters		
status_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
content	String	An XML formatted message with the content's details.

RGSrequestListMetadata		
Input Parameters		
identification	String	This is the unique identifier that is associated with the component that is requesting this function.
signature_algorithm_identifier	String	A string that identifies the signature algorithm that is used to sign the messages.
criteria	String	SQL formatted query to access available metadata.
aus_cert	String	The AUS certificate of the requesting entity.
signature	String	This is the signature of the message, generated by the component requesting

		this message.
Output Parameters		
status_message	String	This is the resulting message of the service. If the message starts by +OK the function succeeded, if it starts by –ERR something wrong has occurred.
content_id	String	The content's unique identifier
metadata		The metadata associated with the content.

Examples

Annex – WSDL descriptions of all the services

AUS – Authentication Service

WSDL location:	D:\WWW\opensdrm\wsdl\ausws.wsdl
targetnamespace:	**http://www.adetti.pt/opensdrmws**

services	bindings	porttypes	messages
opensdrmws	opensdrmwsBinding	opensdrmwsPortType	AUSrequestAuthenticationRequest
			AUSrequestAuthenticationResponse
			AUSrequestComponentSubscriptionRequest
			AUSrequestComponentSubscriptionResponse
			AUSrequestDeleteUserSubscriptionRequest
			AUSrequestDeleteUserSubscriptionResponse
			AUSrequestListOfPGWRequest
			AUSrequestListOfPGWResponse
			AUSrequestModifyUserSubscriptionRequest
			AUSrequestModifyUserSubscriptionResponse
			AUSrequestMutualValidationRequest
			AUSrequestMutualValidationResponse
			AUSrequestUserInfoRequest
			AUSrequestUserInfoResponse
			AUSrequestUserPaymentInfoRequest
			AUSrequestUserPaymentInfoResponse
			AUSrequestUserSubscriptionRequest

service **opensdrmws**

ports **opensdrmwsPort**

binding tns:opensdrmwsBinding

extensibility <soap:address location="http://localhost/opensdrm/AUS/AUS.ws.php"/>

source
```
<service name="opensdrmws">
 <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
  <soap:address location="http://localhost/opensdrm/AUS/AUS.ws.php"/>
 </port>
</service>
```

binding **opensdrmwsBinding**

type tns:opensdrmwsPortType

extensibility <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

operations

AUSrequestModifyUserSubscription

extensibility <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestModifyUserSubscription" style="rpc"/>

input <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

output <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

AUSrequestUserSubscription

extensibility <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestUserSubscription" style="rpc"/>

input <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

output <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

AUSrequestAuthentication

extensibility <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestAuthentication" style="rpc"/>

input <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

output <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

AUSrequestDeleteUserSubscription

extensibility	<soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestDeleteUserSubscription" style="rpc"/>
input	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
output	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

AUSrequestComponentSubscription

extensibility	<soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestComponentSubscription" style="rpc"/>
input	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
output	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

AUSrequestUserInfo

extensibility	<soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestUserInfo" style="rpc"/>
input	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
output	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

AUSrequestUserPaymentInfo

extensibility	<soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestUserPaymentInfo" style="rpc"/>
input	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
output	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

AUSrequestListOfPGW

extensibility	<soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestListOfPGW" style="rpc"/>
input	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
output	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

AUSrequestMutualValidation

extensibility	<soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestMutualValidation" style="rpc"/>
input	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
output	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

AUSrequestWalletVerification

extensibility	<soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestWalletVerification" style="rpc"/>
input	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
output	<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>

used by Service **opensdrmws** in Port **opensdrmwsPort**

source
```
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="AUSrequestModifyUserSubscription">
    <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestModifyUserSubscription" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="AUSrequestUserSubscription">
    <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestUserSubscription" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="AUSrequestAuthentication">
    <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestAuthentication" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="AUSrequestDeleteUserSubscription">
    <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestDeleteUserSubscription" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="AUSrequestComponentSubscription">
    <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestComponentSubscription" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="AUSrequestUserInfo">
    <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestUserInfo" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="AUSrequestUserPaymentInfo">
    <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestUserPaymentInfo" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="AUSrequestListOfPGW">
    <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestListOfPGW" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="AUSrequestMutualValidation">
    <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestMutualValidation" style="rpc"/>
    <input>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="AUSrequestWalletVerification">
  <soap:operation soapAction="http://localhost/opensdrm/AUS/AUS.ws.php/AUSrequestWalletVerification"
style="rpc"/>
  <input>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
</binding>
```

porttype **opensdrmwsPortType**

operations

AUSrequestModifyUserSubscription

input	**tns:AUSrequestModifyUserSubscriptionRequest**
output	**tns:AUSrequestModifyUserSubscriptionResponse**

AUSrequestUserSubscription

input	**tns:AUSrequestUserSubscriptionRequest**
output	**tns:AUSrequestUserSubscriptionResponse**

AUSrequestAuthentication

input	**tns:AUSrequestAuthenticationRequest**
output	**tns:AUSrequestAuthenticationResponse**

AUSrequestDeleteUserSubscription

input	**tns:AUSrequestDeleteUserSubscriptionRequest**
output	**tns:AUSrequestDeleteUserSubscriptionResponse**

AUSrequestComponentSubscription

input	**tns:AUSrequestComponentSubscriptionRequest**
output	**tns:AUSrequestComponentSubscriptionResponse**

AUSrequestUserInfo

input	**tns:AUSrequestUserInfoRequest**
output	**tns:AUSrequestUserInfoResponse**

AUSrequestUserPaymentInfo

input	tns:AUSrequestUserPaymentInfoRequest
output	tns:AUSrequestUserPaymentInfoResponse

AUSrequestListOfPGW

input	tns:AUSrequestListOfPGWRequest
output	tns:AUSrequestListOfPGWResponse

AUSrequestMutualValidation

input	tns:AUSrequestMutualValidationRequest
output	tns:AUSrequestMutualValidationResponse

AUSrequestWalletVerification

input	tns:AUSrequestWalletVerificationRequest
output	tns:AUSrequestWalletVerificationResponse

used by binding opensdrmwsBinding

source
```
<portType name="opensdrmwsPortType">
 <operation name="AUSrequestModifyUserSubscription">
  <input message="tns:AUSrequestModifyUserSubscriptionRequest"/>
  <output message="tns:AUSrequestModifyUserSubscriptionResponse"/>
 </operation>
 <operation name="AUSrequestUserSubscription">
  <input message="tns:AUSrequestUserSubscriptionRequest"/>
  <output message="tns:AUSrequestUserSubscriptionResponse"/>
 </operation>
 <operation name="AUSrequestAuthentication">
  <input message="tns:AUSrequestAuthenticationRequest"/>
  <output message="tns:AUSrequestAuthenticationResponse"/>
 </operation>
 <operation name="AUSrequestDeleteUserSubscription">
  <input message="tns:AUSrequestDeleteUserSubscriptionRequest"/>
  <output message="tns:AUSrequestDeleteUserSubscriptionResponse"/>
 </operation>
 <operation name="AUSrequestComponentSubscription">
  <input message="tns:AUSrequestComponentSubscriptionRequest"/>
  <output message="tns:AUSrequestComponentSubscriptionResponse"/>
 </operation>
 <operation name="AUSrequestUserInfo">
  <input message="tns:AUSrequestUserInfoRequest"/>
  <output message="tns:AUSrequestUserInfoResponse"/>
 </operation>
 <operation name="AUSrequestUserPaymentInfo">
  <input message="tns:AUSrequestUserPaymentInfoRequest"/>
  <output message="tns:AUSrequestUserPaymentInfoResponse"/>
 </operation>
```

```
  <operation name="AUSrequestListOfPGW">
    <input message="tns:AUSrequestListOfPGWRequest"/>
    <output message="tns:AUSrequestListOfPGWResponse"/>
  </operation>
  <operation name="AUSrequestMutualValidation">
    <input message="tns:AUSrequestMutualValidationRequest"/>
    <output message="tns:AUSrequestMutualValidationResponse"/>
  </operation>
  <operation name="AUSrequestWalletVerification">
    <input message="tns:AUSrequestWalletVerificationRequest"/>
    <output message="tns:AUSrequestWalletVerificationResponse"/>
  </operation>
</portType>
```

message **AUSrequestModifyUserSubscriptionRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

name

type **xsd:string**

address

type **xsd:string**

email_address

type **xsd:string**

authentication_type

type **xsd:string**

uid

type **xsd:string**

username

type **xsd:string**

password

type **xsd:string**

arbitrary_data

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestModifyUserSubscription**

source
```
<message name="AUSrequestModifyUserSubscriptionRequest">
 <part name="identification" type="xsd:string"/>
 <part name="signature_algorithm_identifier" type="xsd:string"/>
 <part name="name" type="xsd:string"/>
 <part name="address" type="xsd:string"/>
 <part name="email_address" type="xsd:string"/>
 <part name="authentication_type" type="xsd:string"/>
 <part name="uid" type="xsd:string"/>
 <part name="username" type="xsd:string"/>
 <part name="password" type="xsd:string"/>
 <part name="arbitrary_data" type="xsd:string"/>
 <part name="signature" type="xsd:string"/>
</message>
```

message **AUSrequestModifyUserSubscriptionResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestModifyUserSubscription**

source
```
<message name="AUSrequestModifyUserSubscriptionResponse">
 <part name="result_message" type="xsd:string"/>
</message>
```

message **AUSrequestUserSubscriptionRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

name

type **xsd:string**

address

type **xsd:string**

email_address

type **xsd:string**

authentication_type

type **xsd:string**

username

type **xsd:string**

password

type **xsd:string**

public_key

type **xsd:string**

arbitrary_data

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestUserSubscription**

source
```
<message name="AUSrequestUserSubscriptionRequest">
  <part name="identification" type="xsd:string"/>
  <part name="signature_algorithm_identifier" type="xsd:string"/>
  <part name="name" type="xsd:string"/>
  <part name="address" type="xsd:string"/>
  <part name="email_address" type="xsd:string"/>
  <part name="authentication_type" type="xsd:string"/>
  <part name="username" type="xsd:string"/>
  <part name="password" type="xsd:string"/>
  <part name="public_key" type="xsd:string"/>
  <part name="arbitrary_data" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **AUSrequestUserSubscriptionResponse**

parts **result_message**

type **xsd:string**

user_id

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestUserSubscription**

source
```
<message name="AUSrequestUserSubscriptionResponse">
```

