Web API (V2)

09/2022

Geniki's web API is a SOAP/WSDL web service located at:

https://voucher.taxydromiki.gr/JobServicesV2.asmx

For test purposes, using a test user that we will provide, you can use the web service located at:

https://testvoucher.taxydromiki.gr/JobServicesV2.asmx

> ***Important Notice:***
> *In order to get credentials for the live environment you must first confirm to our sales department that you have implemented, in your solution for the end user, the following actions:*
>
> - *Authorization (Usage of the* **`Authenticate`** *method)*
> - *Voucher creation (Usage of the* **`CreateJob`** *method)*
> - *Print voucher (Usage of* **`GetVouchersPdf`** *method, if using our template. If you are using your own template, validation of the actual printed voucher document will be required)*
> - *Ability to cancel a voucher (Usage of the* **`CancelJob`** *method)*
> - *Finalize created vouchers (Usage of the* **`ClosePendingJobsByDate`**, **`ClosePendingJobs`** *methods)*
>
> ***Order of actions should be as mentioned above**.*
>
> *At first you authenticate with the service, and then you use the authentication key returned to create / print / cancel vouchers or use any other method. It is recommended, in order to gain in client performance, to use the same authentication key until it expires. After it expires, just call* **`Authenticate`** *again for a new key.*
>
> *Just before handing the packages to Geniki* ***you should finalize*** *your created vouchers in order for Geniki shop to be notified and cross check the received packages (**best practice** is to store the last finalization date-time and use the* **`ClosePendingJobsByDate`** *since that date-time to be sure that everything is reported correctly to Geniki)*

The API provides the following methods. The objects that are used are explained later on.

Service Methods:

```
AuthenticateResult Authenticate(string sUsrName, string sUsrPwd, string applicationKey)
```

User name, password and application key are passed and an object with the result code and an authentication key is returned. This authentication key is used with all the other methods until it is expired.

```
CreateJobResult CreateJob(string sAuthKey, Record oVoucher, JobType eType)
```

The voucher or order info is passed in Record object. The record should have been filled with the minimum info for creating a voucher or order (Name, Address, City). Job id, voucher number and subvouchers (if any) are returned (for order job type, voucher number and subvouchers will be empty since no voucher is created when placing the order).

```
GetVoucherJobResult GetVoucherJob(string sAuthKey, long nJobId)
```

The job with the specified id is returned.

```
int ClosePendingJobsByDate(string sAuthKey, DateTime dFr, DateTime dTo)
```

Sends the information to the Geniki shop and closes the pending jobs (vouchers/orders, that are not yet closed) that have been created in the range *dateFrom* – *dateTo*. An error code may be returned (explained later).

```
int ClosePendingJobs(string sAuthKey)
```

As the previous function, but, instead of a date range, it sends to the Geniki shop and closes **only** the pending jobs that have been created **on the same day** it is called. *Does not send or close open jobs created on a previous day [each day ends at midnight 12:00:00 GR time. The next day begins after that.]*.

```
int CancelJob(string sAuthKey, long nJobId, bool bCancel)
```

Cancels or reactivates the job with the specified id. If *bCancel* parameter is true the job is canceled. If *bCancel* is false, the job is made active again. An error code is returned (explained later).

```
GetJobsResult GetJobsFromOrderId(string sAuthKey, string sOrderId)
```

Gets all the jobs referenced by this order id. Can be more than one if the client does not use unique order ids for the jobs he creates.

TrackAndTraceResult TrackAndTrace(string authKey, string voucherNo, string language)

Gets a history of checkpoints for the specified voucher number and its current status (delivered/not delivered). Language can be "el" (for Greek) or "en" (for English). Can, also, be used for tracking the return requests (using the *ReturnSerial* as a voucher number).

