Immowelt
API Dokumentation

Der Anbieter verpflichtet sich, mit Einbindung der Daten in den eigenen Internetauftritt an geeigneter Stelle den Vermerk „Immobilien-Daten bereitgestellt von immowelt.de“ hinzuzufügen und diesen mit www.immowelt.de zu verlinken.

Version 3.1.0
Mittwoch, 25. März 2015

Immowelt-API

Inhaltsverzeichnis

Immowelt-API

1. Einleitung

Die Immowelt stellt Ihnen über verschiedene WebServices eine Standard API zur Verfügung, über die Sie die Immobiliensuche und –anzeige in Ihr bestehendes System sehr einfach integrieren können.

Über diese API erhalten Sie Zugriff auf die Geoinformationen sowie das Immobilienangebot der Immowelt. Für die Verwendung innerhalb Ihrer Applikation ist ein bestimmtes Vorgehen (siehe 2.1.3) beim Abruf der Daten empfehlenswert. Als Ergebnis der API-Aufrufe erhalten Sie die entsprechenden Rohdaten, die Sie nach Ihren Wünschen und Anforderungen in Ihrer Anwendung präsentieren können.

Die Immowelt API ist sprachenunabhängig konzipiert, d. h. Sie können mit jeder Programmiersprache diese API nutzen. Für die Kommunikation zwischen Ihrem System und der Immowelt API werden XML-Nachrichten verwendet. Diese Nachrichten werden über http versendet.

2. API – Auf die Immowelt mit WebServices zugreifen

2.1. Grundlagen

2.1.1. Was benötige ich für die Nutzung der API?

Sie benötigen einen API-Schlüssel für die Authentifizierung, den Sie von uns erhalten und eine Programmiersprache, die WebServices mittels SOAP aufrufen kann.

2.1.2. Welche WebServices gibt es?

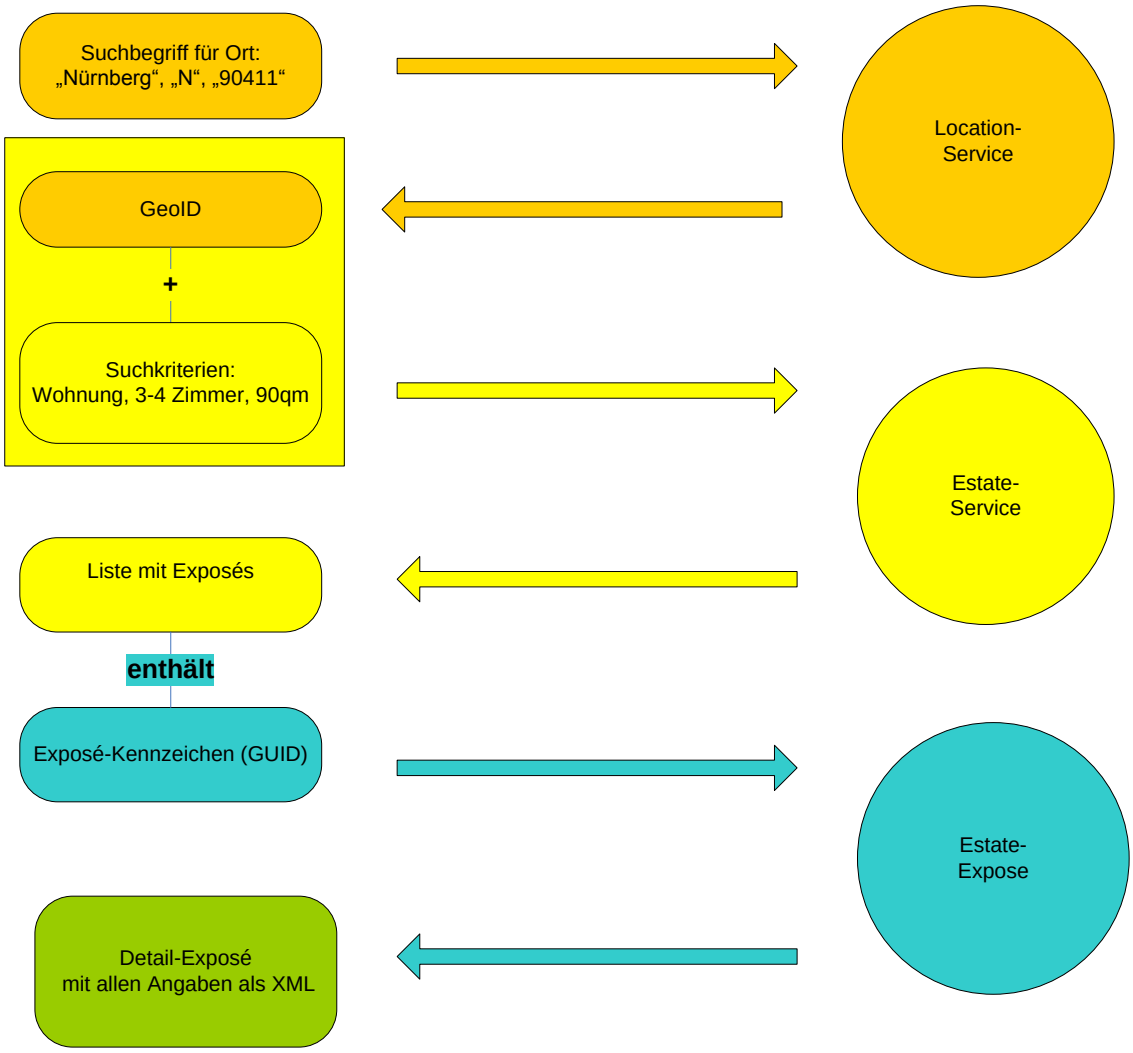
➔ WebService für den Zugriff auf die Geoinformationen der Immowelt AG. Nähere Informationen finden Sie unter Kapitel 3.

➔ WebService für den Zugriff auf die Immobiliensuche mit den Daten des LocationService und weiteren Kriterien, die die Immobiliensuche einschränken. Informationen finden Sie unter Kapitel 4.

➔ Dieser WebService steht Ihnen zur Verfügung, um Detailinformationen zu einem bestimmen Immobilienangebot abzurufen. Dokumentation siehe Kapitel 5.

➔ WebService zum versenden von Kontaktanfragen, Gesuchsanfragen, Empfehlungen, …
Informationen finden Sie unter Kapitel 6.

2.1.3. Wie funktioniert der Datenabruf prinzipiell?

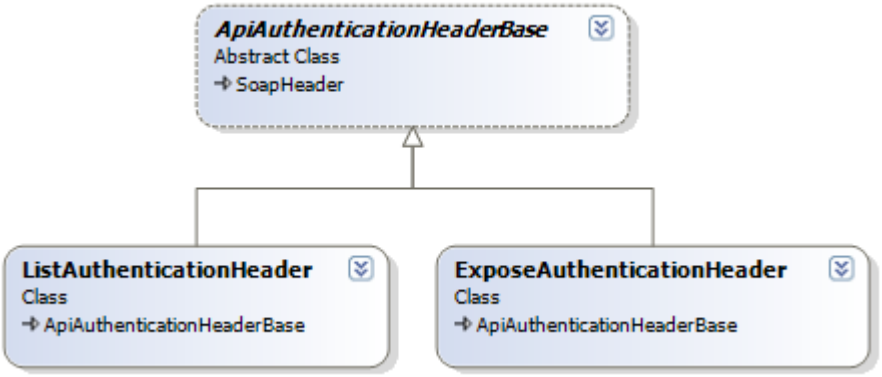


1. Eine *Region* wird durch eine GeoID[1] identifiziert.
   Beispiel: Deutschland hat eine GeoID, ebenso Bayern, Nürnberg und Stadtteile von Nürnberg. Eine Suchanfrage an den LocationService wandelt den Suchbegriff in eine GeoID um.
2. Mit dieser GeoID + Suchkriterien können Sie Listen aus dieser Region abfragen.
   Beispiel: 3-5 Zimmer Wohnung, 90-130qm, 10809
3. Aus der Liste kann ein Exposé für die Detailansicht mit allen Daten ausgesucht werden. Ein Exposé wird durch eine eindeutige Kennziffer (GUID) identifiziert. Mit dieser GUID können die Daten abgerufen werden.

[1] Näheres zur GeoID finden Sie unter Kapitel 7.

2.2. Authentifizierung mittels API-Schlüssel

Für den **Zugriff** auf die **WebServices zum Abfragen** der Daten wird eine Authentifizierung benötigt. Dies geschieht in Form eines **API-Schlüssels**, den Sie von uns erhalten. Momentan sind somit folgende Dienste nur mit einem API-Schlüssel erreichbar.