```
  <part name="result_message" type="xsd:string"/>
  <part name="user_id" type="xsd:string"/>
</message>
```

message **AUSrequestAuthenticationRequest**

parts

identification

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

authentication_type

type **xsd:string**

username

type **xsd:string**

password

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestAuthentication**

source

```
<message name="AUSrequestAuthenticationRequest">
  <part name="identification" type="xsd:string"/>
  <part name="signature_algorithm_identifier" type="xsd:string"/>
  <part name="authentication_type" type="xsd:string"/>
  <part name="username" type="xsd:string"/>
  <part name="password" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **AUSrequestAuthenticationResponse**

parts

result_message

type **xsd:string**

user_id

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestAuthentication**

source

```
<message name="AUSrequestAuthenticationResponse">
```

```
  <part name="result_message" type="xsd:string"/>
  <part name="user_id" type="xsd:string"/>
</message>
```

message **AUSrequestDeleteUserSubscriptionRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

uid

type **xsd:string**

username

type **xsd:string**

password

type **xsd:string**

arbitrary_data

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestDeleteUserSubscription**

source

```
<message name="AUSrequestDeleteUserSubscriptionRequest">
  <part name="identification" type="xsd:string"/>
  <part name="signature_algorithm_identifier" type="xsd:string"/>
  <part name="uid" type="xsd:string"/>
  <part name="username" type="xsd:string"/>
  <part name="password" type="xsd:string"/>
  <part name="arbitrary_data" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **AUSrequestDeleteUserSubscriptionResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestDeleteUserSubscription**

source
```
<message name="AUSrequestDeleteUserSubscriptionResponse">
  <part name="result_message" type="xsd:string"/>
</message>
```

message **AUSrequestComponentSubscriptionRequest**

parts **arbitrary_data**

type **xsd:string**

public_key

type **xsd:string**

password

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestComponentSubscription**

source
```
<message name="AUSrequestComponentSubscriptionRequest">
  <part name="arbitrary_data" type="xsd:string"/>
  <part name="public_key" type="xsd:string"/>
  <part name="password" type="xsd:string"/>
</message>
```

message **AUSrequestComponentSubscriptionResponse**

parts **result_message**

type **xsd:string**

certificate

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestComponentSubscription**

source
```
<message name="AUSrequestComponentSubscriptionResponse">
  <part name="result_message" type="xsd:string"/>
  <part name="certificate" type="xsd:string"/>
</message>
```

message **AUSrequestUserInfoRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

authentication_type

type **xsd:string**

uid

type **xsd:string**

username

type **xsd:string**

password

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestUserInfo**

source

```
<message name="AUSrequestUserInfoRequest">
 <part name="identification" type="xsd:string"/>
 <part name="signature_algorithm_identifier" type="xsd:string"/>
 <part name="authentication_type" type="xsd:string"/>
 <part name="uid" type="xsd:string"/>
 <part name="username" type="xsd:string"/>
 <part name="password" type="xsd:string"/>
 <part name="signature" type="xsd:string"/>
</message>
```

message **AUSrequestUserInfoResponse**

parts **result_message**

type **xsd:string**

uid

type **xsd:string**

name

type **xsd:string**

address

type **xsd:string**

email

type **xsd:string**

other_data_xml

type **xsd:string**

certificate

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestUserInfo**

source
```
<message name="AUSrequestUserInfoResponse">
  <part name="result_message" type="xsd:string"/>
  <part name="uid" type="xsd:string"/>
  <part name="name" type="xsd:string"/>
  <part name="address" type="xsd:string"/>
  <part name="email" type="xsd:string"/>
  <part name="other_data_xml" type="xsd:string"/>
  <part name="certificate" type="xsd:string"/>
</message>
```

message **AUSrequestUserPaymentInfoRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

pgw_identification

type **xsd:string**

user_identification

type **xsd:string**

pvalue

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestUserPaymentInfo**

source
```
<message name="AUSrequestUserPaymentInfoRequest">
  <part name="identification" type="xsd:string"/>
  <part name="signature_algorithm_identifier" type="xsd:string"/>
  <part name="pgw_identification" type="xsd:string"/>
  <part name="user_identification" type="xsd:string"/>
  <part name="pvalue" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **AUSrequestUserPaymentInfoResponse**

parts **result_message**

type **xsd:string**

payclearer

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestUserPaymentInfo**

source
```
<message name="AUSrequestUserPaymentInfoResponse">
  <part name="result_message" type="xsd:string"/>
  <part name="payclearer" type="xsd:string"/>
</message>
```

message **AUSrequestListOfPGWRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

authentication_type

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestListOfPGW**

source
```
<message name="AUSrequestListOfPGWRequest">
  <part name="identification" type="xsd:string"/>
  <part name="signature_algorithm_identifier" type="xsd:string"/>
  <part name="authentication_type" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **AUSrequestListOfPGWResponse**

parts **result_message**

type **xsd:string**

list_of_pgw

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestListOfPGW**

source
```
<message name="AUSrequestListOfPGWResponse">
  <part name="result_message" type="xsd:string"/>
  <part name="list_of_pgw" type="xsd:string"/>
</message>
```

message **AUSrequestMutualValidationRequest**

parts **uid**

type **xsd:string**

login

type **xsd:string**

hash

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestMutualValidation**

source
```
<message name="AUSrequestMutualValidationRequest">
  <part name="uid" type="xsd:string"/>
  <part name="login" type="xsd:string"/>
  <part name="hash" type="xsd:string"/>
</message>
```

message **AUSrequestMutualValidationResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestMutualValidation**

source
```
<message name="AUSrequestMutualValidationResponse">
  <part name="result_message" type="xsd:string"/>
</message>
```

message **AUSrequestWalletVerificationRequest**

parts **uid**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **AUSrequestWalletVerification**

source
```
<message name="AUSrequestWalletVerificationRequest">
  <part name="uid" type="xsd:string"/>
</message>
```

message **AUSrequestWalletVerificationResponse**

parts	**result_message** type **xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **AUSrequestWalletVerification**

source

```
<message name="AUSrequestWalletVerificationResponse">
  <part name="result_message" type="xsd:string"/>
</message>
```

CFS – Configuration Service

WSDL location:	**D:\WWW\opensdrm\wsdl\cfsws.wsdl**
targetnamespace:	**http://www.adetti.pt/opensdrmws**

services	bindings	porttypes	messages
opensdrmws	**opensdrmwsBinding**	**opensdrmwsPortType**	**CFSrequestLocationDeleteRequest**
			CFSrequestLocationDeleteResponse
			CFSrequestLocationStorageRequest
			CFSrequestLocationStorageResponse
			CFSrequestServerLocationRequest
			CFSrequestServerLocationResponse

service **opensdrmws**

ports	**opensdrmwsPort** **binding** **tns:opensdrmwsBinding** **extensibility** `<soap:address location="http://localhost/opensdrm/CFS/CFS.ws.php"/>`

source

```
<service name="opensdrmws">
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm/CFS/CFS.ws.php"/>
  </port>
</service>
```

binding **opensdrmwsBinding**

type **tns:opensdrmwsPortType**

extensibility	<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
operations	**CFSrequestServerLocation**

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/CFS/CFS.ws.php/CFSrequestServerLocation" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

CFSrequestLocationStorage

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/CFS/CFS.ws.php/CFSrequestLocationStorage" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

CFSrequestLocationDelete

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/CFS/CFS.ws.php/CFSrequestLocationDelete" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

used by Service opensdrmws in Port opensdrmwsPort

source
```
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="CFSrequestServerLocation">
  <soap:operation soapAction="http://localhost/opensdrm/CFS/CFS.ws.php/CFSrequestServerLocation" style="rpc"/>
  <input>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="CFSrequestLocationStorage">
  <soap:operation soapAction="http://localhost/opensdrm/CFS/CFS.ws.php/CFSrequestLocationStorage" style="rpc"/>
  <input>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
```

```
  <operation name="CFSrequestLocationDelete">
    <soap:operation soapAction="http://localhost/opensdrm/CFS/CFS.ws.php/CFSrequestLocationDelete" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
</binding>
```

porttype **opensdrmwsPortType**

operations

CFSrequestServerLocation

input	**tns:CFSrequestServerLocationRequest**
output	**tns:CFSrequestServerLocationResponse**

CFSrequestLocationStorage

input	**tns:CFSrequestLocationStorageRequest**
output	**tns:CFSrequestLocationStorageResponse**

CFSrequestLocationDelete

input	**tns:CFSrequestLocationDeleteRequest**
output	**tns:CFSrequestLocationDeleteResponse**

used by binding **opensdrmwsBinding**

source

