WEB.systems
solutions

WEB-API

for UNISERV OE products

Release 4.02

Hotline:

English +49 (0) 7231/936-1010
French +49 (0) 7231/936-2020
German +49 (0) 7231/936-3030

e-mail: hotline@uniserv.com

A solution of

UNISERV GmbH
Rastatter Str. 13
75179 Pforzheim
Germany

Tel.: +49 (0) 7231/936-0
Fax: +49 (0) 7231/936-2500
E-Mail: info@uniserv.com
http://www.uniserv.com

Contents

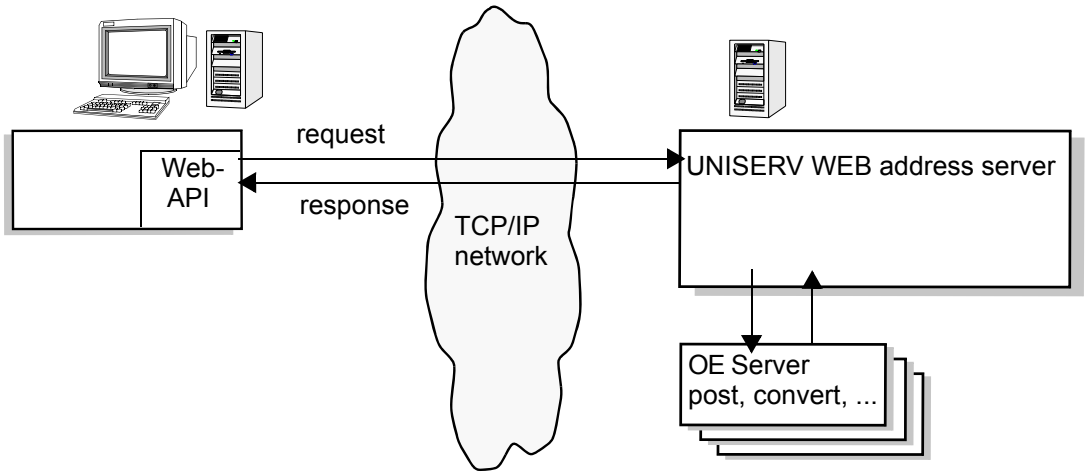
1 UNISERV WEB address server

On one hand, the Internet is being accessed by an increasing number of users and on the other hand, more and more companies become aware of the opportunities offered by this huge marketplace. Whoever discovers the Internet as new channel of distribution will however quickly realise that a shop in the web does not consist of a few nicely designed HTML pages only. Above all, online success also requires intelligent IT solutions which provide error-free customer data at the time as they are entered. Indeed, error-free customer data are essential for credit investigations, allow to assure and speed up delivery, help to reduce costs and strengthen the company's profitability.

In addition to error-free customer data, using the UNISERV address management systems for e-business offers further important features to increase address quality. For instance: online qualification and personalisation of addresses in order to offer products specifically adapted to the customer's 'living environment' (micro-marketing), in order to point out the nearest agency, branch office or distributor, in order to provide the quickest route to reach your location (geo-marketing), or in order to recognise 'old' customers even without customer ID or 'long-term cookie' and to react accordingly as to product offers and terms.

UNISERV provides all address management interfaces required for your e-shop- or e-commerce applications, whether these are based on Microsoft, Linux or any other Unix technology.

System architecture



Based on the UNISERV Adress Service Portal (ASP), your application is able to access the UNISERV WEB address server by way of the WEB-API. This web server collects requests coming from clients, transmits them to the corresponding server of the UNISERV OPEN.edition product line, and returns the results to the clients.

WEB-API encapsulates the requests at the UNISERV WEB address server and provides the server services in a highly user-friendly way, while handling the entire communication between client application and UNISERV WEB address server as well as analysing and processing the received XML document before returning the results to the client application. Where required, a HTTP proxy server is used to set up the communication between the client application and the UNISERV-WEB-address-server. Even when this proxy server requires a user logon, the WEB-API can be used. In this case however, only the so-called 'Basic Authentication' is supported.

1

Using XML allows to provide the request results in an open format, which means the the services of the UNISERV WEB address server can easily be integrated into the user environment.

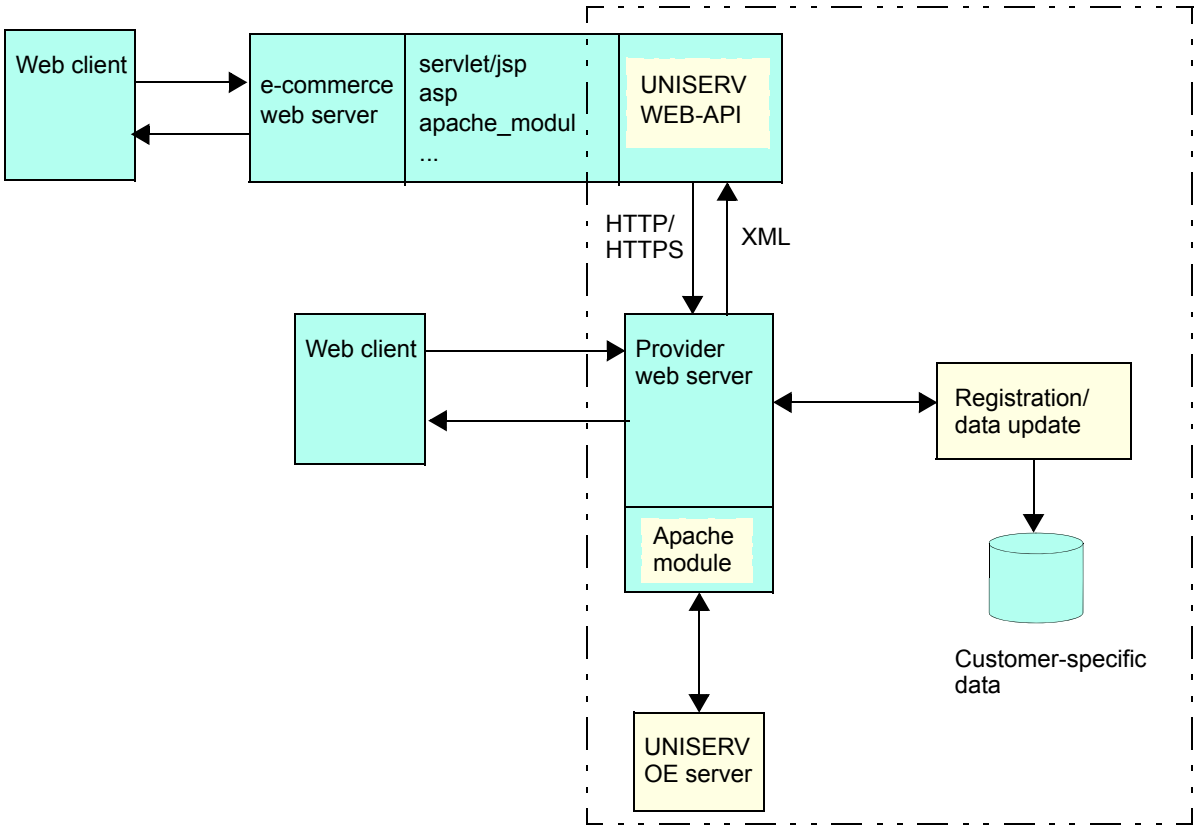
The web API contains interfaces for:

- Java integration
- Perl integration
- a COM interface
- PHP integration

Thus, the UNISERV WEB address server not only allows for quick and easy integration into your application, but also for easy system extensions and flexible customisation according to user requirements. This covers various options from a cost-efficient access to the server at UNISERV (or from any other service provider) to using your own server.

Using standard Internet protocols (HTTP/HTTPS) allows to carry out the communication between client and server in high-security environments protected by firewall computers. A further benefit of these standards being used by the WEB-API is the support for encryption authentication.

Interaction of sub-components



A customer sets up a connection with a provider of an e-commerce web server while indicating his address. Within the corresponding application, this address will then be forwarded by the provider of the e-commerce web server to the UNISERV WEB address server (or to a web server of another service provider), where it is checked and eventually corrected. No direct connection between the customer and the UNISERV WEB address server exists at any time.

In addition, a web client allows the provider of the e-commerce web server to change settings controlling the system reaction to incoming requests for the UNISERV WEB address server.

Open XML format

The open XML format allows the UNISERV OPEN.edition products to be integrated even when there is no WEB-API (for instance if e-commerce is not based on either Java, Perl, PHP or COM).

2 Processing a CSV File

The Asp Batch Client can be used to perform a postal validation for a CSV file.

The client reads the file and passes individual addresses to the Uniserv WEB-address-server for validation. Two result files are generated and returned:

- One file contains lines with the addresses checked and found correct.
- The other contains the addresses that - even with fault-tolerant methods - could not be checked or corrected.

The addresses not checked are provided with a status code, which provides information on the reason why the address could not be validated.

What do I have to do to check my address file?

- Step 1:

  You will find the software solution tailored to your platform under:

  http://asp.uniserv-online.de/download/csv/csv.shtml

  Download the appropriate file and double-click to start the installation.

  To start the client, the 'AspBatchClient' icon is created in the 'ASP Batch Client' program group.

- Step 2:

  Adjust the file <installation directory>/client.xml.

- The file defines the parameters needed to communicate with the Uniserv WEB-address-server, such as the URL, registration data (to be found in the registration e-mail) and, where applicable, the proxy server (consult your IT department).

```
<client-config>

...<uniserv_asp_url>
http://asp.uniserv-online.de/components/address
</uniserv_asp_url>

<uniserv_asp_reg_nr>
</uniserv_asp_reg_nr>

<uniserv_asp_passwd>
</uniserv_asp_passwd>

<!--the name of the proxy server --><proxy_host></proxy_host>

<!--the port number of the proxy server --><proxy_port></proxy_port>

<!--the the authentication data (domain\username:password)
for the proxy server --><proxy_auth></proxy_auth>

....</client-config>
```

  The configuration file also describes the address file: For each address element (to be validated), it specifies its position (column) within the address file.

```
<client-config>

<input_file>
</input_file>

....
```

```
<file_format>
<delimiter>;</delimiter>
<!--describe your input address file:for each existing address element
definethe column position between 1 and n.The both output files,
<valid_output_file> and<not_corrected_output_file> would be written
using the sameformat as the input file -->
<country></country>
<zip></zip>
<city></city>
<!--use either street or streetname + houseNumber -->
<street></street>
<streetName></streetName>
<houseNumber></houseNumber>
<subBuildingName></subBuildingName>
<buildingName></buildingName>
<county></county>
</file_format>
</client-config>
```

The two result files

```
...
<valid_output_file></valid_output_file>
<not_corrected_output_file></not_corrected_output_file>...
```

keep the same format as the input file. Any data not related to the address is simply kept unchanged.

- Step 3:

  Test our example

  For illustration purposes, the file 'address_file.txt' was used as input file:

  ```
  DE;;pforzheim;rastatterstr 13;uniserv gmbh;uniserv ansprechpartner;
  DE;75;pforzheim;westliche;meine firma;meine ansprechpartnerin;
  BE;;borgerhout;de borrekenstraat 122;;;
  GB;sw1a2aa;;10;prime minister;;
  ```

  The configuration file supplied (client.xml) and this input file have been adjusted to each other. All you need to do is to enter valid registration data in the configuration file under uniserv_asp_reg_nr and uniserv_asp_passwd. If your Internet connection requires a proxy server, please also enter the proxy settings accordingly.

  Start the client from the 'ASP BatchClient' program group using the 'AspBatchClient' icon or from the console with the command 'AspBatchClient client.xml'.

  Check the two result files.

  The file 'valid_address_file.txt' should contain the following lines:

  ```
  DE;75179;Pforzheim;Rastatter Str. 13;uniserv gmbh;uniserv ansprech-
  partner;;
  BE;2100;DEURNE (ANTWERPEN);De Borrekensstraat 122;;;;
  GB;SW1A 2AA;LONDON;10 Downing Street;prime minister;;;
  ```

  The file 'not_corrected_address_file.txt' should contain:

```
102-15;DE;75;pforzheim;westliche;meine firma;meine ansprechpart-
nerin;;
```

The status code 102-15 means that this address was not corrected because the street has several sections with different postcodes, which can only be identified through the house number.