- LocationService benötigt ListAuthentication
- EstateService benötigt ListAuthentication
- EstateExpose benötigt ExposeAuthentication

2.3. HowTo - Exposédaten ermitteln und anzeigen

1. Sie haben bereits eine GeoID? Machen Sie bei Punkt 4 weiter.
2. Zu einem gesuchten Ort (Kennzeichen, PLZ, Ortsnamen) per *LocationService.GetLocation* eine GeoID ermitteln.
3. Ist *LocationResponse.Status == LocationResponseStatus.Indefinite* gibt es keinen eindeutigen Treffer zu ihrer Stadt. Suchen Sie aus dem Array *LocationResponse.Location* die Richtige aus.
4. Mit der ermittelten GeoID über *EstateService.GetList* oder *EstateService.GetListAmbit* eine Auflistung von Exposés anfordern. Dazu müssen Sie der Funktion noch Ihre gewünschten Suchkriterien zu einem Objekt mitgeben.
5. Gibt *EstateService.GetList* *EstateServiceListResponseStatus.ResultTooLarge* zurück müssen Sie die Suchkriterien verfeinern. Handelt es sich bei dem gesuchten Ort um einen Stadt- oder Landkreis kann mit *EstateService.GetRegionOverview* eine Übersicht der „Unterregionen" zurückgeliefert werden, die sich nur auf die ausgewählte Unterregion beschränken.

6. *In EstateServiceListResponse.EstatePropertyList* erhalten Sie eine Liste, mit der Sie eine Übersicht über Ihren Datenbestand anzeigen können. Jedes dieser Elemente der Liste erhält eine eindeutige ID für das Exposé (GUID)
7. Mit *EstateExpose.GetEstateExposeByEstateGuid(EstateServiceListResponse.EstatePropertyList[n].guid)* können Sie die Details Ihres Exposés abrufen.

2.4. Rückgabewerte der WebMethoden

Hinweis: Jede Methode der gesicherten WebServices liefert entsprechend zur aufgerufenen Methode eine Antwort zurück. In dieser Antwort ist neben den Daten, die Sie zur weiteren Bearbeitung benötigen, ein Status enthalten. Prüfen Sie, ob der Status „OK" ist. Abweichende Werte und wie Sie darauf reagieren sollen wird bei den entsprechenden Funktionen erklärt.

3. LocationService - Ortseingabe in GeoID umwandeln

3.1. URLs

Web-Service
http://api.immowelt.de/WebServices/LocationService.asmx

WSDL:
http://api.immowelt.de/WebServices/LocationService.asmx?WSDL

3.2. Übersicht

Der Location-Service enthält folgende Methoden:

- *GetLocation*
- *GetLocationWithDdistricts*

3.3. Methode GetLocation

Die Methode *GetLocation* liefert zu einem Ort, einer PLZ, einem Kfz-Kennzeichen oder einem Ortsteil die zugehörige GeoID.

```
LocationResponse GetLocation(string AreaDescription)
```

3.3.1. Parameter

`AreaDescription`	kann ein Ort, eine PLZ, ein Kfz-Kennzeichen oder ein Ortsteil sein (z.B. „10715“, „HH“, „Berlin Kreuzberg“).

3.3.2. Rückgabe

`LocationResponse`

`Status [enum]`	• OK (es konnte eine GeoID ermittelt werden) • Unknown (der Eingabe konnte keine GeoID zugeordnet werden) • Indefinite (es wurden mehre Orte gefunden, z.B bei Eingabe von „Frankfurt“)

Location [array]	Array von einem oder mehreren Location-Elementen, die wiederum aus GeoID und Description bestehen. • Wenn der Status „OK" ist, wird genau ein Location-Element mit einer GeoID und Description zurückgegeben. • Falls der Status „Indefinite" ist, werden mehrere Location-Elemente mit GeoID und Description zurückgegeben.

Bsp. für Eingabe "Berlin"

```
<?xml version="1.0" encoding="utf-8" ?>
<LocationResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://immowelt.de/services">
<Status>OK</Status>
<Location>
      <Location>
            <Description>Berlin</Description>
            <GeoID>10811</GeoID>
            </Location>
      </Location>
</LocationResponse>
```

Bsp. für Eingabe "Frankfurt"

```
<?xml version="1.0" encoding="utf-8" ?>
<LocationResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://immowelt.de/services">
  <Status>Indefinite</Status>
  <Location>
        <Location>
              <Description>Frankfurt (Oder)</Description>
              <GeoID>10812053000</GeoID>
              </Location>
        <Location>
              <Description>Frankfurt am Main</Description>
              <GeoID>10806412000</GeoID>
              </Location>
        </Location>
</LocationResponse>
```

3.4. Methode GetLocationWithDdistricts

```
LocationResponse GetLocationWithDistricts(string AreaDescription, bool
childLocations)
```

GetLocationWithDistricts liefert bei *childLocations* = "true" nicht wie *GetLocation* die übergeordnete GeoID z.B. zu einer PLZ sondern alle GeoIDs, die zu dieser PLZ gehören.
Bei *childLocation* = "false" liefert sie das gleiche Ergebnis wie *GetLocation*.

Bsp.: bei der PLZ 90411 wird nicht die 11-Stellige GeoID zu Nürnberg als übergeordnete GeoID zurückgeliefert, sondern die zu 90411 gehörigen 14-stelligen GeoIDs.

`GetLocation(„90411“)`: Ergebnis-GeoID = 10809564000

`GetLocationWithDistricts(„90411“ , true)`:
Ergebnis-GeoIDs = 10809564000008, 10809564000040, 10809564000051, 10809564000061, 10809564000066, 10809564000090, 10809564000123

3.4.1. Parameter

`AreaDescription`	kann ein Ort, eine PLZ, ein Kfz-Kennzeichen oder ein Ortsteil sein (z.B. „10715“, „HH“, „Berlin Kreuzberg“).
`childLocations`	Legt fest, ob eine übergeordnete GeoID zurückgegeben wird, oder falls es eventuell mehrere GeoIDs zur AreaDescription gibt, eine Liste von Location-Items, die die jeweiligen GeoIDs enthalten.

3.4.2. Rückgabe

`LocationResponse`

`Status [enum]`	• OK (es konnte eine GeoID ermittelt werden) • Unknown (der Eingabe konnte keine GeoID zugeordnet werden) • Indefinite (es wurden mehre Orte gefunden, z.B bei Eingabe von „Frankfurt“)
`Location [array]`	• Array von einem oder mehreren Location-Elementen, die wiederum aus GeoID und Description bestehen.

3.4.3. Anmerkungen

Diese Methode eignet sich vor allem bei der Suche mit PLZ. Oft sind in Großstädten einer PLZ mehrere GeoIDs zugeordnet.

Falls bei einer Suche mit PLZ mehrere GeoIDs zurückkommen ist beim Aufruf des EstateService (z.B. *GetList*) wie nachfolgend beschrieben zu verfahren:

```
EstateServiceListResponse GetList(string GeoID, EstateParameter[] parameter,
int CurrentPage, int PageSize);
```

GetList hat als ersten Übergabeparameter den String GeoID. Es liegen allerdings nach der Location-Suche mehrere GeoIDs vor:

Bilden Sie aus den einzelnen GeoIDs einen String, in dem Sie die GeoIDs Komma separiert miteinander verbinden:

z.B.: GeoID = „10809564000008,10809564000040,10809564000051,10809564000061,10809564000066,10809564000090,10809564000123“

Bitte achten Sie darauf, dass keine Leerzeichen im String enthalten sind!

Übergeben Sie diesen String an die Funktion *GetList*.

4. EstateService - Liste laden

4.1. URLs

Web-Service
http://api.immowelt.de/WebServices/EstateService.asmx

WSDL
http://api.immowelt.de/WebServices/EstateService.asmx?WSDL

4.2. Übersicht

Der EstateService enthält folgende Funktionen:

4.3. Methode GetList

Je nach Region wird hier mit einem Umkreis von 5km (Stadt) oder 10km (Land) gesucht. Die Methode *GetList* ist für die Listensuche die Standardmethode. Wer den Umkreis selber bestimmen will hat die Möglichkeit, die Methode *GetListAmbit* zu verwenden, die unter 4.5 näher beschrieben wird. *GetList* liefert dasselbe Ergebnis wie *GetListAmbit* mit dem Parameter *ambit* = -1.

```
EstateServiceListResponse GetList(string GeoID,
                        EstateParameter[] parameter,
                        int CurrentPage,
                        int PageSize);
```

4.3.1. Parameter

`GeoID [string]`	die GeoID des Ortes in dem gesucht werden soll (bei kleinen Orten bzw. Ortsteilen wird automatisch ein Umkreis hinzugefügt, um das Suchergebnis zu vergrößern)
`EstateParameter [dictionary]`	Beliebig viele Parameter zur Spezifizierung der Suche, z.B. Preis, Größe etc. Ein Parameter besteht aus Key und Value (als string), eine Auflistung des erlaubten Keys befindet sich weiter unten in der Dokumentation.
`CurrentPage [int]`	Anzuzeigende Seite der Liste (die Anzahl der Seiten errechnet sich aus Anzahl der Ergebnisse geteilt durch PageSize)
`PageSize [int]`	Anzahl der Angebote auf einer Seite

4.3.1.1. gültige EstateParameter-Keys

EType	Immobilienart
ECat	Immobilienkategorie
RooMi	Mindestanzahl an Zimmern
RooMa	Höchstanzahl an Zimmern
PriMi	Minimaler Kaufpreis / Miete
PriMa	Maximaler Kaufpreis / Miete
WflMi	Minimale Fläche abhängig von Immobilienart: • Wohnfläche • Bürofläche • Ladenfläche • …