```
<portType name="opensdrmwsPortType">
  <operation name="CFSrequestServerLocation">
    <input message="tns:CFSrequestServerLocationRequest"/>
    <output message="tns:CFSrequestServerLocationResponse"/>
  </operation>
  <operation name="CFSrequestLocationStorage">
    <input message="tns:CFSrequestLocationStorageRequest"/>
    <output message="tns:CFSrequestLocationStorageResponse"/>
  </operation>
  <operation name="CFSrequestLocationDelete">
    <input message="tns:CFSrequestLocationDeleteRequest"/>
    <output message="tns:CFSrequestLocationDeleteResponse"/>
  </operation>
</portType>
```

message **CFSrequestServerLocationRequest**

parts **id**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **CFSrequestServerLocation**

source
```
<message name="CFSrequestServerLocationRequest">
 <part name="id" type="xsd:string"/>
</message>
```

message **CFSrequestServerLocationResponse**

parts **result_message**

type **xsd:string**

location

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **CFSrequestServerLocation**

source
```
<message name="CFSrequestServerLocationResponse">
 <part name="result_message" type="xsd:string"/>
 <part name="location" type="xsd:string"/>
</message>
```

message **CFSrequestLocationStorageRequest**

parts **id**

type **xsd:string**

location

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **CFSrequestLocationStorage**

source
```
<message name="CFSrequestLocationStorageRequest">
 <part name="id" type="xsd:string"/>
 <part name="location" type="xsd:string"/>
</message>
```

message **CFSrequestLocationStorageResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **CFSrequestLocationStorage**

source
```
<message name="CFSrequestLocationStorageResponse">
 <part name="result_message" type="xsd:string"/>
```

```
</message>
```

message **CFSrequestLocationDeleteRequest**

parts	**id** type **xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **CFSrequestLocationDelete**

source	`<message name="CFSrequestLocationDeleteRequest">`
	`<part name="id" type="xsd:string"/>`
	`</message>`

message **CFSrequestLocationDeleteResponse**

parts	**result_message** type **xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **CFSrequestLocationDelete**

source	`<message name="CFSrequestLocationDeleteResponse">`
	`<part name="result_message" type="xsd:string"/>`
	`</message>`

ITS – Protection Tools Service

WSDL location:	**D:\WWW\opensdrm\wsdl\itsws.wsdl**
targetnamespace:	**http://www.adetti.pt/opensdrmws**

services	bindings	porttypes	messages
opensdrmws	**opensdrmwsBinding**	**opensdrmwsPortType**	**ITSaddNewIPMPToolRequest**
			ITSaddNewIPMPToolResponse
			ITSrequestIPMPToolDetailsRequest
			ITSrequestIPMPToolDetailsResponse
			ITSrequestIPMPToolDownloadRequest
			ITSrequestIPMPToolDownloadResponse
			ITSrequestIPMPToolsListRequest
			ITSrequestIPMPToolsListResponse

service **opensdrmws**

ports **opensdrmwsPort**

binding	**tns:opensdrmwsBinding**
extensibility	**<soap:address location="http://localhost/opensdrm/ITS/ITS.ws.php"/>**

source

```
<service name="opensdrmws">
 <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
  <soap:address location="http://localhost/opensdrm/ITS/ITS.ws.php"/>
 </port>
</service>
```

binding **opensdrmwsBinding**

type **tns:opensdrmwsPortType**

extensibility <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

operations

ITSrequestIPMPToolsList

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/ITS/ITS.ws.php/ITSrequestIPMPToolsList" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

ITSrequestIPMPToolDownload

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/ITS/ITS.ws.php/ITSrequestIPMPToolDownload" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

ITSaddNewIPMPTool

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/ITS/ITS.ws.php/ITSaddNewIPMPTool" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

ITSrequestIPMPToolDetails

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/ITS/ITS.ws.php/ITSrequestIPMPToolDetails" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

used by Service **opensdrmws** in Port **opensdrmwsPort**

source

```
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="ITSrequestIPMPToolsList">
```

```
    <soap:operation soapAction="http://localhost/opensdrm/ITS/ITS.ws.php/ITSrequestIPMPToolsList" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="ITSrequestIPMPToolDownload">
    <soap:operation soapAction="http://localhost/opensdrm/ITS/ITS.ws.php/ITSrequestIPMPToolDownload" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="ITSaddNewIPMPTool">
    <soap:operation soapAction="http://localhost/opensdrm/ITS/ITS.ws.php/ITSaddNewIPMPTool" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="ITSrequestIPMPToolDetails">
    <soap:operation soapAction="http://localhost/opensdrm/ITS/ITS.ws.php/ITSrequestIPMPToolDetails" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
</binding>
```

porttype **opensdrmwsPortType**

operations **ITSrequestIPMPToolsList**

input **tns:ITSrequestIPMPToolsListRequest**
output **tns:ITSrequestIPMPToolsListResponse**

ITSrequestIPMPToolDownload

input **tns:ITSrequestIPMPToolDownloadRequest**
output **tns:ITSrequestIPMPToolDownloadResponse**

ITSaddNewIPMPTool

input **tns:ITSaddNewIPMPToolRequest**
output **tns:ITSaddNewIPMPToolResponse**

ITSrequestIPMPToolDetails

input **tns:ITSrequestIPMPToolDetailsRequest**
output **tns:ITSrequestIPMPToolDetailsResponse**

used by binding **opensdrmwsBinding**

source
```
<portType name="opensdrmwsPortType">
  <operation name="ITSrequestIPMPToolsList">
    <input message="tns:ITSrequestIPMPToolsListRequest"/>
    <output message="tns:ITSrequestIPMPToolsListResponse"/>
  </operation>
  <operation name="ITSrequestIPMPToolDownload">
    <input message="tns:ITSrequestIPMPToolDownloadRequest"/>
    <output message="tns:ITSrequestIPMPToolDownloadResponse"/>
  </operation>
  <operation name="ITSaddNewIPMPTool">
    <input message="tns:ITSaddNewIPMPToolRequest"/>
    <output message="tns:ITSaddNewIPMPToolResponse"/>
  </operation>
  <operation name="ITSrequestIPMPToolDetails">
    <input message="tns:ITSrequestIPMPToolDetailsRequest"/>
    <output message="tns:ITSrequestIPMPToolDetailsResponse"/>
  </operation>
</portType>
```

message **ITSrequestIPMPToolsListRequest**

parts

used by PortType **opensdrmwsPortType** in Operation **ITSrequestIPMPToolsList**

source
```
<message name="ITSrequestIPMPToolsListRequest"/>
```

message **ITSrequestIPMPToolsListResponse**

parts **result_message**

type **xsd:string**

ipmp_tools_list

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **ITSrequestIPMPToolsList**

source
```
<message name="ITSrequestIPMPToolsListResponse">
 <part name="result_message" type="xsd:string"/>
 <part name="ipmp_tools_list" type="xsd:string"/>
</message>
```

message **ITSrequestIPMPToolDownloadRequest**

parts **ipmp_tool_id**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **ITSrequestIPMPToolDownload**

source
```
<message name="ITSrequestIPMPToolDownloadRequest">
 <part name="ipmp_tool_id" type="xsd:string"/>
</message>
```

message **ITSrequestIPMPToolDownloadResponse**

parts **result_message**

type **xsd:string**

ipmp_tool_url

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **ITSrequestIPMPToolDownload**

source
```
<message name="ITSrequestIPMPToolDownloadResponse">
 <part name="result_message" type="xsd:string"/>
 <part name="ipmp_tool_url" type="xsd:string"/>
</message>
```

message **ITSaddNewIPMPToolRequest**

parts **ipmptoolid**

type **xsd:string**

ipmptoolurl

type **xsd:string**

ipmptooldesc

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **ITSaddNewIPMPTool**

source
```
<message name="ITSaddNewIPMPToolRequest">
  <part name="ipmptoolid" type="xsd:string"/>
  <part name="ipmptoolurl" type="xsd:string"/>
  <part name="ipmptooldesc" type="xsd:string"/>
</message>
```

message **ITSaddNewIPMPToolResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **ITSaddNewIPMPTool**

source
```
<message name="ITSaddNewIPMPToolResponse">
  <part name="result_message" type="xsd:string"/>
</message>
```

message **ITSrequestIPMPToolDetailsRequest**

parts **ipmptoolid**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **ITSrequestIPMPToolDetails**

source
```
<message name="ITSrequestIPMPToolDetailsRequest">
  <part name="ipmptoolid" type="xsd:string"/>
</message>
```

message **ITSrequestIPMPToolDetailsResponse**

parts **result_message**

type **xsd:string**

ipmptoolid

type **xsd:string**

ipmptoolurl

type **xsd:string**

ipmptooldesc

type	**xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **ITSrequestIPMPToolDetails**

source
```
<message name="ITSrequestIPMPToolDetailsResponse">
  <part name="result_message" type="xsd:string"/>
  <part name="ipmptoolid" type="xsd:string"/>
  <part name="ipmptoolurl" type="xsd:string"/>
  <part name="ipmptooldesc" type="xsd:string"/>
</message>
```

LIS – License Service

WSDL location: **D:\WWW\opensdrm\wsdl\lisws.wsdl**

targetnamespace: **http://www.adetti.pt/opensdrmws**

services	bindings	porttypes	messages
opensdrmws	**opensdrmwsBinding**	**opensdrmwsPortType**	**LISrequestContentKeyStoreRequest**
			LISrequestContentKeyStoreResponse
			LISrequestLicenseCreationRequest
			LISrequestLicenseCreationResponse
			LISrequestLicenseDeleteRequest
			LISrequestLicenseDeleteResponse
			LISrequestLicenseDownloadRequest
			LISrequestLicenseDownloadResponse
			LISrequestLicenseDownloadSpecialRequest
			LISrequestLicenseDownloadSpecialResponse
			LISrequestLicenseListRequest
			LISrequestLicenseListResponse
			LISrequestLicensePassingRequest
			LISrequestLicensePassingResponse
			LISrequestLicenseUpdateRequest
			LISrequestLicenseUpdateResponse

service **opensdrmws**

ports **opensdrmwsPort**

binding	**tns:opensdrmwsBinding**
extensibility	**<soap:address location="http://localhost/opensdrm/LIS/LIS.ws.php"/>**

source
```
<service name="opensdrmws">
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm/LIS/LIS.ws.php"/>
  </port>
</service>
```

binding **opensdrmwsBinding**

type **tns:opensdrmwsPortType**

extensibility `<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>`

operations

LISrequestContentKeyStore

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestContentKeyStore" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

LISrequestLicenseList

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseList" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

LISrequestLicenseDownload

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseDownload" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

LISrequestLicenseDownloadSpecial

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseDownloadSpecial" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

LISrequestLicenseDelete

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseDelete" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

LISrequestLicenseCreation

extensibility	`<soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseCreation" style="rpc"/>`
input	`<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>`
output	`<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>`

LISrequestLicenseUpdate

extensibility	`<soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseUpdate" style="rpc"/>`
input	`<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>`
output	`<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>`

LISrequestLicensePassing

extensibility	`<soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicensePassing" style="rpc"/>`
input	`<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>`
output	`<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>`

used by Service **opensdrmws** in Port **opensdrmwsPort**

source
```
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="LISrequestContentKeyStore">
    <soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestContentKeyStore" style="rpc"/>
    <input>
      <soap:body            use="encoded"            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body            use="encoded"            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="LISrequestLicenseList">
    <soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseList" style="rpc"/>
    <input>
      <soap:body            use="encoded"            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body            use="encoded"            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="LISrequestLicenseDownload">
    <soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseDownload" style="rpc"/>
    <input>
      <soap:body            use="encoded"            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="LISrequestLicenseDownloadSpecial">
  <soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseDownloadSpecial"
style="rpc"/>
  <input>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="LISrequestLicenseDelete">
  <soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseDelete" style="rpc"/>
  <input>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="LISrequestLicenseCreation">
  <soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseCreation" style="rpc"/>
  <input>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="LISrequestLicenseUpdate">
  <soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicenseUpdate" style="rpc"/>
  <input>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
    <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="LISrequestLicensePassing">
  <soap:operation soapAction="http://localhost/opensdrm/LIS/LIS.ws.php/LISrequestLicensePassing" style="rpc"/>
  <input>
   <soap:body           use="encoded"          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body           use="encoded"          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
</binding>
```

porttype **opensdrmwsPortType**

operations

LISrequestContentKeyStore

input	**tns:LISrequestContentKeyStoreRequest**
output	**tns:LISrequestContentKeyStoreResponse**

LISrequestLicenseList

input	**tns:LISrequestLicenseListRequest**
output	**tns:LISrequestLicenseListResponse**

LISrequestLicenseDownload

input	**tns:LISrequestLicenseDownloadRequest**
output	**tns:LISrequestLicenseDownloadResponse**

LISrequestLicenseDownloadSpecial

input	**tns:LISrequestLicenseDownloadSpecialRequest**
output	**tns:LISrequestLicenseDownloadSpecialResponse**

LISrequestLicenseDelete

input	**tns:LISrequestLicenseDeleteRequest**
output	**tns:LISrequestLicenseDeleteResponse**

LISrequestLicenseCreation

input	**tns:LISrequestLicenseCreationRequest**
output	**tns:LISrequestLicenseCreationResponse**

LISrequestLicenseUpdate

input	**tns:LISrequestLicenseUpdateRequest**
output	**tns:LISrequestLicenseUpdateResponse**

LISrequestLicensePassing

input **tns:LISrequestLicensePassingRequest**
output **tns:LISrequestLicensePassingResponse**

used by binding **opensdrmwsBinding**

source