- Step 4:

  Start your validation procedure. To do so, adjust the ’client.xml’ configuration file to your address file. Then start the client from the 'ASP BatchClient' program group using the 'AspBatchClient' icon or from the console with the command 'AspBatchClient client.xml'.

If you have any questions, please send us an e-mail to hotline.asp@uniserv.com.

3 WEB-API

With WEB-API, UNISERV currently offers interfaces for Java, Perl, PHP and COM. Disregarding the underlying programming language, using the WEB-API can be divided into the following steps:

– Creation of an interface object
– Initialisation of the interface object
– Setting the input arguments (properties)
– Execution of check function
– Evaluation of return values
– Readout of output arguments (properties)

3.1 Standard Properties

Disregarding the programming language used, WEB-API provides the following standard properties.

Property	Description	Input	Output	Restrictions
RequestErrorMsg	Additional description text in case of errors (UNI_REQUEST_ERROR only)		X	
RequestHint	Additional information for return value		X	
Language	Language for messages (d, e or f)	X		Sets the language for messages. If this property is not set, the server messages are returned in the language used to register the service. Also, if this property is not set, the error messages of the individual client components are returned in english.
Transliteration	Sets whether the result is output in the original character set or in the (limited) ISO-8859-1 character set. Transliteration = false, default setting	X	X	
Address properties				

3

Property	Description	Input	Output	Restrictions
AddressStatus	Classifies the address as correct, corrected, ambiguous or erroneous		X	
AddressStatusHint	Additional information for AddressStatus		X	
AddressMode	Sets whether the address is entered using line-oriented or field-oriented properties. – The field-oriented input mode UNI_FIELD_MODE uses properties such as 'City', 'Zip', 'Street', ... – The line-oriented input mode UNI_LINE_MODE uses the properties 'Line1', 'Line2', ... Default: UNI_FIELD_MODE	X		
BuildingName	Building name		X	GB only
CarRegistration	License plate country code		X	
City	Town name	X	X	
CityDistrict	Town district		X	FR: Contains the place of residence when that and the delivery town (City) are different.
CityQuality	Similarity measure (0 - 100) for the town. This describes the quality of a search result, with 100 meaning full concordance after standardization. On the other hand, a similarity measure of 70 or less means the result is dubious.		X	
CommunityCode	Municipality key, administration code		X	
Country	Country code (2 or 3 digits acc. to ISO or license plate)	X	X	Valid output only 2 digits acc. to ISO.
CountryName	English name of the country		X	
County	County in Great Britain, province in Spain, Italy and the Netherlands, canton in Switzerland	X	X	Valid in entry only in CA, GB and USA.
HouseNumber	House number	X	X	Valid in entry only if StreetInputMode = UNI_FIELD_MODE ist

Property	Description	Input	Out-put	Restrictions
HouseNumberMatch	Sets whether a house number validation is to be performed (= TRUE) or not (= FALSE). In a house number validation, the server returns UNI_SUSPECT / UNI_UNKNOWN_HOUSE_NUMBER when the house number passed does not exist. House number validation is disabled by default.	X	X	AT and DE only
Line1	Address line for line-oriented input. AddressMode must be set to UNI_LINE_MODE.	X	X	AT, BE, CH, DE, DK, ES, FI, FR, GB, IT, LU, NL, NO, PT, SE
Line2	Refer to Line1	X	X	Refer to Line1
Line3	Refer to Line1	X	X	Refer to Line1
Line4	Refer to Line1	X	X	Refer to Line1
Line5	Refer to Line1	X	X	Refer to Line1
Line6	Refer to Line1	X	X	Valid in entry only for GB
Organisation	Company name		X	GB only
POBoxNumber	PO box number (see section 3.4.3).	X	X	
POBoxWithoutNumber	Indicates whether or not the address is a PO box address without PO box number (see section 3.4.3).	X	X	
Street	Street name with house number and/or P.O.Box number	X	X	Valid in entry only if StreetInputMode = UNI_LINE_MODE
Street1	Further address line	X		GB only
Street2	Further address line	X		GB only
StreetInputMode	Sets whether the entered street information is available in the property Street alone (UNI_LINE_MODE) or in the properties StreetName and HouseNumber separately (UNI_FIELD_MODE). Default setting is UNI_LINE_MODE.	X		
StreetName	Street name without house number and/or P.O.Box number	X	X	Valid in entry only if StreetInputMode = UNI_FIELD_MODE ist

Property	Description	Input	Output	Restrictions
StreetQuality	Similarity measure (0 - 100) for the street. This describes the quality of a search result, with 100 meaning full concordance after standardization. On the other hand, a similarity measure of 65 or less means the result is dubious.		X	
SubBuildingName	Access/apartment		X	GB only
Zip	Postcode	X	X	
Coordinates properties				Requires corresponding license
CoordinateStatus	Additional information for the properties XCoordinate and YCoordinate		X	
XCoordinate	Latitude		X	
YCoordinate	Longitude		X	
Freight code properties				D only, requires corresponding license
Cargo	First 11 digits of freight code		X	
CargoStatus	Additional information for property Cargo		X	
KGS properties				D only, requires corresponding license
KGS22	22-digit district/municipality key		X	
KGS22Status	Additional information for property KGS22		X	
Personicx™ Geo-Properties				D only, requires corresponding license
PrizmCounty	Regierungsbezirk (administrative region)		X	
PrizmDistrict	Kreis (administrative area)		X	
PrizmHouseholds	Number of private households in statistical district		X	
PrizmMilieu	GeoCluster		X	
PrizmMunicipality	Municipality		X	
PrizmRegioType	Regiotype		X	
PrizmResidentalEnv	Evaluation of residential environment		X	
PrizmState	Bundesland (federal entity)		X	
PrizmStatisticalArea	Statistical district		X	

Property	Description	Input	Out-put	Restrictions
PrizmStatus	Additional information for the Per-sonicx™ Geo properties		X	
Bank-Properties				All bank properties require a corresponding license
Account	Account no. to be checked	X		
AccountStatus	Status of account no. verification		X	
BankCity	The 'postally valid' town name, i.e the name used by the post to de-liver mail.		X	
BankCode	Bank code	X	X	
BankCountry		X		
BankName	Bank name	X	X	
BankNetwork	Network used by the bank (e.g. giro centres and savings banks).		X	
BankStatus	Classifies bank reference as cor-rect, ambiguous or erroneous.		X	
BankType	Specifies whether the bank is a head office (UNI_BANK_HEADOFFICE) or a branch office (UNI_BANK_BRANCHOFFICE).		X	
BankZip	Postcode of bank town.		X	
ClearingRegion	Area/region of operation for pay-ment transactions, e.g. Baden-Württemberg.		X	
SWIFT	SWIFT code ('Society for World-wide Interbank Financial Tele-communication'). This allows associated banks to exchange in-formation in order to simplify cross-border money transfers by way of an identification code - BIC (Bank Identifier Code).		X	
IBAN Properties				All IBAN properties require a corresponding license
IBAN	”International Bank Account Number” to be checked	X		
IBANStatus	Status of IBAN verification		X	
Credit card properties				All credit card properties require a corresponding li-cense
CreditCardNetwork	Credit card network, e.g. Visa, Eu-rocard		X	

Property	Description	Input	Out-put	Restrictions
CreditCardNr	Credit card no. to be checked	X		
CreditCardNrStatus	Status of credit card no. verifica-tion		X	

3.2 Methods

clear — Resets all properties of the client component to default settings. In case of multiple use of a component instance, this method allows to ensure that all properties are set to sensible values, even when the client application does not explicitly reset the properties or does not set them to specific values.

init — Initialisation of component. This method has to be called before any other methods are called or properties are set or read. The specific syntax for the selected programming language is described in the corresponding chapters for that language.

validate — Calling the validation procedures using the previously set properties.

3.3 Meaning of Return Values

When called, all methods provide a return value to classify the result of the call of the method. A further feature of the interface object (Property hint) provides a more detailed description of the return value.

Return values

The return value of the validate() method does not give information on the validation process or on the data quality. This information is provided by the respective status.

Return value	Request Hint	Explanation
UNI_REQUEST_OK		Method was executed without errors
UNI_REQUEST_AMS_ERROR		No verification was carried out, AMS-related
	UNI_REQUEST_ERROR	An AMS-internal error has occurred
	UNI_REQUEST_XML_ERROR	A WEB-API -internal error has occurred
	UNI_REQUEST_BAD_URL_ERROR	The entered AMS address is invalid
	UNI_REQUEST_COM_ERROR	Communication error
UNI_REQUEST_ACCESS_DENIED		No verification was carried out due to missing rights
	UNI_REQUEST_BAD_CUSTOMER_NR	Registration number is invalid
	UNI_REQUEST_BAD_CUSTOMER_PASSWD	Password is invalid

3.4 Postal Validation

Postal address validation requires the town and/or the postcode. further address elements are optional. The address can be entered in either lines or fields.

In field-oriented mode, AddressMode has to be set to UNI_FIELD_MODE (default value). In this case, properties such ’Zip, City, Street…’ are passed to the server and postally validated.

In line-oriented mode, AddressMode has to be set to UNI_LINE_MODE. In this case, the properties ’Line1’ to ’Line6’ are passed to the server, structured and postally validated. In addition to the individual address properties, the output also includes the formatted address. For details on the countries for which address formatting is available, please refer section 3.1

In field-oriented mode, two modes are available to enter the street:

Line mode — The entire street information (street name with house no. and/or P.O. Box no.) is passed in the Street property.

Field mode — The street information is split up between the StreetName and HouseNumber properties.

The street entry mode is set in the StreetInputMode property. Default: line mode.

To enter UK addresses, refer to see section 3.4.1

Address properties are only available if the validate method has returned 'UNI_REQUEST_OK'. The AddressStatus and AddressStatusHint properties are used to classify the address as described below.

Address Status	AddressStatusHint	Explanation
UNI_OK		The passed address was correct.
	UNI_WAS_CORRECT	Address was correct.
	UNI_CORRECTED	Address was corrected.
	UNI_STR_NOT_CHECKED	Street not checked due to missing street directory.
	UNI_VALID_POBOX_ADDRESS	The address is a valid PO box address.
	UNI_POBOX_NUMBER_NOT_CHECKED	The address is a PO box address. However, the PO box was not checked because the reference data does not include PO box data.
UNI_SUSPECT		Address was changed with uncertainty.
	UNI_SUSPECT_ZIP	The first two postcode digits were changed.
	UNI_SUSPECT_CITY	Town was found by way of archived town name.
	UNI_SUSPECT_STREET	Street was found by way of old street name.
	UNI_SUSPECT_POBOX	No longer used!
	UNI_SUSPECT_ADDRESS	Not enough information.
	UNI_STR_NO_INPUT	No street entered, address is incomplete.

Address Status	AddressStatusHint	Explanation
	UNI_UNKNOWN_HOUSE_NUMBER	With house number validation enabled, value returned if the house number passed does not exist.
	UNI_HNO_NO_INPUT	Pas de no. de maison indiqué, adresse incomplète.
UNI_SELECTION		Address is ambiguous. Selection list mode[*1)].
	UNI_AMBIGUOUS_CITY	Town selection list was created.
	UNI_AMBIGUOUS_STREET	Street selection list was created.
	UNI_AMBIGUOUS_BOX	House no. selection list was created.
	UNI_AMBIGUOUS_ORGANISATION	Company selection list was created :
	UNI_AMBIGUOUS_BUILDING	Building selection list was created.
	UNI_AMBIGUOUS_SUB_BUILDING	Building access selection list was created (GB only).
	UNI_AMBIGUOUS_DEPENDENT_STR	Dependent street selection list was created (GB only).
	UNI_AMBIGUOUS_DEPENDENT_LOCALITY	Dependent locality selection list was created (GB only).
	UNI_AMBIGUOUS_CITY_DISTRICT	Town district selection list was created (GB only).
	UNI_AMBIGUOUS_ADDRESS	Address selection list was created.
	UNI_AMBIGUOUS_GOVERNMENT_DEPARTMENT	Public authority selection list was created (CA only).
	UNI_AMBIGUOUS_UNIT_POBOX	PO box No. or House No. supplement selection list was created (CA only).
	UNI_AMBIGUOUS_ROUTE_SERVICE	Route Services selection list was created (CA only).

