

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. By using the API you agree to follow any guidelines given and not to abuse its usage.

Service Methods:

AuthenticateResult **Authenticate**(string sUserName, 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.

GetPickupJobStatusResult **GetPickupJobStatus**(string authKey, long jobId)

Gets the current status along with any vouchers created of a job of type "JobType.Pickup". JobId parameter is the "jobId" value returned, in the "**CreateJobResult**", when the job is initially created.

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.

CreateVoucherPickupOrderResult CreateVoucherPickupOrder(**string** authKey, **string** voucherNumber, **DateTime** pickupDate, **DayQuarter** dayQuarter)

Creates a voucher pickup order request for a specified date/day-quarter.
Day quartered enum can have the following values:

```
enum DayQuarter {  
    Morning,  
    LateMorning,  
    Afternoon,  
    Evening  
}
```

GetVoucherPickupOrderResult GetVoucherPickupOrder(**string** authKey, **string** voucherNumber)

Gets the details/status of a voucher pickup order.

UpdateVoucherPickupOrderResult UpdateVoucherPickupOrder(**string** authKey, **string** voucherNumber, **DateTime** pickupDate, **DayQuarter** dayQuarter)

Updates the date/day-quartered of a voucher pickup order.

CancelVoucherPickupOrderResult CancelVoucherPickupOrder(**string** authKey, **string** voucherNumber)

Marks a voucher pickup order as canceled by the requester.

CreateCODReleaseResult CreateCODRelease(**string** AuthKey, **string** voucherNo, **decimal** codAmount)

Creates a COD (Cash On Delivery) amount Release for a voucher number with an already existing COD service.
The method must take as parameter the voucher number and the new Cash on Delivery amount [≥ 0].

CreateVoucherReturnOrderResult CreateVoucherReturnOrder(**string** authKey, **string** voucherNumber, **string** reason)

Creates a voucher return order for an existing voucher. It takes as parameters the voucher number (must be an existing voucher of the authenticated user) and a reason for user reference (reason can be empty – if not, the delivery shop is informed on the reason of the return).

Warning: This is a blocking action for the transport/delivery of the voucher, and can not be undone. The destination shop can not deliver to the recipient, and has to return the package to the sender. If the voucher is already delivered before the return order is created, the voucher return order is not honored.

GetVoucherReturnOrderResult GetVoucherReturnOrder(**string** authKey, **string** voucherNumber)

Gets the created voucher return order of a voucher, if one exists.

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). See, also, the checkpoints table for checkpoints' status-codes.

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). See, also, the checkpoints table for checkpoints' status-codes.

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/compatible models. The method returns a single *voucher ZPL file (utf-8)*. The only format argument value accepted is "Sticker1015" and it will return a label with 10cm Height X 15cm Width.

GetShopsResult GetShopsList(string authKey)

Gets a list of Geniki Taxydromiki shops.

GetLockersListResult GetLockersList(string authKey)

Gets a list of Geniki Taxydromiki lockers.

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 paid 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 paid 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, but can not be used to create client voucher jobs, since its not active.

Create3rdPartyShipperResult Create3rdPartyShipper(string sAuthKey, ThirdPartyShipperDetails shipperDetails)

A request towards the sales department to create a new Geniki client and add the client code as eligible 3rd party shipper of the authenticated user in order to be able to be used in the method "CreateClientVoucherJob".

When the field *ServicePointCode* of the *shipperDetails* is not empty, and it is set to a 2-digit code of an existing Geniki Shop [see: *Shop.Code* field in *GetShopsResult* of *GetShopsList* method], the 3rd-party-shipper client is created to be serviced by the corresponding shop, and is activated within 15 minutes.

Get3rdPartyShipperResult Get3rdPartyShipper(string sAuthKey, string creationUid)

A get method, using as input the *CreationUID* of the *ThirdPartyShipper* to retrieve the current info of the third party shipper. When the returned fields *Shipper.Active* is true and also *Shipper.Code* has a value, this value [it is the client code] can be used in the method "CreateClientVoucherJob".