TrackDeliveryStatusResult TrackDeliveryStatus(string authKey, string voucherNo, string language)

Gets the current delivery status of a voucher (delivered/not delivered) along with the latest destination/shop code. Language can be "el" (for Greek) or "en" (for English). Can, also, be used for tracking the return requests (using the *ReturnSerial* as a voucher number).

void GetVouchersPdf(string authKey, string[] voucherNumbers, MediaFormat format, ExtraInfoFormat extraInfoFormat)

Get a pdf document to print on Geniki's supported paper types, for the specified vouchers. This method differs from the others in that it does not return a SOAP message as result. In case of success it returns an "application/pdf" document and in case of failure a "text/plain" document with the error result code (see later). "*voucherNumbers*" argument is an array with the voucher numbers you need to generate a PDF document for. "*format*" argument can have either of the values "Flyer" or "Sticker", depending on the paper type that will be used for printing on. Lastly *extraInfoFormat* should always have the value "None". You can use an HTTP GET request if this is easier, like this:

https://voucher.taxydromiki.gr/JobServicesV2.asmx/GetVouchersPdf?authKey=key&voucherNumbers=voucherNo1&voucherNumbers=voucherNo2...&voucherNumbers=voucherNoN&format=Flyer&extraInfoFormat=None

void GetVoucherPdf(string authKey, string voucherNo, MediaFormat format, ExtraInfoFormat extraInfoFormat)

Get a pdf document for the specified voucher. Same as above (GetVouchersPdf) but for a single *voucherNo*.
The HTTP Get request can be constructed like this:

https://voucher.taxydromiki.gr/JobServicesV2.asmx/GetVoucherPdf?authKey=key&voucherNo=voucherNo1&format=Flyer&extraInfoFormat=None

void GetVoucherZPL(string authKey, string voucherNo, MediaFormatZPL format)

Get a ZPL document to print for the specified voucher. Right now it works for Zebra model GC420 but it can also work for similar models. The method returns a single *voucher ZPL file (utf-8)*. The format argument value must be "ZPL1015" which means that it will return a label with 10cm Height X 15cm Width.

GetShopsResult GetShopsList(string authKey)

Gets a list of Geniki Taxydromiki shops.

Advanced Service Methods:

The authenticated user needs to be granted the special permission [RR] to use the following methods:

`CreateReturnRequestResult CreateReturnRequest(string sAuthKey, ReturnRecord oReturnRecord, int nDaysValid, string sRecipientId, bool bPayedBySender)`

The *oReturnRecord* is required in order to create a direct return request. Total days (since the request date) that the return coupon is valid to be used (*nDaysValid*) can be either 0 (zero=infinite) or any positive number of days. The recipient id (*sRecipientId*) may be empty or it can be provided when the authenticated user has defined more than one recipients (e.g. different warehouses), it is authorized and predefined by sales department, to deliver the returned items to. If the return voucher transportation fee is to be payed by the sender, set *bPayedBySender* to *True*, otherwise set it to *False* in order to charge the fee to your own account. The created return request coupon serial number is returned by the call.

`CreateReturnRequestResult CreateVoucherReturnRequest(string sAuthKey, string sVoucherNumber, int nDaysValid, string sRecipientId, bool bPayedBySender)`

The *sVoucherNumber* number is required in order to create a voucher return request. The voucher should be an already existing voucher created by the authenticated user. Total days (since the request date) that the return coupon is valid to be used (*nDaysValid*) can be either 0 (zero=infinite) or any positive number of days. The recipient id (*sRecipientId*) may be empty or it can be provided when the authenticated user has defined more than one recipients (e.g. different warehouses), it is authorized and predefined by sales department, to deliver the returned items to. If the return voucher transportation fee is to be payed by the sender, set *bPayedBySender* to *True*, otherwise set it to *False* in order to charge the fee to your own account. The created return request coupon serial number is returned by the call.

`CreateReturnRequestResult UpdateReturnRequestParams(string sAuthKey, string sReturnSerial, int nDaysValid, string sRecipientId, bool bPayedBySender)`

The *sReturnSerial* is required in order to update the usage parameters *nDaysValid, sRecipientId* and *bPayedBySender* of an existing return request. Updating the parameters is possible any time before the return request coupon is used in a shop. The updated return request coupon is returned by the call.

`GetReturnRequestResult GetReturnRequest(string sAuthKey, string returnSerial)`

The *returnSerial* is required in order to get the return request information. The already existing return request coupon information is returned by the call, or an error result otherwise.

