

---

# Naviscale API

---

*Willem Engel*  
*willem@naviscale.com*  
*Version 1*

## **Samenvatting**

Dit document bevat de technische beschrijving van de beschikbare API opties voor het Naviscale systeem.

# Inhoudsopgave

<b>1</b>	<b>Introductie</b>	<b>4</b>
<b>2</b>	<b>Front End</b>	<b>5</b>
2.1	Positie . . . . .	5
2.1.1	Parameters . . . . .	5
2.1.2	Voorbeeld . . . . .	5
2.2	Weergave . . . . .	6
2.2.1	Parameters . . . . .	6
2.2.2	Voorbeeld . . . . .	6
<b>3</b>	<b>Management</b>	<b>7</b>
3.1	Webservice . . . . .	8
3.2	Authenticatie . . . . .	8
3.3	Test Functies . . . . .	8
3.3.1	Testen van de SOAP koppeling . . . . .	8
3.3.2	Testen van de SOAP authenticatie . . . . .	8
3.4	Classes . . . . .	9
3.4.1	Locatie . . . . .	9
3.4.2	Response . . . . .	10
3.4.3	Categorie . . . . .	10
3.5	Locatie Beheer . . . . .	11
3.5.1	Toevoegen . . . . .	11
3.5.2	Wijzigen . . . . .	11
3.5.3	Verwijderen . . . . .	11
3.5.4	Ophalen van een enkele locatie . . . . .	12
3.5.5	Ophalen van een meedere locaties . . . . .	12
3.6	Afbeeldingen en Fotos . . . . .	13
<b>A</b>	<b>Voorbeelden</b>	<b>14</b>
A.1	PHP . . . . .	14
A.1.1	Auth Header . . . . .	14
A.1.2	Locatie . . . . .	14
A.1.3	SOAP koppeling . . . . .	15
A.1.4	Locatie Toevoegen . . . . .	15
A.1.5	Objecten Ophalen . . . . .	15
A.1.6	Return Values . . . . .	15
A.2	ASP.Net . . . . .	16

A.2.1	Visual Studio . . . . .	16
A.2.2	SOAP koppeling . . . . .	16
A.2.3	Locatie Toevoegen . . . . .	16
A.2.4	Objecten Ophalen . . . . .	16

## 1 Introductie

Naviscale bestaat uit twee delen, de gebruikers kant (**Frontend**), en de beheerders kant (**Management**).

De frontend bevat enkele opties om de kaart te manipuleren onafhankelijk van de in de Management module ingestelde standaard opties. Dit zijn opties voor de positie van de kaart, het zoomlevel, de sidebar, enzovoort.

De Management module heeft een uitgebreide webservice welke aan te spreken is met elk programma dat het *SOAP* protocol ondersteunt. Deze webservice biedt opties voor het beheren van alle op de kaart geplaatste locaties.

## 2 Front End

Opties voor de front end kunnen als HTTP GET parameters meegegeven worden aan de kaart.

### 2.1 Positie

#### 2.1.1 Parameters

- ref** Geef een referentie nummer (vrij in te vullen voor elk object) op, waar de kaart bij het laden op gecentreerd wordt.
- point** Geef een point id op, waar de kaart bij het laden op gecentreerd wordt. Het point id is de unieke index van een locatie in het Naviscale systeem.
- lat,lng** Geef een latitude en longitude op waar de kaart bij het laden op gecentreerd wordt. Werkt alleen indien beide waarden geldige getallen zijn.
- zoom** Geef een zoom niveau op waar de kaart bij het laden op gecentreerd wordt. Niveau **0** is volledig uitgezoomed, **17** is volledig ingezoomed.

