



Dokumentation for KraK Webservice

**Version 2.2
September 2009**

Content

<i>Document version</i>	4
<i>Introduction</i>	4
<i>Web services</i>	5
Login	5
Tele search	5
Company search	5
Road search	5
Address search	5
Address geocoding	5
Routing	5
<i>Login procedures</i>	6
1. Username/password → ticket	6
2. Username/IP → ticket	7
3. Username/IP (deprecated)	8
Setting up the SOAP header correctly	9
<i>Methods</i>	10
Tele search	10
Company search	13
Road search	16
Address search	17
Address Geoding	22
Routing	25
<i>Objects</i>	28
Address	28
AddressEx	29
MapReference	31
Point	31
AddressCoordinate	31
Company	32
ContactInfo	33
Tele	34
Road	35
RouteInfo	36
Route	37
RouteDescription	37



Eniro Danmark A/S

Sydmarken 44A
DK - 2860 Søborg

Telefon +45 88 38 38 00

Telefax +45 88 38 38 10
eniro@eniro.dk

CVR-nr. DK 18 93 69 84

Bank 2149-3484466784
www.eniro.dk



Part	38
Segment	38
<i>Frequently asked questions</i>	39

Document version

Version	Date	Description
2.0	April, 2009	Major update of version 1. Basically, everything has been rewritten. Throw out any previous documentation.
2.1	June, 2009	The section "Login procedures" has been updated with information on how to set up the SOAP header correctly. The info has been added as the web service descriptions (ASMX/WSDL) are not crystal clear in terms of which parameters should be included in the SOAP header.
2.2	September, 2009	Documentation of method GetAddressBy has been added.

Introduction

The purpose of this document is to provide a technical description of the Krak web service framework. The document contains a description of the different login procedures to obtain web service access and additionally a web service method and object reference. The structure of method/object definitions and sample code is based on C# .Net syntax.

Web services

The web services are exposed through several logically divided web service end points. They are all described in brief below.

Login

This web service contains method to ensure that only authorized users can access the Krak web services.

Service end point: <http://login.webservice.krak.dk/TicketCentral.aspx>

Tele search

This web service contains methods to search among almost 6 millions registered Danish phone numbers. By registered phone numbers means phone numbers with name and address of the owner. The web service is primarily used to ease population of forms on the web and in desktop systems.

Service end point: <http://basicservices.webservice.krak.dk/telesearch.aspx>

Company search

This web service contains methods to search among 680.000 Danish companies. The result includes CVR-no. The web service is primarily used to ease population of forms in B2B solutions.

Service end point: <http://basicservices.webservice.krak.dk/companysearch.aspx>

Road search

This web service contains methods to search among 125.000 Danish roads. It is often used in combination with address search.

Service end point: <http://basicservices.webservice.krak.dk/roadsearch.aspx>

Address search

This web service contains methods to search among 2.5 million Danish addresses. Supports structured address input as well as parsing and validation.

Service end point 1: <http://basicservices.webservice.krak.dk/addresssearch.aspx>

Service end point 2: <http://basicservices2.webservice.krak.dk/addresssearch.aspx>

Address geocoding

This web service contains methods to geocode an address. Supports coordinates in UTM 32 ETRS89/WGS84 and geographic coordinate system.

Service end point 1: <http://basicservices.webservice.krak.dk/map.aspx>

Service end point 2: <http://basicservices.webservice.krak.dk/mapping.aspx>

Routing

This web service contains methods to obtain route descriptions, estimated travelling time and travel distance between two or more addresses.

Service end point: <http://basicservices.webservice.krak.dk/routearch.aspx>

The content of the web services is described in detail in the following sections.

Login procedures

There are three ways of performing login in order to get access to Krak web service methods:

1. Login with **username and password to obtain a ticket** that will be used in every subsequent method request. The duration of the ticket is 30 minutes. Make sure to renew it before it expires! Use this login procedure when your application is installed in numerous locations.
2. Login with **username and IP to obtain a ticket** that will be used in every subsequent method request. The duration of the ticket is 30 minutes. Make sure to renew it before it expires! Use this login procedure when your applications can be associated with a static IP range e.g. a web server.
3. Login with **username and IP**. With this method you do call a login procedure before calling the methods. **This login procedure is deprecated and should not be used any longer. It is mentioned only because of backward compatibility issues!**

All three login procedures are described in details below.

If you experience [**KSF-00015**] **Invalid SoapHeader** or just have problems setting up the SOAP header, please refer to section “Setting up the SOAP header correctly”.

1. Username/password → ticket

This login procedure is based on tickets. The ticket is used for authenticating the web service method caller. A ticket can be obtained by calling the web service method – GetTicketByUser - and supplying a username and a password. This method is located in the following web service: <http://login.webservice.krak.dk/TicketCentral.asmx>.

Next, a Krak SOAP header must be set up by supplying a product number and the obtained ticket. With this information the web service framework will be able to check if the caller is authorized to call a given web service method.