WflMa	Maximale Fläche abhängig von Immobilienart: • Wohnfläche • Bürofläche • Ladenfläche • ...
GflMi	Minimale Grundstücksfläche
GflMa	Maximale Grundstücksfläche
QPriMi	Minimaler Quadratmeter-Preis
QPriMa	Maximale Quadratmeter-Preis
FroMi	Minimale Fensterfront (Ladenflächen)
FroMa	Maximale Fensterfront (Ladenflächen)
XfaMi	Minimale x-fache Miete (Rendieteobjekte)
XfaMa	Maximale x-fache Miete (Rendieteobjekte)
ESR	1: Kauf-Objekte 2: Miet-Objekte
EPos	Lage des Objekts
EqID	Ausstattungsmerkmale

4.3.1.2. Immobilienarten

1	Wohnungen
2	Häuser
3	Grundstücke
4	Büro-/Praxisflächen
5	Ladenflächen
6	Hallen/Industrieflächen
7	Gewerbe-Grundstücke
8	Renditeobjekte
9	Sonstiges
10	Gastronomie/Hotels
11	Land-/Forstwirtschaft
12	Ferienimmobilien
13	Garage/Stellplatz
15	Wohnen auf Zeit
16	Wohngemeinschaften
17	Typenhäuser
19	Dachflächen

4.3.1.3. Kategorien

1	Einfamilienhaus
2	Doppelhaushälfte
3	Reihenendhaus
4	Reihenmittelhaus
5	Zweifamilienhaus
6	Doppelhaus
7	Mehrfamilienhaus (nur in Verbindung mit der Immobilienart 2: Häuser)
8	Bungalow
10	Burg / Schloss
11	Außen-Stellplatz
12	Tiefgarage
13	Garage
14	Carport
15	Doppelparker
16	Gemeinschaftsgarage
17	Gewerbeflächen
18	Zimmer
19	Wohnung
20	Appartement
21	Haus
22	Loft
23	Maisonette
24	Bauernhaus
25	Penthouse
26	Terrassenwohnung
27	Etagenwohnung
28	Stadthaus
29	Bürofläche
30	Praxis
31	Bürohaus
32	Ausstellungsfläche
33	Mehrfamilienhaus (nur in Verbindung mit der Immobilienart 8: Renditeobjekte)
34	Wohn-Geschäftshaus
35	Geschäftshaus
36	Bürogebäude
37	SB-Markt
38	Einkaufszentrum
39	Wohnanlage

40	Verbrauchermarkt
41	Industrieanlage
42	Gastronomie
43	Hotel
44	Pension
45	Freizeitobjekt
46	Bauernhof
47	Reiterhof
48	Forstwirtschaft
49	Landwirtschaftliche Fläche
50	Weingut
51	Doppelhaushälfte
52	Einfamilienhaus
53	Reihenhaus

4.3.1.4. Lage des Objektes

1	1A - Lage
2	1B - Lage
3	Randlage
4	Nähe Zentrum

4.3.1.5. Ausstattungsmerkmale

2	Balkon
3	Dachgeschoss
4	Dachterrasse
5	Einliegerwohnung
9	Souterrain
10	Laderampe
11	Kabelkanäle
12	voll klimatisiert
13	Deckenbeleuchtung
14	Alarmanlage
15	Altbau (bis 1945)
16	rollstuhlgerecht
17	Bad/WC getrennt
18	EDV-Arbeitsplatzbeleuchtung
19	Dachboden
20	Kabelanschluß
21	Doppelboden
22	EDV-Verkabelung

23	frei
24	Einbauküche
25	Kantine
26	Erdgeschoss
27	Raumaufteilung flexibel
28	Garage
29	Garten
30	getr. Damen- u. Herren WC´s
31	Kapitalanlage
32	voll unterkellert
33	Lastenaufzug
34	Neubau
35	Erstbezug
36	unerschlossen
37	teilweise erschlossen
38	voll erschlossen
39	Ofenheizung
40	Personenaufzug
41	renoviert
43	Rolltor
44	Sonnenschutz
45	Sprinkleranlage
46	Stellplatz
47	Terrasse
48	Tiefgarage
49	Wintergarten
50	Zentralheizung
52	Bad mit Dusche
53	Bad mit Wanne
54	Bad mit Fenster
56	offene Küche
57	Speisekammer
59	Fliesenboden
60	Steinboden
61	Teppichboden
62	Parkettboden
63	Dielenboden
64	Kunststoffboden
65	Estrich
66	Laminat
69	Etagenheizung
71	Fernheizung
72	Fußbodenheizung
73	offener Kamin
74	Kachelofen
76	Öl
77	Gas

78	Elektro
79	Solar
80	Erdwärme
81	Holz
82	Kohle
83	Regenwassernutzung
85	teilweise klimatisiert
86	kontrollierte Be- und Entlüftungsanlage
88	Carport
89	Parkhaus
90	Duplex
91	Gartenanteil
93	teilweise möbliert
94	voll möbliert
97	Sauna
98	Wellnessbereich
99	Swimming Pool
100	Solarium
103	Sat
104	Antenne
106	Sporteinrichtungen
107	Brauereibindung
108	Gastterrasse
109	Bar
110	Hotelrestaurant
112	Loggia
114	Catering
115	Reinigung
116	Einkauf
117	Wachdienst
118	Hausmeister
121	Kamera
122	Polizeiruf
123	Empfang
124	Pförtner
125	Kartenzugangskontrolle
128	teilweise unterkellert
129	nicht unterkellert
130	als Wohnraum nutzbar
132	renovierungsbedürftig
133	neuwertig
134	saniert
135	baufällig
136	Rohbau
137	entkernt
138	Abrißobjekt
139	projektiert

141	Nachbarschaft (§34)
142	Außengebiet (§35)
143	Bebauungsplan
144	kein Bauland
145	Bauerwartungsland
151	Fernblick
152	Seeblick
153	Berge
155	WG geeignet
156	Wasch-Trockenraum
157	Abstellraum
158	Kelleranteil
159	Hobbyraum
160	Dach ausgebaut
161	Dach ausbaufähig
163	Hebebühne
164	Kran
165	Teeküche
166	LKW-Zufahrt
167	antistatischer Teppichboden
171	vermietet
172	frei werdend
174	Holzfenster
175	Kunststofffenster
176	Sprossenfenster
177	Aluminiumfenster
179	Holzhaus
180	Massivhaus
181	Fertighaus
182	Passivhaus
183	Niedrigenergie
185	ISDN
186	Analog
187	DSL
189	Denkmalschutz-Afa
190	Sanierungs-Afa
191	betreutes Wohnen
192	Altenpflege
193	Ausbauhaus
194	Bausatzhaus
195	schlüsselfertig ohne Bodenplatte
196	schlüsselfertig mit Bodenplatte
197	schlüsselfertig mit Keller
198	Massiv
199	Holz
200	Krüppelwalmdach
201	Mansarddach

202	Pultdach
203	Satteldach
204	Walmdach
205	Wohnberechtigungsschein
206	Baugenehmigung
207	Wohnen
208	Freizeit
209	Pacht
210	Erbpacht
211	Gäste WC
212	Holzdielen
213	Gartenmitbenutzung
214	Gemeinschaftspool
215	Städtische Wasserversorgung
216	Eigene Wasserversorgung
217	Städtische Stromversorgung
218	Eigene Stromversorgung
219	DVBT
220	gepflegt
221	Blockheizkraftwerk
222	seniorengerechtes Wohnen
223	Flachdach
224	unter 20°
225	20 - 30°
226	30 - 40°
227	40 - 50°
228	über 50°
229	Fassade
230	Süden
231	Südwest/ Südost
232	Westen/ Osten
233	Dachziegel
234	Wellblechdach
235	andere
236	schattenfrei
237	Zwangsversteigerung
238	Haustiere erlaubt

4.3.2. Rückgabewert

`EstateServiceListResponse`

`Status [enum]`	• OK (gültiges Suchergebnis) • ResultTooLarge (es wurden mehr als 4000 Ergebnisse gefunden). Es werden aber weiterhin die Objekte die per

	PageSize und CurrentPage angefordert wurden zurück gegeben. • SearchCriteriaMissing (es wurde entweder keine GeoID oder keine Parameter übergeben)
`EstatePropertyList [array]`	Array von EstateProperty Elementen, in denen alle Daten eines Listeneintrages enthalten sind: • Category (z.B. Wohnung zur Miete) • City (Stadt des Objekts) • Description (Beschreibungstext/Überschrift) • LandArea (Grundstücksfläche) • LivingArea (abhängig von der Immobilienart: Wohnfläche, Bürofläche, Ladenfläche, ...) • PreviewImage (Url des Thumbnails) • Price (Preis/Miete; enthält auch die Währung) • UrlContact (vollständige Url zum Kontaktformular) • UrlExpose (vollständige Url zum Expose) • UrlImages (vollständige Url zu den Bildern) • Zip (Postleitzahl des Objekts) • Equipment: wenn Equipment-Ids vorhanden sind werden diese hier mit übergeben, wenn nicht steht dieses Feld auf null • GeoID • GeoDescription • Laengengrad (ist nur gesetzt, wenn das Objekt in einer Karte angezeigt werden darf) • Breitengrad (ist nur gesetzt, wenn das Objekt in einer Karte angezeigt werden darf) • ExactMapPin (mögliche Werte: True, wenn eine exakte Angabe des Längen- und Breitengrades des Objektes vorliegt; False, wenn die Angaben zu Längen- und Breitengrad nicht exakt oder nicht vorhanden sind.)
`TotalCount [int]`	Insgesamt gefundene Ergebnisse
`CurrentPage [int]`	Anzuzeigende Seite der Liste (die Anzahl der Seiten errechnet sich aus TotalCount geteilt durch PageSize)
`PageSize [int]`	Anzahl der Angebote in der EstatePropertyList

4.3.3. Exceptions

`UnauthorizedAccessException`	Fehlerhafte Authentifizierung oder keine Zugriffsrechte.

