|  |  |  |  |
| --- | --- | --- | --- |
| **Documentation control.** | | | |
| **Version** | **Date** | **Pages impacted** | **Comments and reasons for change** |
| 1.0 | 03/04/2018 |  | Initial version, considered stable after the implementation of the GALA 2.1.0 software.  On version 1.3.0\_v3.6 of the same document, only the error codes added in this version are updated. |
| 2.0 | 11/09/2019 |  | Include the numUniddes and rentaUnidades fields in the XML of the annual sales declaration  Modify:  Id = /> by id = /> page 39, 46, 48, 52, 55  Year = /> per year = /> page 52  Number = "" by number = "" page 55  Operation = “” /> per operation = “” /> page 55  New error codes included |
| 3.0 | 06/04/2021 |  | Include the changes that will be implemented after migrating the WS to be invoked through the API Manager. A new OAuth authentication method is added, as well as the functional description, the input fields necessary for the invocation and the output fields. |

INDEX

[INTRODUCTION 7](#_Toc69817513)

[**DOCUMENT OBJECTIVE** 7](#_Toc69817514)

[Technical aspects 7](#_Toc69817515)

[**Mandatory requests** 7](#_Toc69817516)

[INFORMATION EXCHANGE 10](#_Toc69817517)

[**SENDING INFORMATION (SW 🡪 AENA SERVER).** 10](#_Toc69817518)

[PROTOCOL DEFINITION 10](#_Toc69817519)

[**VEHICLE RENTING MANAGEMENT WEB SERVICE** 11](#_Toc69817520)

[**WHITE LISTS MANAGEMENT WEB SERVICE** 29](#_Toc69817521)

[ERRORS MANAGEMENT 32](#_Toc69817522)

[XML FILES 38](#_Toc69817523)

[**SERVICES INVENTORY DECLARATION FILE** 38](#_Toc69817524)

[**MONTHLY SALES DECLARATION FILE.** 43](#_Toc69817525)

[**Annual sales declaration file.** 52](#_Toc69817526)

[**White list declaration file.** 56](#_Toc69817527)

[CERTIFICATION PROCESS 58](#_Toc69817528)

[**PROCESS DEFINITION** 58](#_Toc69817529)

[**CERTIFICATION PROCESS STEPS** 58](#_Toc69817530)

[ANNEX 59](#_Toc69817531)

[**STANDARD TABLES** 59](#_Toc69817532)

[**TIPO\_PAGO** 59](#_Toc69817533)

[**TIPO\_CLIENTE** 60](#_Toc69817534)

[**TIPO\_VENTA\_CONTRATO** 60](#_Toc69817535)

[**TIPO\_IMPUESTO** 61](#_Toc69817536)

[**AIRPORTS** 61](#_Toc69817537)

[**EXAMPLES OF INVOCATION IN JSON** 64](#_Toc69817538)

# INTRODUCTION

## **DOCUMENT OBJECTIVE**

Aena has established a procedure for management and control in SAVIA of, vehicle renting tenant contracts, and of commercial sales registration.

To that end, it’s necessary that SW been installed in airports, is able to send in real time, contract information, to AENA.

Document purpose is first to fix technical and functional guidelines that tenant software has to ensure, to be able to, operate in AENA’s airports, and, then, to ensure with SAVIA a correct data exchange.

# Technical aspects

Due to systems fast evolution, technical requirements listed in this version of the document are susceptible to be subsequently updated, on the basis, of new market trends.

## **Mandatory requests**

Two systems for integration of vehicles renting tenants are available.

A system based on a list of web services with public availability. Which provides the required functionality to send, the performed vehicles renting contracts.

A second system of web nature, which permits to perform the same operations, but doesn’t enable integration with each in use tenant own systems. Access to this second system is more restricted, being allowed only through VPN

Connectivity requirements for GALA’s web application.

Tenant is expected to request, its VPN user creation, by sending the appropriate form to [saviacomercial@aena.es](mailto:saviacomercial@aena.es)

Connectivity requests for GALA web service application

Tenant can invoke API published through public internet, making available two working environments.

The Apis will be exposed in the API Manager for consumption by third parties (external users). To consume them, it will be necessary to be correctly authenticated and authorized, the application user must have been registered in the API Manager and given the relevant credentials.

Every time an application or service user calls one of the APIs, a session token (OAUTH2) previously obtained by calling the API get token must be passed through the header. It will have an expiration of 15 minutes.

New call sequences from a consumer of GALA's web services.