```
<portType name="opensdrmwsPortType">
  <operation name="LISrequestContentKeyStore">
    <input message="tns:LISrequestContentKeyStoreRequest"/>
    <output message="tns:LISrequestContentKeyStoreResponse"/>
  </operation>
  <operation name="LISrequestLicenseList">
    <input message="tns:LISrequestLicenseListRequest"/>
    <output message="tns:LISrequestLicenseListResponse"/>
  </operation>
  <operation name="LISrequestLicenseDownload">
    <input message="tns:LISrequestLicenseDownloadRequest"/>
    <output message="tns:LISrequestLicenseDownloadResponse"/>
  </operation>
  <operation name="LISrequestLicenseDownloadSpecial">
    <input message="tns:LISrequestLicenseDownloadSpecialRequest"/>
    <output message="tns:LISrequestLicenseDownloadSpecialResponse"/>
  </operation>
  <operation name="LISrequestLicenseDelete">
    <input message="tns:LISrequestLicenseDeleteRequest"/>
    <output message="tns:LISrequestLicenseDeleteResponse"/>
  </operation>
  <operation name="LISrequestLicenseCreation">
    <input message="tns:LISrequestLicenseCreationRequest"/>
    <output message="tns:LISrequestLicenseCreationResponse"/>
  </operation>
  <operation name="LISrequestLicenseUpdate">
    <input message="tns:LISrequestLicenseUpdateRequest"/>
    <output message="tns:LISrequestLicenseUpdateResponse"/>
  </operation>
  <operation name="LISrequestLicensePassing">
    <input message="tns:LISrequestLicensePassingRequest"/>
    <output message="tns:LISrequestLicensePassingResponse"/>
  </operation>
</portType>
```

message **LISrequestContentKeyStoreRequest**

parts **key**

type **xsd:string**

cid

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestContentKeyStore**

source
```
<message name="LISrequestContentKeyStoreRequest">
  <part name="key" type="xsd:string"/>
  <part name="cid" type="xsd:string"/>
</message>
```

message **LISrequestContentKeyStoreResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestContentKeyStore**

source
```
<message name="LISrequestContentKeyStoreResponse">
  <part name="result_message" type="xsd:string"/>
</message>
```

message **LISrequestLicenseListRequest**

parts **uid**

type **xsd:string**

cid

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestLicenseList**

source
```
<message name="LISrequestLicenseListRequest">
  <part name="uid" type="xsd:string"/>
  <part name="cid" type="xsd:string"/>
</message>
```

message **LISrequestLicenseListResponse**

parts **result_message**

type **xsd:string**

list_of_licenses

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestLicenseList**

source
```
<message name="LISrequestLicenseListResponse">
  <part name="result_message" type="xsd:string"/>
  <part name="list_of_licenses" type="xsd:string"/>
```

```
</message>
```

message **LISrequestLicenseDownloadRequest**

parts **uid**

type **xsd:string**

cid

type **xsd:string**

authData

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestLicenseDownload**

source
```
<message name="LISrequestLicenseDownloadRequest">
 <part name="uid" type="xsd:string"/>
 <part name="cid" type="xsd:string"/>
 <part name="authData" type="xsd:string"/>
</message>
```

message **LISrequestLicenseDownloadResponse**

parts **result_message**

type **xsd:string**

license

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestLicenseDownload**

source
```
<message name="LISrequestLicenseDownloadResponse">
 <part name="result_message" type="xsd:string"/>
 <part name="license" type="xsd:string"/>
</message>
```

message **LISrequestLicenseDownloadSpecialRequest**

parts **uid**

type **xsd:string**

cid

type **xsd:string**

authData

	type **xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **LISrequestLicenseDownloadSpecial**

source
```
<message name="LISrequestLicenseDownloadSpecialRequest">
  <part name="uid" type="xsd:string"/>
  <part name="cid" type="xsd:string"/>
  <part name="authData" type="xsd:string"/>
</message>
```

message **LISrequestLicenseDownloadSpecialResponse**

parts **result_message**

type **xsd:string**

license

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestLicenseDownloadSpecial**

source
```
<message name="LISrequestLicenseDownloadSpecialResponse">
  <part name="result_message" type="xsd:string"/>
  <part name="license" type="xsd:string"/>
</message>
```

message **LISrequestLicenseDeleteRequest**

parts **uid**

type **xsd:string**

cid

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestLicenseDelete**

source
```
<message name="LISrequestLicenseDeleteRequest">
  <part name="uid" type="xsd:string"/>
  <part name="cid" type="xsd:string"/>
</message>
```

message **LISrequestLicenseDeleteResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestLicenseDelete**

source
```
<message name="LISrequestLicenseDeleteResponse">
```

```
 <part name="result_message" type="xsd:string"/>
</message>
```

message **LISrequestLicenseCreationRequest**

parts	**uid**		
		type	**xsd:string**
	cid		
		type	**xsd:string**
	licdata		
		type	**xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **LISrequestLicenseCreation**		

source
```
<message name="LISrequestLicenseCreationRequest">
 <part name="uid" type="xsd:string"/>
 <part name="cid" type="xsd:string"/>
 <part name="licdata" type="xsd:string"/>
</message>
```

message **LISrequestLicenseCreationResponse**

parts	**result_message**		
		type	**xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **LISrequestLicenseCreation**		

source
```
<message name="LISrequestLicenseCreationResponse">
 <part name="result_message" type="xsd:string"/>
</message>
```

message **LISrequestLicenseUpdateRequest**

parts	**uid**		
		type	**xsd:string**
	cid		
		type	**xsd:string**
	licdata		
		type	**xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **LISrequestLicenseUpdate**		

source
```
<message name="LISrequestLicenseUpdateRequest">
  <part name="uid" type="xsd:string"/>
  <part name="cid" type="xsd:string"/>
  <part name="licdata" type="xsd:string"/>
</message>
```

message **LISrequestLicenseUpdateResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestLicenseUpdate**

source
```
<message name="LISrequestLicenseUpdateResponse">
  <part name="result_message" type="xsd:string"/>
</message>
```

message **LISrequestLicensePassingRequest**

parts **uid_src**

type **xsd:string**

uid_target

type **xsd:string**

cid

type **xsd:string**

rights

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **LISrequestLicensePassing**

source
```
<message name="LISrequestLicensePassingRequest">
  <part name="uid_src" type="xsd:string"/>
  <part name="uid_target" type="xsd:string"/>
  <part name="cid" type="xsd:string"/>
  <part name="rights" type="xsd:string"/>
</message>
```

message **LISrequestLicensePassingResponse**

parts **result_message**

type **xsd:string**

used by	PortType **opensdrmwsPortType** in Operation **LISrequestLicensePassing**
source	