4.4. Methode GetListWithSort

Überladung von *GetList*. Diese Methode hat einen zusätzlichen Parameter *ls,* mit dem die Sortierung gesteuert werden kann. Rückgabewert und Exceptions siehe *GetList*.

```
EstateServiceListResponse GetListWithSort(string GeoID,
                                          EstateParameter[] parameter,
                                          int CurrentPage,
                                          int PageSize,
                                          ListSort ls)
```

4.4.1. Parameter

`GeoID [string]`	die GeoID des Ortes, in dem gesucht werden soll
`EstateParameter [array]`	Beliebig viele Parameter zur Spezifizierung der Suche, z.B. Preis, Größe etc. Ein Parameter besteht aus Key und Value (als string), eine Auflistung des erlaubten Keys finden Sie unter 4.3.1.x.
`CurrentPage [int]`	Anzuzeigende Seite der Liste (die Anzahl der Seiten errechnet sich aus Anzahl der Ergebnisse geteilt durch PageSize)
`PageSize [int]`	Anzahl der Angebote auf einer Seite
`ls [ListSort]`	Listensortierung mittels des enums ListSort

price	Sortierung nach Preis
pricedesc	Sortierung nach Preis absteigend
wohnflaeche	Sortierung nach Flaeche
wohnflaechedesc	Sortierung nach Flaeche absteigend
city	Sortierung nach der Ort
citydesc	Sortierung nach der Ort absteigend
distance	Sortierung nach der Entfernung
createdate	Sortierung nach dem Einstelldatum
createdatedesc	Sortierung nach dem Einstelldatum absteigend

4.5. Methode GetListAmbit

Überladung von *GetList*. Hat einen zusätzlichen Parameter *ambit,* mit dem die Umgebungssuche gesteuert werden kann. Rückgabewert und Exceptions siehe *GetList*.

```
EstateServiceListResponse GetListAmbit(string GeoID,
                                       EstateParameter[] parameter,
                                       int CurrentPage,
                                       int PageSize,
                                       int ambit);
```

4.5.1. Parameter

`GeoID [string]`	die GeoID des Ortes in dem gesucht werden soll.
`EstateParameter [array]`	Beliebig viele Parameter zur Spezifizierung der Suche, z.B. Preis, Größe etc. Ein Parameter besteht aus Key und Value (als string), eine Auflistung des erlaubten Keys finden Sie unter 4.3.1.x.
`CurrentPage [int]`	Anzuzeigende Seite der Liste (die Anzahl der Seiten errechnet sich aus Anzahl der Ergebnisse geteilt durch PageSize)
`PageSize [int]`	Anzahl der Angebote auf einer Seite
`ambit [int]`	Umkreissuche 0 – deaktiviert -1 – wird die gleiche Funktionalität wie GetList aufgerufen (je nach Region wird hier mit einem Umkreis von 5km (Stadt) oder 10km(Land) gesucht. >= 1 – Umkreis in km.

4.6. Methode GetListAmbitWithSort

Überladung von *GetList.* Hat zwei zusätzliche Parameter *ambit* und *ls* mit dem die Umgebungssuche gesteuert werden kann. Rückgabewert und Exceptions siehe GetList.

```
EstateServiceListResponse GetListAmbitWithSort(string GeoID,
                                               EstateParameter[] parameter,
                                               int CurrentPage,
                                               int PageSize,
                                               int ambit,
                                               ListSort ls)
```

4.6.1. Parameter

<table>
<tr><td>GeoID [string]</td><td>die GeoID des Ortes in dem gesucht werden.</td></tr>
<tr><td>EstateParameter [array]</td><td>Beliebig viele Parameter zur Spezifizierung der Suche, z.B. Preis, Größe etc.
Ein Parameter besteht aus Key und Value (als string), eine Auflistung des erlaubten Keys finden Sie unter 4.3.1.x.</td></tr>
<tr><td>CurrentPage [int]</td><td>Anzuzeigende Seite der Liste (die Anzahl der Seiten errechnet sich aus Anzahl der Ergebnisse geteilt durch PageSize)</td></tr>
<tr><td>PageSize [int]</td><td>Anzahl der Angebote auf einer Seite</td></tr>
<tr><td>Ambit [int]</td><td>Umkreissuche
<table>
<tr><td>0</td><td>deaktiviert</td></tr>
<tr><td>-1</td><td>wird die gleiche Funktionalität wie GetList aufgerufen (je nach Region wird hier mit einem Umkreis von 5km (Stadt) oder 10km(Land) gesucht.</td></tr>
<tr><td>>= 1</td><td>Umkreis in km.</td></tr>
</table></td></tr>
<tr><td>ls [ListSort]: enum</td><td>Listensortierung mittels des enums ListSort
<table>
<tr><td>price</td><td>Sortierung nach Preis</td></tr>
<tr><td>pricedesc</td><td>Sortierung nach Preis absteigend</td></tr>
<tr><td>wohnflaeche</td><td>Sortierung nach Wohnflaeche</td></tr>
<tr><td>wohnflaechedesc</td><td>Sortierung nach Wohnflaeche absteigend</td></tr>
<tr><td>city</td><td>Sortierung nach der Stadt</td></tr>
<tr><td>citydesc</td><td>Sortierung nach der Stadt absteigend</td></tr>
<tr><td>distance</td><td>Sortierung nach der Entfernung</td></tr>
<tr><td>createdate</td><td>Sortierung nach dem Einstelldatum</td></tr>
<tr><td>createdatedesc</td><td>Sortierung nach dem Einstelldatum absteigend</td></tr>
</table></td></tr>
</table>

4.7. Methode GetListByLatLong

Diese Methode entspricht fast der Methode GetListAmbitWithSort. Allerdings wird hier nicht die GeoID übergeben, sondern der Längen- und Breitengrad.
Mit dieser Methode kann man also zu einem bestimmten Standpunkt eine Umkreissuche ausführen.

```
EstateServiceListResponse GetListByLatLong(double Lat,
                                           double Long,
                                           EstateParameterList parameter,
                                           int CurrentPage,
                                           int PageSize,
                                           int ambit,
                                           ListSort ls)
```

4.7.1. Parameter

<table>
<tr><td>Lat [double]</td><td>Breitengrad</td></tr>
<tr><td>Long [double]</td><td>Längengrad</td></tr>
<tr><td>EstateParameter [array]</td><td>Beliebig viele Parameter zur Spezifizierung der Suche, z.B. Preis, Größe etc.
Ein Parameter besteht aus Key und Value (als string), eine Auflistung des erlaubten Keys finden Sie unter 4.3.1.x.</td></tr>
<tr><td>CurrentPage [int]</td><td>Anzuzeigende Seite der Liste (die Anzahl der Seiten errechnet sich aus Anzahl der Ergebnisse geteilt durch PageSize)</td></tr>
<tr><td>PageSize [int]</td><td>Anzahl der Angebote auf einer Seite</td></tr>
<tr><td>Ambit [int]</td><td>Umkreissuche
<table>
<tr><td>0</td><td>deaktiviert</td></tr>
<tr><td>-1</td><td>wird die gleiche Funktionalität wie GetList aufgerufen (je nach Region wird hier mit einem Umkreis von 5km (Stadt) oder 10km(Land) gesucht.</td></tr>
<tr><td>>= 1</td><td>Umkreis in km.</td></tr>
</table></td></tr>
<tr><td>ls [ListSort]: enum</td><td>Listensortierung mittels des enums ListSort
<table>
<tr><td>price</td><td>Sortierung nach Preis</td></tr>
<tr><td>pricedesc</td><td>Sortierung nach Preis absteigend</td></tr>
<tr><td>wohnflaeche</td><td>Sortierung nach Wohnflaeche</td></tr>
<tr><td>wohnflaechedesc</td><td>Sortierung nach Wohnflaeche absteigend</td></tr>
<tr><td>city</td><td>Sortierung nach der Stadt</td></tr>
<tr><td>citydesc</td><td>Sortierung nach der Stadt absteigend</td></tr>
<tr><td>distance</td><td>Sortierung nach der Entfernung</td></tr>
<tr><td>createdate</td><td>Sortierung nach dem Einstelldatum</td></tr>
<tr><td>createdatedesc</td><td>Sortierung nach dem Einstelldatum absteigend</td></tr>
</table></td></tr>
</table>

4.8. Methode GetListByEstateGuids

Diese Methode liefert in einer Liste die Objekte zu den übergebenen EstateGuids.

```
EstateServiceListResponse GetListByEstateGuids(string EstateGuids)
```

4.8.1. Parameter

`EstateGuids [string]`	Eine **kommaseparierte** Liste aus einzelnen EstateGuids.