#### 2.1.2 Voorbeeld

```
map.aspx?lat=51.046466&lng=13.728619&zoom=13
```

Kaart starten op een andere locatie

## 2.2 Weergave

### 2.2.1 Parameters

- map** Geef het standaard kaart type op. Geldige opties zijn:
- 0 Gewone kaart (*standaard*)
  - 1 Sateliet kaart
  - 2 Combinatie van sateliet en gewone kaart
- bar** Geef op of de sidebar in- of uitgeklaapt moet zijn. Geldige opties zijn:
- 0 Ingeklaapt
  - 1 Uitgeklaapt (*standaard*)
- tab** Geef op welk tabblad in de sidebar actief moet zijn. Geldige opties zijn:
- category** Het tabblad met de categorieën en kaart opties (*standaard*)
  - search** Het tabblad met zoek opties
- objs** Geef op of de objecten zichtbaar zijn. Geldige opties zijn:
- 0 Verbergen
  - 1 Weergeven (*standaard*)
- pois** Geef op of de overige locaties (*POI's*) zichtbaar zijn. Geldige opties zijn:
- 0 Verbergen
  - 1 Weergeven (*standaard*)

### 2.2.2 Voorbeeld

```
map.aspx?bar=0&pois=0
```

Kaart starten met verborgen sidebar en POI's

### 3 Management

De management API bestaat uit een beveiligde *SOAP* webservice waarmee alle locaties beheerd kunnen worden. Voor de veiligheid is de webservice alleen bereikbaar via het *HTTPS* protocol.

Deze webservice is ontwikkeld om websitebouwers de mogelijkheid te geven om elk willekeurig Content Management Systeem te koppelen met de Naviscale kaart. De objecten op de website worden dan automatisch op de Naviscale kaart geplaatst, gewijzigd en verwijderd.

Een korte beschrijving van het *SOAP* protocol is te vinden op Wikipedia.

<http://en.wikipedia.org/wiki/SOAP>

### 3.1 Webservice

De webservice is te vinden op:

```
https://maps.naviscale.com/Management/Webservice.asmx
```

Deze pagina toont ook een automatisch gegenereerde beschrijving van alle beschikbare functies, hun parameters, en responses.

XML Service definities (*WSDL*) zijn te vinden op:

```
https://maps.naviscale.com/Management/Webservice.asmx?WSDL
```

Deze definities kunnen gebruikt worden om de koppeling automatisch te genereren, bijvoorbeeld met *Microsoft Visual Studio*.

### 3.2 Authenticatie

Voor gebruik van de webservice is het noodzakelijk een **Auth Header** mee te sturen in de *SOAP Request*.

De header ziet er als volgt uit:

```
<Header >
  <AuthHeader xmlns="naviscale.com">
    <Username>string</Username >
    <Password>string</Password >
  </AuthHeader >
</Header >
```

Naviscale SOAP Auth Header

Gebruikersnaam en wachtwoord zijn gelijk aan die voor de management pagina.

### 3.3 Test Functies

Voor het testen van de webservice zijn twee functies beschikbaar. Een functie voor het testen van de *SOAP* koppeling in het algemeen, en een voor het testen van de authenticatie.

#### 3.3.1 Testen van de SOAP koppeling

De simpelste test is een functie zonder parameters en authenticatie, die een *Hello World* response geeft.

De uitgebreide functie beschrijving staat op:

```
Webservice.asmx?op=HelloWorld
```

#### 3.3.2 Testen van de SOAP authenticatie

→ Sectie 3.2

De volgende test functie geeft de mogelijkheid om de **Auth Header** te testen. Deze functie verwacht een lege request met alleen de authenticatie header, en geeft de gebruikersnaam, of een foutmelding, als response.

De uitgebreide functie beschrijving staat op:

```
Webservice.asmx?op=HelloWorldAuth
```

## 3.4 Classes

### 3.4.1 Locatie

De *Locatie* is de basis van de applicatie. Een locatie ziet er als volgt uit:

```
<Location>
  <PointId>int</PointId>
  <Lat>decimal</Lat>
  <Lng>decimal</Lng>
  <Name>string</Name>
  <Address>string</Address>
  <CategoryId>int</CategoryId>
  <CategoryName>string</CategoryName>
  <Active>boolean</Active>
  <ObjectRefId>int</ObjectRefId>
  <ObjectPrice>decimal</ObjectPrice>
  <ObjectText>string</ObjectText>
  <ObjectURL>string</ObjectURL>
</Location>
```

#### Naviscale Locatie

**PointId** Uniek identificatie nummer binnen de Naviscale applicatie. Dit nummer wordt bij het toevoegen automatisch aangemaakt.