Address Status	AddressStatusHint	Explanation
UNI_BAD_INPUT		No unambiguous address based on the input was found.
	UNI_BAD_ADDRESS_MODE	In case of line-oriented input for a country for which address formatting is not available. Refer to Line1 in the property table in section 3.1
	UNI_UNKNOWN_ZIP	Unknown postcode.
	UNI_UNKNOWN_CITY	Unknown town.
	UNI_UNKNOWN_STREET	Unknown street.
	UNI_UNKNOWN_ADDRESS	Unknown address.
	UNI_UNKNOWN_COUNTRY	The input country code is unknown. Valid entries are 2 or 3 letter country codes according to ISO.
		No selection list mode:
	UNI_AMBIGUOUS_CITY	Ambiguous town
	UNI_AMBIGUOUS_STREET	Ambiguous street.
	UNI_AMBIGUOUS_BOX	Ambiguous house no.
	UNI_AMBIGUOUS_ORGANISATION	Ambiguous company (GB only).
	UNI_AMBIGUOUS_BUILDING	Ambiguous building.
	UNI_AMBIGUOUS_SUB_BUILDING	Ambiguous building access (GB only).
	UNI_AMBIGUOUS_DEPENDENT_STR	Ambiguous dependent street (GB only).
	UNI_AMBIGUOUS_DEPENDENT_LOCALITY	Ambiguous dependent locality (GB only).
	UNI_AMBIGUOUS_CITY_DISTRICT	Ambiguous town district (GB only).
	UNI_AMBIGUOUS_ADDRESS	Other ambiguity. This mainly occurs when processing British addresses.
	UNI_AMBIGUOUS_GOVERNMENT_DEPARTMENT	Public authority selection list was created (CA only).

Address Status	AddressStatusHint	Explanation
	UNI_AMBIGUOUS_UNIT_POBOX	PO box No. or House No. supplement selection list was created (CA only).
	UNI_AMBIGUOUS_ROUTE_SERVICE	Route Services selection list was created (CA only).
	UNI_UNKNOWN_POBOX_NUMBER	Invalid PO box number.
	UNI_LACK_OF_POBOX_NUMBER	'POBoxWithoutNumber' was set for the input, but PO box numbers have to be specified for that town.
	UNI_CITY_WITHOUT_POBOX	One input address was a PO box address, but there are no PO boxes in that town.
	UNI_AMBIGUOUS_POBOX_NUMBER	The PO box number is ambiguous, as no postcode was specified for the town.
	UNI_MIXED_STREET_POBOX	The input contains both a street and a PO box. This combination is not permitted and is therefore rejected by the server.
UNI_AMS_ERROR		No validation, AMS-related.
	UNI_NO_DATA	No data for input country.
	UNI_ERROR	OE-internal error.
UNI_ACCESS_DENIED		No validation, missing rights.
	UNI_NO_LICENSE	No license for postal validation.

*1) Registration includes the Create Selection List option. Otherwise, an error is returned in case of ambiguous addresses (UNI_BAD_INPUT).

3.4.1 Postal validation of addresses in the UK

British addresses can be made up of different elements, some of which may not necessarily exist. An address can contain the following address elements, their order having to be observed:

- Organisation
- Sub-Building
- Building
- Building Number
- Dependent Street
- Street
- Dependent Locality
- Locality
- City
- County
- ZIP

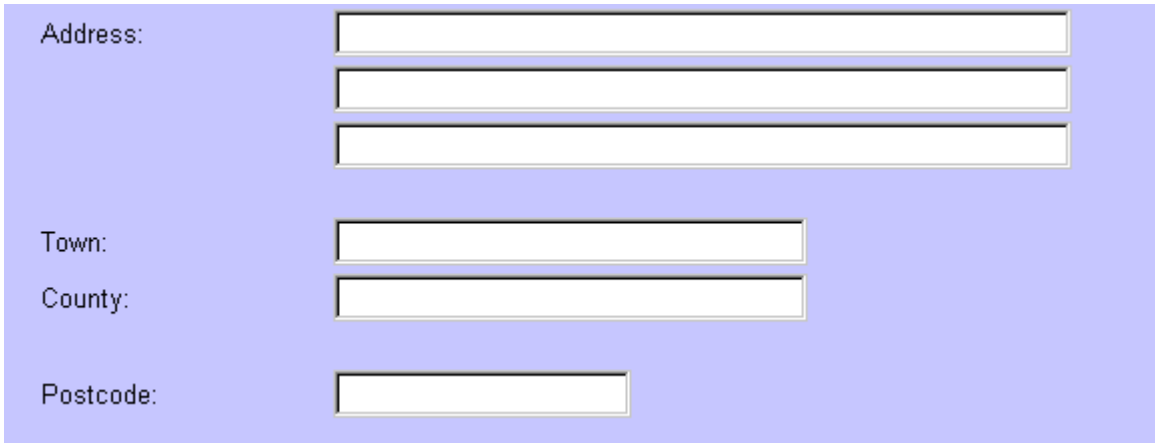
For the verification of British addresses, UNISERV recommends to work in line mode only (StreetInputMode = UNI_LINE_MODE) which is also the default setting.

For the postal validation of an address, the address elements ‘City‘, ‘County‘ and ‘ZIP‘ (where required) have to be assigned to input properties having the same name. Further address elements can be assigned to the input properties ‘Street‘, ‘Street1‘ and ‘Street2‘. However, the order indicated above has to be observed in order to obtain sensible results from the postal validation.

After calling the validate method, the result of the postal validation is provided in the following output properties: ‘Organisation‘, ’SubBuildingName’ ‘BuildingName‘, ‘Street[*1)]‘, ‘CityDistrict‘, ‘City‘, ‘County‘ and ‘Zip‘.

*1) ’Street’ also includes ’Dependant Street’ where applicable.

A typical British address form could be designed as follows:



In that context, the input fields have to be assigned to the following input properties:

1. Address line Property ‘Street‘
2. Address line Property ‘Street1‘
3. Address line Property ‘Street2‘
4. Town Property ‘City‘
5. County Property ‘County‘
6. Postcode Property ‘Zip‘

The first three lines can be used to enter ’any’ address elements, provided the required order is observed. In the lines 4 to 6, only fixed address elements are to be entered by the user.

3.4.1.1 Rapid Entry

The rapid entry option enables searching for a complete address with incomplete data. The only input information required is the postcode (’Zip’ property) and the house number (’HouseNumber’ property). All the input properties have to be empty. Rapid entry requires field mode (StreetInputMode = UNI_FIELD_MODE, refer to section 3.1).

3.4.2 Selection Lists

Depending on the type of registration, a selection list will be created (selection list mode) or not in case of ambiguous addresses. The three selection lists below are available for all countries:

- Town selection list
- Street selection list
- House number selection list

For the UK, the following selection lists are also available:

- Company selection list (Organisation)
- Building selection list (Building)
- Building access selection list (Sub Building)
- Dependent street selection list (Dependent Street)
- Dependent locality selection list (Dependent Locality)
- Town district selection list (City District)
- Address selection list (Address)

Additionally, the following selection lists are available for Canada:

- Public authority selection list (Government Department)
- Building selection list (Building)
- PO box No. or House No. supplement selection list (Unit Pobox)
- Route Services selection list
- Address selection list (Address)

The maximum length of selection lists is set during the registration. Should a selection list for an ambiguous address contain more entries than this registered limit, the selection list will **not** be trimmed to the set length, but an error UNI_BAD_INPUT will be returned instead.

Methods for processing selection lists

getLength	Returns length of selection list.
getArgumentNames	Returns the argument names of the list. Each selection list contains a status argument which clearly identifies individual lines.
getArgumentValue	Returns the value of an individual argument for the desired position within the list (the position lying between 0 and length of list).

To obtain all address properties, select an address from the selection list and resume the postal validation. For details on selecting an address from the selection list, also refer to the programming language section. Please note that several consecutive selection lists can be created (e.g. first a town selection list and then a street selection list).

3.4.3 PO Box Validation

As of WEB-API Rel. 3.00, the validation of PO box addresses is possible in several countries. Please note that **either** a street address **or** a PO box address can be validated within a request. A validation of mixed addresses (both street address and PO box address) is not possible within a singles request.

A PO box address is defined as such by setting the input properties 'POBoxNumber' and 'POBoxWithoutNumber' or by passing the country-specific PO box term and the PO box number in the 'Street' input property.

In the output, the PO box information is provided in the output properties ’POBoxNumber’ and ’POBoxWithoutNumber’ as well as, together with the country-specific PO box term and the PO box number, in the ’Street’ output property.

Depending on the environment, this allows to store the information both in separate database fields (for street addresses and PO box addresses) and in common fields for both address formats.

3

3.5 Coordinates and Freight Code

Coordinates contain the longitudes and latitudes in decimal notation multiplied by 100,000. For instance, the geo-decimal notation 18,050,000 represents a value of 180 degrees and 30 minutes. The underlying date is stored in WGS84.

The freight code contains information on the route of a freight consignment. 14 digits have to be encoded and indicated on the label, both as bar code and as combination of digits. The code consists of two parts:

1. The first part (digits 1-11) represents the geographical information which consists of the postcodes of a street number and of the house number. This part can either be determined dynamically for each postage optimisation or it can be permanently stored in the address database.
2. The second part (digits 12-14) consists of the product code (type of consignment) and a check digit. This part can only be set and determined at the time of the postage optimisation.

GeoCoding properties are available for unambiguous addresses only. The property CoordinateStatus is used to classify the validity of the properties XCoordinate and YCoordinate. In the same way, the property CargoStatus is used to classify the validity of the property Cargo.

Possible values:

Value	Explanation
UNI_STAT_NOT_SUPPORTED	The corresponding information is not supported for the indicated country.
UNI_STAT_NO_LICENSE	The corresponding information is not licensed for the indicated registration ID.
UNI_STAT_SERVICE_NOT_AVAILABLE	The information could not be determined due to an internal error of the WEB-address-server.
UNI_STAT_NOT_AVAILABLE	The information could not be determined for the indicated address although the service is licensed for the indicated registration ID and the indicated country is supported.
UNI_STAT_HOUSE_NUMBER	The information is valid for the single house number.
UNI_STAT_STREET_DIVISION	The information is valid for the street section.
UNI_STAT_STREET	The information is valid for the street.
UNI_STAT_ZIP_CITY_DISTRICT	The information is valid for the postcode and the town district.
UNI_STAT_CITY_DISTRICT	The information is valid for the town district.
UNI_STAT_ZIP	The information is valid for the postcode.
UNI_STAT_CITY	The information is valid for the entire town

3.6 Personicx™ Geo

In the light of nearly 40 million private households in Germany, it is impossible to obtain comprehensive personalised information for each one of them.

However, an analysis of the residential environment of an address allows to determine whether the person belonging to the address is a potential customer or not.

For that purpose, Personicx™ Geo is used as regional segmentation system providing typical/characteristic information on the smallest possible geographical level for all of Germany.

Personicx™ Geo data can be combined in a highly flexible way and are particularly used for analyses in the fields of database marketing and customer relationship management (CRM).

In Germany, the Personicx™ Geo Lifestyle micro-market segmentation is used by companies from a wide variety of industries including banks, insurance companies, telecommunications, media, FMCG/retail, automotive and others. Personicx™ Geo provides essential information used for strategic core fields such as customer acquisition, customer loyalty or risk management. Furthermore, they allow to significantly improve marketing and sales control.

Personicx™ Geo properties are available for unambiguous addresses only and require a corresponding license. The property PrizmStatus is used to classify the validity of the remaining properties.