Finally, the wanted web service method can be called, supplying it with the necessary parameter(s). If the authentication of the caller succeeds and he is authorized to call the method, the method will be invoked and a result is returned.

Example

The following ASP.NET example demonstrates how to obtain a ticket based on username/password and use this ticket plus a product number to call the GetCompanyByTn web service method.

You must be a registered customer with access to a product containing the GetCompanyByTn method to use this sample. If you have access to another product this sample can still serve as inspiration, where the only needed changes are the product number assignment and the web service method call at the end.

Two web service references added to the project are required:

<http://login.webservice.krak.dk/TicketCentral.asmx> and <http://basicservices.webservice.krak.dk/companysearch.asmx>. These references should be named TicketCentralProxy and CompanySearchProxy respectively.

```
// Get a ticket based on username and password
TicketCentralProxy.TicketCentral tc = new TicketCentralProxy.TicketCentral();
TicketCentralProxy.AuthTicket at = tc.GetTicketByUser("<Username>", "<Password>",
"en-US");

// Prepare webservice
CompanySearchProxy.CompanySearch proxy = new CompanySearchProxy.CompanySearch();

// Setup SOAP header
proxy.KrakSoapHeaderValue = new CompanySearchProxy.KrakSoapHeader();
proxy.KrakSoapHeaderValue.ticket = at.ticket;
proxy.KrakSoapHeaderValue.product = "IIP-TFORM-OPH1";

// Call webservice method
CompanySearchProxy.Company[] results;
results = proxy.GetCompanyByTn("<phoneNumber>");
```

2. Username/IP → ticket

This login procedure is based on tickets. The ticket is used for authenticating the web service method caller. Calling the web service method - `GetTicketByUserIP` - and supplying a valid username matching the IP address of the caller, a ticket can be obtained. This method is located in the following web service: <http://login.webservice.krak.dk/TicketCentral.asmx>.

Next, a Krak SOAP header must be set up by supplying a product number and the obtained ticket. With this information the web service framework will be able to check if the caller is authorized to call a given web service method.

Finally, the wanted web service method can be called, supplying it with the necessary parameter(s). If the authentication of the caller succeeds and he is authorized to call the method, the method will be invoked and a result is returned.

Example

The following ASP.NET example demonstrates how to obtain a ticket based on username/IP and use this ticket plus a product number to call the `GetCompanyByTn` web service method.

You must be a registered customer with access to a product containing the `GetCompanyByTn` method to use this sample. If you have access to another product this sample can still serve as inspiration, where the only needed changes are the product number assignment and the web service method call at the end.

Two web service references added to the project are required:

<http://login.webservice.krak.dk/TicketCentral.asmx> and
<http://basicservices.webservice.krak.dk/companysearch.asmx>. These references should be named `TicketCentralProxy` and `CompanySearchProxy` respectively.

```
// Get a ticket based on username and password
TicketCentralProxy.TicketCentral tc = new TicketCentralProxy.TicketCentral();
TicketCentralProxy.AuthTicket at = tc.GetTicketByUser("<Username>", "en-US");

// Prepare webservice
CompanySearchProxy.CompanySearch proxy = new CompanySearchProxy.CompanySearch();

// Setup SOAP header
proxy.KrakSoapHeaderValue = new CompanySearchProxy.KrakSoapHeader();
proxy.KrakSoapHeaderValue.ticket = at.ticket;
proxy.KrakSoapHeaderValue.product = "IIP-TFORM-OPH1";

// Call webservice method
CompanySearchProxy.Company[] results;
results = proxy.GetCompanyByTn("<phoneNumber>");
```

3. Username/IP (deprecated)

As opposed to the two other login procedures this login procedure is not based on tickets. It is instead based only on a username matching an IP address. **Notice, this method is no longer supported and exists only because of backward compatibility.**

To call a web service method, set up a Krak SOAP header and supply it with a valid username. Then call the web service method with the necessary parameter(s). If the authentication of the caller succeeds and he is authorized to call the method, the method will be invoked and a result is returned.

The following ASP.NET example demonstrates how to call the GetCompanyByTn web service method. You must be a registered customer with access to the GetCompanyByTn method to use this sample. If you have access to another method this sample can still serve as inspiration, where the only needed change is the web service method call at the end.

A web service reference added to the project is required:

<http://basicservices.webservice.krak.dk/companysearch.asmx>. This reference should be named CompanySearchProxy.

```
// Prepare webservice
CompanySearchProxy.CompanySearch proxy = new
CompanySearchProxy.CompanySearch();

// Setup SOAP header
proxy.KrakSoapHeaderValue = new CompanySearchProxy.KrakSoapHeader();
proxy.KrakSoapHeaderValue.username = "<Username>";

// Call webservice method
CompanySearchProxy.Company[] results;
results = proxy.GetCompanyByTn("<phoneNumber>");
```

Setting up the SOAP header correctly

Some developers have faced a problem in setting up the SOAP header correctly. According to the ASMX/WSDL the SOAP header should contain one or more of the following info: ticket, product and username regardless of what login procedure being used – see screen dumps of ASMX and WSDL XML structure below:

ASMX:

```
<KrakSoapHeader xmlns="http://webservice.krak.dk/">  
  <ticket>string</ticket>  
  <product>string</product>  
  <username>string</username>  
</KrakSoapHeader>
```

WSDL:

```
- <s:complexType name="KrakSoapHeader">  
  - <s:sequence>  
    <s:element minOccurs="0" maxOccurs="1" name="ticket" type="s:string" />  
    <s:element minOccurs="0" maxOccurs="1" name="product" type="s:string" />  
    <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string" />  
  </s:sequence>  
</s:complexType>
```

However, supplying all three parameters or a wrong combination of parameters will lead to the **[KSF-00015] Invalid SoapHeader** error. Use the table below to see which parameters you should add to the Krak SOAP header.

Login procedure	Ticket	Product	Username
1. Username/password → ticket	Yes	Yes	No
2. Username/IP → ticket	Yes	Yes	No
3. Username/IP	No	Yes	Yes

Methods

Tele search

GetTeleByTn

Search by telephone owners by telephone number.

```
Tele[] GetTeleByTn (  
    string telephoneNumber  
)
```

Parameters	telephoneNumber: Phone number e.g. "88383800"
Return values	Tele[]
Remarks	

GetTeleByName

Search for telephone owners by their name.

```
Tele[] GetTeleByName (  
    string name  
)
```

Parameters	name: Name of the telephone owner e.g. "Lars Olsen"
Return values	Tele[]
Remarks	

GetTeleByNameRoadName

Search for telephone owners by their name and road name.

```
Tele[] GetTeleByNameRoadName (  
    string name  
    string roadName  
)
```

Parameters	name: Name of the telephone owner e.g. "Lars Olsen" roadName: Name of the road where owner is located e.g. "Lærkevej".
Return values	Tele[]
Remarks	Both parameters are required.

GetTeleByNameRnPc

Search for telephone owners by road name and postal code.

```
Tele[] GetTeleByRnPc (  
    string roadName  
    string postalCode  
)
```

Parameters	roadName: Name of the road where owner is located e.g. "Lærkevej". postalCode: Code of the postal districts where owner belongs to e.g. "4700".
Return values	Tele[]
Remarks	Both parameters are required.

GetTeleByNamePostalCode

Search for telephone owners by their name and postal code.

```
Tele[] GetTeleByNamePostalCode (  
    string name  
    string postalCode  
)
```

Parameters	name: Name of the telephone owner e.g. "Lars Olsen" postalCode: Code of the postal districts where owner belongs to e.g. "4700".
Return values	Tele[]
Remarks	Both parameters are required.

GetTeleByRnHnnPc

Search for telephone owners by road name, house number and postal code.

```
Tele[] GetTeleByRnHnnPc (  
    string roadName  
    int houseNumberNumeric  
    string postalCode  
)
```

Parameters	roadName: Name of the road where owner is located e.g. "Lærkevej". houseNumberNumeric: The numeric part of a house number e.g. 42 postalCode: Code of the postal districts where owner belongs to e.g. "4700".
Return values	Tele[]
Remarks	All parameters are required.

GetTeleByNameAddr

Search for telephone owners by name and address of owner.

```
Tele[] GetTeleByNameAddr (  
    string name  
    string address  
)
```

Parameters	name: Name of the telephone owner e.g. "Lars Olsen" address: Address where owner belongs to e.g. "Sydmarken 44A, 2860 Søborg".
Return values	Tele[]
Remarks	Both parameters are required. As a minimum the address must include road name and postal code. Examples of valid inputs are "Sydmarken 2860", "Sydmarken 44A 2860", "Sydmarken 44A, 2860" and "Sydmarken 44A, 2860 Søborg".

Company search

GetCompanyByCn

Search for companies by company name.

```
Company[] GetCompanyByCn (  
    string name  
)
```

Parameters	name: Name of the company e.g. "Eniro Danmark A/S" or "Eniro".
Return values	Company[]
Remarks	

GetCompanyByTn

Search for companies by their telephone number.

```
Company[] GetCompanyByTn (  
    string telephoneNumber  
)
```

Parameters	telephoneNumber: Phone number e.g. "88383800".
Return values	Company[]
Remarks	

GetCompanyByRnHnnPc

Search for companies by road name, house number and postal code.

```
Company[] GetCompanyByRnHnnPc (  
    string roadName  
    int houseNumberNumeric  
    string postalCode  
)
```

Parameters	roadName: Name of the road where company is located e.g. "Lærkevej". houseNumberNumeric: The numeric part of a house number e.g. 42 postalCode: Code of the postal districts of company e.g. "4700".
Return values	Company[]
Remarks	All parameters are required.

GetCompanyByAddr

Search for companies by address.

```
Company[] GetCompanyByAddr (  
    string address  
)
```

Parameters	address: Address where company is located.
Return values	Company[]
Remarks	Only road name of address is required. Examples of valid values are “Sydmarken 44A 2860 Søborg”, “Sydmarken 44” and “Sydmarken”.