4.8.2. Rückgabewert

`EstateServiceListResponse`

`Status [enum]`	• OK (gültiges Suchergebnis) • ResultTooLarge (es wurden mehr als 4000 Ergebnisse gefunden). Es werden aber weiterhin die Objekte die per PageSize und CurrentPage angefordert wurden zurück gegeben. • SearchCriteriaMissing (es wurde entweder keine GeoID oder keine Parameter übergeben)
`EstatePropertyList [array]`	Array von EstateProperty Elementen, in denen alle Daten eines Listeneintrages enthalten sind: • Category (z.B. Wohnung zur Miete) • City (Stadt des Objekts) • Description (Beschreibungstext/Überschrift) • LandArea (Grundstücksfläche) • LivingArea (abhängig von der Immobilienart: Wohnfläche, Bürofläche, Ladenfläche, ...) • PreviewImage (Url des Thumbnails) • Price (Preis/Miete; enthält auch die Währung) • UrlContact (vollständige Url zum Kontaktformular) • UrlExpose (vollständige Url zum Expose) • UrlImages (vollständige Url zu den Bildern) • Zip (Postleitzahl des Objekts) • Equipment: wenn Equipment-Ids vorhanden sind werden diese hier mit übergeben, wenn nicht steht dieses Feld auf null • GeoID • GeoDescription • Laengengrad (ist nur gesetzt, wenn das Objekt in einer Karte angezeigt werden darf) • Breitengrad (ist nur gesetzt, wenn das Objekt in einer

	Karte angezeigt werden darf) • ExactMapPin (mögliche Werte: True, wenn eine exakte Angabe des Längen- und Breitengrades des Objektes vorliegt; False, wenn die Angaben zu Längen- und Breitengrad nicht exakt oder nicht vorhanden sind.)
`TotalCount [int]`	Insgesamt gefundene Ergebnisse
`CurrentPage [int]`	Anzuzeigende Seite der Liste (die Anzahl der Seiten errechnet sich aus TotalCount geteilt durch PageSize)
`PageSize [int]`	Anzahl der Angebote in der EstatePropertyList

4.9. Methode GetRegionOverview

Wenn beim Aufruf der Liste mit *GetList* der *Status* „ResultTooLarge" zurückgegeben worden ist, kann man mit dieser Methode einen Überblick der Resultate in den Unter-Regionen erhalten. Liefert *GetRegionOverview* den *Status* OK zurück, aber keine Ergebnismenge, gibt es für die Region, die durchsucht wird, keine weiteren Ortschaften/Stadtteile.

```
GetRegionOverview(string GeoId, EstateParameterList parameter)
```

4.9.1. Parameter

Die Übergabe-Parameter entsprechen denen der Methode *GetList.*

4.9.2. Rückgabewert

Zurückgegeben wird ein Array von *RegioOverviewItem,* in denen jeweils die Elemente *GeoID, Description* und *Count* enthalten sind.

Weiterhin wird ein *Status* „OK" oder „GeoIDImprecise" zurückgegeben. Letzteren erhält man, wenn die übergebene GeoID keine Stadt darstellt, so ist es z.B. nicht möglich, einen Überblick der Immobilien in einem Bundesland zu erhalten.

Wurde mit *GetRegionOverview* bereits eine Ergebnismenge abgerufen, kann mit der GeoID des *RegioOverviewItem* kein weiterer Aufruf mehr erfolgen, auch wenn „ResultTooLarge" zurückgeben wird. In diesem Fall besteht nur die Möglichkeit die Suche über die *EstateParameter* weiter einzuschränken und/oder die Funktion *GetListAmbit* zu benutzen um mit dem Parameter ambit den Umkreis weiter einzuschränken.

4.9.3. Exceptions

`UnauthorizedAccessException`	Fehlerhafte Authentifizierung, oder keine Zugriffsrechte.
`ArgumentException`	Es wurde ein Parameter übergeben. Bitte die Nachricht der Exception dazu beachten.

4.10. Methode GetEstateParameterList

Mittels dieser Methode kann man sich die entsprechenden EstateParameter zu Stichworten liefern lassen. „Haus kaufen bis 500000€" liefert z.B. die folgenden *EstateParameter*:

- EType=2
- ESR=1
- PriMa=500000

```
GetEstateParameterList(string description)
```

4.10.1. Parameter

Der Übergabe-Parameter *description* ist ein String Parameter und enthält die einzelnen Stichworte, z.B.:
„Haus kaufen bis 500000€" oder
„Grundstück mieten bis 1000€ Fläche 2000qm"

4.10.2. Rückgabewert

Zurückgegeben wird ein Array von *EstateParameter*.
Dieses Array kann man dann z.B. wieder als Übergabeparameter für die Funktion *GetList* verwenden.
GetList(string GeoID, **EstateParameter[] parameter**, int CurrentPage, int PageSize)

```
...
string searchFor = "90411";

string description = "Haus kaufen bis 500000€";

EstateParameter[] result = estateService.GetEstateParameterList(description);

EstateServiceListResponse result2 = GetListSearch(searchFor, result);
...
```

4.11. Methode GetListCustomerProjects

Mittels dieser Methode kann man sich auch die Objekte zurückgeben lassen, die in der DB als „Projekt extern" markiert sind. Mittels der Markierung „Projekt extern" können Objekte gepflegt werden, die nicht in der Immowelt angezeigt werden, sondern z.B. nur in von Partnern angelegten Immowelt-Includes.

```
EstateServiceListResponse GetListCustomerProjects(string GeoID,
                                                  EstateParameterList parameter,
                                                  ListSort ls,
                                                  bool IncludeIwObj,
                                                  int CurrentPage,
                                                  int PageSize,
                                                  string Intranet)
```

4.11.1. Parameter

<table>
<tr><td>GeoID [string]</td><td>die GeoID des Ortes in dem gesucht werden soll</td></tr>
<tr><td>parameter [array]</td><td>Beliebig viele Parameter zur Spezifizierung der Suche, z.B. Preis, Größe etc.
Ein Parameter besteht aus Key und Value (als string), eine Auflistung des erlaubten Keys finden Sie unter 4.3.1.</td></tr>
<tr><td>ls [ListSort]: enum</td><td>Listensortierung mittels des enums ListSort

<table><tr><td>price</td><td>Sortierung nach Preis</td></tr><tr><td>pricedesc</td><td>Sortierung nach Preis absteigend</td></tr><tr><td>wohnflaeche</td><td>Sortierung nach Wohnflaeche</td></tr><tr><td>wohnflaechedesc</td><td>Sortierung nach Wohnflaeche absteigend</td></tr><tr><td>city</td><td>Sortierung nach der Stadt</td></tr><tr><td>citydesc</td><td>Sortierung nach der Stadt absteigend</td></tr><tr><td>distance</td><td>Sortierung nach der Entfernung</td></tr><tr><td>createdate</td><td>Sortierung nach dem Einstelldatum</td></tr><tr><td>createdatedesc</td><td>Sortierung nach dem Einstelldatum absteigend</td></tr></table></td></tr>
<tr><td>IncludeIwObj [bool]</td><td>Mit diesem Parameter legt man fest, ob nur Objekte, die in externen Projekten (Customer-Projekten) angezeigt werden sollen, geladen werden oder ob auch die Objekte die in der Immowelt angezeigt werden mit ausgegeben werden.
Falls dieser Parameter auf „true" gesetzt ist, ist ein gültiger API-Key mit zugehöriger Projekt- oder Adress-ID nötig. Wenn dies nicht der Fall ist wird eine UnauthorizedAccessException geworfen.</td></tr>
<tr><td>CurrentPage [int]</td><td>Anzuzeigende Seite der Liste (die Anzahl der Seiten errechnet sich aus Anzahl der Ergebnisse geteilt durch PageSize)</td></tr>
<tr><td>PageSize [int]</td><td>Anzahl der Angebote auf einer Seite</td></tr>
<tr><td>Intranet[string]</td><td>Optionales Feld zum Abfragen des Intranet-Feldes in der DB.</td></tr>
</table>

	Wenn man diesen Parameter nicht nutzen will, übergibt man einfach NULL oder einen Leer-String („“).