Possible values:

Value	Explanation
UNI_STAT_NOT_SUPPORTED	The corresponding information is not supported for the indicated country.
UNI_STAT_NO_LICENSE	The corresponding information is not licensed for the indicated registration ID.
UNI_STAT_SERVICE_NOT_AVAILABLE	The information could not be determined due to an internal error of the WEB-address-server.
UNI_STAT_NOT_AVAILABLE	The information could not be determined for the indicated address although the service is licensed for the indicated registration ID and the indicated country is supported.
UNI_STAT_HOUSE_NUMBER	The information is valid for the single house number.
UNI_STAT_STREET_DIVISION	The information is valid for the street section.
UNI_STAT_STREET	The information is valid for the street.
UNI_STAT_ZIP_CITY_DISTRICT	The information is valid for the postcode and the town district.
UNI_STAT_CITY_DISTRICT	The information is valid for the town district.
UNI_STAT_ZIP	The information is valid for the postcode.
UNI_STAT_CITY	The information is valid for the entire town

As a general rule, the more complete the entered address information is, the more detailed the results provided by Personicx™ Geo will be. Thus, keep in mind that a number of larger cities require entering the street and occasionally even the house number to obtain comprehensive results.

The Personicx™ Geo logic requires, where possible, complete entries of the address elements to be supplemented with statistical data. The assignment of a GeoCluster will illustrate the importance of this point. A GeoCluster can only be assigned to a very small geographical unit - called micro-cell. In a town or a street with several different GeoClusters, there will be no precise assignment as this would contradict the fundamental idea of precision marketing. The corresponding information fields will therefore not be filled in such a case. Similarly, the same applies to the evaluation of the residential environment. If there are two or more residential environments in street, Personicx™ Geo will not carry out an assignment unless the house number is provided.

3.6.1 PrizmRegioType

The Regiotype is determined for each municipality in Germany by way of its centrality and its urbanisation level. The calculation of the centrality is based on a weighted evaluation of a number of variables such as availability of medical care, employment data, public administration, supplies, educational facilities and culture. Major centres have the highest centrality and medium centres have an above-average centrality.

Possible values:

12	Agglomeration area - major centre
13	Agglomeration area - medium centre
14	Agglomeration area - urban-rural trading zone
22	Urban area - major centre
23	Urban area - medium centre
24	Urban area - urban-rural trading zone
32	Rural area - major centre
33	Rural area - medium centre
34	Rural area - municipality

Agglomeration area - major centre

A major centre is assigned to an agglomeration area if the density of population - together with the surrounding municipalities - exceeds 400 inhabitants per km² and if the total population exceeds 500,000 inhabitants.

Agglomeration area - medium centre

A medium centre is assigned to an agglomeration area if the density of population - together with the surrounding municipalities - exceeds 400 inhabitants per km² and the total population including the surrounding municipalities exceeds 500,000 inhabitants, or if the share of its commuters towards a major centre of the agglomeration area exceeds 10% of its working population subject to social insurance contributions.

Agglomerationsraum - urban-rural trading zone

A municipality is classified as urban-rural trading zone in an agglomeration area if its centrality is not above average and if the share of its commuters towards a centre of the agglomeration area exceeds 10% of its working population subject to social insurance contributions.

Urban area - major centre

A major centre is assigned to an urban area if the density of population - together with the surrounding municipalities - ranges from 250 to 400 inhabitants per km² and its total population ranges from 100,000 to 500,000 inhabitants.

Urban area - medium centre

A medium centre is assigned to an urban area if the density of population - together with the surrounding municipalities - ranges from 250 to 400 inhabitants per km² and its total population ranges from 100,000 to 500,000 inhabitants, or if the share of its commuters towards a major centre of the urban area exceeds 10% of its working population subject to social insurance contributions.

Urban area - urban-rural trading zone

A municipality is classified as urban-rural trading zone in an urban area if its centrality is not above average and if the share of its commuters towards a centre of the urban area exceeds 10% of its working population subject to social insurance contributions.

Rural area - major centre

A major centre is assigned to a rural area if the density of population - together with the surrounding municipalities - is below 250 inhabitants per km² and its total population does not exceed 100,000 inhabitants.

Rural area - medium centre

A medium centre is assigned to a rural area if the density of population - together with the surrounding municipalities - is below 250 inhabitants per km² and its total population does not exceed 100,000 inhabitants and, additionally, if there are no clear commuter relations with a major centre of an agglomeration area or urban area.

Rural area - municipality

A municipality is classified as rural municipality in a rural area if its centrality is not above average and if there are no clear commuter relations with a centre of an agglomeration area or urban area.

3.6.2 PrizmMilieu

A GeoCluster is based on specific socio-demographic and socio-economic area data which are condensed to "milieu areas" by way of sophisticated statistical methods. GeoClusters are characterised by different household and lifestyle structures.

Possible values:

11	Highly established upper class
12	Established bourgeois GeoCluster
13	Established status-oriented GeoCluster
14	Advance-oriented middle class
15	Active urban GeoCluster
16	Advance-oriented urban GeoCluster
17	Well-established middle class
18	Urban middle class
19	Young established middle class
20	Traditional middle class
21	Petty-bourgeois working class GeoCluster
22	Active petty-bourgeois GeoCluster
23	Status-oriented working class GeoCluster
24	Urban traditional working class GeoCluster
25	Rural conformist GeoCluster
26	Urban nonconformist GeoCluster
27	Simple working class GeoCluster
28	Simple petty-bourgeois GeoCluster
29	Older low-status GeoCluster
88	Undefined
99	Business area

Highly established upper class

The typical education level in this GeoCluster is a university degree. Typical professions include self-employed, managers and liberal professions.

All age groups as of 35 years are represented, in particular the 55 years and older age group. People in this GeoCluster generally have less children than average and are mainly married. The typical household consists of one or two persons and lives in a condominium.

The net monthly income is 4,000 € and more.

Financial products such as warrants, shares, investment funds and credit cards are strongly used. Typical representatives of this GeoCluster are customers at large banks and have a private health insurance, a life insurance over 100,000 € as well as real-estate for old-age protection.

Established bourgeois GeoCluster

The typical education level in this GeoCluster is upper secondary education. Typical professions include managers, salaried employees and self-employed.

The dominant age groups in this GeoCluster range from 25 to 55. People in this GeoCluster have less children than average and they are married, divorced or separated. The typical household consists of one or two persons and lives in a condominium.

The net monthly income is 3,000 € and more.

Warrants, investment funds and credit cards are strongly used. Typical representatives of this GeoCluster a life insurance over 100,000 € as well as real-estate for old-age protection.

Established status-oriented GeoCluster

The typical education level in this GeoCluster is upper secondary education. Typical professions include managers, civil servants, salaried employees and self-employed.

The dominant age groups in this GeoCluster range from 35 to 63. People in this GeoCluster have less children than average and they are married. The typical household consists of one or two persons and lives in a condominium or in a rented or owned house.

The net monthly income is 3,000 € and more.

Financial products, mainly warrants and shares, are used more than average, while the use of credit cards is only average. Typical representatives of this GeoCluster have a private health insurance, a life insurance over 100,000 € as well as real-estate for old-age protection.

Advance-oriented middle class

The typical education level in this GeoCluster is secondary and upper secondary education. Typical professions include managers, salaried employees and self-employed.

The dominant age groups in this GeoCluster range from 35 to 55. People in this GeoCluster typically have two or more children and they are married. The typical household consists of four or more persons and usually lives in an owned house.

The net monthly income is 3,000 € and more.

Financial products such as warrants, shares and credit cards are used above average. Typical representatives of this GeoCluster have a private health insurance, a life insurance over 100,000 € as well as real-estate for old-age protection.

Active urban GeoCluster

The typical education level in this GeoCluster is upper secondary education. Typical professions include managers, salaried employees and university students.

The dominant age group in this GeoCluster is 55 and higher. People in this GeoCluster have few or no children. The typical household consists of one or two persons and lives in a condominium.

The net monthly income is 3,000 € and more.

Financial products such as shares and investment funds are frequently used in this GeoCluster. Credit cards are used more than average. Typical representatives of this GeoCluster favour instalment sales, usually have a private health insurance and rely on real-estate for old-age protection.

Advance-oriented urban GeoCluster

The typical education level in this GeoCluster is upper secondary and university education. Typical professions include managers, liberal professions, self-employed and university students.

The dominant age group in this GeoCluster is below 35. People in this GeoCluster typically have no children and they are unmarried. The typical household consists of one person and lives in a rented flat.

The income distribution is very heterogeneous. The net monthly income is either below 1,000 € or above 4,000 €.

Financial products such as warrants and credit cards are frequently used. Typical representatives of this GeoCluster are often customers of large banks and of the Deutsche Bank in particular. They usually have a private health insurance and use their personal credit line for investments.

Well-established middle class

Typical professions in this GeoCluster are civil servants and housewives/housemen.

All group ages are well-represented in this GeoCluster. People in this GeoCluster typically have more children than average and they are married. The typical household consists of many person and lives in an owned house.

The net monthly income lies between 3,000 € and 4,000 €.

The use of financial products is average and even below average for credit cards. Typical representatives of this GeoCluster are customers of cooperative banks and frequently have a private health insurance.

Urban middle class

The typical education level in this GeoCluster is upper secondary education. Typical professions include managers, salaried employees and university students.

All group ages are well-represented in this GeoCluster. People in this GeoCluster typically have less children than average and they are more often single, divorced or separated than married. The typical household consists of one or two persons and lives in a condominium.

The net monthly income lies between 2,000 € and 4,000 €.

More than average, typical representatives of this GeoCluster favour shares and investment funds and strongly use credit cards, in particular Diners. They are customers at large banks and often have a private health insurance. For investments, they favour instalment sales or use their personal credit line.

Young established middle class

The typical education level in this GeoCluster is secondary and lower secondary education. Typical professions include civil servants and housewives/housemen.

The dominant age groups in this GeoCluster range from 25 to 45. People in this GeoCluster typically have two or more children and are married. The typical household consists of four or more persons and usually lives in a rented or owned house.

The net monthly income is 3,000 € and more.

In this GeoCluster, bank credits are frequently used and the possession of shares and investment funds is slightly above average. The typical representatives of this GeoCluster are more likely to be customers of cooperative banks, they have a building insurance and rely on real-estate for old-age protection.

Traditional middle class

The typical education level in this GeoCluster is secondary and lower secondary education. Typical professions include civil servants, workers and housewives/housemen.

The dominant age groups in this GeoCluster range from 35 to 45. People in this GeoCluster typically have two or more children and they are married. The typical household consists of four or more persons and usually lives in an owned house.

The net monthly income ranges from 1,000 € to 8,000 €.

In this GeoCluster, bank credits are frequently used and typical representatives are likely to be customers of cooperative banks. They often have a building insurance and a legal protection insurance and they rely on real-estate for old-age protection.

Petty-bourgeois working class GeoCluster

The typical education level in this GeoCluster is secondary and lower secondary education. Typical professions include civil servants, workers and housewives/housemen.

The dominant age groups in this GeoCluster range from 25 to 55. People in this GeoCluster typically have many children (often three or more) and they are married. The typical household consists of four or more persons and usually lives in a rented or owned house.

The net monthly income ranges from 1,000 € and 3,000 €.

In this GeoCluster, bank credits are frequently used. The typical representatives of this GeoCluster are usually customers of cooperative banks and often have a building insurance.

Active petty-bourgeois GeoCluster

The typical education level in this GeoCluster is lower secondary education. Typical professions include civil servants, workers and housewives/housemen, with many part-time jobs.

The dominant age groups in this GeoCluster range from 25 to 45. People in this GeoCluster typically have many children (often three or more) and they are married. The typical household consists of four or more persons and usually lives in an owned house.

The net monthly income ranges from 2,000 € and 4,000 €.

Financial products such as warrants, shares and investment funds are used more than average. Bank credits are very frequent, however neither instalment sales nor investments covered by personal credit lines. The state health insurance scheme is dominant.

Status-oriented working class GeoCluster

The typical education level in this GeoCluster is lower or higher secondary education with only few university degrees. Typical professions include workers and housewives/housemen.

The dominant age groups in this GeoCluster range from 18 to 55. People in this GeoCluster typically have two or more children and they are divorced or separated. The typical household consists of four or more persons and usually lives in a condominium.

The net monthly income ranges from 2,000 € and 7,000 €.