```
<message name="LISrequestLicensePassingResponse">
  <part name="result_message" type="xsd:string"/>
</message>
```

MDS – Media Delivery Service

WSDL location:	**D:\WWW\opensdrm\wsdl\mdsws.wsdl**
targetnamespace:	**http://www.adetti.pt/opensdrmws**

services	bindings	porttypes	messages
opensdrmws	**opensdrmwsBinding**	**opensdrmwsPortType**	**MDSrequestContentDeleteRequest**
			MDSrequestContentDeleteResponse
			MDSrequestContentDeliveryRequest
			MDSrequestContentDeliveryResponse
			MDSrequestContentDownloadRequest
			MDSrequestContentDownloadResponse
			MDSrequestContentStorageRequest
			MDSrequestContentStorageResponse
			MDSrequestDirectoryListRequest
			MDSrequestDirectoryListResponse
			MDSrequestUpdateRequest
			MDSrequestUpdateResponse

service **opensdrmws**

ports **opensdrmwsPort**

binding	**tns:opensdrmwsBinding**
extensibility	**<soap:address location="http://localhost/opensdrm/MDS/MDS.ws.php"/>**

source

```
<service name="opensdrmws">
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm/MDS/MDS.ws.php"/>
  </port>
</service>
```

binding **opensdrmwsBinding**

type	**tns:opensdrmwsPortType**
extensibility	<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
operations	**MDSrequestContentStorage**

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestContentStorage" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

MDSrequestContentDelivery

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestContentDelivery" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

MDSrequestContentDownload

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestContentDownload" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

MDSrequestContentDelete

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestContentDelete" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

MDSrequestDirectoryList

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestDirectoryList" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

MDSrequestUpdate

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestUpdate" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

used by	Service **opensdrmws** in Port **opensdrmwsPort**

source

```
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="MDSrequestContentStorage">
  <soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestContentStorage" style="rpc"/>
  <input>
   <soap:body              use="encoded"              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body              use="encoded"              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="MDSrequestContentDelivery">
  <soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestContentDelivery" style="rpc"/>
  <input>
   <soap:body              use="encoded"              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body              use="encoded"              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="MDSrequestContentDownload">
  <soap:operation              soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestContentDownload"
style="rpc"/>
  <input>
   <soap:body              use="encoded"              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body              use="encoded"              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="MDSrequestContentDelete">
  <soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestContentDelete" style="rpc"/>
  <input>
   <soap:body              use="encoded"              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body              use="encoded"              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="MDSrequestDirectoryList">
  <soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestDirectoryList" style="rpc"/>
  <input>
   <soap:body              use="encoded"              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body           use="encoded"           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="MDSrequestUpdate">
  <soap:operation soapAction="http://localhost/opensdrm/MDS/MDS.ws.php/MDSrequestUpdate" style="rpc"/>
  <input>
   <soap:body           use="encoded"           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body           use="encoded"           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
</binding>
```

porttype **opensdrmwsPortType**

operations

MDSrequestContentStorage

input	**tns:MDSrequestContentStorageRequest**
output	**tns:MDSrequestContentStorageResponse**

MDSrequestContentDelivery

input	**tns:MDSrequestContentDeliveryRequest**
output	**tns:MDSrequestContentDeliveryResponse**

MDSrequestContentDownload

input	**tns:MDSrequestContentDownloadRequest**
output	**tns:MDSrequestContentDownloadResponse**

MDSrequestContentDelete

input	**tns:MDSrequestContentDeleteRequest**
output	**tns:MDSrequestContentDeleteResponse**

MDSrequestDirectoryList

input	**tns:MDSrequestDirectoryListRequest**
output	**tns:MDSrequestDirectoryListResponse**

MDSrequestUpdate

input	**tns:MDSrequestUpdateRequest**
output	**tns:MDSrequestUpdateResponse**

used by binding **opensdrmwsBinding**

source
```
<portType name="opensdrmwsPortType">
  <operation name="MDSrequestContentStorage">
    <input message="tns:MDSrequestContentStorageRequest"/>
    <output message="tns:MDSrequestContentStorageResponse"/>
  </operation>
  <operation name="MDSrequestContentDelivery">
    <input message="tns:MDSrequestContentDeliveryRequest"/>
    <output message="tns:MDSrequestContentDeliveryResponse"/>
  </operation>
  <operation name="MDSrequestContentDownload">
    <input message="tns:MDSrequestContentDownloadRequest"/>
    <output message="tns:MDSrequestContentDownloadResponse"/>
  </operation>
  <operation name="MDSrequestContentDelete">
    <input message="tns:MDSrequestContentDeleteRequest"/>
    <output message="tns:MDSrequestContentDeleteResponse"/>
  </operation>
  <operation name="MDSrequestDirectoryList">
    <input message="tns:MDSrequestDirectoryListRequest"/>
    <output message="tns:MDSrequestDirectoryListResponse"/>
  </operation>
  <operation name="MDSrequestUpdate">
    <input message="tns:MDSrequestUpdateRequest"/>
    <output message="tns:MDSrequestUpdateResponse"/>
  </operation>
</portType>
```

message **MDSrequestContentStorageRequest**

parts

cid

type **xsd:string**

filetype

type **xsd:string**

protocol

type **xsd:string**

location

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestContentStorage**

source
```
<message name="MDSrequestContentStorageRequest">
  <part name="cid" type="xsd:string"/>
  <part name="filetype" type="xsd:string"/>
  <part name="protocol" type="xsd:string"/>
  <part name="location" type="xsd:string"/>
</message>
```

message **MDSrequestContentStorageResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestContentStorage**

source

```
<message name="MDSrequestContentStorageResponse">
  <part name="result_message" type="xsd:string"/>
</message>
```

message **MDSrequestContentDeliveryRequest**

parts **content_id**

type **xsd:string**

user_id

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestContentDelivery**

source

```
<message name="MDSrequestContentDeliveryRequest">
  <part name="content_id" type="xsd:string"/>
  <part name="user_id" type="xsd:string"/>
</message>
```

message **MDSrequestContentDeliveryResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestContentDelivery**

source

```
<message name="MDSrequestContentDeliveryResponse">
  <part name="result_message" type="xsd:string"/>
</message>
```

message **MDSrequestContentDownloadRequest**

parts **content_id**

type **xsd:string**

user_id

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestContentDownload**

source
```
<message name="MDSrequestContentDownloadRequest">
 <part name="content_id" type="xsd:string"/>
 <part name="user_id" type="xsd:string"/>
</message>
```

message **MDSrequestContentDownloadResponse**

parts **result_message**

type **xsd:string**

url

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestContentDownload**

source
```
<message name="MDSrequestContentDownloadResponse">
 <part name="result_message" type="xsd:string"/>
 <part name="url" type="xsd:string"/>
</message>
```

message **MDSrequestContentDeleteRequest**

parts **content_id**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestContentDelete**

source
```
<message name="MDSrequestContentDeleteRequest">
 <part name="content_id" type="xsd:string"/>
</message>
```

message **MDSrequestContentDeleteResponse**

parts **result_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestContentDelete**

source
```
<message name="MDSrequestContentDeleteResponse">
 <part name="result_message" type="xsd:string"/>
</message>
```

message **MDSrequestDirectoryListRequest**

parts

used by PortType **opensdrmwsPortType** in Operation **MDSrequestDirectoryList**

source
```
<message name="MDSrequestDirectoryListRequest"/>
```

message **MDSrequestDirectoryListResponse**

parts **result_message**

type **xsd:string**

clist

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestDirectoryList**

source
```
<message name="MDSrequestDirectoryListResponse">
 <part name="result_message" type="xsd:string"/>
 <part name="clist" type="xsd:string"/>
</message>
```

message **MDSrequestUpdateRequest**

parts **cid**

type **xsd:string**

filetype

type **xsd:string**

protocol

type **xsd:string**

location

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **MDSrequestUpdate**

source
```
<message name="MDSrequestUpdateRequest">
 <part name="cid" type="xsd:string"/>
 <part name="filetype" type="xsd:string"/>
 <part name="protocol" type="xsd:string"/>
 <part name="location" type="xsd:string"/>
</message>
```

message **MDSrequestUpdateResponse**

parts	**result_message**
	type **xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **MDSrequestUpdate**

source
```
<message name="MDSrequestUpdateResponse">
 <part name="result_message" type="xsd:string"/>
</message>
```

PGW – Billing Service

WSDL location:	**D:\WWW\opensdrm\wsdl\pgwws.wsdl**
targetnamespace:	**http://www.adetti.pt/opensdrmws**

services	bindings	porttypes	messages
opensdrmws	**opensdrmwsBinding**	**opensdrmwsPortType**	**PGWrequestCOSSubscribeRequest**
			PGWrequestCOSSubscribeResponse
			PGWrequestPaymentCaptureRequest
			PGWrequestPaymentCaptureResponse
			PGWrequestPaymentClearenceRequest
			PGWrequestPaymentClearenceResponse

service **opensdrmws**

ports	**opensdrmwsPort**
	binding **tns:opensdrmwsBinding**
	extensibility **<soap:address location="http://localhost/opensdrm/PGW/PGW.ws.php"/>**

source
```
<service name="opensdrmws">
 <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
  <soap:address location="http://localhost/opensdrm/PGW/PGW.ws.php"/>
 </port>
</service>
```

binding **opensdrmwsBinding**

type **tns:opensdrmwsPortType**

extensibility	<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

operations

PGWrequestPaymentClearence

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/PGW/PGW.ws.php/PGWrequestPaymentClearence" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

PGWrequestPaymentCapture

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/PGW/PGW.ws.php/PGWrequestPaymentCapture" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

PGWrequestCOSSubscribe

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/PGW/PGW.ws.php/PGWrequestCOSSubscribe" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

used by — Service **opensdrmws** in Port **opensdrmwsPort**

source