**Lat, Lng** De latitude en longitude van de locatie.

**Name** De naam van de locatie.

**Address** Het adres van de locatie, als vrij in te voeren string. Het standaard formaat voor het adres is als volgt:

Straatnaam 00, 0000AA Plaats, Land

Het is niet noodzakelijk dit formaat te gebruiken. Voor het automatisch opzoeken van latitude en longitude tijdens het toevoegen van een locatie is het echter wel aan te raden, om verkeerde resultaten uit te sluiten.

→ Sectie 3.4.3

**CategoryId** Het identificatie nummer van de categorie waar de locatie onder valt. Categorie-nummers zijn als volgt ingedeeld:

1000 - 1999 Objecten

2000 - 2999 Points of Interest

3000 - 3999 Fotos

**CategoryName** De naam van de categorie van de locatie. Categorie naam en nummer zijn aan elkaar gekoppeld.

**Active** Boolean waarde die bepaald of de locatie zichtbaar is op de kaart.

**ObjectRefId** Vrij op te geven *Referentie Id* om het koppelen van de locatie aan een bestaande database makkelijker te maken. Alle functies in de webservice kunnen via deze referentie aangeroepen worden, waardoor het niet nodig is zelf koppelingen op te slaan.

De gebruiker is zelf verantwoordelijk voor het bijhouden van deze referenties, er zijn vanuit Naviscale geen beperkingen en controles op dubbele referenties en dergelijke.

**ObjectPrice** De prijs van het aan de locatie gekoppelde object.

**ObjectText** Korte beschrijving van het aan de locatie gekoppelde object. Er is beperkt ruimte beschikbaar, lange beschrijvingen passen niet goed in de pagina.

**ObjectURL** Een *URL* met meer informatie over het aan de locatie gekoppelde object.

Gebruik deze link om te verwijzen naar het object op de eigen pagina van de makelaar. Zorg ervoor dat volledige links gebruikt worden, inclusief qualificatie zoals `http://` of `https://`.

### 3.4.2 Response

De meeste functies in de webservice geven een standaard *Response* na het uitvoeren.

```
<Response>
  <Success>boolean</Success>
  <Message>string</Message>
  <Location>
    ...
  </Location>
</Response>
```

Naviscale Response

**Success** Boolean die aangeeft of de functie gelukt is of niet.

**Message** Kort status bericht met het resultaat van de functie. Bevat bij niet gelukte acties de foutmelding.

→ Sectie 3.4.1 **Location** Bevat bij toevoegen, wijzigen, en opehalen de *Locatie* die het resultaat is van de functie.

### 3.4.3 Categorie

Alle locaties zijn ingedeeld in categorieën. Elke categorie kan meerdere subcategorieën hebben.

```
<Cat>
  <CatId>int</CatId>
  <Name>string</Name>
  <Categories>
    <Cat>
      ...
    </Cat>
    <Cat>
      ...
    </Cat>
    <Cat>
      ...
    </Cat>
  </Categories>
</Cat>
```

Naviscale Categorie

**CatId** Het identificatienummer van de categorie. Categorie-nummers zijn als volgt ingedeeld:

1000 - 1999 Objecten

2000 - 2999 Points of Interest

3000 - 3999 Fotos

**Name** De naam van de categorie.

**Categories** Een array met alle subcategorieën in deze categorie.

## 3.5 Locatie Beheer

### 3.5.1 Toevoegen

Een locatie toevoegen gaat via de functie `LocationSave`, zoals beschreven op de webservice:

```
Webbservice.asmx?op=LocationSave
```

→ Sectie 3.4.1

`LocationSave` verwacht een `locatie`, waarbij een paar parameters extra belangrijk zijn:

- PointId** Laat het `PointId` leeg of 0 voor het toevoegen van een nieuwe locatie.
- Lat, Lng** Latitude en longitude zijn optioneel als het adres is opgegeven. De geografische positie zal automatisch berekend worden op basis van het adres.
- Name** Dit veld is verplicht.
- Address** Een adres is verplicht indien geen *latitude* en *longitude* zijn opgegeven.
- CategoryId** Indien geen of een ongeldig *CategoryId* is opgegeven, zal de categorie bepaald worden aan de hand van de opgegeven *CategoryName*.
- CategoryName** Heeft alleen nut als er geen geldig *CategoryId* is opgegeven. De naam wordt vergeleken met de bestaande categorieën. Indien de naam nog niet voor komt wordt deze opgeslagen en doorgestuurd naar Naviscale. Die zal de naam dan koppelen aan een bestaande of eventueel een nieuwe categorie. De locatie zal tot die tijd in de categorie 1000 (*Objecten*) geplaatst worden.