1. Request security token (POST)

Endpoint: <https://globalservicesqa.aena.es/gettoken/v1.0.0/token>

Technical description: Get token

Request body:

{

    "grant\_type": "password", (compulsory)

     "username": "<username>", (compulsory)

     "password": "<password>", (compulsory)

     "scope": "device\_1"

}

Response body:

{

      "access\_token": "9fc558e8-f173-3f2a-a224-7d61049d14cb",

     "refresh\_token": "07fb8da0-006d-3384-9ee9-4faf1caa1b01",

     "scope": "default",

     "token\_type": "Bearer",

     "expires\_in": 3600

}

**The authentication credentials must be provisioned to the different users for the consumption of the “gettoken” Api.**

1. API call and concrete operation with the provided credentials.

In the call to these APIs, the session token (OAUTH2) previously requested in the call to the API described above (gettoken), must be added to the request Header.

Header:

{

"Authorization": "Bearer 9fc558e8-f173-3f2a-a224-7d61049d14cb” (access\_token recovered in gettoken request)

(headers….)

}

1. Call to the web service and specific operation.

* **Development**:
  + The different published Endpoints that will be targeted depending on the operation to be invoked for Contract Management will be the following:
    - **Parking validation for unrelated contracts to AENA:**  [https://globalservicesqa.aena.es/AlquilerVehiculos/v1.0/gestionarParkingContratoAjeno](%20https://globalservicesqa.aena.es/AlquilerVehiculos/v1.0/gestionarParkingContratoAjeno)
    - **Contract closure:**  [https://globalservicesqa.aena.es/AlquilerVehiculos/v1.0/cierreContrato](%20https://globalservicesqa.aena.es/AlquilerVehiculos/v1.0/cierreContrato)
    - **Contract Modification:**  [https://globalservicesqa.aena.es/AlquilerVehiculos/v1.0/modificaContrato](https://scqlapp0003.sscc.ae.aena.es:9443/AlquilerVehiculos/v1.0/modificaContrato)
    - **Contract creation:**  [https://globalservicesqa.aena.es/AlquilerVehiculos/v1.0/altaContrato](https://scqlapp0003.sscc.ae.aena.es:9443/AlquilerVehiculos/v1.0/altaContrato)
    - **Posterior modifications to contract closure:**  [https://globalservicesqa.aena.es/AlquilerVehiculos/v1.0/modificaPosteriorCierreContrato](https://scdlapp0029.sscc.ae.aena.es:8243/AlquilerVehiculos/v1.0/modificaPosteriorCierreContrato)
* The different published Endpoints that will be targeted depending on the operation to be invoked for the Management of White Lists will be the following:
  + - **Global WhiteList Management**:  [https://globalservicesqa.aena.es/ListaBlanca/v1.0/gestionarListaBlancaGlobal](https://scqlapp0003.sscc.ae.aena.es:9443/ListaBlanca/v1.0/gestionarListaBlancaGlobal)
    - **Incremental WhiteList Management**:  [https://globalservicesqa.aena.es/ListaBlanca/v1.0/gestionarListaBlancaIncremental](https://scqlapp0003.sscc.ae.aena.es:9443/ListaBlanca/v1.0/gestionarListaBlancaIncremental)
* **Production: Pending to report**

The definition of the operations published in the service are defined below in this document.

INFORMATION EXCHANGE

## **SENDING INFORMATION (SW 🡪 AENA SERVER).**

Contracts are sent, to AENA’s server, through web services published in public internet.

At AENA’S request, software must contemplate, the “regeneration of transaccions” from any previous registered operation, with same data than the existing one in the original transaction, which includes actual date and time.To do so, operations order to perfom the “regeneration of transactions” must be the same than the one previously performed, by starting with the contract creation, and subsequently performing the appropriate operation to restore the transaction to its previous status.

For example, if a contract has to be regenerated when its current status is closed, first creation has to be invoked, then closure has to be performed, it doesn’t exist any short cut that enables to regenerate directly a closed contract.

# PROTOCOL DEFINITION

WSO2 API MANAGER is in charge of exposing services to the outside in a secure way and mapping the messages through a mediator.

The services described below are of type REST with a resource of type POST in which a SOAP transformation is performed and have an input and output mediation sequence, used to map the different messages.

An API will be published to manage the rental of vehicles with the following operations:

* Contract creation.
* Contract modification
* Contract closure
* Posterior modifications to contract closure
* Parking validation for unrelated contracts to AENA

In addition, there will be another API that will allow the management of white lists, where the following operations are offered:

* Process global whitelist
* Process incremental whitelist

In order to consume each of these APIs, it is necessary to previously make a request to the access token API that will provide the Access authorization token for the different APIS.