4.11.2. Rückgabewert

`EstateServiceListResponse`

`Status [enum]`	• OK (gültiges Suchergebnis) • ResultTooLarge (es wurden mehr als 4000 Ergebnisse gefunden, es wird keine Liste angezeigt) • SearchCriteriaMissing (es wurde entweder keine GeoID oder keine Parameter übergeben)
`EstatePropertyList [array]`	Array von EstateProperty Elementen, in denen alle Daten eines Listeneintrages enthalten sind: • Category (z.B. Wohnung zur Miete) • City (Stadt des Objekts) • Description (Beschreibungstext/Überschrift) • LandArea (Grundstücksfläche) • LivingArea (abhängig von der Immobilienart: Wohnfläche, Bürofläche, Ladenfläche, ...) • PreviewImage (Url des Thumbnails) • Price (Preis/Miete; enthält auch die Währung) • UrlContact (vollständige Url zum Kontaktformular) • UrlExpose (vollständige Url zum Expose) • UrlImages (vollständige Url zu den Bildern) • Zip (Postleitzahl des Objekts) • Equipment: wenn Equipment-Ids vorhanden sind werden diese hier mit übergeben, wenn nicht steht dieses Feld auf null • GeoID • GeoDescription • Laengengrad (ist nur gesetzt, wenn das Objekt in einer Karte angezeigt werden darf) • Breitengrad (ist nur gesetzt, wenn das Objekt in einer Karte angezeigt werden darf) • ExactMapPin (mögliche Werte: True, wenn eine exakte Angabe des Längen- und Breitengrades des Objektes vorliegt; False, wenn die Angaben zu Längen- und Breitengrad nicht exakt oder nicht vorhanden sind.)
`TotalCount [int]`	Insgesamt gefundene Ergebnisse
`CurrentPage [int]`	Anzuzeigende Seite der Liste (die Anzahl der Seiten errechnet sich aus TotalCount geteilt durch PageSize)
`PageSize [int]`	Anzahl der Angebote in der EstatePropertyList

4.12. Methode GetOffererList

Diese Methode liefert zu einer *ProjektID* alle Anbieter, die unter dieser *ProjektID* zusammengefasst sind.

```
EstateServiceProjectOffererListResponse GetOffererList(string ProjektID,
                                                       bool ProjektExtern)
```

4.12.1. Parameter

`ProjektID [string]`	Die ProjektID, unter der wieder mehrere Anbieter zusammengefasst sein können. Bei API-Anbindungen die mittels eines API-Keys auf die API zugreifen ist dieser Parameter nicht nötig (es kann „null" oder ein Leerstring übergeben werden), da anhand des API-Keys eine feste Bindung an eine ProjektID oder AnbieterID gegeben ist. Die Ergebnismenge bezieht sich nur auf Objekte, die unter dieser ProjektID gespeichert sind.
`ProjektExtern [bool]`	Wenn dieser Parameter auf true gesetzt ist liefert diese Methode auch diejenigen Objekte, die nur in externen Projekten angezeigt werden sollen. Falls dieser Parameter auf „true" gesetzt ist, ist ein gültiger API-Key mit zugehöriger Projekt- oder Adress-ID nötig. Wenn dies nicht der Fall ist wird eine *`UnauthorizedAccessException`* geworfen.

4.12.2. Rückgabewert

`EstateServiceProjectOffererListResponse`

`Status [enum]`	• OK: gültiges Suchergebnis • FALSE: ungültiges Suchergebnis • NoOfferer: zu dieser ProjektID existiert kein Anbieter
`OffererList [List<Offerer>]`	Liste mit Offerer: `Offerer:` • **`AdrID`**: die Adress-ID des jeweiligen Anbieters. • **`OffererName`**: der Name des Anbieters. • **`ObjectCount`**: die Anzahl an Objekten, die der Anbieter besitzt.

	• **`MieteCount`**: Anzahl an Mietobjekten • **`KaufCount`**: Anzahl an Kaufobjekten

4.13. Methode GetCountEstateType

GetCountEstateType kumuliert alle Objekte die unter einer *ProjektID* oder einer *Adress-ID* erfasst sind nach den EstateTypes (Immobilienarten).

```
EstateServiceCountEstateTypeResponse GetCountEstateType(string StartGeoID,
                                                        bool CustomerProject,
                                                        string ProjektID,
                                                        string AdrID)
```

4.13.1. Parameter

`StartGeoID [string] -optional`	Dieser Parameter ist optional. Man kann hier eine übergeordnete GeoID angeben, so dass die Ergebnismenge nur Objekte beinhaltet, die sich innerhalb dieser GeoID-Region befinden. Falls dieser Parameter nicht verwendet werden will, muss „null" oder ein Leerstring übergeben werden.
`CustomerProject [bool]`	Wenn dieser Parameter auf true gesetzt ist liefert diese Methode auch diejenigen Objekte, die nur in externen Projekten angezeigt werden sollen. Falls dieser Parameter auf „true" gesetzt ist, ist ein gültiger API-Key mit zugehöriger Projekt- oder Adress-ID nötig. Wenn dies nicht der Fall ist wird eine *`UnauthorizedAccessException`* geworfen.
`ProjektID [string] -optional`	Die ProjektID, unter der wieder mehrere Anbieter zusammengefasst sein können. Bei API-Anbindungen die mittels eines API-Keys auf die API zugreifen ist dieser Parameter nicht nötig (es kann „null" oder ein Leerstring übergeben werden), da anhand des API-Keys eine feste Bindung an eine ProjektID oder AnbieterID gegeben ist. Die Ergebnismenge bezieht sich nur auf Objekte, die unter dieser ProjektID gespeichert sind.
`AdrID [string] -optional`	Die Adress-ID, unter der ein Anbieter in der Immowelt erfasst ist. Bei API-Anbindungen die mittels eines API-Keys auf die API zugreifen ist dieser Parameter nicht nötig (es kann „null" oder ein Leerstring übergeben werden), da anhand des API-Keys eine feste Bindung an eine ProjektID oder AnbieterID gegeben ist. Die Ergebnismenge bezieht sich nur auf Objekte, die unter dieser AdrID gespeichert sind.

4.13.2. Rückgabewert

`EstateServiceCountEstateTypeResponse`

`Status [enum]`	• OK: gültiges Suchergebnis • FALSE: ungültiges Suchergebnis
`EstateTypeCountItemList [List<EstateTypeCountItem>]`	Liste mit EstateTypeCountItems: `EstateTypeCountItem:` • **`EstateType:`** die Immobilienart, nach der die Ergebnisse kumuliert werden (als ID). • **`EstateTypeDescription:`** die Immobilienart in ausgeschriebener Form. • **`ObjectCount:`** die Anzahl an Objekten die in die Immobilienart-Kategorie fallen. • **`MieteCount:`** Anzahl an Mietobjekten • **`KaufCount:`** Anzahl an Kaufobjekten

4.14. Methode GetCountGeoID

GetCountGeoID kumuliert alle Objekte die unter einer Adress-ID oder einer Projekt-ID erfasst sind nach den Regionen.

```
EstateServiceCountGeoIDResponse GetCountGeoID(string StartGeoID,
                                              bool CustomerProject,
                                              CountRegion region,
                                              EstateParameterList parameter,
                                              string ProjektID,
                                              string AdrID)
```

4.14.1. Parameter

`StartGeoID [string] -optional`	Dieser Parameter ist optional. Man kann hier eine übergeordnete GeoID angeben, so dass die Ergebnismenge nur Objekte beinhaltet, die sich innerhalb dieser GeoID-Region befinden. Falls dieser Parameter nicht verwendet werden will, muss „null" oder ein Leerstring übergeben werden.
`CustomerProject [bool]`	Wenn dieser Parameter auf true gesetzt ist, liefert diese Methode auch diejenigen Objekte, die nur in externen Projekten angezeigt werden sollen. Falls dieser Parameter auf „true" gesetzt ist, ist ein gültiger API-Key mit zugehöriger Projekt- oder Adress-ID nötig. Wenn dies nicht der Fall ist wird eine *`UnauthorizedAccessException`* geworfen.
`region [CountRegion] enum`	Mittels dieses Parameters kann man die Granulierung der Kumulation der Ergebnis-Counts angeben: • **`Bundeslaender`**: Die Ergebnisse werden auf Bundeslandebene kumuliert. • **`Regierungsbezirk`**: Die Ergebnisse werden auf Regierungsbezirk-Ebene kumuliert. • **`Landkreis`**: Die ERgebnisse werden auf Landkreisebene kumuliert. • **`Stadt`**: Die Ergebnisse werden auf Stadt/Ort-Ebene kumuliert. • **`Stadtteil`**: Die Ergebnisse werden auf Stadtteil-Ebene kumuliert.
`parameter [array]`	Beliebig viele Parameter zur Spezifizierung der Suche, z.B. Preis, Größe etc. Ein Parameter besteht aus Key und Value (als string), eine Auflistung des erlaubten Keys finden Sie unter 4.3.1.

ProjektID [string] -optional	Die ProjektID, unter der wieder mehrere Anbieter zusammengefasst sein können. Bei API-Anbindungen die mittels eines API-Keys auf die API zugreifen ist dieser Parameter nicht nötig (es kann „null" oder ein Leerstring übergeben werden), da anhand des API-Keys eine feste Bindung an eine ProjektID oder AnbieterID gegeben ist. Die Ergebnismenge bezieht sich nur auf Objekte, die unter dieser ProjektID gespeichert sind.
AdrID [string] -optional	Die Adress-ID, unter der ein Anbieter in der Immowelt erfasst ist. Bei API-Anbindungen die mittels eines API-Keys auf die API zugreifen ist dieser Parameter nicht nötig (es kann „null" oder ein Leerstring übergeben werden), da anhand des API-Keys eine feste Bindung an eine ProjektID oder AnbieterID gegeben ist. Die Ergebnismenge bezieht sich nur auf Objekte, die unter dieser AdrID gespeichert sind.