GetCompanyByCnAddr

Search for companies by their name and address.

```
Company[] GetCompanyByCnAddr (  
    string name  
    string address  
)
```

Parameters	name: Name of the company e.g. “Eniro Danmark A/S” or “Eniro” address: Address where company is located.
Return values	Company[]
Remarks	Only one of the parameters is required e.g. search by name or address or both. Only road name of address is required. Examples of valid values are “Sydmarken 44A 2860 Søborg”, “Sydmarken 44” and “Sydmarken”.

GetCompanyByCVR

Search for companies by their CVR-number.

```
Company[] GetCompanyByCVR (  
    long CVRNumber  
)
```

Parameters	CVRNumber: CVR of the company e.g. 12345678
Return values	Company[]
Remarks	

GetCompanyByCVRP

Search for companies by their CVR P-number.

```
Company[] GetCompanyByCVRP (  
    long productionUnitNumber  
)
```

Parameters	productionUnitNumber: CVR P of the company e.g. 1234567890
Return values	Company[]
Remarks	The returned array size should be 0 or 1.

GetCompanyByKn

Search for companies by their Krak number.

```
Company[] GetCompanyByKn (  
    long krakNumber  
)
```

Parameters	krakNumber: Krak ID of the company e.g. 123456.
Return values	Company[]
Remarks	The returned array size should be 0 or 1.

GetCompanyByCnPc

Search for companies by name and postal code.

```
Company[] GetCompanyByCnPc (  
    string name  
    string postalCode  
)
```

Parameters	name: Name of the company e.g. "Eniro Danmark A/S" or "Eniro" postalCode: Code of the postal districts where company is located e.g. "4700".
Return values	Company[]
Remarks	Only one of the parameters is required e.g. search by name or postal code or both

GetCVRPNumberByCVRNumber

Search for production units by CVR number.

```
long[] GetCVRPNumberByCVRNumber (
    long CVRNumber
)
```

Parameters	CVRNumber: CVR of the company e.g. 12345678
Return values	long[]
Remarks	

Road search

GetRoadByRnHnnPc

Search road by road name, house number and postal code.

```
Address GetRoadByRnHnnPc (
    string roadName,
    int houseNumberNumeric,
    int postalCode
)
```

Parameters	roadName: Name of the road where company is located e.g. "Lærkevej". houseNumberNumeric: The numeric part of a house number e.g. 42 postalCode: Code of the postal districts e.g. "4700".
Return values	Road []
Remarks	All parameters are required.

GetRoadByRnPc

Search road by road name and postal code.

```
Address GetRoadByRnHnnPcPln (
    string roadName,
    int postalCode,
)
```

Parameters	roadName: Name of the road where company is located e.g. "Lærkevej". postalCode: Code of the postal districts to e.g. "4700".
Return values	Road []
Remarks	Both parameters are required

Address search

GetAddressByPd

Search postal districts by postal district.

```
Address GetAddressByPd (  
    string postalDistrict  
)
```

Parameters	postalDistrict: Name of postal district e.g. "Odense" or "Odense C"
Return values	Address []
Remarks	Only Address.PostalCode and Address.PostalDistrict contain values

GetAddressByPc

Search postal districts by postal code.

```
Address GetAddressByPc (  
    int postalCode  
)
```

Parameters	postalCode: Code of postal district e.g. 5000.
Return values	Address []
Remarks	Only Address.PostalCode and Address.PostalDistrict contain values

GetAddressByRnHnnPc

Search address by road name, house number and postal code.

```
Address GetAddressByRnHnnPc (  
    string roadName,  
    int houseNumberNumeric,  
    int postalCode  
)
```

Parameters	roadName: Name of the road e.g. "Lærkevej". houseNumberNumeric: The numeric part of a house number e.g. 42 postalCode: Code of the postal district e.g. "4700".
Return values	Address []
Remarks	All parameters are required.

GetAddressByRnPc

Search address by road name and postal code.

```
Address GetAddressByRnPc (  
    string roadName,  
    int postalCode  
)
```

Parameters	roadName: Name of the road e.g. "Lærkevej". postalCode: Code of the postal district e.g. "4700".
Return values	Address []
Remarks	Both parameters are required.

GetAddressByRnrHnn

Search address by road number and house number.

```
Address GetAddressByRnrHnn (  
    long roadNumber,  
    int houseNumberNumeric  
)
```

Parameters	roadNumber: Krak road id of address e.g. 107100. houseNumberNumeric: The numeric part of a house number e.g. 42.
Return values	Address []
Remarks	Both parameters are required.