`GetReturnRequestResult GetReturnRequestOfVoucher(string sAuthKey, string sVoucherNumber)`

The *sVoucherNumber* number is required in order to get the return request information. The voucher should be an already existing voucher created by the authenticated user. The already existing voucher return request coupon information is returned by the call, or an error result otherwise.

`void GetReturnRequestsPdf(string authKey, string[] returnSerials)`

Get a pdf document for the specified RR serials. This method differs from the others in that it does not return a SOAP message as result. In case of success it returns an "application/pdf" document and in case of failure a "text/plain" document with the error result code (see later). "*returnSerials*" argument is an array with the return serials you need to generate a PDF document for. You can use an HTTP GET request if this is easier, like this:

https://voucher.taxydromiki.gr/JobServicesV2.asmx/GetReturnRequestsPdf?authKey=key&returnSerials=serialNo1&returnSerials=serialNo2...&returnSerials=serialNoN

`void GetReturnRequestPdf(string authKey, string returnSerial)`

Get a pdf document for the specified RR serial. Same as above (GetReturnRequestsPdf) but for a single *returnSerial*. The HTTP Get request can be constructed like this:

https://voucher.taxydromiki.gr/JobServicesV2.asmx/GetReturnRequestPdf?authKey=key&returnSerial=serialNo

---

The authenticated user needs to be granted the special permissions [3rd-Party-Shippers/Grouping] to use the following methods:

`Add3rdPartyShipperResult Add3rdPartyShipper(string sAuthKey, string clientCode, string vatId)`

The *clientCode* and the client's *vatId* are required. This client code is added to the eligible 3rd party shippers of the authenticated user in order to be able to be used in the method "CreateClientVoucherJob". Vat Identification Number must begin with the 2-digit country code uppercase (e.g. EL for Greece). Creation date and active status of the 3rd party shipper are returned.

When testing, only the following sample clients can be used, any other code/vat-id will return an error:

Client Code	Is an active code?	Vat Identification Number	Name
1DM10001	Yes	NL123456789B01	Test Shipper1
1DM10002	Yes	EL123456702	Test Shipper2
1DM10003	Not Active (*)	EL123456703	Test Shipper3
1DM10004	Yes	EL123456704	Test Shipper4
1DM10005	Yes	AU12345678X	Test Shipper5

(*) 1DM10003 client can be added to 3rd party shippers of the authenticated user, bu can not be used to create client voucher jobs, since its not active.

`CreateJobResult CreateClientVoucherJob(string sAuthKey, string clientCode, Record oVoucher)`

The requested *clientCode* is passed in a string. The *voucher* is passed in a Record object. The record should have been filled with the minimum info for creating a voucher (Name, Address, City). The created Job id, voucher number and subvouchers (if any) are returned.
An error is returned if the *clientCode* is not in the 3rd party shippers of the authenticated user, or if the *clientCode* is not an active shipper.

`CreateGroupedVoucherJobResult CreateGroupedVoucherJob(string sAuthKey, Record oVoucher, string[] groupedVoucherNumbers)`

The grouping *voucher* is passed in the Record object. The record should have been filled with the minimum info for creating a voucher (Name, Address, City). The existing vouchers' numbers, that are to be grouped, are passed in a string array. The grouping voucher number and the created grouping records are returned.

Objects

Record		
Field	Type	Description
`OrderId`	`string`	User defined id
`Name`	`string`	Recipient name
`Address`	`string`	Recipient address
`City`	`string`	Recipient city
`Telephone`	`string`	Recipient telephone
`Zip`	`string`	Recipient zip code
`Weight`	`decimal`	Voucher weight
`Pieces`	`int`	Voucher pieces. If greater than one, subvouchers are created with different voucher numbers
`Comments`	`string`	Comments can be any length, but up to 40 (255 for orders) characters are inserted in the system
`Services`	`string`	Any services the voucher might have, delimited with comma (see *"Geniki Services"* paragraph that follows)
`CodAmount`	`string`	The amount of the cod. If greater than zero it must always have the service "αμ" or "αν"
`InsAmount`	`decimal`	The amount to be insured. If greater that zero it must have the service "ασ"
`VoucherNo`	`string`	Geniki's unique voucher number
`SubCode`	`string`	If you are using cost centers, this is the code of the cost center
`BelongsTo`	`string`	If it is a subvouher, this field contains the number of the parent voucher
`DeliverTo`	`string`	Used with pickup jobs. Denotes the recipient of a pickup job. If it's empty recipient is the client who created the job.
`ReceivedDate`	`DateTime`	The date that the package is supposed to be received by Geniki's shop. Cannot be smaller than today.
`ContentsDescription`	`string`	A description of the packages' contents. Requested for abroad deliveries.
`SendAndReturnRecipient`	`string`	The recipient id for the returning voucher of send and return services. If empty then the returning voucher is delivered to the originating sender (default).