### 3.5.2 Wijzigen

Een locatie wijzigen gaat via de functie `LocationSave`, zoals beschreven op de webservice:

```
Webbservice.asmx?op=LocationSave
```

→ Sectie 3.5.1

Het wijzigen van een locatie gaat, behalve voor het `PointId` op de zelfde manier als het toevoegen..

- PointId** Moet het identificatie nummer zijn van de te wijzigen locatie. Indien het identificatie nummer niet bekend is, kan deze opgezocht worden door de locatie op te halenaan de hand van het referentie id.

→ Sectie 3.5.4

### 3.5.3 Verwijderen

Het verwijderen van een locatie kan op basis van het `PointId`, `ObjectId`, of het `RefId`.

Let op dat verwijderen een definitieve actie is. Als de locatie alleen maar van de kaart verborgen moet worden kan de `Actief` parameter worden gebruikt.

<code>PointId</code>	<code>Webbservice.asmx?op=LocationDeleteByPointId</code>
<code>ObjectId</code>	<code>Webbservice.asmx?op=LocationDeleteByObjectId</code>
<code>RefId</code>	<code>Webbservice.asmx?op=LocationDeleteByRefId</code>

### 3.5.4 Ophalen van een enkele locatie

Het ophalen van een locatie kan op basis van het `PointId`, `ObjectId`, of het `RefId`.

→ Sectie 3.4.2

De opgevraagde locatie komt in de vorm van een `Response`.

<code>PointId</code>	<code>Webservice.asmx?op=LocationGetByPointId</code>
<code>ObjectId</code>	<code>Webservice.asmx?op=LocationGetByObjectId</code>
<code>RefId</code>	<code>Webservice.asmx?op=LocationGetByRefId</code>

### 3.5.5 Ophalen van een meerdere locaties

Voor het ophalen van meerdere locaties zijn de de volgende functies beschikbaar:

Alle Locaties	<code>Webservice.asmx?op=LocationsGet</code>
Alle Objecten	<code>Webservice.asmx?op=LocationsGetObjects</code>
Alle POI's	<code>Webservice.asmx?op=LocationsGetPOIs</code>
Alle Fotos	<code>Webservice.asmx?op=LocationsGetPhotos</code>

→ Sectie 3.4.1

Alle functies geven een array van locaties terug, zoals in het voorbeeld hier onder.

```
<LocationsGetResult>
  <Location>
    ...
  </Location>
  <Location>
    ...
  </Location>
  <Location>
    ...
  </Location>
</LocationsGetResult>
```

Naviscale Locatie Array

### 3.6 Afbeeldingen en Fotos

Het is mogelijk om aan elke locatie, onafhankelijk van de categorie een foto te koppelen.

De fotos worden gekoppeld door een *URL* naar een *JPEG* bestand te sturen, in combinatie met een *PointId*, *ObjectId*, of het *RefId*. De maximale bestandsgrootte per afbeelding is 1.5Mb. De functie geeft een standaard response om aan te geven of de actie geslaagd is of niet, met eventueel de foutmelding.

→ Sectie 3.4.2

! → De Naviscale server zal de afbeelding downloaden vanaf de aangegeven *URL*, let dus op dat deze vrij toegankelijk is.