GetAddressByTn

Search address by telephone number.

```
Address[] GetAddressByTn (  
    string telephoneNumber  
)
```

Parameters	telephoneNumber: Phone number e.g. "88383800".
Return values	Address []
Remarks	

GetAddressByRnHnnHnaPcCn

Search address by road name, house number numeric, house number alphabetic, postal code and city name.

```
Address GetAddressByRnHnnHnaPcCn (  
    string roadName,  
    string houseNumberNumeric,  
    string houseNumberCharacter,  
    string postalCode,  
    string cityName,  
)
```

Parameters	roadName: Name of the road e.g. "Lærkevej". houseNumberNumeric: The numeric part of a house number e.g. "42". houseNumberCharacter: The alphabetic part of a house number e.g. "A". postalCode: Code of the postal district e.g. "4700". cityName: Name of the city e.g. Næstved.
Return values	AddressEx[]
Remarks	Only road name of address is required

GetMapReferenceByRnrHnn

Get Krak map reference by Krak road number and numeric house number.

```
Address GetMapReferenceByRnrHnn (  
    long roadNumber,  
    int houseNumberNumeric  
)
```

Parameters	roadNumber: Krak road id of address e.g. 107100. houseNumberNumeric: The numeric part of a house number e.g. 42.
Return values	MapReference[]
Remarks	Both parameters are required.

ParseNSearchAddress

Search addresses by semi structured/concatenated address elements.

```
Address[] ParseNSearchAddress (  
    string address  
)
```

Parameters	address: Semi structured address e.g. "Sydmarken 44A 2860 Søborg".
Return values	Address[]
Remarks	Only road name of address is required. Examples of valid values are "Sydmarken 44A 2860 Søborg", "Sydmarken 44" and "Sydmarken".

ValidateAddressByRnHnnPc

Validate address by road name, house number and postal code.

```
Address ValidateAddressByRnHnnPc (  
    string roadName,  
    int houseNumberNumeric,  
    int postalCode  
)
```

Parameters	roadName: Name of the road e.g. "Lærkevej". houseNumberNumeric: The numeric part of a house number e.g. 42 postalCode: Code of the postal district e.g. "4700".
Return values	Address
Remarks	Use validation when you just want to validate that an address exists. Do not use it for lookup and parsing. The method returns only one address – not an array! All parameters are required.

Address Geoding

GetPoint

Address geocoding by road number, numeric house number and alpha house number. Returns UTM32 ETRS89 coordinates.

```
Point GetPoint (  
    long roadNumber,  
    int houseNumberNumeric,  
    string houseNumberCharacter  
)
```

Parameters	roadNumber: Krak road id of address e.g. 107100. houseNumberNumeric: The numeric part of a house number e.g. 42. houseNumberCharacter: The character part of a house number e.g. "A".
Return values	Point
Remarks	houseNumberCharacter should be set to "" when no house number character is present e.g. "42".

GetGPSPoint

Address geocoding by road number, numeric house number and alpha house number. Returns lat/long coordinates.

```
Point GetGPSPoint (  
    long roadNumber,  
    int houseNumberNumeric,  
    string houseNumberCharacter  
)
```

Parameters	roadNumber: Krak road id of address e.g. 107100. houseNumberNumeric: The numeric part of a house number e.g. 42. houseNumberCharacter: The character part of a house number e.g. "A".
Return values	Point
Remarks	houseNumberCharacter should be set to "" when no house number character is present e.g. "42".

GetPoints

Address geocoding of multiple addresses. Each address is defined by Krak road number, numeric house number and alpha house number. Returns UTM32 ETRS89 coordinates.

```
Point GetGPSPoint (  
    long roadNumber,  
    int houseNumberNumeric,  
    string houseNumberCharacter  
)
```

Parameters	roadNumber: Krak road id of address e.g. 107100. houseNumberNumeric: The numeric part of a house number e.g. 42. houseNumberCharacter: The character part of a house number e.g. "A".
Return values	Point []
Remarks	houseNumberCharacter should be set to "" when no house number character is present e.g. "42".

GetCoordinatesByAddress

Address geocoding by road name, numeric house number and postal code. Returns UTM32 ETRS89 coordinates.

```
AddressCoordinate GetCoordinatesByAddress (  
    long roadNumber,  
    int houseNumberNumeric,  
    string houseNumberCharacter  
)
```

Parameters	roadNumber: Krak road id of address e.g. 107100. houseNumberNumeric: The numeric part of a house number e.g. 42. houseNumberCharacter: The character part of a house number e.g. "A".
Return values	AddressCoordinate
Remarks	Returns two sets of coordinates; one for the address location and one for the access-by-road location. houseNumberCharacter should be set to "" when no house number character is present e.g. "42".