The typical representatives of this GeoCluster frequently own warrants, usually have a private health insurance and are unlikely to be customers of large banks.

Urban traditional working class GeoCluster

The typical education level in this GeoCluster is lower or higher secondary education, with only few university degrees. Typical professions include workers, civil servants and apprentices.

The dominant age groups in this GeoCluster are 18 to 35 and above 70. People in this GeoCluster typically have no children and they are single, divorced or separated. The typical household consists of one person and usually lives in a rented flat.

The net monthly income is up to 2,000 €.

The typical representatives of this GeoCluster have little or no financial/insurance products and are unlikely to be customers of large banks.

Rural conformist GeoCluster

This GeoCluster is mainly found in the five new states. The typical education level in this GeoCluster is a university degree. Typical professions include workers, civil servants and apprentices.

All group ages are well-represented in this GeoCluster. People in this GeoCluster typically have more children than average, live with a life partner and are usually married or widowed. The typical household consists of three or more persons and lives in an owned house.

The net monthly income is up to 2,000 €.

Except for bank credits, the typical representatives of this GeoCluster use little or no financial/insurance products. They usually are customers of large banks and often have a building insurance. The state health insurance scheme is dominant.

Urban nonconformist GeoCluster

The typical education level in this GeoCluster is a university degree. Typical professions include liberal professions, self-employed and university students.

The dominant age groups in this GeoCluster range from 18 to 35. People in this GeoCluster typically have no children and they are single, divorced or separated. The typical household consists of one person and usually lives in a rented flat.

The net monthly income in this GeoCluster is distributed very unevenly: It is either below 1,000 € or it ranges from 4,000 € to 7,000 €.

The typical representatives of this GeoCluster often own warrants and credit cards, are usually customers of large banks, have few insurances and consider the public age-protection scheme insufficient.

Simple working class GeoCluster

This GeoCluster is mainly found in the five new states. The typical education level in this GeoCluster is secondary education or a university degree. Typical professions include civil servants, workers and pensioners.

The dominant age groups in this GeoCluster are 18 to 25 and above 55. People in this GeoCluster typically have slightly less children than average and they are single, divorced or separated. The typical household consists of one or two persons and usually lives in a rented flat.

The net monthly income lies below 1,000 €.

Financial products, credit cards and insurances are seldom used. The state health insurance scheme is dominant.

Simple petty-bourgeois GeoCluster

This GeoCluster is mainly found in the five new states. The typical education level in this GeoCluster is a university degree. Typical professions include civil servants and pensioners.

The dominant age groups in this GeoCluster are 18 to 25 and above 55. People in this GeoCluster typically have few children and have a life partner or they are widowed. The typical household consists of two persons and usually lives in a rented flat.

The net monthly income lies below 1,000 €.

Financial products, credit cards and insurances are seldom used. People in this GeoCluster rely on state health insurance and consider the consider the public age-protection scheme sufficient.

Older low-status GeoCluster

This GeoCluster is mainly found in the five new states. The typical education level in this GeoCluster is a university degree. The typical profession is pensioner.

The dominant age groups in this GeoCluster 63 and higher. People in this GeoCluster typically have no (more) children and are often widowed. The typical household consists of one or two persons and usually lives in a rented flat.

The net monthly income lies below 2,000 €.

Financial products, credit cards and insurances are seldom used. People in this GeoCluster consider the public age-protection scheme sufficient.

Business Area

In addition to social GeoClusters, business areas in Germany (West and East) are also classified according to their dominant economical sector. Thus, areas characterised by business/ economical activities rather than by private households are evaluated separately regarding their social GeoCluster.

3

3.6.3 PrizmResidentalEnv

The evaluation of residential environments is based on the lifestyle segmentation of GeoClusters and regiotypes as well as on the dominant dimensions of age/period of life, private purchasing power and type of building/development in the street section. Information regarding the evaluation of the residential environment is provided for all (currently 1.3 million) street sections in Germany.

Possible values:

1 Extremely good

2 Rather very good

3 Above average

4 Average

5 Below average

9 No evaluation

Residential environment extremely good

The established GeoCluster is the dominant lifestyle segment of this class, in particular in the large city suburbs and in the medium centres of urban areas. These areas are typically characterised by houses with one or two units and by a very high purchasing power.

Residential environment rather very good

The established GeoCluster and the upper middle-class GeoCluster are the dominant lifestyle segments of this class in Germany. These areas are typically characterised by houses with one or two units and terrace houses as well as a high purchasing power.

Residential environment above average

The middle-class GeoClusters are the dominant lifestyle segments of this class in Germany (see GeoCluster). In West German large cities, the advance-oriented GeoCluster is also strongly represented. These areas are typically characterised by smaller multiple dwelling units and an above-average purchasing power.

Residential environment average

The lower middle-class GeoCluster and the upper working class GeoCluster are the dominant lifestyle segments of this class in Germany (see GeoCluster). These areas are typically characterised by multiple dwelling units and an average purchasing power.

Residential environment below average

The working class GeoClusters are the dominant lifestyle segments of this class in Germany (see GeoCluster). These areas are typically characterised by multiple dwelling units, blocks of flats and a below-average purchasing power.

3.7 Bank Reference

3.7.1 Bank Reference Verification

The bank server checks the integrity of a bank reference (bank code, name and town) as well as the validity of a bank account number (optional). Fault-tolerant algorithms are used to check the bank reference (code, name, town) against a reference table provided by the Deutsche Bundesbank. If an incomplete or erroneous input leads to several equally probable credit institutions, a result list is returned.

To check a bank reference, the BankName and/or BankCode properties have to be set. To check a bank account number, the Account input property has to be set in addition.

Bank properties are only available if the 'validate' method has returned UNI_REQUEST_OK as return value. The BankStatus property is used to classify the validity of the bank reference.

Possible values:

Value	Explanation
UNI_STAT_VALID	Bank reference is unique; the Bank properties are available.
UNI_STAT_AMBIGUOUS	Several alternatives were found and a result list was created. The result list contains all bank properties. No further selection is required. For accessing the result list under the programming language used, refer to the corresponding programming language section.
UNI_STAT_INVALID	No valid bank reference was found.
UNI_STAT_NOT_AVAILABLE	The input could not be checked due to an internal error of the WEB-address-server.
UNI_STAT_SERVICE_NOT_AVAILABLE	A valid bank reference could not be determined due to an internal error of the WEB-address-server.
UNI_STAT_NO_LICENSE	The passed registration ID does not include a license for bank reference information.
UNI_STAT_NO_INPUT	The input does not contain a bank reference.

The AccountStatus property is used to classify the validity of the bank account number.

Possible values:

Value	Explanation
UNI_STAT_VALID	The account no. is a valid account for the specified credit institution.
UNI_STAT_INVALID	The account no. is not a valid account for the specified credit institution.
UNI_STAT_NOT_CHECKED	The bank account number could not be checked due to the missing plausibility check method for the specified credit institution.
UNI_STAT_NO_INPUT	The input does not contain an account number.

3.7.2 IBAN Verification

The International Bank Account Number (IBAN) was proposed in Brussels by the European banks, savings banks and people's banks associations in the summer 2001. The IBAN is a a code consisting of up to 34 digits which will be used to automate cross-border money transfers within the European Union as of next year. Basically, the IBAN contains in a country-specific way the bank code and the account number. To check an IBAN, the IBAN property is used.

IBAN properties are only available if the 'validate' method has returned UNI_REQUEST_OK as return value. The IBANStatus property is used to classify the validity of the IBAN.

Possible values:

Value	Explanation
UNI_STAT_VALID	The IBAN is valid.
UNI_STAT_INVALID	The IBAN is not a valid number.
UNI_STAT_NOT_CHECKED	The IBAN could not be checked due to the missing plausibility check method.
UNI_STAT_NOT_AVAILABLE	The input could not be checked due to an internal error of the WEB-address-server.
UNI_STAT_SERVICE_NOT_AVAILABLE	The IBAN could not be checked due to an internal error of the WEB-address-server.
UNI_STAT_NO_LICENSE	The passed registration ID does not include a license for IBAN information.
UNI_STAT_NO_INPUT	The input does not contain an IBAN.

3.7.3 Credit card verification

The bank server carries out plausibility checks for credit cards of various providers. The digits of the credit card number are split into four groups used among others to specify the network

(e.g. Visa, Eurocard). The verification procedure is international and can be applied to any credit card. To check a credit card number, the CreditCardNr property is used.

Credit card properties are only available if the 'validate' method has returned UNI_REQUEST_OK as return value. The CreditCardNrStatus property is used to classify the validity of the credit card number.

Possible values:

Value	**Explanation**
UNI_STAT_VALID	The credit card number is valid.
UNI_STAT_INVALID	The credit card number is not a valid number.
UNI_STAT_NOT_CHECKED	The credit card number could not be checked due to the missing plausibility check method.
UNI_STAT_NOT_AVAILABLE	The input could not be checked due to an internal error of the WEB-address-server.
UNI_STAT_SERVICE_NOT_AVAILABLE	The credit card number could not be checked due to an internal error of the WEB-address-server.
UNI_STAT_NO_LICENSE	The passed registration ID does not include a license for credit card information.
UNI_STAT_NO_INPUT	The input does not contain a credit card number.

4

4 WEB-API for Java

Please also refer to the general description in section 3.

4.1 Properties and Methods

```
package uniserv.components;

public class UniAspOe {

    //corresponds to 'init' method (see also section 3.2)

    public UniAspOe(String addressServerURL,String customerNr, String customerPasswd);

    //The argument proxyAuthorization is available in the following format:
    //domain\username:password
    public void setProxyAuthorization (String proxyAuthorization);

    public void setTransliteration(boolean enabled);
    public boolean getTransliteration();

    public void setLanguage(String enabled);

    public void setHouseNumberMatch(boolean enabled);
    public boolean getHouseNumberMatch();

    public int getRequestHint();
    public String getRequestErrorMsg();

    //Postal validation

    public void setCountry(String country);
    public void setAddressMode (int addressMode);
    public void setZip(String zip);
    public void setCity (String city);
    public void setStreetInputMode(int inputMode);
    public void setStreet(String street);
    public void setStreet1(String street);
    public void setStreet2(String street);
    public void setStreetName(String street);
    public void setHouseNumber(String street);
    public void setCounty(String street);
    public void setPOBoxNumber(String poboxNumber);
    public void setPOBoxWithoutNumber(boolean enabled);
    public void setLine1(String line1);
    public void setLine2(String line2);
    public void setLine3(String line3);
    public void setLine4(String line4);
    public void setLine5(String line5);
    public void setLine6(String line6);

    public int getAddressStatus();
    public int getAddressStatusHint();
    public String getAddressErrorMsg();
```

```
public String getOrganisation();
public String getSubBuildingName();
public String getBuildingName();
public String getZip();
public String getCommunityCode();
public int getCityQuality();
public String getCity();
public String getStreet();
public int getStreetQuality
public String getStreetName();
public String getHouseNumber();
public String getCityDistrict();
public String getCountry();
public String getCountryName();
public String getCarRegistration();

public String getPOBoxNumber();
public boolean getPOBoxWithoutNumber();

public String getLine1();
public String getLine2();
public String getLine3();
public String getLine4();
public String getLine5();
public String getLine6();

public void clear();
public int validate();

//Selection list
public SelectionList get SelectionList();

//Resume postal validation with selected address from selection list
public int validate(String status);

//Co-ordinate
public int getCoordinateStatus();
public String getXCoordinate();
public String getYCoordinate();

//Freight code
public int getCargoStatus();
public String getCargo();

//Personicx Geo
public String getPrizmStatus();
public String getPrizmState();
public String getPrizmCounty();
public String getPrizmDistrict();
public String getPrizmMunicipality();
public String getPrizmStatisticalArea();
public int getPrizmHouseholds();
public int getPrizmRegioType();
public int getPrizmMilieu();
public int getPrizmResidentalEnv();

//KGS22
public int getKGS22Status();
```


```
    public String getKGS22();

    // Bank reference
    public void setBankCountry (String bankCountry);
    public void setBankName (String bankName);
    public void setBankCode (String bankCode);
    public void setAccount (String account);
    public int getBankStatus();
    public int getAccountStatus();
    public String getBankName();
    public String getBankCode();
    public String getClearingRegion();
    public String getBankNetwork();
    public String getBankType();
    public String getSWIFT();
    public String getBankCity();
    public String getBankZip();

    // Bank reference result list
    public BankList getBankList();

    // IBAN
    public void setIBAN (String iban);
    public int getIBANStatus;

    // Credit card
    public void setCreditCardNr (String creditCard);
    public int getCreditCardNrStatus();
    public String getCreditCardNetwork();
}
public class SelectionList {
    public Enumeration getArgumentNames();
    public String getArgumentValue(String argumentName, int listPosition);
    public int getLength();
    }

    // Bank reference result list
    public class BankList {
    public int getLength();
    public Enumeration getArgumentNames();
    public String getArgumentValue (String name, int position);
}
```

4.2 Sample Program

In the sample programs (demo/PostalValidation.java, Bank.java), the input data is set as address or bank properties and then passed for validation. The output is generated according to the return value:

- Both the corrected address and the corresponding geocoding data and Personicx™ Geo information are displayed, with fields marked 'Suspect' if they were changed significantly or with uncertainty. The returned address is formatted according to country-specific requirements which is particularly important for addresses in the UK.
- Selection lists are displayed in case of ambiguous addresses.