VoucherJob		
Field	Type	Description
`Id`	`int`	Distinct job id
`Type`	`JobType`	The type of the job (see enumeration below)
`Voucher`	`Record`	The main voucher that this job created
`ReturningVoucher`	`Record`	If the job is of type "SendAndReturn", this field holds the returning voucher created on server
`SubVouchers`	`Record[]`	The subvouchers created by this job, if the main voucher had more than one pieces
`Date`	`DateTime`	Date the job was created
`OrderId`	`string`	User defined id
`IsClosed`	`bool`	Is job closed?
`IsCanceled`	`bool`	Has the job been canceled?
`Status`	`string`	Job status (empty for now)
`StatusDate`	`DateTime`	The date of the status
`User`	`string`	The user who created the job

Type of the job can be any of the following enumerations

```
enum JobType {
        Voucher,        // A normal shipment
        Pickup,         // An order to go to a client and pickup a package to be returned to you
        SendAndReturn   // Sent some documents to a client and get back from him some other documents
}
```

CreateJobResult		
Field	Type	Description
`Result`	`int`	The result code (explained later)
`JobId`	`int`	Distinct job id
`Voucher`	`string`	The main voucher number that this job created
`SubVouchers`	`Record[]`	The subvoucher records created by this job, if the main voucher had more than one piece

GetVoucherJobResult		
Field	Type	Description
Result	int	The result code (explained later)
JobId	int	Distinct job id
Job	VoucherJob	The voucher job

GetJobsResult		
Field	Type	Description
Result	int	The result code (explained later)
Jobs	VoucherJob[]	Array of jobs

AuthenticateResult		
Field	Type	Description
Result	int	The result code (explained later)
Key	string	The authentication key to be used with all functions (except Authenticate)

TrackAndTraceResult		
Field	Type	Description
Result	int	The result code (explained later)
Checkpoints	Checkpoint[]	An array holding the checkpoints history of a voucher
Status	string	The current status of the tracked package (DELIVERED/IN TRANSIT)
DeliveryDate	DateTime	The date that the package was delivered
Consignee	string	The person who signed upon delivery

TrackDeliveryStatusResult		
Field	Type	Description
Result	int	The result code (explained later)
ShopCode	string	The destination/shop code
Status	string	The current status of the tracked package (DELIVERED/IN TRANSIT/IN RETURN)
DeliveryDate	DateTime	The date that the package was delivered
Consignee	string	The person who signed upon delivery

Checkpoint		
Field	Type	Description
Status	string	Checkpoint's status description
StatusDate	DateTime	The date and time the status occurred
Shop	string	The shop associated with the checkpoint

GetShopsListResult		
Field	Type	Description
Result	int	The result code (explained later)
Shops	Shop[]	An array holding the shops of Geniki Taxydromiki

Shop		
conField	Type	Description
Code	string	Shop code
Code2	string	Shop secondary code
Name	string	Shop description
State	string	Shop state
City	string	Shop city
Address	string	Shop address
Telephone	string	Shop telephone
Zip	string	Shop zip
Email	string	Shop email
Longitude	decimal	Shop longitude
Latitude	decimal	Shop latitude
SubShop	boolean	If true the shop is a subshop
Active	boolean	If true the shop is active