```
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="PGWrequestPaymentClearence">
    <soap:operation soapAction="http://localhost/opensdrm/PGW/PGW.ws.php/PGWrequestPaymentClearence" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="PGWrequestPaymentCapture">
    <soap:operation soapAction="http://localhost/opensdrm/PGW/PGW.ws.php/PGWrequestPaymentCapture" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
  <operation name="PGWrequestCOSSubscribe">
    <soap:operation soapAction="http://localhost/opensdrm/PGW/PGW.ws.php/PGWrequestCOSSubscribe" style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
    </output>
  </operation>
</binding>
```

porttype **opensdrmwsPortType**

operations **PGWrequestPaymentClearence**

input **tns:PGWrequestPaymentClearenceRequest**
output **tns:PGWrequestPaymentClearenceResponse**

PGWrequestPaymentCapture

input **tns:PGWrequestPaymentCaptureRequest**
output **tns:PGWrequestPaymentCaptureResponse**

PGWrequestCOSSubscribe

input **tns:PGWrequestCOSSubscribeRequest**
output **tns:PGWrequestCOSSubscribeResponse**

used by binding **opensdrmwsBinding**

source

```
<portType name="opensdrmwsPortType">
  <operation name="PGWrequestPaymentClearence">
    <input message="tns:PGWrequestPaymentClearenceRequest"/>
    <output message="tns:PGWrequestPaymentClearenceResponse"/>
  </operation>
  <operation name="PGWrequestPaymentCapture">
    <input message="tns:PGWrequestPaymentCaptureRequest"/>
    <output message="tns:PGWrequestPaymentCaptureResponse"/>
  </operation>
  <operation name="PGWrequestCOSSubscribe">
    <input message="tns:PGWrequestCOSSubscribeRequest"/>
    <output message="tns:PGWrequestCOSSubscribeResponse"/>
  </operation>
</portType>
```

message **PGWrequestPaymentClearenceRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

data

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **PGWrequestPaymentClearence**

source
```
<message name="PGWrequestPaymentClearenceRequest">
  <part name="identification" type="xsd:string"/>
  <part name="signature_algorithm_identifier" type="xsd:string"/>
  <part name="data" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **PGWrequestPaymentClearenceResponse**

parts **result_message**

type **xsd:string**

transaction_number

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **PGWrequestPaymentClearence**

source
```
<message name="PGWrequestPaymentClearenceResponse">
  <part name="result_message" type="xsd:string"/>
  <part name="transaction_number" type="xsd:string"/>
</message>
```

message **PGWrequestPaymentCaptureRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

tid

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **PGWrequestPaymentCapture**

source
```
<message name="PGWrequestPaymentCaptureRequest">
 <part name="identification" type="xsd:string"/>
 <part name="signature_algorithm_identifier" type="xsd:string"/>
 <part name="tid" type="xsd:string"/>
 <part name="signature" type="xsd:string"/>
</message>
```

message **PGWrequestPaymentCaptureResponse**

parts **result_message**

type **xsd:string**

transaction_number

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **PGWrequestPaymentCapture**

source
```
<message name="PGWrequestPaymentCaptureResponse">
 <part name="result_message" type="xsd:string"/>
 <part name="transaction_number" type="xsd:string"/>
</message>
```

message **PGWrequestCOSSubscribeRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

certificate

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **PGWrequestCOSSubscribe**

source
```
<message name="PGWrequestCOSSubscribeRequest">
 <part name="identification" type="xsd:string"/>
 <part name="signature_algorithm_identifier" type="xsd:string"/>
```

```
  <part name="certificate" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **PGWrequestCOSSubscribeResponse**

parts	**result_message** type **xsd:string** **cert** type **xsd:string**
used by	PortType **opensdrmwsPortType** in Operation **PGWrequestCOSSubscribe**

source	`<message name="PGWrequestCOSSubscribeResponse">`
	`  <part name="result_message" type="xsd:string"/>`
	`  <part name="cert" type="xsd:string"/>`
	`</message>`

RGS – Registration Service

WSDL location:	**D:\WWW\opensdrm\wsdl\rgsws.wsdl**
targetnamespace:	**http://www.adetti.pt/opensdrmws**

services	bindings	porttypes	messages
opensdrmws	**opensdrmwsBinding**	**opensdrmwsPortType**	**RGSrequestContentRegistrationRequest**
			RGSrequestContentRegistrationResponse
			RGSrequestListAvailableContentRequest
			RGSrequestListAvailableContentResponse
			RGSrequestListMetadataRequest
			RGSrequestListMetadataResponse
			RGSrequestMetadataRegistrationRequest
			RGSrequestMetadataRegistrationResponse

service **opensdrmws**

ports	**opensdrmwsPort** **binding** **tns:opensdrmwsBinding** **extensibility** **<soap:address location="http://localhost/opensdrm/RGS/RGS.ws.php"/>**
source	`<service name="opensdrmws">`

```
  <port name="opensdrmwsPort" binding="tns:opensdrmwsBinding">
    <soap:address location="http://localhost/opensdrm/RGS/RGS.ws.php"/>
  </port>
</service>
```

binding **opensdrmwsBinding**

type **tns:opensdrmwsPortType**

extensibility `<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>`

operations

RGSrequestContentRegistration

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/RGS/RGS.ws.php/RGSrequestContentRegistration" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

RGSrequestMetadataRegistration

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/RGS/RGS.ws.php/RGSrequestMetadataRegistration" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

RGSrequestListAvailableContent

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/RGS/RGS.ws.php/RGSrequestListAvailableContent" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

RGSrequestListMetadata

extensibility	**<soap:operation soapAction="http://localhost/opensdrm/RGS/RGS.ws.php/RGSrequestListMetadata" style="rpc"/>**
input	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**
output	**<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://www.adetti.pt/opensdrmws"/>**

used by Service **opensdrmws** in Port **opensdrmwsPort**

source

```
<binding name="opensdrmwsBinding" type="tns:opensdrmwsPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="RGSrequestContentRegistration">
    <soap:operation          soapAction="http://localhost/opensdrm/RGS/RGS.ws.php/RGSrequestContentRegistration"
style="rpc"/>
    <input>
      <soap:body            use="encoded"            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="RGSrequestMetadataRegistration">
  <soap:operation soapAction="http://localhost/opensdrm/RGS/RGS.ws.php/RGSrequestMetadataRegistration"
style="rpc"/>
  <input>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="RGSrequestListAvailableContent">
  <soap:operation soapAction="http://localhost/opensdrm/RGS/RGS.ws.php/RGSrequestListAvailableContent"
style="rpc"/>
  <input>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
 <operation name="RGSrequestListMetadata">
  <soap:operation soapAction="http://localhost/opensdrm/RGS/RGS.ws.php/RGSrequestListMetadata" style="rpc"/>
  <input>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </input>
  <output>
   <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.adetti.pt/opensdrmws"/>
  </output>
 </operation>
</binding>
```

porttype **opensdrmwsPortType**

operations	**RGSrequestContentRegistration**

input **tns:RGSrequestContentRegistrationRequest**
output **tns:RGSrequestContentRegistrationResponse**

RGSrequestMetadataRegistration

input **tns:RGSrequestMetadataRegistrationRequest**
output **tns:RGSrequestMetadataRegistrationResponse**

RGSrequestListAvailableContent

input **tns:RGSrequestListAvailableContentRequest**
output **tns:RGSrequestListAvailableContentResponse**

RGSrequestListMetadata

input **tns:RGSrequestListMetadataRequest**
output **tns:RGSrequestListMetadataResponse**

used by binding **opensdrmwsBinding**

source

```
<portType name="opensdrmwsPortType">
  <operation name="RGSrequestContentRegistration">
    <input message="tns:RGSrequestContentRegistrationRequest"/>
    <output message="tns:RGSrequestContentRegistrationResponse"/>
  </operation>
  <operation name="RGSrequestMetadataRegistration">
    <input message="tns:RGSrequestMetadataRegistrationRequest"/>
    <output message="tns:RGSrequestMetadataRegistrationResponse"/>
  </operation>
  <operation name="RGSrequestListAvailableContent">
    <input message="tns:RGSrequestListAvailableContentRequest"/>
    <output message="tns:RGSrequestListAvailableContentResponse"/>
  </operation>
  <operation name="RGSrequestListMetadata">
    <input message="tns:RGSrequestListMetadataRequest"/>
    <output message="tns:RGSrequestListMetadataResponse"/>
  </operation>
</portType>
```

message **RGSrequestContentRegistrationRequest**

parts

identification

type **xsd:string**

signature_algorithm

type **xsd:string**

hash

type **xsd:string**

file

type **xsd:string**

additional_data

type **xsd:string**

aus_cert

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **RGSrequestContentRegistration**

source
```
<message name="RGSrequestContentRegistrationRequest">
  <part name="identification" type="xsd:string"/>
  <part name="signature_algorithm" type="xsd:string"/>
  <part name="hash" type="xsd:string"/>
  <part name="file" type="xsd:string"/>
  <part name="additional_data" type="xsd:string"/>
  <part name="aus_cert" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **RGSrequestContentRegistrationResponse**

parts **status_message**

type **xsd:string**

content_id

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **RGSrequestContentRegistration**

source
```
<message name="RGSrequestContentRegistrationResponse">
  <part name="status_message" type="xsd:string"/>
  <part name="content_id" type="xsd:string"/>
</message>
```

message **RGSrequestMetadataRegistrationRequest**

parts **identification**

type **xsd:string**

signature_algorithm

type **xsd:string**

content_id

type **xsd:string**

metadata

type **xsd:string**

aus_cert

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **RGSrequestMetadataRegistration**

source

```
<message name="RGSrequestMetadataRegistrationRequest">
  <part name="identification" type="xsd:string"/>
  <part name="signature_algorithm" type="xsd:string"/>
  <part name="content_id" type="xsd:string"/>
  <part name="metadata" type="xsd:string"/>
  <part name="aus_cert" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **RGSrequestMetadataRegistrationResponse**

parts **status_message**

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **RGSrequestMetadataRegistration**

source