– The return value is displayed in case of an error.
– In case an unambiguous bank reference is found, all bank properties are displayed. In case a result list is returned, it is displayed entirely.

4

4.3 Installation

After having unpacked the archive `jaspXXX.zip`, the directory named `lib` will contain the JAR file named `uniaspoe.jar`. Add this file to your development or runtime environment.

The interface documentation is to be found in the `doc` directory.

5 COM Interface

Please also refer to the general description in section 3.

With COM (and its successors DCOM and COM+), Microsoft has set a protocol which defines how components can be made available for other applications (or components) and how data data can be exchanged between a component and a user of this component.

UNISERV provides a COM component which allows to use the services of the UNISERV WEB address server. This component not only enables access from applications created for instance with Visual C++ or Visual Basic, but also using the services within script languages such as Visual Basic for Applications (VBA) or Visual Basic Script (VBS).

For use in IIS applications (Active Server Pages), a second component (specifically adapted to IIS security settings) is installed.

The COM interface requires Microsoft Internet Explorer 4.0 or higher and uses the IE settings for connecting to the Internet. This in particular also applies to the use of Proxy servers.

5.1 Properties

Standard properties of the COM interface.

Property	Access
Account	write only
AccountStatus	read only
AddressErrorMsg	read only
AddressMode	write only
AddressStatus	read only
AddressStatusHint	read only
BankCity	read only
BankCode	read / write
BankCountry	write only
BankName	read / write
BankNetwork	read only
BankStatus	read only
BankType	read only
BankZip	read only
BuildingName	read only
Cargo	read only
CargoStatus	read only

Property	Access
CarRegistration	read only
City	read / write
CityDistrict	read only
CityQuality	read only
ClearingRegion	read only
CommunityCode	read only
CoordinateStatus	read only
Country	read / write
CountryName	read only
County	read / write
CreditCardNetwork	read only
CreditCardNr	write only
CreditCardNrStatus	read only
ErrorMsg	read only
Hint	read only
HouseNumber	read / write
HouseNumberMatch	read / write
IBAN	write only
IBANStatus	read only
KGS22	read only
KGS22Status	read only
Language	write only
Line1	read / write
Line2	read / write
Line3	read / write
Line4	read / write
Line5	read / write
Line6	read / write
Organisation	read only
POBoxNumber	read / write
POBoxWithoutNumber	read / write
PrizmCounty	read only
PrizmDistrict	read only
PrizmHouseholds	read only
PrizmMilieu	read only

Property	Access
PrizmMunicipality	read only
PrizmRegioType	read only
PrizmResidentalEnv	read only
PrizmState	read only
PrizmStatisticalArea	read only
PrizmStatus	read only
Street	read / write
Street1	write only
Street2	write only
StreetInputMode	write only
StreetName	read / write
StreetQuality	read only
SWIFT	read only
Transliteration	read / write
XCoordinate	read only
YCoordinate	read only
Zip	read / write

Extended properties of the COM interface

Property	Access	Description
ReturnValue	read only	In addition, this property contains the return value of the last call of the method
registrationID	write only	User identification for the access to the UNISERV WEB address server
password	write only	Password for the access to the UNISERV WEB address server
serverURL	write only	URL of the UNISERV WEB address server
SelListLength	read only	Length of current selection list (when value > 0)
ProxyAuthorization	write only	Optional user identification for HTTP proxy servers in the format Username:Password. Whenever this property is set, its contents will be additionally transmitted when executing a request.
XMLServerResponse	read only	Contains the XML reply generated by the WEB-address-server for the last request transmitted (for test purposes only)
BankListLength	read only	Length of the result list of the current bank verification

Property	Access	Description
ValidateStatus	write only	Status of preceding validation in case of stateless address validation (see section 5.5).

5.2 Methods

Method	Explanation
clear	This method can be called to reset all properties to their default settings.
initialize	The initialisation method has to be called after creating a COM object before properties are accessed or other methods are called. By calling this method, the internal data structures of the object are initialised and the object itself is put into a defined state.
validate	The validation method has to be called after setting the properties and starts the verification of the input properties. The check result is stored in the corresponding properties and can be retrieved from the client application.
select	Selects an entry from the current selection list and continues the postal address validation previously started by calling the method validate.
getSelectionItem	Returns an element of the current selection list of the address verification. The vbsGetSelectionItem method is to be used in a VBScript environment, as it provides the desired information directly as result.
getBankItem	Returns an element of the result list of the bank verification. The vbsGetBankItem method is to be used in a VBScript environment, as it provides the desired information directly as result.
validateWithStatus	Like the validate method. Additionally uses the ValidateStatus property (see section 5.5).

5.3 Address Verification Selection Lists

In order to use the COM interface in as many environments as possible, a number of conventions have to be complied to when passing arguments to methods. Thus, the COM interface can be used in script languages as well as in development environments generating directly executable files.

The address verification is started after setting the input properties by calling the validate method. After a successful execution of the validate method (return value is UNI_OK), a selection list is created if the AddressStatus output property was set to UNI_SELECTION. The selection list can be processed in the client application. The AddressStatusHint output property provides the type of the selection list.

The following selection lists are available:

Property AdressStatusHint	Selection list
UNI_AMBIGUOUS_CITY	Town selection list
UNI_AMBIGUOUS_STREET	Street selection list
UNI_AMBIGUOUS_BOX	House number selection list
UNI_AMBIGUOUS_ORGANISATION	Company selection list

5

Property AdressStatusHint	Selection list
UNI_AMBIGUOUS_BUILDING	Building selection list
UNI_AMBIGUOUS_SUB_BUILDING	Building access selection list
UNI_AMBIGUOUS_DEPENDENT_STR	Dependent street selection list
UNI_AMBIGUOUS_DEPENDENT_LOCALITY	Dependent locality selection list
UNI_AMBIGUOUS_CITY_DISTRICT	Town district selection list
UNI_AMBIGUOUS_ADDRESS	Address selection list
UNI_AMBIGUOUS_ROUTE_SERVICE	Route Services selection list (CA only)
UNI_AMBIGUOUS_UNIT_POBOX	House No. supplement selection list (CA only)
UNI_AMBIGUOUS_GOVERNMENT_DEPARTMENT	Public authority selection list

The property SelListLength allows to determine the number of entries in the selection list. Using the method getSelectionItem, the various elements of the selection list can then be transferred from the COM interface to the client application. To do so, both the position within the selection list and the type of the element to be transferred have to be passed to this method. The desired element is always returned as string.

ItemId	Selection list
UNI_SELECTION_ITEM_CITY	Town selection list, address selection list
UNI_SELECTION_ITEM_DISTRICT	Town selection list
UNI_SELECTION_ITEM_MIN_MAX_ZIP	Town selection list
UNI_SELECTION_ITEM_ZIP	House number selection list, Company selection list, building selection list, building access selection list, dependent street selection list, dependent locality selection list, town district selection list, address selection list, Route Services selection list, house no. supplement selection list, public authority selection list
UNI_SELECTION_ITEM_BOX	House number selection list
UNI_SELECTION_ITEM_STREET	Street selection list, House number selection list, Route Services selection list, house no. supplement selection list
UNI_SELECTION_ITEM_AMBIGUOUS	Street selection list
UNI_SELECTION_ITEM_ORGANISATION	Company selection list, address selection list
UNI_SELECTION_ITEM_BUILDING_NAME	Building selection list, address selection list, public authority selection list

ItemId	Selection list
UNI_SELECTION_ITEM_SUB_BUILDING_NAME	Building access selection list
UNI_SELECTION_ITEM_DEPENDENT_STREET	Dependent street selection list
UNI_SELECTION_ITEM_DEPENDENT_LOCALITY	Dependent locality selection list
UNI_SELECTION_ITEM_CITY_DISTRICT	Town district selection list, address selection list
UNI_SELECTION_ITEM_STREETNAME	Address selection list
UNI_SELECTION_ITEM_HOUSENUMBER	Address selection list
UNI_SELECTION_ITEM_STATUS	All selection lists
UNI_SELECTION_ITEM_COUNTRY_NAME	Town selection list
NI_SELECTION_ITEM_COUNTY	Town selection list
UNI_SELECTION_ITEM_ROUTE_SERVICE	Route Services selection list
UNI_SELECTION_ITEM_UNIT_POBOX	House no. supplement selection list
UNI_SELECTION_ITEM_GOVERMENT_DEPARTMENT	Public authority selection list

Interacting with the user, the client application is able to determine the correct entry from the selection list and then to resume the address check by way of the method select. The method select in turn is able to generate a selection list to be processed accordingly.

5.4 Bank Verification Selection Lists

The verification of a bank reference or credit institution may return ambiguous results, i.e. the BankStatus output property is set to UNI_STAT_AMBIGUOUS after a successful execution of the validate method. In this case, the BankListLength output property will provide the length of the result list. The getBankItem method enables the client application to retrieve the various elements of the result list. To do so, the selected element and its position in the result list have to be passed to this method. The selected element is returned as string.

The following elements of the result list can be retrieved by the client application:

– UNI_BANKSEL_BANKNAME
– UNI_BANKSEL_BANKCODE
– UNI_BANKSEL_BANKNETWORK
– UNI_BANKSEL_BANKTYPE
– UNI_BANKSEL_SWIFT
– UNI_BANKSEL_BANKCITY
– UNI_BANKSEL_BANKZIP
– UNI_BANKSEL_CLEARINGREGION
– UNI_BANKSEL_ACCOUNTSTATUS

5

5.5 Stateless Address Validation

In WEB applications, it may not always be possible to use a COM object beyond page limits. For that reason, the address validation can also be run in a stateless mode. To do so, the required information has to be temporarily saved and passed to the address validation by the application at a later stage.

This procedure only concerns cases where a selection list is returned, in which case it consists of the following steps:

- Create a COM object
- Set input properties
- Call validate method
- Determine and temporarily save the selection list elements, including the element UNI_SELECTION_ITEM_STATUS
- Release the COM object
- .....
- Create a new COM object
- Set the input properties as for the first validation; in addition, assign the previously determined value of the UNI_SELECTION_ITEM_STATUS element of the selected entry to the ValidateStatus property.
- Call the validateWithStatus method
- If no selection is returned, the output properties can be processed directly; otherwise, the selection list has to be processed accordingly.

5.6 Sample Programs

During the installation of the COM client component, various sample programs for Visual Basic (sample.bas), Visual C++ (sample.cpp), and Microsoft Word (aspletter.dot) as well as IIS applications (Active Server Pages) (sample.asp) are copied into the installation directory and made available in the Windows Start menu.

To use the component in Visual Basic, you have to activate the option 'References' in the menu 'Project' and then the option 'UNISERV ASP COM 2.0 Type Library' in the window 'References'. To use the component in Microsoft Office applications under VBA, the component has to be activated by way of the window 'References' of the menu 'Tools' in the Visual Basic Editor.