4.14.2. Rückgabewert

EstateServiceCountGeoIDResponse

Status [enum]	▪ OK: gültiges Suchergebnis ▪ FALSE: ungültiges Suchergebnis
RegionCountItemList [List<RegionCountItem>]	Liste mit RegioCountItem: RegionCountItem: • **GeoID**: die GeoID der Region, nach der die Ergebnisse kumuliert werden. • **GeoID_Description**: die Beschreibung der GeoID • **ObjectCount**: die Anzahl an Objekten die in die Region fallen. • **MieteCount**: Anzahl an Mietobjekten • **KaufCount**: Anzahl an Kaufobjekten

4.15. V2 Versionen der Listen-Funktionen

Es gibt für alle Listen-Funktionen im Estate-Service eine V2-Version.

```
GetListV2(string GeoID, EstateParameterList parameter, int CurrentPage,
          int PageSize, bool UseBigImage)

GetListWithSortV2(string GeoID, EstateParameterList parameter,
                  int CurrentPage, int PageSize, ListSort ls,
                  bool UseBigImage)

GetListAmbitV2(string GeoID, EstateParameterList parameter, int CurrentPage,
                int PageSize, int ambit, bool UseBigImage)

GetListAmbitWithSortV2(string GeoID, EstateParameterList parameter,
                  int CurrentPage,int PageSize, int ambit, ListSort ls,
                  bool UseBigImage)

GetListCustomerProjectsV2(string GeoID, EstateParameterList parameter,
                  ListSort ls, bool IncludeIwObj, int CurrentPage,
                  int PageSize, string Intranet, bool UseBigImage)

GetListByDescriptionV2(string description, int CurrentPage, int PageSize,
                        int ambit, ListSort ls, bool UseBigImage)

GetListByLatLongV2(double Lat, double Long, EstateParameterList parameter,
                   int CurrentPage, int PageSize, int ambit, ListSort ls,
                   bool UseBigImage)

GetSonderwerbeformenV2(string GeoID, EstateParameterList parameter,
                        bool UseBigImage)

GetListByEstateGuidsV2(string EstateGuids, bool UseBigImage)
```

Mittels dieser Funktionen kann man nun per zusätzlichen Parameter "UseBigImage" angeben,
ob man für die Liste die Thumbnails oder die großen Bilder haben möchte.

UseBigImage=false => Thumbnails
UseBigImage=true => große Bilder

4.16. Beispiel in C# zur Nutzung des EstateService

```
EstateService estateService = new EstateService ();
LocationService ls = new LocationService ();

void Search( string areaDescription )
{
      LocationResponse lr = ls.GetLocation (areaDescription);

      if (lr.Status == LocationResponseStatus.OK)
      {
            GetItems( -1, lr.Location[0].GeoID);
      }
}

private int GetItems( int ambit, string geoID)
{
      int pageSize = 5;
      int currentPage = 1;

      //define the search parameters.
      EstateParameter[] estateParams = new EstateParameter[3];
      estateParams[0] = new EstateParameter();
      estateParams[0].Key = EstateParameterKey.EType;
      estateParams[0].Value = "1";
      estateParams[1] = new EstateParameter();
      estateParams[1].Key = EstateParameterKey. RooMi;
      estateParams[1].Value = "1";
      estateParams[2] = new EstateParameter();
      estateParams[2].Key = EstateParameterKey. RooMa;
      estateParams[2].Value = "4";

      //call GetListAmbit with ambit = -1 for an equal call
      //to GetList ... or define the ambit and get a smaller
      //result-set.
      EstateServiceListResponse eslr = estateService.GetListAmbit(
                              geoID, estateParams, currentPage,
                              pageSize, ambit);

      if( eslr.Status == EstateServiceListResponseStatus.ResultTooLarge )
      {
      //GetList/GetListAmbit returns ResultTooLarge,
      //you have to define even more estateParams and/or use
      //GetListAmbit with param ambit > -1.
      //You can call GetRegionOverview to test if you are
      //searching for a city with districts or a region with
      //administrative district.
      //The EstateServiceRegionOverviewResponse.Items are
      //showing a list of possible districts.
            ShowDdistricts( geoID, estateParams);
            DisplayResultTooLargeMessage();
```

```
            return 0;
      }
      else if( eslr.Status == EstateServiceListResponseStatus.OK)
      {
            count.Text = eslr.TotalCount.ToString ();
            BindItemsToResultView();
            return 1;
      }

      return -1;
}

public void ShowDistricts(    string geoID,
                              EstateParameterList estateParams)
{
      EstateServiceRegionOverviewResponse esror =
estateService.GetRegionOverview (geoID, estateParams);

      if (esror.Status == EstateServiceRegionOverviewResponseStatus.OK)
      {
      //With the GeoIDs of this items there are no further
      //districts, so calling GetRegionOverview on one of this
      //item will always result in an empty result-set.
      //If you want to search exactly for one of this Items
      //without an ambit, call Search esror.Items with ambit = -1

            BindItemsToDistrictSelectionView(esror.Items[i].GeoID);
      }
}

public void OnClickDistrict()
{
      //Set ApiKey for LocationService
      ls.ListAuthenticationHeaderValue = new ListAuthenticationHeader();
       ls.ListAuthenticationHeaderValue.ApiKey = "APIKEY";

      //Set ApiKey for EstateService
      estateService.ListAuthenticationHeaderValue =
      new ListAuthenticationHeader();
      estateService.ListAuthenticationHeaderValue.ApiKey = "APIKEY";

      string geoID = GetSelectedDistrict();

      // Call GetItems with no ambit to exactly get the number of
      //Items specified by the result.
      GetItems( 0, geoID);
}
```

4.17. Beispiel in php

```
<?php

$client = new
SoapClient('http://api.immowelt.de/WebServices/EstateService.asmx?WSDL'
);

$authHeader = new SoapHeader('http://immowelt.de/services',
'ListAuthenticationHeader', array('ApiKey' => 'YOURAPIKEY'));

// define estate parameters
$estateParameter[] = array('Key' => 'EType', 'Value' => '1,2,3');
$estateParameter[] = array('Key' => 'ESR', 'Value' => '1');
$estateParameter[] = array('Key' => 'PriMa', 'Value' => '50000');

// define search parameters
$getEstateListParam = array(
          'GeoID' => '10801055046',
          'parameter' => $estateParameter,
          'CurrentPage' => 1,
          'PageSize' => 2,
          'ambit' => ''
);

// get result from "GetListAmbit"
// call GetListAmbit with ambit = -1 for an equal call to GetList
// or define the ambit and get a smaller result-set.

$response = $client->__soapCall('GetListAmbit', array('params' =>
$getEstateListParam), NULL, $authHeader);

if ($response->GetListAmbitResult->Status == 'OK') {
          print_r($response->GetListAmbitResult);
          echo($response->GetListAmbitResult->TotalCount .' Ergebnisse
gefunden');

// render Result ...

}

?>
```

5. EstateExpose - Aufruf eines Exposes

5.1. URLS

Web-Service-URL
http://api.immowelt.de/WebServices/EstateExpose.asmx

WSDL-URL
http://api.immowelt.de/WebServices/EstateExpose.asmx?WSDL

5.2. Übersicht

Der Service EstateExpose enthält folgende Funktionen:

- *GetEstateExposeByEstateGuid*
- *GetEstateExposeByOnlineID*
- *GetImpressumByAdressGuid*

5.3. Methode GetEstateExposeByEstateGuid

Mit dem Ergebnis aus *EstateService.GetList* können Sie die Details des Exposés abfragen.

```
EstateExpose.GetEstateExposeByEstateGuid(string EstateGuid)
```

5.3.1. Parameter

`EstateGuid [string]`	Die eindeutige ID eines Exposés.

5.3.2. Rückgabewert

`ExposeServiceReponse`

`Status [enum]`	Ok, wenn kein Fehler aufgetreten ist. Invalid, wenn die ExposeGuid nicht mehr gültig ist.
`XmlExpose [string]`	Enthält die Daten des Exposes in Xml-Format.

5.3.3. Exceptions

`UnauthorizedAccessException`	Fehlerhafte Authentifizierung, oder keine Zugriffsrechte.

5.4. Methode GetEstateExposeByOnlineID

Mit der *OnlineID* können Sie die Details des Exposés abfragen.

```
ExposeServiceResponse GetEstateExposeByOnlineID(string OnlineID)
```

5.4.1. Parameter

`OnlineID [string]`	Die eindeutige OnlineID eines Exposés.

5.4.2. Rückgabewert

`ExposeServiceReponse`

`Status [enum]`	Ok, wenn kein Fehler aufgetreten ist. Invalid, wenn die ExposeGuid nicht mehr gültig ist.
`XmlExpose [string]`	Enthält die Daten des Exposes in Xml-Format.

5.4.3. Exceptions

`UnauthorizedAccessException`	Fehlerhafte Authentifizierung, oder keine Zugriffsrechte.