```
<message name="RGSrequestMetadataRegistrationResponse">
  <part name="status_message" type="xsd:string"/>
</message>
```

message **RGSrequestListAvailableContentRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

criteria

type **xsd:string**

aus_cert

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **RGSrequestListAvailableContent**

source
```
<message name="RGSrequestListAvailableContentRequest">
 <part name="identification" type="xsd:string"/>
 <part name="signature_algorithm_identifier" type="xsd:string"/>
 <part name="criteria" type="xsd:string"/>
 <part name="aus_cert" type="xsd:string"/>
 <part name="signature" type="xsd:string"/>
</message>
```

message **RGSrequestListAvailableContentResponse**

parts **status_message**

type **xsd:string**

content

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **RGSrequestListAvailableContent**

source
```
<message name="RGSrequestListAvailableContentResponse">
 <part name="status_message" type="xsd:string"/>
 <part name="content" type="xsd:string"/>
</message>
```

message **RGSrequestListMetadataRequest**

parts **identification**

type **xsd:string**

signature_algorithm_identifier

type **xsd:string**

content_id

type **xsd:string**

aus_cert

type **xsd:string**

signature

type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **RGSrequestListMetadata**

source
```
<message name="RGSrequestListMetadataRequest">
  <part name="identification" type="xsd:string"/>
  <part name="signature_algorithm_identifier" type="xsd:string"/>
  <part name="content_id" type="xsd:string"/>
  <part name="aus_cert" type="xsd:string"/>
  <part name="signature" type="xsd:string"/>
</message>
```

message **RGSrequestListMetadataResponse**

parts **status_message**

type **xsd:string**

content_id

type **xsd:string**

metadata

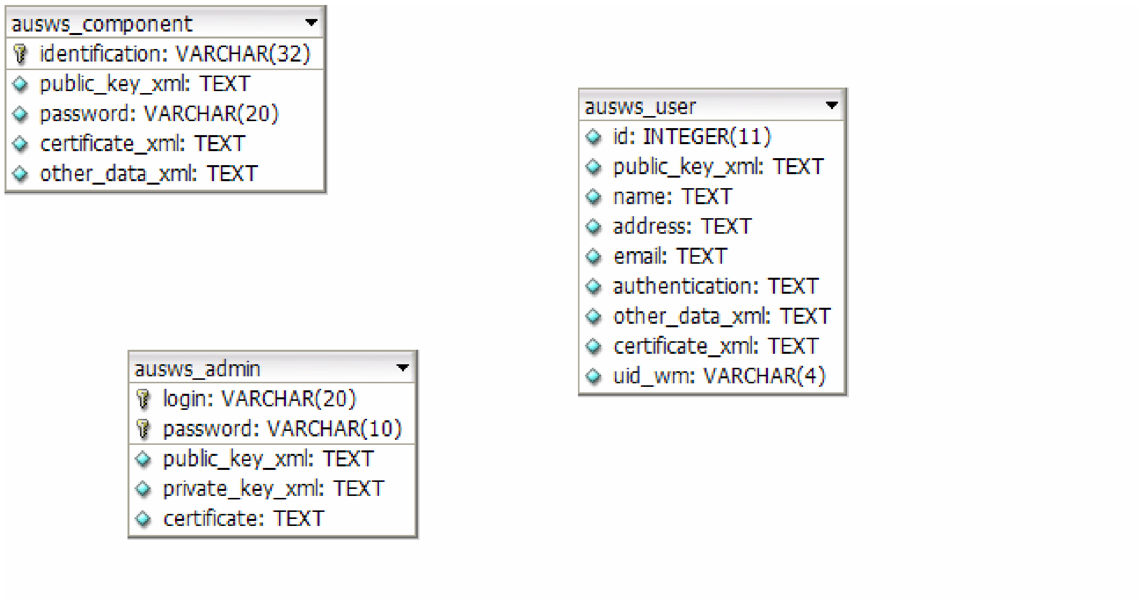
type **xsd:string**

used by PortType **opensdrmwsPortType** in Operation **RGSrequestListMetadata**

source
```
<message name="RGSrequestListMetadataResponse">
  <part name="status_message" type="xsd:string"/>
  <part name="content_id" type="xsd:string"/>
  <part name="metadata" type="xsd:string"/>
</message>
```

Annex – Relational Database schema

AUS – Authentication Service



ausws_admin

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(10)**	PK	NN				
public_key_xml	TEXT		NN				
private_key_xml	TEXT		NN				
certificate	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	login password

ausws_component

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
identification	**VARCHAR(32)**	PK	NN				
public_key_xml	TEXT		NN				
password	VARCHAR(20)		NN				
certificate_xml	TEXT						
other_data_xml	TEXT						

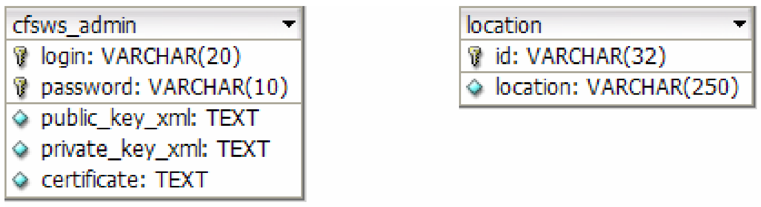
IndexName	IndexType	Columns
PRIMARY	PRIMARY	identification

ausws_user

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
identification	**VARCHAR(32)**	PK	NN				
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(20)**	PK	NN				
id	INTEGER(11)		NN				
public_key_xml	TEXT		NN				
name	TEXT						
address	TEXT						
email	TEXT						
authentication	TEXT						
other_data_xml	TEXT						
certificate_xml	TEXT						
uid_wm	VARCHAR(4)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	identification login password

CFS – Configuration Service



cfsws_admin

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(10)**	PK	NN				
public_key_xml	TEXT		NN				
private_key_xml	TEXT		NN				
certificate	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	login password

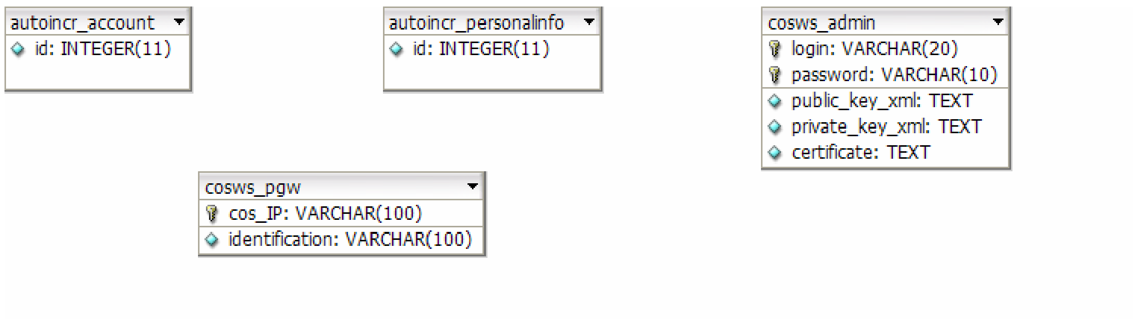
location

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**VARCHAR(32)**	PK	NN				

location	VARCHAR(250)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

COS – Commerce Service



autoincr_account

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	INTEGER(11)		NN				

autoincr_personalinfo

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	INTEGER(11)		NN				

cosws_admin

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(10)**	PK	NN				
public_key_xml	TEXT		NN				
private_key_xml	TEXT		NN				
certificate	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	login password

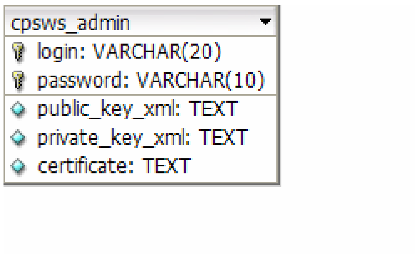
cosws_pgw

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
cos_IP	**VARCHAR(100)**	PK	NN				
identification	VARCHAR(100)		NN				

IndexName	IndexType	Columns

PRIMARY	PRIMARY	cos_IP

CPS – Content Preparation Service

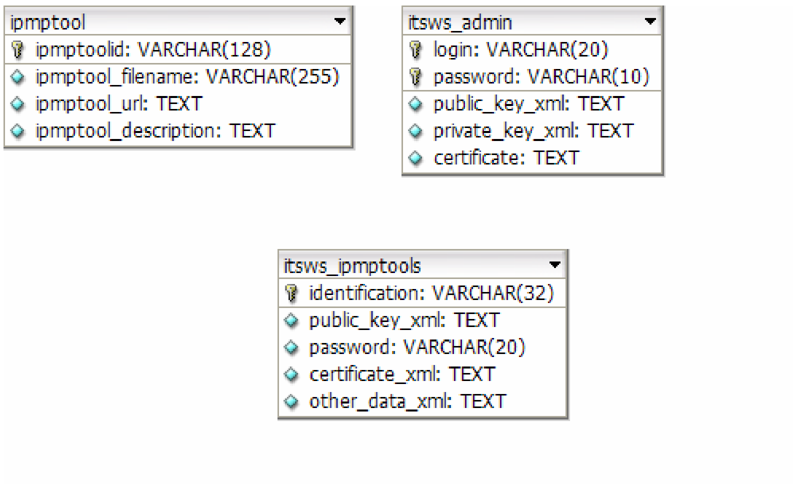

cpsws_admin
login: VARCHAR(20)
password: VARCHAR(10)
public_key_xml: TEXT
private_key_xml: TEXT
certificate: TEXT

cpsws_admin

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(10)**	PK	NN				
public_key_xml	TEXT		NN				
private_key_xml	TEXT		NN				
certificate	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	login password

ITS – Protection Tools Service


ipmptool
ipmptoolid: VARCHAR(128)
ipmptool_filename: VARCHAR(255)
ipmptool_url: TEXT
ipmptool_description: TEXT

itsws_admin
login: VARCHAR(20)
password: VARCHAR(10)
public_key_xml: TEXT
private_key_xml: TEXT
certificate: TEXT

itsws_ipmptools
identification: VARCHAR(32)
public_key_xml: TEXT
password: VARCHAR(20)
certificate_xml: TEXT
other_data_xml: TEXT

ipmptool