5.7 Installation

The COM interface is installed by running the file CASPxxx.EXE. You will be prompted to enter a temporary directory for the installation files. This temporary directory can be deleted once the installation procedure is completed.

During the installation, you will be prompted to enter the installation directory. You can either accept the pre-set entry or select any other directory. The files will then be copied into the desired installation directory and the COM interface is registered in the Windows Registry.

In addition, you will be prompted to enter your registration ID, your password as well as the URL of the WEB-address-server. The entered information is stored in the Registry and used by the test/sample programs provided with the delivery.

The UNISERV COM interface can then be tested using a simple client. To do so, run the program file TESTCOMP.EXE in the installation directory, for instance by double-clicking on it in the Windows Explorer.

The input fields on the left side of the dialog window are used to set input properties. The command button ’Check’ is used to start the postal validation. The result will then be displayed on the right side of the window.

In case an existing application uses an older version of the COM interface (lower than version 1.10) and is not upgraded/adapted, the new version has to be installed in another directory!

6

6 WEB-API for Perl

Please also refer to the general description in section 3.

WEB-API for Perl consists of the modules Uniserv::UniAspOe.pm for the address validation as such and Uniserv::SelectionList.pm for the handling of lists. Both are written as object-oriented modules and each one of them provides a method new() allowing to create new object instances. All other methods are object methods.

6.1 Properties and Methods

Module Uniserv::UniAspOe.pm

Method	Description
sub new($server_url, $customer_number, $password, $proxy_IP) { }	Creates a new object.
sub setProxyAuthorization ($proxy_domain, $proxy_user, $proxy_passwd) { }	Optional user ID for HTTP proxy servers in the format $proxy_domain\$proxy_user:$proxy_passwd
sub setTransliteration($enabled) {}	Sets the property Transliteration.
sub setLanguage($language) {}	Sets the property Language.
sub setAddressMode($addressMode) {}	Sets the property AddressMode. This property is used to control the client's address input mode (field-oriented or line-oriented). Possible values: – UNI_FIELD_MODE – UNI_LINE_MODE Default value: UNI_FIELD_MODE.
sub setZip($zip) { }	Sets the property Zip of an object instance.
sub setCity($city) { }	Sets the property City of an object instance.
sub setStreetInputMode($street_mode) { }	Sets the property StreetInputMode of an object instance. $street_mode can have the value 0 (street and house number in input property Street) or 1 (street and house number in input properties StreetName and HouseNumber separately).
sub setStreet($street) { }	Sets the property Street of an object instance.
sub setStreetName($street_name) { }	Sets the property StreetName of an object instance.
sub setHouseNumber($house_number) { }	Sets the property HouseNumber of an object instance.

Method	Description
sub setHouseNumber-Match($house_number_match) {}	Sets the HouseNumberMatch property. This property is used to set whether or not a house number validation is to be performed. Possible values for '$house_number_match': 'true' (for house number validation) or 'false'. The latter is the default setting. If house number validation is enabled, 'AddressStatus' is set to UNI_SUSPECT and 'AddressStatusHint' to UNI_UNKNOWN_HOUSE_NUMBER whenever the house number in not valid.
sub setStreet1($street1) { }	Sets the property Street1 of an object instance.
sub setStreet2($street2) { }	Sets the property Street2 of an object instance.
sub setCounty($county) { }	Sets the property County of an object instance.
sub setPOBoxNumber($pobox_number) { }	Sets the property POBoxNumber of an object instance.
sub setPOBoxWithoutNumber($enabled) { }	Sets the property POBoxWithoutNumber of an object instance.
sub setCountry($country) { }	Sets the property Country of an object instance.
sub setLine1($line1) {}	Sets the property Line1 of an object instance
sub setLine2($line2) {}	Sets the property Line2 an object instance
sub setLine3($line3) {}	Sets the property Line3 an object instance
sub setLine4($line4) {}	Sets the property Line4 an object instance
sub setLine5($line5) {}	Sets the property Line5 an object instance
sub setLine6($line6) {}	Sets the property Line6 an object instance
sub setBankName($bank_name) {}	Sets the property BankName of an object instance
sub setBankCode($bank_code) {}	Sets the property BankCode of an object instance
sub setBankCountry($bank_country) {}	Sets the property BankCountry of an object instance
sub setAccount($account) {}	Sets the property Account of an object instance
sub setCreditCardNr($credit_card_nr) {}	Sets the property CreditCardNr of an object instance

Method	Description
sub setIBAN($iban) {}	Sets the property IBAN of an object instance
sub validate ($mit_oder_ohne_Auswahlliste, $status_number) { }	Carries out (resumes) the validation of the currently set input properties. It is possible to set whether selection lists are to be created or not. If selection lists are created, the method has to be called again after processing the selection list, with $status_number having to refer to the desired entry in the selection lists.
sub getRequestErrorMsg { }	Reads the property RequestErrorMsg
sub getRequestHint { }	Reads the property RequestHint
sub getAddressStatus { }	Reads the property AddressStatus
sub getAddressStatusHint { }	Reads the property AddressStatusHint
sub getAddressErrorMsg { }	Reads the property AddressErrorMsg
sub getAddressSelectionList { }	Returns an object instance for a selection list (AddressSelectionList)
sub getTransliteration {}	Reads the property Transliteration.
sub getZip { }	Reads the property Zip.
sub getStreet { }	Reads the property Street.
sub getStreetQuality { }	Reads the property StreetQuality. Provides the similarity measure for the street. Possible values range from 0 to 100.
sub getStreetName { }	Reads the property StreetName.
sub getHouseNumberMatch { }	Reads the property HouseNumberMatch. Default: 'false'
sub getHouseNumber { }	Reads the property HouseNumber.
sub getCityDistrict { }	Reads the property CityDistrict.
sub getCommunityCode { }	Reads the property CommunityCode.
sub getCounty { }	Reads the property County.
sub getOrganisation { }	Reads the property Organisation.

Method	Description
sub getBuildingName { }	Reads the property BuildingName.
sub getCountry { }	Reads the property Country.
sub getCity { }	Reads the property City.
sub getCityQuality { }	Reads the property CityQuality. Provides the similarity measure for the town. Possible values range from 0 to 100.
sub getPOBoxNumber { }	Reads the property POBoxNumber
sub getPOBoxWithoutNumber { }	Reads the property POBoxWithoutNumber
sub getCountryName { }	Reads the Property CountryName.
sub getCarRegistration { }	Reads the Property CarRegistration.
sub getLine1() {}	Reads the property Line1.
sub getLine2() {}	Reads the property Line2.
sub getLine3() {}	Reads the property Line3.
sub getLine4() {}	Reads the property Line4.
sub getLine5() {}	Reads the property Line5.
sub getLine6() {}	Reads the property Line6.
sub getCoordinateStatus { }	Reads the property CoordinateStatus.
sub getXCoordinate { }	Reads the property XCoordinate.
sub getYCoordinate { }	Reads the property YCoordinate.
sub getCargoStatus { }	Reads the property CargoStatus.
sub getCargo { }	Reads the property Cargo.
sub getKGS22Status { }	Reads the property KGS22Status.
sub getKGS22 { }	Reads the property KGS22.

Method	Description
sub getPrizmStatus { }	Reads the property PrizmStatus.
sub getPrizmState { }	Reads the property PrizmState.
sub getPrizmCounty { }	Reads the property PrizmCounty.
sub getPrizmDistrict { }	Reads the property PrizmDistrict.
sub getPrizmMunicipality { }	Reads the property PrizmMunicipality.
sub getPrizmStatisticalArea { }	Reads the property PrizmStatisticalArea.
sub getPrizmHouseholds { }	Reads the property PrizmHouseholds.
sub getPrizmRegioType { }	Reads the property PrizmRegioType.
sub getPrizmMilieu { }	Reads the property PrizmMilieu.
sub getPrizmResidentalEnv { }	Reads the property PrizmResidentalEnv.
sub getBankList { }	Returns an object instance for a list (BankList)
sub getBankStatus { }	Reads the property BankStatus
sub getBankName { }	Reads the property BankName
sub getBankCode { }	Reads the property BankCode
sub getAccountStatus { }	Reads the property AccountStatus
sub getClearingRegion { }	Reads the property ClearingRegion
sub getBankNetwork { }	Reads the property BankNetwork
sub getBankType { }	Reads the property BankType
sub getSWIFT { }	Reads the property SWIFT
sub getBankCity { }	Reads the property BankCity
sub getBankZip { }	Reads the property BankZip

Method	Description
sub getIBANStatus { }	Reads the property IBANStatus
sub getCreditCardNrStatus { }	Reads the property CreditCardNrStatus
sub getCreditCardNetwork { }	Reads the property CreditCardNetwork
sub clear { }	Resets all properties to default settings.

Module Uniserv::SelectionList.pm

Method	Description
sub new($list_ref) { }	Creates a new object instance to handle lists. Every method getAddressSelectionList(), getBankList() of the module Uniserv::UniAspOe.pm returns a corresponding reference $list_ref to be used as input argument for this method call.
sub getArgumentNames { }	Returns the names of the arguments in the list.
sub getLength { }	Returns the length of the list.
sub getArgumentValue($argument_name, $position_in_Auswahlliste) { }	Returns the value of an element in the list. Note: the first position in the selection list is 1.

6.2 Application Programs

Sample programs

Two sample programs, app_html and app_console, are provided with the installation files. The program app_html shows how to use the Perl client component in a web browser while the program app_console illustrates how to use the Perl client component in a console application.

The following has to be taken into account when using/running the sample programs:

- If the Internet connection uses a proxy server, its IP address ($proxyIP) and its port ($proxyPort) has to be entered accordingly in the sample program used. For the proxy authentication, the parameters $proxy_domain, $proxy_user and $proxy_passwd need to be entered in the sample program.
- In both sample programs, the property StreetInputMode is set to line mode. If street and house number are to be entered separately, the mode needs to be switched accordingly ($street_mode = 1)
- Depending on whether selection list for the postal validation are desired or not, the $choice variable is to be set to ’yes’ or ’no‘. In the sample programs, it is set to ’yes‘.

The sample program app_html allows to enter data in a form. A selection contains the countries for which a postal validation is supported. In the corresponding CGI program, the entered data are set as properties and checked.

Based on the return value, a further HTML page containing the result is created:

– Both the corrected address and the corresponding geocoding data and Personicx™ Geo information are displayed, with address elements marked 'Suspect' if they were changed significantly or with uncertainty.

– In case of an ambiguous address, selection lists are displayed and the validation of the address is resumed. The bank list however is returned/displayed as final result.

– In case of an error, the corresponding return code is displayed.

In the sample program app_console, all input properties are set interactively and transmitted to the WEB-address-server for validation. If required, a postal selection list is created, provided the parameter for the creation of selection lists was set to 'yes' when calling the method validate(). After the selection, the validation is continued. The validation result is output in the same way as by the sample program app_html.

Structure of an application program

Application programs are developed according to the following scheme:

1. Using the method new(), create a new object of Uniserv::UniAspOe.pm.
2. Using the methods setXXX() of the module Uniserv::UniAspOe.pm, the values of the desired input properties are set.
3. Call the method validate() of the module Uniserv::UniAspOe.pm to start the validation. If postal selection lists are desired, the first parameter has to be set to 'yes'. The return value of the validation is either the final result or a selection list.
4. If there is no list, the methods getXXX() of the module Uniserv::UniAspOe.pm are used to determine the values of the output properties.
5. However, if a list*1) is returned:
   - Using the method new(), create a new object of Uniserv::SelectionList.pm; the input parameter for the new() method is obtained by way of the method getAddressSelectionList() or getBankList() of the module Uniserv::UniAspOe.pm.
   - Using the method getLength() of the module Uniserv::SelectionList.pm, determine the length of the list.
   - Using the method getArgumentNames() of the module Uniserv::SelectionList.pm, determine the attribute names of the different elements in the list.
   - Using the module Uniserv::SelectionList.pm, determine all elements of the list.
   - In case of a selection list, make selection to resume postal validation; in case of a bank reference verification, the bank list is the final result.
   - Resume the validation of the input address by calling again the method validate() of the module Uniserv::UniAspOe.pm, with the status number of the selected entry used as in-

*1) Selection list for postal validation or bank list for bank reference verification

put parameter for the validate() method. The return value of the validation now is either the final result or a further selection list.