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.

The authenticated user needs to be granted the special permissions [Affiliate Advanced] to use the following methods:

AffiliateTrackingPairResult CreateAffiliateTrackingPair(**string** authKey, **string** voucherNumber, **string** trackingId)

The *voucherNumber* and the affiliate *trackingId* fields are required. This method creates a bond/pair using an existing voucher number, already created by the user, and a **unique** affiliate tracking-id. After creating the Affiliate Tracking Pair, Geniki can use it for transport tracking. The tracking-ids are validated against rules given by each affiliate company.

AffiliateTrackingPairResult GetAffiliateTrackingPair(**string** authKey, **string** code)

The *code* field is required, and its value can be either a voucher-number or an affiliate tracking-id. This method returns the affiliate tracking pair created for the requested code.

Objects

Important notice: When developing for your api client you should never error out on new fields. At any time, and without prior notice, new fields may be added to the following objects, but at no time fields or objects will be removed.

Record		
Field	Type	Description
OrderId	string	User defined id
Name	string	Recipient name
Address	string	Recipient address
City	string	Recipient city
Country	string	Recipient country (English country name or ISO 3166 Alpha 2-code, requested for GE service shipments)
Email	String	Recipient email (requested for GE service shipments)
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, in euro, of the cod. If greater than zero it must always have the service "aμ" or "av"
InsAmount	decimal	The amount to be insured. If greater than zero it must have the service "ao"
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 subvoucher, 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

GetPickupJobStatusResult		
Field	Type	Description
Result	int	The result code (explained later)
JobId	long	The id of the pickup type job
Status	String	The status of the order (Pending / Done / Canceled)
StatusDate	DateTime	The updated date of the status.
Vouchers	String[]	An array holding the voucher numbers (if any) for a "done" order

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

CreateCODReleaseResult		
Field	Type	Description
Result	int	The result code (explained later)
VoucherNo	string	The main voucher number of the shipment job
CodAmount	decimal	The new amount, in euro, of the cod.

CreateVoucherReturnOrderResult		
Field	Type	Description
Result	int	The result code (explained later)
Order	VoucherReturnOrder	The created voucher return order

GetVoucherReturnOrderResult		
Field	Type	Description
Result	int	The result code (explained later)
Order	VoucherReturnOrder	An existing voucher return order

VoucherReturnOrder		
Field	Type	Description
VoucherNumber	string	The voucher number
UniqueID	string	Voucher return order uuid [used only as an internal reference]
Reason	string	Reason of the voucher return order

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
ReturningServiceVoucher	string	The returning service voucher created upon delivery
DeliveredAt	string	The delivery target (RECIPIENT / RETURN TO SENDER)

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
ReturningServiceVoucher	string	The returning service voucher created upon delivery
DeliveredAt	String	The delivery target (RECIPIENT / RETURN TO SENDER)

Checkpoint		
Field	Type	Description
StatusCode	string	Checkpoint's immutable status code (see: Checkpoints Status Codes table)
Status	string	Checkpoint's status description text
StatusDate	DateTime	The date and time the status occurred
Shop	string	The shop associated with the checkpoint

CreateVoucherPickupOrderResult		
Field	Type	Description
Order	VoucherPickupOrder	Voucher pickup order created

GetVoucherPickupOrderResult		
Field	Type	Description
Order	VoucherPickupOrder	Voucher pickup order

UpdateVoucherPickupOrderResult		
Field	Type	Description
Order	VoucherPickupOrder	Voucher pickup order updated

CancelVoucherPickupOrderResult		
Field	Type	Description
Order	VoucherPickupOrder	Voucher pickup order canceled