The request must be made with the authentication of a user and once this action has been carried out, it will return a JSON with the Access Token of authorization, with a limited duration of 15 minutes, for the different APIS.

Both the access user with the token request endpoint varies depending on the environment.

The definitions of the APIS are included in the following annex:

* 8.7 Examples of invocation of operations in JSON.

## **VEHICLE RENTING MANAGEMENT WEB SERVICE**

An API is published to receive information related to managed vehicles renting tenants information.

In numeric fields, in all items defined as amounts or percentages., decimal separators must be imperatively points, and not comas. Not informed optional fields, are not included in SOAP message to be sent.

Contract creation.

This transaction is requested in real-time when a car renting company is registering a contract creation.

The following table briefly explains the required information to create a contract.

Format and restrictions are defined, through an XSD file, and, public interface, is defined through a WSDL file, documented in the annex. All fields, are mandatory, unless otherwise specified

|  |  |
| --- | --- |
| FIELD | DESCRIPTION |
| clave\_contrato | Vehicle renting contract key. It’s the field which identifies the contract.  Format: a maximum of 16 character strings. Tenant must be identified by the 2 initial characters. Airport’s Commercial area provides this identifier/prefix when SAVIA’s system is going to be used. |
| cod\_iata | Airport identification IATA code.  Format: 3 capital letters |
| fecha\_apertura | Contract creation date  Format: As specified, format must be YYYY-MM-DDThh:mm:ss. |
| fecha\_cierre | Closure contract date. It’s a forecasted contract closure date  Format: As it has been specified, it must be YYYY-MM-DDThh:mm:ss. |
| id\_arrendatario | SAVIA’s tenant identification. Airport’s Commercial area provides this code when SAVIA’s system is going to be used.  Format: Numeric value with a maximum of 6 positive digits integer |
| id\_contrato | SAVIA’s contract identifier. Airport’s Commercial area provides this code when SAVIA s system is going to be used.  Format: a maximum of 30 character strings. |
| id\_local | SAVIA’s commercial place identifier. Airport’s Commercial area provides this code when SAVIA’s system is going to be used.  Format: a maximum of 50 character strings. |
| matricula | Car number plate linked to the contract  Format: a maximum of 20 character strings. |
| tipo\_cliente | Client type standard code.  **Not Mandatory**  Defined in standard table: TIPO\_CLIENTE.  Format: a maximum of 2 character strings. |
| origen | Passenger departure airport IATA code.  **Not Mandatory.**  Format: a maximum of 3 character strings. |
| tipos\_pago | List of types of payment which has divided the collection of the gross amount of the contract.  **Not Mandatory**  The structure of this element is displayed at the end of this table. |
| imp\_bruto | Total of all gross amounts of declared services  Format: numeric value, integer, or decimal with a maximum of 7 positive integer digits and 3 decimals.  If in the request there is a breakdown at a service level, then this field should be validated against the sum of all declared services gross amounts. |
| imp\_neto | Total of all net amount of declared services  Format: numeric value, integer or decimal, with a maximum of 7 positive integer digits and 3 decimals  **Not Mandatory**.  If in the request this amount is informed and there is a breakdown at a service level, then this field should be validated against the sum of all declared services net amounts. |
| imp\_impuesto | Total of all tax amount of declared services.  Format: numeric value, integer or decimal with a maximum of 6 positive integer digits and 3 decimals.  **Not Mandatory**.  If in the request this amount is informed and there is a breakdown at a service level, then this field should be validated against the sum of all tax amounts of all declared services**.** |
| Services[] | This structure is repetitive: It shows a list of contracted services.  If there is only one service informed, it must be related to the vehicle renting contract in itself. The remaining services on the list of services are extra services.  **Not Mandatory**  Structure of this field is described in next table. |
| cod\_clienteaenaclub | Client club AENA code  Format: 7 characters string.  **Not mandatory** |