- Repeat the steps above until no selection list is returned.

6. The method clear() of the module Uniserv::UniAspOe.pm allows to reset the values of the input properties in order to carry out a new validation.

6.3 Installation

The WEB-API requires the following modules which can be downloaded under: http://www.cpan.org/modules/by-module:

– LWP / libwww-perl
– XML / XML parser
– Unicode / Unicode string

For encrypted requests, the module Crypt::SSLeay is required.

– Net/Crypt::SSLeay

As usual, the WEB-API is integrated in the Perl standard library:

– perl Makefile.PL
– make test
– make install

7 WEB-API for PHP

Please also refer to the general description under section 3. A large number of today's dynamic web applications are based on PHP. Available on most operating systems and web servers, PHP is easy to learn and can be used for a wide variety of purposes. UNISERV provides a PHP module that allows to use the UNISERV WEB-address-server services.

7.1 Restrictions and Requirements

WEB-API for PHP requires PHP Version 4.0.4 or higher. All inputs/outputs are in ISO-8859-1 or UTF-8 character set. The entire implementation is provided in the `webapi.inc.php` file. This file needs to be included in the corresponding PHP source file by way of the 'require' instruction.

```
require "webapi.inc.php";
```

after which the class can be instanced and used:

```
// create WEB-API object
$a = new uniaspoe;

// perform initialization
$a->initialize();
$a->clear();

// set account and control properties
$a->serverURL = "http://asp.uniserv-online.de/components/address";
$a->registrationID = "yourID";
$a->password = "yourPassword";
$a->Language = "e";

// set address input properties
$a->Country = "DE";
$a->Street  = "rastatterstr. 13";
$a->City    = "pforzheim";
$a->Zip     = "75179";

$a->validate();
if (($a->AddressStatus == UNI_OK) ||
    ($a->AddressStatus == UNI_SUSPECT))
{
   echo "<H2>Result:</H2>";
   echo "<pre>Street: $a->Street\n";
   echo "ZIP:    $a->Zip\n";
   echo "City:   $a->City\n";
}

// release WEB-API object
$a->clear();
unset ($a);
```

7.2 Properties and Methods

In addition to the standard properties, WEB-API for PHP provides the following properties:

serverURL	URL of the UNISERV-WEB-address-server
registrationID	User ID for accessing the UNISERV-WEB-address-server
password	Password for accessing the UNISERV-WEB-address-server
ProxyServer	Optional name of a proxy server
ProxyPort	Optional port of a proxy server
ProxyAuthorization	User name / password for proxy server, separated by ":"
addressSelectionLength	Length of the current selection list (if value > 0)
addressSelectionItems	Current selection list (two-dimensional array)
bankSelectionLength	Length of the result list of the current bank reference validation
ReturnValue	Additionally, the property also contains the return value of the last method call
utf8Properties	Sets the character set to be used to describe and read properties. If ISO-8859-1 is used, this property has to be set to FALSE (default). For UTF-8, the property is set to TRUE. Please note that this property needs to be set first. Switching from one character set to the other within a request (set input, run validation, read output) is not possible!

WEB-API for PHP provides the following (public) methods.

initialize

Performs the initialisation of an object instance. This method has to be called after creating an object instance and before any other methods are called or properties set.

clear

Allows to reset all properties of an object instance, for instance when several validations are to be run in succession. The proxy server properties (ProxyAuthorization, ProxyServer and ProxyPort) and the server properties (serverURL, registrationID and password) are not reset.

validate

This method transmits the input properties to the server and starts the validation.

validateWithStatus

This method transmits the input properties to the server and resumes the validation in case of a stateless validation. To do so, a previously determined status information has to be passed so that the validation can be resumed at the point where it was interrupted. The information used for that purpose is the one provided under the STATUS key for each entry of the selection list.

7

select

This method selects an entry of the current address validation selection list and resumes the validation. If no further selection list is returned, the output properties can then be processed; otherwise, the selection list has to be reprocessed accordingly.

getAddressSelectionItems

This method returns an array with the keys of the 'addressSelectionItems' array. These key values then enable the client application to access the array.

getBankSelectionItems

This method returns an array with the keys of the 'bankSelectionItems' array. These key values then enable the client application to access the array.

7.3 Address Validation Selection Lists

After having set the input properties, the address validation is started or resumed by calling the 'validate' or 'select' methods respectively. If, after having successfully called the 'validate' or 'select' method (return value = UNI_OK), the output property 'AddressStatus' is set to UNI_SELECTION, a selection list is created. This selection list can be processed by the client application. The output property 'AddressStatusHint' provides the type of the selection list.

The property 'addressSelectionLength' allows to determine the number of entries in a selection list. The output property 'addressSelectionItems' provides the various elements of the result list to the client application as two-dimensional array. The 'getAddressSelectionItems' method allows to determine the key values for accessing the array.

Example

```
echo "<table>";
$myitems = $a->getAddressSelectionItems();
for ($i = 0; $i < $a->addressSelectionLength; $i = $i + 1)
{
   echo "<tr>";
   foreach ($myitems as $item)
   {
      if ($item == "STATUS")
      {
         echo '<td><input type="radio" name="in_status" value="'.
              $a->addressSelectionItems[$i][$item].'"';
         if ($i == 0)
            echo 'CHECKED></td>';
          else
             echo '></td>';
      }
      else
         echo '<td>'.$a->addressSelectionItems[$i][$item].'</td>';
   }
   echo "</tr>";
}
echo "</table>";
```

7.4 Bank Validation Result Lists

The validation of a bank reference may return an ambiguous result, i.e., after having successfully called the 'validate' method, the output property 'BankStatus' is set to UNI_STAT_AMBIGUOUS. In this case, the output property 'bankSelectionLength' provides the length of the result list and the output property 'bankSelectionItems' provides the various elements of the result list to the client application as two-dimensional array.

The method 'getBankSelectionItems' allows to determine the key values for accessing the array.

Example

```
...
echo "<table>";
$myitems = $a->getBankSelectionItems();
for ($i = 0; $i < $a->bankSelectionLength; $i = $i + 1)
{
   echo "<tr>";
   foreach ($myitems as $item)
   {
      echo '<td>'.$a->bankSelectionItems[$i][$item].'</td>';
   }
   echo "</tr>";
}
echo "</table>";
```

7.5 Stateless Address Validation

In web applications, it is usually not possible to use an object beyond page limits. Therefore, it is also possible to run the address validation as a stateless procedure. To do so, the application has to temporarily save the relevant information and re-pass to the address validation when it resumed.

This sequence of operations only concerns situations where a selection list is returned and consists of the following steps:

- Create an object
- Set input properties
- Call 'validate' method
- Determine and save selection list elements, including the element with the STATUS key
- Release the object
- .....
- Create a new object
- Call the 'validateWithStatus' method with the previously saved status value of the entry selected from the selection list
- If no further selection list is returned, the output properties can then be processed; otherwise, the selection list has to be reprocessed accordingly.

7

7.6 Installation

In addition to the implementation of the WEB-API for PHP, the file `phpaspXXX.zip` also contains two small sample applications that can be used to get started, namely `testpage.php` (for field-oriented validation) and `testpagelm.php` (for line-oriented validation). To be able to use them, the files need to be copied to a document directory of the web server, after which they are immediately available.

A Appendix

A.1 Remarks to release changes

Basic release 1.0 as of 37/00

Release 1.01 as of 46/00

Availability of geocoding information, geographical coordinates and cargo code.

Release 1.02 as of 08/01

- Improved postal validation for addresses from the UK.
- New properties:
  - Organisation
  - BuildingName
  - StreetName
  - Housenumber
  - Street1
  - Street2
  - StreetInputMode
  - CityDistrict
  - County
- Property Country is also readable
- New method: clear
- New selection lists (for the UK only):
  - Organisation selection list
  - Building selection list
  - Subbuilding selection list
  - Dependent street selection list
  - Dependent locality selection list
  - City district selection list
  - Address selection list

Release 1.03 as of 14/01

Personicx® Geo now available. See section 3.6.

Perl-API

The previous modules Address.pm, SelectionList.pm as well as Parsen.pm were restructured to Uniserv::Address.pm, Uniserv::SelectionList.pm and Uniserv::Parsen.pm by placing them under the directory Uniserv. the file Makefile.PL under the directory Uniserv and the affected codes in all files were changed accordingly. Existing client programs need to be adapted accordingly.

A

Release 1.03-001 as of 29/01

WEB-API also supports access to the WEB address management server via a proxy server requiring a user identification. The ProxyAuthorization property is provided in the format: domain\username:password

Release 1.10

- In addition to the postal validation of addresses and the retrieval of geocoding information, the WEB-API provides a possibility to check a bank reference, a credit card number or an IBAN.
- The methods used to validate the input properties only return the request status to the UNISERV WEB-address-server; the actual result of the verification/validation is provided through the following output properties: AddressStatus, CoordinateStatus, CargoStatus, PrizmStatus, BankStatus, CrediCardStatus, and IBANStatus.
- New properties:
  - RequestHint, RequestErrorMsg
  - AddressStatus, AddressStatusHint, AddressErrorMessage
  - BankCountry, BankName, BankCode, Account, BankStatus, AccountStatus, ClearingRegion, BankNetwork, BankType, SWIFT, BankCity, BankZip
  - CreditCardNr, CreditCardNrStatus, CreditCardNetwork
  - IBAN, IBANStatus

Release 1.11 as of 46/04

In addition to the expert systems, *postWorld* is also used for postal validation. Valid country codes are those with 2 or 3 letters according to ISO.

- New address properties:
  - CountryName
  - CarRegistration
- New selection lists (Canada only):
  - Public authority selection list.
  - PO box no. or house no. supplement selection list.
  - 'Route Service' selection list.
- New values of the AddressStatusHint property:
  - UNI_UNKNOWN_COUNTRY
  - UNI_STR_NO_INPUT
  - UNI_STR_NOT_CHECKED

Release 2.00 as of 06/05

- Communication between client components and UNISERV-WEB-address-server is now bidirectional in UTF-8.
- New address property: Transliteration.
- HTTPS support for WEB-API COM interface reviewed.

Release 2.00-001 as of 18/05

New properties KGS22 and KGS22Status.

Release 2.00-002 as of 30/05

New property: Language.

Release 3.00 as of 39/05

- WEB-API for PHP
- PO box validation
- New properties: POBoxNumber and POBoxWithoutNumber
- New values of the AddressStatusHint property:
  - UNI_VALID_POBOX_ADDRESS
  - UNI_POBOX_NUMBER_NOT_CHECKED
  - UNI_UNKNOWN_POBOX_NUMBER
  - UNI_LACK_OF_POBOX_NUMBER
  - UNI_CITY_WITHOUT_POBOX
  - UNI_AMBIGUOUS_POBOX_NUMBER
  - UNI_MIXED_STREET_POBOX

Release 4.00 as of 39/06

- Line-oriented address formatting and validation
- New properties: Line1 – Line6, AddressMode
- New value of the property AddressStatusHint: UNI_BAD_ADDRESS_MODE

Release 4.01 as of 02/07

- New properties HouseNumberMatch, CityQuality, StreetQuality
- New value of the AddressStatusHint property: UNI_UNKNOWN_HOUSE_NUMBER
- WEB-API for PHP: The new property utf8Properties allows to read and write properties in UTF-8 (UNICODE support).

Release 4.02 as of 32/07

- New address property
  - CommunityCode
- New value of the property AddressStatusHint
  - UNI_HNO_NO_INPUT

A

A.2 Form sheet for messages to UNISERV

Fax: ++49(0)7231/936-2500

To
UNISERV GmbH
Rastatter Str. 13

D-75179 Pforzheim

Suggestions and corrections for UNISERV user manual

WEB-API

Release: 4.02 dated: 32/07

If errors are detected in this manual, we kindly request you to inform us by using this form sheet.

We also look forward to receive any suggestions or improvements.

Sender:
Name
Company/Department
Address/Telephone

Change suggestions (please indicate the page number):

Page	Remarks