CreateReturnRequestResult		
Field	Type	Description
ReturnSerial	string	The return request coupon serial number
ReturnFrom	string	The return from information (voucher/return-record)
ReturnType	string	The return type (VoucherRR/DirectRR)
Recipient	string	The requested recipient id (may be null)
DaysValid	int	The requested days for a valid use of the return coupon [0 = infinite]
IsPayedBySender	bool	Set to *true* only when the sender has to pay the return transportation fee
CreatedOn	DateTime	The creation date and time of the return request

GetReturnRequestResult		
Field	Type	Description
ReturnSerial	string	The return request coupon serial number
ReturnFrom	string	The return from information (voucher/return-record)
ReturnType	string	The return type (VoucherRR/DirectRR)
Recipient	string	The requested recipient id (may be null)
DaysValid	int	The requested days for a valid use of the return coupon [0 = infinite]
IsPayedBySender	bool	*True* only when the sender has to pay the return transportation fee
CreatedOn	DateTime	The creation date and time of the return request
ReturningVoucher	string	The voucher that was created when the return coupon was used
ReturnedOn	DateTime	The date and time the return coupon was used

ReturnRecord		
Field	Type	Description
Name	string	The return-from sender name
Address	string	The return-from address
City	string	The return-from city
Telephone	string	The return-from telephone
Zip	string	The return-from zip code
Pieces	int	Number of pieces
Weight	decimal	Total weight of the return
Comments	string	Additional comments to print on the voucher
OrderId	string	User defined id

Add3rdPartyShipperResult		
Field	Type	Description
Shipper	ThirdPartyShipper	The information of the added third party shipper.

ThirdPartyShipper		
Field	Type	Description
Active	bool	True when the shipper can be used for sending new vouchers.
Code	string	The code of the third party shipper.
CreatedOn	DateTime	The date and time the third party shipper was added.

CreateGroupedVoucherJobResult		
Field	Type	Description
Voucher	string	The grouping voucher number.
GroupedVouchers	GroupingRecord[]	A grouping records array.

GroupingRecord		
Field	Type	Description
Voucher	string	Grouped Voucher number
GroupingVoucher	string	Grouping Voucher number
GroupingShop	string	Id of the grouping shop.
JobId	Long	Job Id of the grouping job.

Error Codes		
Number	Description	Functions that can return it
0	Ok	All functions
1	Authentication failed	**Authenticate**
2	Not implemented	None right now
3	No data	**CreateJob** if null record is passed **TrackAndTrace** if null or empty voucher number is passed **CreateReturnRequest** if not existing/empty voucher number
4	Invalid operation	**CancelJob** if the job is closed **CreateReturnRequest** if the return request coupon is already used for a return
5	Max voucher No. reached	**CreateJob** if the server has reached the max voucher number. Should not happen (still many millions to go), but if it does you should really contact us
6	Max subvoucher No. reached	As above for subvoucher numbers
700	Validation failed	**CreateJob** if any of the required fields (Name, Address, City) is empty
701	Validation failed	**CreateJob**, if cod service set with no cod amount
702	Validation failed	**CreateJob**, if cod amount set with no cod service
703	Validation failed	**CreateJob**, if cod amount limit is exceeded
704	Validation failed	**CreateJob**, if insurance service set with no insurance amount
705	Validation failed	**CreateJob**, if insurance amount set with no insurance service
706	Validation failed	**CreateJob**, if received date is smaller than today
711	Validation failed	**CreateJob**, if recipient id is not empty, and the voucher does not have a send-and-return service.
712	Validation failed	**CreateJob**, **CreateReturnRequest** if recipient id used does not exist.
8	SQL error	All functions, internal error
9	Doesn't exist	**GetVoucherJob** or **CancelJob**, when specified job does not exist **TrackAndTrace**, if specified voucher number is not found
10	Not authorized	**GetVoucherJob, CancelJob**, when the requesting user has not right to access the specified job **TrackAndTrace**, when the requesting user has no right to access the specified voucher
11	Invalid key	All functions except *Authenticate,* when the used authentication key is not valid or it is expired. This is normal, the authentication key may expire after a period and you have to authenticate again.
12	Run-time error	All functions, internal error
13	Job canceled	**GetVoucherJob, CancelJob**, if the job is canceled. For GetVoucherJob it should be treated as an error but rather as a status. The info of the job is still returned
14	Server busy	**CreateJob, ClosePendingJobs, ClosePendingJobsByDate** when temporarily the operation can't be carried out.
15	Request limit reached	All functions, when a limit for requests per some time is set for the calling user and this limit has been exceeded

(*) **CreateJob** errors can also occur when using the methods: **CreateReturnRequest, CreateClientJob, CreateGroupedVoucherJob**.
TrackAndTrace errors can also occur when using the method: **TrackDeliveryStatus**.