**Service entity structure[]**

|  |  |
| --- | --- |
| ELEMENTO | DESCRIPCIÓN |
| cod\_familia | Family identifier of associated service  AENA’s Commercial area provides it; It is used by RAC tenant, in its declaration of services. (to see annex 6.1).  Format: Integer value, with a maximum of 6 digits. |
| cod\_subfamilia | Subfamily identifier of associated service  AENA’s Commercial area provides it, It is used by RAC tenant, when it applies (to see annex 6.1).  Format: Integer value, with a maximum of 6 digits  . |
| imp\_bruto | Gross amount declared for the service  Format: Integer or decimal numeric value, with a maximum of 7 positive integer digits and 3 decimals.  Fields: **imp\_bruto**, **imp\_neto,** e **imp\_impuesto**, are validated through the following equation: gross amount = net amount+ taxes. |
| imp\_neto | Net amount declared for the service  Format: Integer or decimal numeric value with a maximum of 7 positive integer digits and 3 decimals.  Fields: **imp\_bruto**, **imp\_neto** e **imp\_impuesto**, are validated through the following equation: gross amount = net amount + taxes |
| imp\_impuesto | Tax amount declared for the service  Format: Integer or decimal, numeric value with a maximum of 7 positive integer digits and 3 decimals.  Field: **imp\_bruto**, **imp\_neto** and **imp\_impuesto**, are validated through the following equation: gross amount = net + taxes. |
| tipo\_impuesto | Tax type code for the service  Defined in standard table: TIPO\_IMPUESTO.  **Not Mandatory**.  Format: Numeric value, with a maximum of 2 positive integer digits |
| porc\_impuesto | Tax percentage applied to the service  **Not Mandatory**.  Format: Integer or decimal numeric value, with a maximum of 3 positive integer digits, and 2 decimals. Values must be included in the rank [0, 100]. |
| descripción\_servicio | Service description.  Format: a maximum of 25 character strings |

**tipos\_pago entity estructure[]**

|  |  |
| --- | --- |
| FIELD | DESCRIPTION |
| tipo\_pago[] | The field tipos\_pago consist in one or more tipo\_pago fields. This is a repetitive structure. Each tipo\_pago is made up by the next two items: |
| tipo | Normalized code for each kind of type of payment.  See normalized tables: TIPO\_PAGO  Format:Numeric value, with a maximum of 2 positive integer digits. |
| importe | Gross amount declared por the the type of payment. Its value is less than or equal to the gross amount of the contract.  Format: numeric or decimal integer value, consisting of a maximum of 7 digits for the integer part and 3 for the decimal part. |

This transaction returns a response with the following information:

|  |  |
| --- | --- |
| FIELD | DESCRIPTION |
| Result | Successful creation is communicated through this message  Possible values are:   * OK: when transaction is successful. * WARNING: when creation is done correctly, but an error in the communication occurs with the parking opening barrier opening, in the exit of the car. * Failed, creation couldn’t be processed.   Format: a maximum of 10 character strings |
| descripción | This field provides a brief description of the orientation of the operation result.  If the result field is OK, it indicates that the operation was successful. If it is a ERROR, provide a description for the type of error. If the result is a WARNING, provide a description of the reason for the WARNING, although the cod\_error field will indicate a value of 001 means that the operation was successful (the operation was indeed correctly, but has failed communication with the parking for the opening of the barrier.) |
| id\_transaccion | Associated transaction code  Possible values are:   * A valid Id, which informs that transaction, is successfull. * “Without valid identifier”:   + Result is OK, operation has been processed, but monitoring transaction is missing   + If result is WARNING operation has been executed successfully but there is a communication failure with the parking, and transaction can’t be monitored.   + Result is failed, and transaction with failure details is missing   Format: a maximum of 50 character strings |
| cod\_error | This field provides the appropriate code to the result of the operation.  There is a code for a correct result; other codes refer to the different causes that can cause an error.  Format: String 3 characters. |

On regards to “ WARNING” result:

A “Warning” notification tells us that, renting contract exists in our systems, but parking vehicle exit can’t be achieved correctly. Reasons of such issue are mainly the following ones:

1. The parking can’t enable the vehicle exit, as there is no register of its entrance. Error description sent back with the response message tells us: “Parking service hasn’t found any ticket with the same plate number”. Main reasons being:
   1. Contract has not been updated correctly, plate number has been wrongly introduced. In such a case, operator in charge to introduce contract information has to correct plate number information, to next enable, the vehicle to exit.
   2. There is a failure in parking barriers to read plate numbers.
2. There is a system failure in parking management. Returned error description tells us “Parking service returns a failure”.
3. From a done modification Iata-exit-parking value got wrong. Error description provided in the response message tells us “There is a failure in the parking service – Returned exit iata code does not match with the value stored in Savia’s system. In such a case operator in charge of introducing contract data should correct Iata-exit –parking value, once done the vehicle is allowed to exit.

Depending on a, i, ii, and b number of issues, there are standard contingency measures being implemented. If incidents are punctual, in such a case the use of AENA’s web application is not a valid solution, such situation can be solved through a fictitious contract modification with no data update, which generates a parking barrier opening request, being managed by an employee with its identification card, …