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
ipmptoolid	**VARCHAR(128)**	PK	NN				

ipmptool_filename	VARCHAR(255)		NN				
ipmptool_url	TEXT		NN				
ipmptool_description	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	ipmptoolid

itsws_admin

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(10)**	PK	NN				
public_key_xml	TEXT		NN				
private_key_xml	TEXT		NN				
certificate	TEXT		NN				

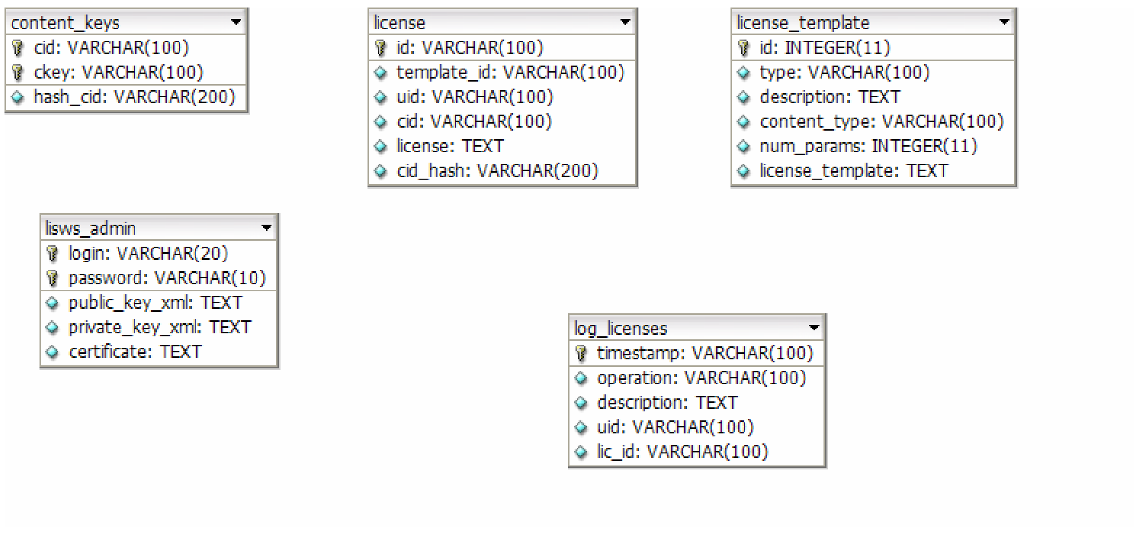
IndexName	IndexType	Columns
PRIMARY	PRIMARY	login password

itsws_ipmptools

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
identification	**VARCHAR(32)**	PK	NN				
public_key_xml	TEXT		NN				
password	VARCHAR(20)		NN				
certificate_xml	TEXT						
other_data_xml	TEXT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	identification

LIS – License Service



content_keys
- cid: VARCHAR(100)
- ckey: VARCHAR(100)
- hash_cid: VARCHAR(200)

license
- id: VARCHAR(100)
- template_id: VARCHAR(100)
- uid: VARCHAR(100)
- cid: VARCHAR(100)
- license: TEXT
- cid_hash: VARCHAR(200)

license_template
- id: INTEGER(11)
- type: VARCHAR(100)
- description: TEXT
- content_type: VARCHAR(100)
- num_params: INTEGER(11)
- license_template: TEXT

lisws_admin
- login: VARCHAR(20)
- password: VARCHAR(10)
- public_key_xml: TEXT
- private_key_xml: TEXT
- certificate: TEXT

log_licenses
- timestamp: VARCHAR(100)
- operation: VARCHAR(100)
- description: TEXT
- uid: VARCHAR(100)
- lic_id: VARCHAR(100)

content_keys

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
cid	**VARCHAR(100)**	PK	NN				
ckey	**VARCHAR(100)**	PK	NN				
hash_cid	VARCHAR(200)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	cid ckey

license

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**VARCHAR(100)**	PK	NN				
template_id	VARCHAR(100)		NN				
uid	VARCHAR(100)		NN				
cid	VARCHAR(100)		NN				
license	TEXT		NN				
cid_hash	VARCHAR(200)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

license_template

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER(11)**	PK	NN				
type	VARCHAR(100)		NN				
description	TEXT		NN				
content_type	VARCHAR(100)		NN				
num_params	INTEGER(11)		NN				

license_template	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

lisws_admin

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(10)**	PK	NN				
public_key_xml	TEXT		NN				
private_key_xml	TEXT		NN				
certificate	TEXT		NN				

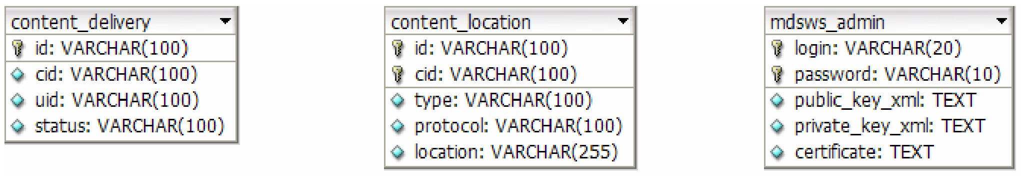
IndexName	IndexType	Columns
PRIMARY	PRIMARY	login password

log_licenses

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
timestamp	**VARCHAR(100)**	PK	NN				
operation	VARCHAR(100)		NN				
description	TEXT		NN				
uid	VARCHAR(100)		NN				
lic_id	VARCHAR(100)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	timestamp

MDS – Media Delivery Service



content_delivery
- id: VARCHAR(100)
- cid: VARCHAR(100)
- uid: VARCHAR(100)
- status: VARCHAR(100)

content_location
- id: VARCHAR(100)
- cid: VARCHAR(100)
- type: VARCHAR(100)
- protocol: VARCHAR(100)
- location: VARCHAR(255)

mdsws_admin
- login: VARCHAR(20)
- password: VARCHAR(10)
- public_key_xml: TEXT
- private_key_xml: TEXT
- certificate: TEXT

content_delivery

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**VARCHAR(100)**	PK	NN				
cid	VARCHAR(100)		NN				
uid	VARCHAR(100)		NN				
status	VARCHAR(100)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

content_location

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**VARCHAR(100)**	PK	NN				
cid	**VARCHAR(100)**	PK	NN				
type	VARCHAR(100)		NN				
protocol	VARCHAR(100)		NN				
location	VARCHAR(255)		NN				

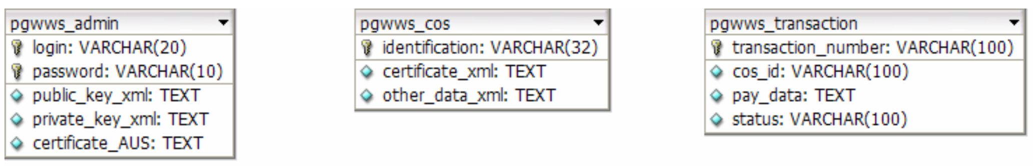
IndexName	IndexType	Columns
PRIMARY	PRIMARY	id cid

mdsws_admin

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(10)**	PK	NN				
public_key_xml	TEXT		NN				
private_key_xml	TEXT		NN				
certificate	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	login password

PGW – Billing Service



pgwws_admin

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(10)**	PK	NN				
public_key_xml	TEXT		NN				
private_key_xml	TEXT		NN				
certificate_AUS	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	login password

pgwws_cos

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
identification	**VARCHAR(32)**	PK	NN				
certificate_xml	TEXT						
other_data_xml	TEXT						

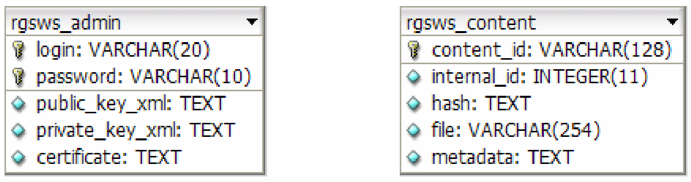
IndexName	IndexType	Columns
PRIMARY	PRIMARY	identification

pgwws_transaction

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
transaction_number	**VARCHAR(100)**	PK	NN				
cos_id	VARCHAR(100)		NN				
pay_data	TEXT		NN				
status	VARCHAR(100)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	transaction_number

RGW – Registration Service



rgsws_admin

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
login	**VARCHAR(20)**	PK	NN				
password	**VARCHAR(10)**	PK	NN				
public_key_xml	TEXT		NN				
private_key_xml	TEXT		NN				
certificate	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	login password

rgsws_content

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
content_id	**VARCHAR(128)**	PK	NN				
internal_id	INTEGER(11)						
hash	TEXT		NN				
file	VARCHAR(254)		NN				
metadata	TEXT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	content_id