UTM2GEO

Converts an UTM coordinate (UTM 32/ETRS89) to a Geographic longitude/lattitude coordinate.

```
Point UTM2GEO (  
    double northing,  
    double easting  
)
```

Parameters	northing: The northing of a UTM 32 coordinate e.g. 6220000,00. eastng: The eastng of a UTM 32 coordinate e.g. 550000,00.
Return values	Point
Remarks	

GEO2UTM

Converts a Geographic longitude/lattitude coordinate to a UTM coordinate (UTM 32/ETRS89).

```
Point GEO2UTM (  
    double longitude,  
    double lattitude  
)
```

Parameters	longitude: The longitude of a geo coordinate e.g. 9,1234. lattitude: The lattitude of a geo coordinate e.g. 56,1711.
Return values	Point
Remarks	

Routing

GetRouteInfo

Get distance and travelling time between two addresses.

```
RouteInfo GetRouteInfo (  
    long RoadNumber1  
    int houseNumberNumeric1  
    string houseNumberCharacter1  
    long RoadNumber2  
    int houseNumberNumeric2  
    string houseNumberCharacter2  
)
```

Parameters	<p>roadNumber1: Krak road id e.g. 107100 for origin address.</p> <p>houseNumberNumeric1: The numeric part of a house number e.g. 42 for origin address.</p> <p>houseNumberCharacter1: The character part of a house number e.g. "A" for origin address.</p> <p>roadNumber2: Krak road id e.g. 107100 for destination address.</p> <p>houseNumberNumeric2: The numeric part of a house number e.g. 42 for destination address.</p> <p>houseNumberCharacter2: The character part of a house number e.g. "A" for destination address.</p>
Return values	RouteInfo
Remarks	houseNumberCharacter1 and houseNumberCharacter2 should be set to "" when no house number character is present e.g. "42".

GetRoute

Get distance, travelling time and route direction/description between two addresses.

```
Route GetRoute (  
    long RoadNumber1  
    int houseNumberNumeric1  
    string houseNumberCharacter1  
    long RoadNumber2  
    int houseNumberNumeric2  
    string houseNumberCharacter2  
)
```

Parameters	<p>roadNumber1: Krak road id e.g. 107100 for origin address.</p> <p>houseNumberNumeric1: The numeric part of a house number e.g. 42 for origin address.</p> <p>houseNumberCharacter1: The character part of a house number e.g. "A" for origin address.</p> <p>roadNumber2: Krak road id e.g. 107100 for destination address.</p> <p>houseNumberNumeric2: The numeric part of a house number e.g. 42 for destination address.</p> <p>houseNumberCharacter2: The character part of a house number e.g. "A" for destination address.</p>
Return values	Route
Remarks	houseNumberCharacter1 and houseNumberCharacter2 should be set to "" when no house number character is present e.g. "42".

GetRouteVia

Get distance, travelling time and route direction/description between two or more addresses.

```
RouteInfo GetRouteVia (  
    addressKeys[] addressKey  
)
```

Parameters	<p>addressKey: Object that includes roadNumber, houseNumberNumeric and houseNumberCharacter.</p> <p>roadNumber: Krak road id e.g. 107100 for destination address.</p> <p>houseNumberNumeric: The numeric part of a house number e.g. 42 for destination address.</p> <p>houseNumberCharacter: The character part of a house number e.g. "A" for destination address.</p>
Return values	Route
Remarks	houseNumberCharacter should be set to "" when no house number character is present e.g. "42".

Objects

Description of objects used by the web service methods.

Address

The address object is used for holding information about an address. Depending on the specific use of the object not all properties may contain useful information.

```
class Address {  
    string RoadName  
    long RoadNumber  
    string PlaceName  
    string PostalCode  
    string PostalDistrict  
    int HouseNumberNumericFrom  
    string HouseNumberCharacterFrom  
    string Floor  
    string Door  
    int MunicipalityCode  
    int CountyCode  
}
```

Property	Description
RoadName	Road name (vejnavn) e.g. Lærkevej
RoadNumber	Krak road ID. This property is often used as input when calling routing and geocoding web services.
PlaceName	Place name (stednavn). An address can have a place name associated to it in addition to the postal district. The property rarely contains a value as it is only used when an address otherwise would be non-unique.
PostalCode	Postal code (postnummer) e.g. 4000
PostalDistrict	Postal district (postdistrikt), e.g. Roskilde
HouseNumberNumericFrom	The numeric part of the house number. If, for example, the house number for a given address is 42A, then this property will hold "42".
HouseNumberCharacterFrom	The character part of the house number. If, for example, the house number for a given address is 42A, then this property will hold "A".
Floor	The floor (etage) at which the address exists, e.g. 3.
Door	The door (side) at which the address exists, e.g. TH
MunicipalityCode	Code for the municipality (kommunekode) to which the address belongs, e.g. 101 (Copenhagen).
CountyCode	DEPRECATED! Contains no value. Use the municipalitycode to look up the county/region in a static table.