Geniki Services

Voucher record `Services` field can have the following (two-character-code) values, separated with comma if more than one services need to be selected (field value is case insensitive).

1Σ	SPECIAL ΠΡΩΙΝΗ ΠΑΡΑΔΟΣΗ	EARLY MORNING DELIVERY
3Σ	ΑΥΘΗΜΕΡΟΝ ΠΟΛΗΣ	SAME DAY DELIVERY (INTRACITY)
5Σ	ΠΑΡΑΔΟΣΗ ΣΑΒΒΑΤΟ	SATURDAY DELIVERY
ΑΜ	ΑΝΤΙΚΑΤΑΒΟΛΗ ΜΕΤΡΗΤΟΙΣ	COD (cash payment)
ΑΝ	ΑΝΤΙΚΑΤΑΒΟΛΗ ΑΞΙΟΓΡΑΦΑ	COD (cheque payment)
ΒΡ	D2SP, ΒΑΣΙΚΗ RECEPTION	D2SP SERVICE (reception delivery)
ΑΡ	D2SP, ΑΝΤΙΚΑΤΑΒΟΛΗ RECEPTION	D2SP COD (cash payment – reception delivery)
ΑΣ	ΑΣΦΑΛΙΣΗ	INSURANCE
ΔΔ	ΔΙΚΑΙΟΛΟΓΗΤΙΚΑ ΔΙΑΓΩΝΙΣΜΩΝ	SUBMISSION OF TENDER DOCUMENTATION
ΕΔ	ΕΙΔΙΚΗ ΧΡΕΩΣΗ	SPECIAL RATE
ΕΜ	ΠΑΡΑΛΑΒΗ ΔΙΚΑΙΟΛΟΓΗΤΙΚΩΝ	RETURN OF PROOF OF DELIVERY or RETURN OF SIGNED RECEIPT
ΕΨ	ΕΙΔΗ ΨΥΓΕΙΟΥ	REFRIGERATED GOODS
ΠΡ	ΠΑΡΑΛΑΒΗ ΠΡΩΤΟΚΟΛΟΥ	RETURN OF PROTOCOL NUMBER
ΠΚ	ΕΠΙΣΤΡΟΦΗ ΠΑΚΕΤΟΥ	EXCHANGE PACKAGE
ΤΝ	ΑΕΡΟΜΕΤΑΦΟΡΑ	NEXT DAY DELIVERY TO ISLANDS
ΧΠ	ΧΡΕΩΣΗ ΠΑΡΑΛΗΠΤΗ	CASH COLLECT (RECEIVER PAYS TRANSPORT FEES)
Τ1	ΤΑΧΥΔΡ. ΚΙΒΩΤΙΟ Ν1 - 5 (ΕΩΣ 2 KG)	POSTAL BOX N1 - 5 (UP TO 2 KG)
Τ6	ΤΑΧΥΔΡ. ΚΙΒΩΤΙΟ Ν6 - 7 (ΕΩΣ 4KG)	POSTAL BOX N6 - 7 (UP TO 4KG)
ΤΕ	ΤΑΧΥΔΡ. ΚΙΒΩΤΙΟ ΕΓΓΡΑΦΩΝ (ΕΩΣ 2 KG)	POSTAL BOX for DOCUMENTS (UP TO 2 KG)
ΥΠ	VIP ΠΑΡΑΔΟΣΗ	VIP DELIVERY
ΦΡ	ΕΜΠΟΡΕΥΜΑΤΙΚΗ ΜΕΤΑΦΟΡΑ	ECONOMY SEA FREIGHT SERVICE TO CYPRUS