VoucherPickupOrder		
Field	Type	Description
VoucherNumber	string	Voucher number to pickup
UniqueID	string	Voucher pickup unique id [uuid]
IsActive	bool	Voucher pickup active marker [when true the pickup is to be scheduled]
ClientCode	string	The voucher client code (autofilled) that the pickup is to be made from.
PickupDate	string	Voucher pickup scheduled to date
DayQuarter	string	Voucher pickup scheduled to day quaterd
IsFinalized	string	True when voucher pickup has been made
FinalizedAt	DateTime	Voucher pickup date when pickup has been made

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

Shop		
Field	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 (reception)
Active	boolean	If true the shop is active
WorkingHours	string	Working hours of shop/reception

GetLockersListResult		
Field	Type	Description
Result	int	The result code (explained later)
Lockers	LockerInfo[]	An array holding the lockers of Geniki Taxydromiki

LockerInfo		
Field	Type	Description
Id	string	Locker reference id
Name	string	Locker description
Vendor	string	Locker vendor
Address	string	Locker address
City	string	Locker city
Country	string	Locker country
Longitude	decimal	Locker longitude
Latitude	decimal	Locker latitude

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]
IsPaidBySender	bool	Set to <i>true</i> 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]
IsPaidBySender	bool	<i>True</i> 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.
CreationUid	string	Creation request UUID for the client.

ThirdPartyShipperDetails		
Field	Type	Description
VatId	string	The vat-id of the client to be created.
Name	string	The name of the client to be created.
Address	string	The address of the client to be created.
City	string	The city of the client to be created.
Telephone	String	The telephone of the client to be created.
Zip	String	The zip code of the client to be created.
ServicePointCode	String	The 2-digit code of the Geniki shop that will service the shipper [see: <i>Shop.Code</i> field in <i>GetShopsResult</i> of <i>GetShopsList</i> method] - If this field is set, the 3rd-party-shipper client is created to be serviced by the corresponding shop, and is activated within 15 minutes.

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.

AffiliateTrackingPairResult		
Field	Type	Description
Pair	AffiliateTrackingPair	Affiliate tracking pair

AffiliateTrackingPair		
Field	Type	Description
TrackingId	string	Affiliate tracking id
Voucher	string	Voucher number
InsertedAt	DateTime	Date and time of the pair creation

Checkpoints Status Codes

Status Code	Meaning
C_NW	Shipment label created/printed
C_A1	Arrival at Service Point
C_A3	Out for Delivery
C_E1	Return to sender
C_E2	Cancel previous checkpoint: C_E1
C_L1	Routing to other service point
C_L2	Cancel previous checkpoint: C_L1
C_EA_AG	Attempted Delivery - Unknown recipient
C_EA_AK	Attempted Delivery - Damaged
C_EA_AP	Attempted Delivery - Refusal to receive
C_EA_AS	Attempted Delivery - Absent recipient
C_EA_DA	Attempted Delivery - Shipment routing
C_EA_DD	Attempted Delivery - Not Distributed
C_EA_DP	Attempted Delivery - Delivery in 2-3 days
C_EA_EP	Attempted Delivery - Return
C_EA_KS	Attempted Delivery - Delivery Rescheduled
C_EA_LA	Attempted Delivery - Shipment Missorted / Misrouted
C_EA_LD	Attempted Delivery - Wrong Address
C_K8	Departure from Service Point
C_K9	Cancel previous checkpoint: C_K8
C_KK	Registration of shipment details
C_W2	Shipment Delivered
C_W3	Shipment Collected / PickedUp
C_H1	Arrival at hub
C_H2	Departure from hub
C_P1	Registration of delivery details
C_P4	Shipment canceled
C_D2	On hold - Awaiting Pick Up
C_S0	Int'l Shipment Status: Out for Delivery
C_S1	Int'l Shipment Status: In Transit
C_S2	Int'l Shipment Status: Delivered
C_S3	Int'l Shipment Status: Returned
C_S4	Int'l Shipment Status: Parcel is on hold
C_S5	Int'l Shipment Status: Shipment Cancelled
C_S6	Int'l Shipment Status: Shipment Lost
C_S7	Int'l Shipment Status: Shipment Rerouting
C_S8	Int'l Shipment Status: Shipment Cancelled
C_S9	Int'l Shipment Status: On hold - Awaiting Pick Up
C_SA	Int'l Shipment Status: Shipment Damaged
C_SB	Int'l Shipment Status: Shipment Destroyed
C_SC	Int'l Shipment Status: Shipment label created/printed
C_SD	Int'l Shipment Status: Arrival at HUB
C_SE	Int'l Shipment Status: Delivery Rescheduled
C_SF	Int'l Shipment Status: Attempted Delivery