AddressEx

The addressEx object is an extension of the Address object. Examples of additional info are address position and relations to the Danish administrative districts e.g. municipalities, regions and courts. Depending on the specific use of the object not all properties may contain useful information.

```
class Address {
    long RoadNumber
    string RoadName
    int HouseNumberNumeric
    string HouseNumberCharacter
    string PlaceName
    string PostalCode
    string PostalDistrict
    int MunicipalityCode
    string MunicipalityName
    int MunicipalityRoadNumber
    int RegionCode
    string RegionName
    int CourtNumber
    string CourtName
    long RoadNumberOld
    int HouseNumberNumericOld
    string HouseNumberCharacterOld
    string PlaceNameOld
    string PostalCodeOld
    string PostalDistrictOld
    int MunicipalityCodeOld
    string MunicipalityNameOld
    int CountyCode
    string CountyName
    int CourtNumberOld
    string CourtNameOld
    double RoadCoordinateX
    double RoadCoordinateY
    double PlacementCoordinateX
    double PlacementCoordinateY
    double RoadLongitude
    double RoadLatitude
    double PlacementLongitude
    double PlacementLatitude
    int Precision
}
```

Property	Description
RoadNumber	Krak road ID. This property is often used as input when calling routing and geocoding web services.
RoadName	Road name (vejnavn) e.g. Lærkevej
HouseNumberNumeric	The numeric part of the house number. If, for example, the house number for a given address is 42A, then this property will hold "42".
HouseNumberCharacter	The character part of the house number. If, for example, the house number for a given address is 42A, then this property will hold "A".
PlaceName	Place name (stednavn). An address can have a place name associated to it in addition to the postal district. The property rarely contains a value as it is only used when an address otherwise would be non-unique.
PostalCode	Postal code (postnummer) e.g. 4000

PostalDistrict	Postal district (postdistrikt), e.g. Roskilde
MunicipalityCode	Code for the municipality (kommunekode) to which the address belongs, e.g. 101 (Copenhagen).
MunicipalityName	Name of the municipality to which the address belongs.
MunicipalityRoadNumber	Road ID according to the municipality.
RegionCode	Code for the region to which the address belongs.
RegionName	Name of the region to which the address belongs.
CourtNumber	Code/number for the court to which the address belongs.
CourtName	Name of the court to which the address belongs.
RoadNumberOld	DEPRECATED!
HouseNumberNumericOld	DEPRECATED!
HouseNumberCharacterOld	DEPRECATED!
PlaceNameOld	DEPRECATED!
PostalCodeOld	DEPRECATED!
PostalDistrictOld	DEPRECATED!
MunicipalityCodeOld	DEPRECATED!
MunicipalityNameOld	DEPRECATED!
CourtNumberOld	DEPRECATED!
CourtNameOld	DEPRECATED!
CountyCode	DEPRECATED! Use the municipalityCode or regionCode.
CountyName	DEPRECATED! Use the municipalityName or RegionName.
RoadCoordinateX	The x/Easting part of the address location. Use this position for routing.
RoadCoordinateY	The y/Northing part of the address location. Use this position for routing.
PlacementCoordinateX	The x/Easting part of the address location. Use this position for displaying the address on a map.
PlacementCoordinateY	The y/Northing part of the address location. Use this position for displaying the address on a map.
RoadCoordinateLongitude	The longitude of the address location. Use this position for routing.
RoadCoordinateLatitude	The latitude of the address location. Use this position for routing.
PlacementLongitude	The longitude part of the address location. Use this position for displaying the address on a map.
PlacementLatitude	The latitude part of the address location. Use this position for displaying the address on a map.

MapReference

The map reference object is used for holding information about a specific position in a Krak map book.

```
class MapReference {  
    string Book  
    string City  
    string Page  
    string Grid  
}
```

Property	Description
Book	Name of Krak map book e.g. "København og Omegn"
City	City name
Page	Page number e.g. 136
Grid	Grid reference e.g. F6

Point

The Point object is used for holding a geographic position.

```
class Point {  
    double X  
    double Y  
}
```

Property	Description
X	The x/Easting part of a coordinate.
Y	The y/Northing part of a coordinate.

AddressCoordinate

The AddressCoordinate object holds two address-related locations; one for the door location of the address and one for the access-by-road location.

```
class PlacementLocation {  
    Point PlacementLocation  
    Point AccessRoad  
}
```

Property	Description
PlacementLocation	A Point object holding the address location.
AccessRoad	A Point object holding the access-by-address location.

Company

The company object is used for holding information about a company. Depending on the specific use of the object not all properties may contain useful information.

```
class Company {  
    string GUID  
    string CompanyName  
    long CVRNumber  
    long ProductionUnitNumber  
    long KrakNumber  
    Address Address  
    ContactInfo ContactInfo  
}
```

Property	Description
GUID	Reserved.
CompanyName	The name of the company.
CVRNumber	The CVR number of the company.
ProductionUnitNumber	The CVR P number of the company.
KrakNumber	Krak's internal number for identifying a company.
Address	An Address object containing the address of the company.
ContactInfo	A ContactInfo object containing contact information such as phone numbers and email address of the company.

ContactInfo

The contact info object is used for holding contact information associated, for example, with a company or telephone registrant. Depending on the specific use of the object not all properties may contain useful information.

```
class ContactInfo {  
    string TelephoneNumber  
    string MobileTelephoneNumber  
    string FaxNumber  
    string EmailAddress  
    string URL  
}
```

Property	Description
TelephoneNumber	Phone number.
MobileTelephoneNumber	Mobile phone number.
FaxNumber	Fax number.
EmailAddress	E-mail address.
URL	Website URL.