Example C# code

```
private void ExampleCode() {
    //JobServicesV2 is the referenced web service class
    JobServicesV2 services = new JobServicesV2();
    AuthenticateResult authResult = services.Authenticate("UserName", "Password", "AppKey");
    if(authResult.Result != 0) {
        //Could not get key
        return;
    }

    Record voucher = new Record {
        OrderId = "00001",
        Name = "Test name",
        Address = "Test address",
        City = "Test city",
        Telephone = "2109999999",
        Zip = "12345",
        Comments = "Test comment",
        SubCode = "",
        Weight = 12.34m,
        Pieces = 3,
        Services = "αν",
        CodAmount = 1234.56m
    };

    CreateJobResult result = services.CreateJob(authResult.Key, voucher, JobType.Voucher);
    if(result.Result != 0) {
        //Error creating voucher
        return;
    }

    //Print them, store them, whatever...
    string voucherNumber = result.Voucher;

    foreach(Record subVoucher in result.SubVouchers) {
        string subVoucherNumber = subVoucher.VoucherNo;
        string name = subVoucher.Name;
        //...
        //...
    }

    //Not necessary to call with each print.
    //Can (and should, to avoid uneccessary burden) be called when all printing is done.
    services.ClosePendingJobs(authResult.Key);
    services.Dispose();
}
```

Example PHP code

Note: if coding under WAMP, don't forget to add the extension php_soap. Also the file has to be saved as UTF-8 for the greek services.

```
<html>
    <body>
        <?php
            try {
                $soap = new SoapClient("https://testvoucher.taxydromiki.gr/JobServicesV2.asmx?WSDL");

                echo "----------- Authenticating -----------<br>";
                $oAuthResult = $soap->Authenticate(
                    array(
                        'sUsrName' => 'UserName',
                        'sUsrPwd' => 'Password',
                        'applicationKey' => 'AppKey'
                    )
                );
                print_r($oAuthResult);
                echo "<BR>";

                if ($oAuthResult->AuthenticateResult->Result != 0) {
                    echo "Error authenticating!!<br>";
                    return;
                }
                echo "Key = " . $oAuthResult->AuthenticateResult->Key . "<br>";
                echo "Authentication OK<br>";

                echo "----------- Creating a voucher -----------<br>";
                $oVoucher = array(
                    'OrderId' => '00001',
                    'Name' => 'Test name',
                    'Address' => 'Test address',
                    'City' => 'Test city',
                    'Telephone' => '2109999999',
                    'Zip' => '12345','Destination' => "",
                    'Courier' => "",
                    'Pieces' => 3,
                    'Weight' => 12,
                    'Comments' => 'Test comment',
```

```
                'Services' => "αν",
                'CodAmount' => 1234.56,
                'InsAmount' => 0,
                'VoucherNumber' => "",
                'SubCode' => "",
                'BelongsTo' => "",
                'DeliverTo' => "",
                'ReceivedDate' => "2012-01-01"
            );

            $xml = array(
                'sAuthKey' => $oAuthResult->AuthenticateResult->Key,
                'oVoucher' => $oVoucher,
                'eType' => "Voucher"
            );
            echo "----------- Result of the voucher creation -----------<br>";
            print_r($xml);
            echo "<BR>";

            $oResult = $soap->CreateJob($xml);
            print_r($oResult);
            echo "<BR>";

            if($oResult->CreateJobResult->Result != 0) {
                echo "Error Creating a voucher!!<br>";
                return;
            }

            echo "----------- Track and Trace a voucher -----------<br>";
            $xml = array (
                'authKey' => $oAuthResult->AuthenticateResult->Key,
                'voucherNo' => $oResult->CreateJobResult->Voucher,
                'language' => 'el'
            );

            $TT = $soap->TrackAndTrace($xml);
            print_r($TT);
            echo "<BR>";

            echo "----------- Closing Pending Jobs -----------<br>";
            $soap->ClosePendingJobs(
                array('sAuthKey' => $oAuthResult->AuthenticateResult->Key)
            );
        } catch(SoapFault $fault) {
            echo $fault;
        }
    ?>
  </body>
</html>
```