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 CreateCODRelease if not existing/empty voucher number is used CreateVoucherReturnOrder if not existing/empty voucher number is used CreateVoucherPickupOrder/UpdateVoucherPickupOrder if not existing/empty voucher number is used Get/CreateAffiliateTrackingPair if not existing/empty parameters are used
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
710	Validation failed	CreateCODRelease , if voucher does not have a COD service, or new COD service amount is not valid (< 0) CreateVoucherPickupOrder/UpdateVoucherPickupOrder , if pickup date is before today. CreateClientVoucherJob , if clientCode is invalid [or third party sender does not exist / is not active] Get/CreateAffiliateTrackingPair , if parameters values are invalid
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 , CreateVoucherPickupOrder / UpdateVoucherPickupOrder / CreateVoucherReturnOrder , GetVoucherReturnOrder 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**, **CreateClientVoucherJob**, **CreateGroupedVoucherJob**, **CreateCODRelease**, **CreateVoucherPickupOrder**, **CreateVoucherReturnOrder**, **CreateAffiliateTrackingPair**
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 extra 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)
B2	ΠΑΡΑΔΟΣΗ ΣΕ ΔΙΕΥΘ. ΠΑΡΑΛΗΠΤΗ ΝΗΣΙΑ ⁽¹⁾	RECIPIENT LOCATION DELIVERY ISLANDS ⁽¹⁾
BP	D2SP, ΒΑΣΙΚΗ RECEPTION ⁽¹⁾	D2SP SERVICE (reception delivery) ⁽¹⁾
AP	D2SP, ΑΝΤΙΚΑΤΑΒΟΛΗ RECEPTION ⁽¹⁾	D2SP COD (cash payment - reception delivery) ⁽¹⁾
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
TN	ΑΕΡΟΜΕΤΑΦΟΡΑ	NEXT DAY DELIVERY TO ISLANDS
ΧΠ	ΧΡΕΩΣΗ ΠΑΡΑΛΗΠΤΗ	CASH COLLECT (RECEIVER PAYS TRANSPORT FEES)
T1	ΤΑΧΥΔΡ. ΚΙΒΩΤΙΟ N1 - 5 (ΕΩΣ 2 KG)	POSTAL BOX N1 - 5 (UP TO 2 KG)
T6	ΤΑΧΥΔΡ. ΚΙΒΩΤΙΟ N6 - 7 (ΕΩΣ 4KG)	POSTAL BOX N6 - 7 (UP TO 4KG)
TE	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
GE	GTEEC - GT EUROPE E-COMMERCE	GTEEC - GT EUROPE E-COMMERCE

Remarks: At the case that no extra service is selected (so an empty Services field is sent), the normal shipment service is set. Additionally, instead of an empty Services field, the three-character-code service 'STD' can be selected, which is, also, translated to 'no-extra-service' as well.

--

(1) Services may be eligible only for some destinations. More info on www.taxydromiki.com

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 = "αv",
        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 unnecessary 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 = $soap->Authenticate(
                array(
                    'sUserName' => 'UserName',
                    'sUserPwd' => 'Password',
                    'applicationKey' => 'AppKey'
                )
            );
            print_r($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' => "av",
        '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>

```