<i>PointId</i>	<code>Webservice.asmx?op=LocationSetImageByPointId</code>
<i>ObjectId</i>	<code>Webservice.asmx?op=LocationSetImageByObjectId</code>
<i>RefId</i>	<code>Webservice.asmx?op=LocationSetImageByRefId</code>

## A Voorbeelden

### A.1 PHP

#### A.1.1 Auth Header

→ Sectie 3.2

Voor het inloggen is een **Auth Header** nodig. Dat gaat het makkelijkste met een simpele class.

```
class AuthHeader
{
    var $Username;
    var $Password;

    function __construct( $name, $pass )
    {
        $this->Username = $name;
        $this->Password = $pass;
    }
}
```

AuthHeader class

#### A.1.2 Locatie

→ Sectie 3.4.1

Wanneer er gewerkt wordt met de **Locatie** is het handig om ook daarvoor een class aan te maken

```
class Location
{
    var $PointId;
    var $Lat;
    var $Lng;
    var $Name;
    var $Address;
    var $CategoryId;
    var $CategoryName;
    var $Active;
    var $ObjectRefId;
    var $ObjectPrice;
    var $ObjectText;
    var $ObjectURL;

    function __construct( $name, $address, $categoryname )
    {
        $this->Name = $name;
        $this->Address = $address;
        $this->CategoryName = $categoryname;
    }
}
```

Locatie class

### A.1.3 SOAP koppeling

*PHP* heeft ingebouwde *SOAP* functionaliteit. Die gebruiken we voor het opzetten van de koppeling.

```
// SOAP Client starten
$client = new
    SoapClient (
        "https://maps.naviscale.com/Management/WebService.asmx?
            WSDL",
        array( "trace" => 1 ) // Voor debug
    );

// Nieuwe auth header aanmaken met login gegevens
$AuthHeader = new AuthHeader( "Gebruiker", "Wachtwoord" );
// AuthHeader aan de SOAP headers toevoegen
$header = new SoapHeader( "naviscale.com", "AuthHeader",
    $AuthHeader, false );
$client->__setSoapHeaders( array($header) );
```

SOAP koppeling maken

### A.1.4 Locatie Toevoegen

Als de koppeling is opgezet is het eenvoudig de functies aan te roepen. Een nieuwe woning toevoegen kan als volgt worden gedaan:

```
$result = $client->LocationSave( array(
    "Location" => new Location(
        "Test Woning", // Naam
        "Straatnaam 00, 0000AA Plaats, Land", // Adres
        "Woonhuis") // Categorie
    ) );
```

Nieuw object toevoegen

Dit is uiteraard afhankelijk van de *Location* class, en de functionaliteit die daar in geschreven is. Het kan handig zijn om de datatypes te controleren, en bijvoorbeeld meerdere constructors aan te maken. Dit valt echter buiten het kader van deze handleiding.

### A.1.5 Objecten Ophalen

Als tweede voorbeeld de functie om objecten op te vragen vanaf de server.

```
$result = $client->LocationsGetObjects( array() );
```

Alle objecten opvragen

### A.1.6 Return Values

→ Sectie 3.4.2

Alle functies geven een *return value*, die in de vorige voorbeelden wordt opgeslagen in `$return`.

Voor de meeste functies is dit een **response**. De waarde van de **response** is gemakkelijk te zien met de functie `var_dump`.

```
var_dump( $result );
```

Response

## A.2 ASP.Net

### A.2.1 Visual Studio

Het opzetten van een *SOAP* koppeling gaat het gemakkelijkst met *Microsoft Visual Studio*.

Om de koppeling op de zetten moet een nieuwe *WebReference* aan het project worden toegevoegd. De benodigde classes worden automatisch aangemaakt.

### A.2.2 SOAP koppeling

Als de classes zijn aangemaakt is het opzetten van een soap koppeling erg gemakkelijk.

```
ManagementWebService service = new ManagementWebService
{
    Authentication = new ManagementWebService.AuthHeader
    {
        Username = "Gebruiker",
        Password = "Wachtwoord"
    }
};
```

SOAP koppeling maken

### A.2.3 Locatie Toevoegen

Als de koppeling is opgezet is het eenvoudig de functies aan te roepen. Een nieuwe woning toevoegen kan als volgt worden gedaan:

```
ManagementWebService.Response result = service.LocationSave(
    new ManagementWebService.Location
    {
        Name = "Test Woning",
        Address = "Straatnaam 00, 0000AA Plaats, Land",
        CategoryName = "Woonhuis"
    }
);
```

Nieuw object toevoegen

### A.2.4 Objecten Ophalen

Als tweede voorbeeld de functie om objecten op te vragen vanaf de server.

```
ManagementWebService.Location[] result = service.
    LocationsGetObjects();
```

Alle objecten opvragen