Tele

The tele object is used for holding information about a telephone registrant. Depending on the specific use of the object not all properties may contain useful information.

```
class Tele {  
    string CompanyName  
    string GUID  
    string FirstName  
    string LastName  
    string KrakTeleRegistrationNumber  
    string KrakNumber  
    ContactInfo ContactInfo  
    Address Address  
}
```

Property	Description
CompanyName	If the telephone registrant is a company, then this property holds the company name, otherwise it will be empty.
GUID	Reserved.
FirstName	If the telephone registrant is a person, then this property holds the first name of the person, otherwise it will be empty.
LastName	If the telephone registrant is a person, then this property holds the last name of the person, otherwise it will be empty.
KrakTeleRegistrationNumber	Krak's internal number for identifying a telephone registrant.
KrakNumber	Krak's internal number for identifying a company.
Address	An Address object containing the address of the company.
ContactInfo	A ContactInfo object containing contact information such as phone numbers and email address of the company.

Road

The road object is used for holding information about a road. Depending on the specific use of the object not all properties may contain useful information.

```
class Road {  
    string RoadName  
    long RoadNumber  
    string PlaceName  
    string PostalCode  
    string PostalDistrict  
}
```

Property	Description
RoadName	Road name (vejnavn) e.g. Lærkevej
RoadNumber	Krak road ID. This property is often used as input when calling routing and geocoding web services.
PlaceName	Place name (stednavn). An address can have a place name associated to it in addition to the postal district. The property rarely contains a value as it is only used when an address otherwise would be non-unique.
PostalCode	Postal code (postnummer) e.g. 4000
PostalDistrict	Postal district (postdistrikt), e.g. Roskilde

RouteInfo

The RouteInfo object is used for holding information about a route in terms of travelling time and distance. It does not contain a route description!

```
class RouteInfo {  
    double TotalLength  
    double TotalTime  
}
```

Property	Description
TotalLength	Total length of route in meters.
TotalTime	Total travelling time of route in minutes.

Route

The Route object is used for holding information about a route description. It also includes general information about total travelling time and distance.

```
class Route {  
    Address[] Addresses  
    RouteDescription[] RouteDescriptions  
}
```

Property	Description
Addresses	An array of Address object . As a minimum the array contains the origin and destination address.
RouteDescription	An array of RouteDescription object . A route without via points has an array size of 1.

RouteDescription

The RouteDescription object is holding the route description and the total length and travelling time of the route.

```
class RouteDescription {  
    Part[] Parts  
    double TotalLength  
    double TotalTime  
}
```

Property	Description
Parts	A part is a general, overall and logical part of a route in terms of road classification. A part contains an array of the Part object .
TotalLength	Total length of route in meters.
TotalTime	Total travelling time of route in minutes.

Part

The Part object contains a collection of logically grouped road segments. A Part always starts from the origin address and ends when the route enters a logically different road category e.g. a highway or a distinct road number such as “E47” and “O4”. When leaving the highway the part stops and a new one starts based on another group of road segments.

```
class Part {
    Segment[] Segments
    double TotalLength
    double TotalTime
    string RouteNumber
    double Length
    double Time
}
```

Property	Description
Segments	A collection of the Segment object .

Segment

A Segment object contains routing information about the road segment.

```
class Segment {
    int Direction
    double Length
    double TotalLength
    double TotalTime
    int RoadID
    int RoadType
    int NextTurn
    int PrevTurn
    double Time
    double Weight
    string RouteNumber
    int RoundCount
    string RoadName
    int RoadTypePrev
    int RoadIDPrev
    string Icon
    string TextDescription
}
```

Due to the complexity of the Segment object we encourage you to contact Krak before you implement the GetRoute method.

Contact info can be found on <http://www.krakwebservices.dk>

Frequently asked questions

How do I geocode an address?

Step 1 is to lookup, validate or parse your address. Depending on your address input you should use methods like `GetAddressByRnHnnPc`, `ValidateAddress` and `ParseNSearchAddress`. All the methods will return an `Address` object that contains a `Road-ID` for the address.

Step 2 is to do the geocoding with a method like `GetCoordinateByAddress`. This method requires the `Road-ID` you just obtained from step 1 and the numeric and character part for the house number.

Can I geocode to latitude/longitude rather than UTM32 ETRS89?

Yes. Use method `GetGPSPoint` instead of `GetCoordinateByAddress`.

How do I get the distance between two addresses?

Step 1 is to lookup, validate or parse your address. Depending on your address input you should use methods like `GetAddressByRnHnnPc`, `ValidateAddress` and `ParseNSearchAddress`. All the methods will return an `Address` object that contains a `Road-ID` for the address.

Step 2 is to do the distance calculation with method `GetRouteInfo`. This method requires the `Road-ID` you just obtained from step 1 and the numeric and character part for the house number.