5.4.4. Format der Xml-Antwort

ExposeServiceResponse.XmlExpose wird in nachfolgendem Format zurückgegeben:

```
<expose>
      <address AdrGuid="4DA3551A0FB26641A5F401CB2B2287C8">
            <mobil>Handy-Nummer</mobil>
            <fax>Fax-Nummer</fax>
            <phone>Telefon-Nummer</phone>
            <email>E-Mail-Adresse</email>
            <city>Stadt</city>
            <zip>Postleitzahl/zip>
            <street>Straße</street>
            <contactperson>Ansprechpartner</contactperson>
            <salutation>Anrede Ansprechpartner</salutation>
            <company>Firmenname</company>
            <linktopartnerpage>URL der Immowelt-Partner-Seite</linktopartnerpage>
      </address>
      <estate id="" guid="" onlineid="" type-id="" type-description="[IMMOBILIENART]" salestype="[MIETEKAUF]"
            category-id="" category-description="[IMMOBILIEN-KATEGORIE]">
            <!-- innerhalb von Estate befinden sich 1 bis n Items: -->

            <!-- entwdeder einfaches Item, das aus Title und Descriptions besteht -->
            <item id="[ITEMID]">
                  <title>Überschrift, z.B. Kaltmiete</title>
                  <description>Wert, z.B. 500 Euro</description>
            </item>
            <!-- oder Item mit Title und 1 bis n Unter-Items (descriptionitem), die wiederum aus Title
            und Description bestehen →
            <item id="[ITEMID]">
                  <title>Überschrift</title>
                  <descriptionitem>
                        <title>Untergeordnete überschrift 1</title>
```

```
                  <description>Wert</description>
            </descriptionitem>
            <descriptionitem>
                  <title>Untergeordnete überschrift 1</title>
                  <description>Wert</description>
            </descriptionitem>
      </item>
<estate>

<images>
      <thumbnail>http://pics1.immowelt.de/immo-ek/ObjBilder/aaa.jpg</thumbnail>
      <image id="0">
            <source>http://bilder.immowelt.de/immo-tek/ObjBilder/bbb.jpg</source>
            <description>Bild 1</description>
            <source_thumbnail>http://thumbs1.immowelt.de/immo-tek/100_bbb.jpg</source_thumbnail>
            <source_XXL>http:// pics1.immowelt.de/immo-ek/ObjBilder/xxl.jpg </source_XXL>
      </image>
      <image id="1">
            <source>http://bilder.immowelt.de/immo-tek/ObjBilder/ccc.jpg</source>
            <description>Bild 2</description>
            <source_thumbnail>http://thumbs1.immowelt.de/immo-tek/100_bbb.jpg</source_thumbnail>
            <source_XXL>http:// pics1.immowelt.de/immo-ek/ObjBilder/xxl.jpg </source_XXL>
      </image>
      <image id="2">
            <source>http://bilder.immowelt.de/immo-tek/ObjBilder/ddd.jpg</source>
            <description>Bild 3</description>
            <source_thumbnail>http://thumbs1.immowelt.de/immo-tek/100_bbb.jpg</source_thumbnail>
            <source_XXL>http:// pics1.immowelt.de/immo-ek/ObjBilder/xxl.jpg </source_XXL>
      </image>
</images>

<attachments>
      <Document id="0">
            <source>http://files.immowelt.de/0/F/4/7/8553B1A55A74F0.pdf</source>
            <description>Grundriss EG</description>
      </Document>
```

Immowelt-API

```
      <Document id="1">
            <source>http://files.immowelt.de/0/F/4/7/8553Bydfggy1A55A74F0.pdf</source>
            <description>Kostenaufstellung</description>
      </Document>
      ...
</attachments>

<youtubelinks>
      <Document id="0">
            <source>http://www.youtube.com/v/2DYEkxyy7S4</source>
            <description>Deura Rohbau in 3 Tagen mit Liapor Massivwänden</description>
      </Document>
      …
</youtubelinks>

<extensions>
      <environmentmap visible="false" />
      <financecalculator visible="false" />
      <contactformular visible="true" />
      <energyperformance visible="true"
                  EnergiePassArt="Heizwärmebedarf (HWB)"
                  EnergiePassWert="30,5600"
                  EnergiePassWertKlasse="Klasse B"
                  EnergiePassInclWasser="False" />
      <energyperformanceat visible="true"
                  value="0,8900"
                  valueText="Klasse B"
                  type="Gesamtenergieeffizienz (fGEE)" />
      <energyperformancelist visible="true">
            <energyperformanceitem visible="true"
                  Norm="EnergieAusweis vorhanden"
                  AusweisId="1"
                  Art=""
                  AusweisArtId="0"
                  GebaeudeTyp=""
                  GebaeudeTypId="0"
                  AusstellDatum=""
                  GueltigBisDatum=""
                  Wert1="30,56"
```

```
                        Wert1Typ="Heizwärmebedarf (HWB)"
                        Wert1KlasseId="4"
                        Wert1Klasse="Klasse B"
                        Wert1InclWarmWasser="False"
                        Wert2="0,89"
                        Wert2Typ="Gesamtenergieeffizienz (fGEE)"
                        Wert2KlasseId="4" Wert2Klasse="Klasse B"
                        Wert2InclWarmWasser="False"
                        Baujahr="0"
                        PrimaerEnergieTraeger=""
                        Bezeichnung="" />
            <energyperformanceitem visible="true"
                        Norm="EnergieAusweis vorhanden"
                        AusweisId="1"
                        Art=""
                        AusweisArtId="0"
                        GebaeudeTyp=""
                        GebaeudeTypId="0"
                        AusstellDatum=""
                        GueltigBisDatum=""
                        Wert1="52,00"
                        Wert1Typ="Heizwärmebedarf (HWB)"
                        Wert1KlasseId="3"
                        Wert1Klasse="Klasse A"
                        Wert1InclWarmWasser="False"
                        Wert2="0,95"
                        Wert2Typ="Gesamtenergieeffizienz (fGEE)"
                        Wert2KlasseId="5"
                        Wert2Klasse="Klasse C"
                        Wert2InclWarmWasser="False"
                        Baujahr="0"
                        PrimaerEnergieTraeger=""
                        Bezeichnung="EAusweis2" />
      </energyperformancelist>
      <slideshow visible="false" />
</extensions>

<GeoData>
      <GeoID>10809162000029</GeoID>
      <laengengrad>11,5421</laengengrad>
      <breitengrad>48,1175</breitengrad>
```

```
      </GeoData>
<expose>
```

***energyperformance:** **Veraltet!** EnergiePassArt, EnergiePassWert, EnergiePassInclWasser sind nur bei visible="true" vorhanden. EnergiePassArt kann dabei folgende zwei Werte annehmen:

- Bedarfsausweis
- Verbraucherausweis

Falls energyperformanceat visible=true ist auch noch das Attribut EnergiePassWertKlasse vorhanden. EnergiePassArt hat dann den Wert: Heizwärmebedarf (HWB)

EnergyPassClass:
Energieklassen für Wohngebäude (1-9, 0 bedeutet „keine" Klasse)

EnergyPassBitMask:
Repräsentiert die Darstellung für "Energieverbrauch für Warmwasser enthalten" (Nichtwohngebäuden -> Verbrauchsausweis -> „Endenergieverbrauch nach Wärme und Strom getrennt"):

1 – „Energieverbrauch für Warmwasser enthalten" für Endenergieverbrauch Wärme
2 – "Energieverbrauch für Warmwasser enthalten" für Endenergieverbrauch Strom
3 – beide (1 und 2)
0 – keine

EnergyPassHeat:
Endenergiebedarf / Endenergieverbrauch Wärme (Nichtwohngebäuden)

EnergyPassPower:
Endenergiebedarf / Endenergieverbrauch Strom (Nichtwohngebäuden)

***energyperformanceat (nur bei österreichischen Objekten):** **Veraltet!** value, type sind nur bei visible="true" vorhanden. type kann dabei abhängig von value folgende Werte annehmen:

- Klasse A++
- Klasse A+
- Klasse A
- Klasse B

- Klasse C
- Klasse D
- Klasse E
- Klasse F
- Klasse G

EnergyPerformanceList		*Liste von Energieausweisen*
	Art	*Art des Energieausweises: Verbrauchsausweis /Bedarfsausweis*
	AusweisArtId	*ID der Art des Energieausweises*
	AusstellDatum	*Ausstellungsdatum des Energieausweises*
	Baujahr	*Baujahr laut Energieausweis*
	Bezeichnung	*Frei wählbare Bezeichnung des Energieausweises*
	GebaeudeTyp	*Wohngebäude oder Nicht-Wohngebäude*
	GebaeudeTypId	*ID des Gebäudetyps*
	GueltigBisDatum	*Gültigkeits Datum des Energieausweises*
	Norm	*Richtilinie nach welcher der Energieausweis erstellt wurde.* *Alte Richtlinie : ausgestellt vor 31.04.2014* *Neue Richtilinie: ausgestellt nach 01.05.2014* *Nicht Nötig: nicht notwendig (Denkmalschutz)* *Bei AT: Energieausweis vorhanden*
	AusweisId	*ID der Ausweises, entspricht dem Wert von Norm*
	PrimaerEnergie Traeger	*Hauptenergieträger kann hier angegeben werden*
	Wert1	*Erster Energiewert*
	Wert1Klasse	*Energiewertklasse zu Wert1*
	Wert1KlasseId	*Verwendete GrenzwertId zu der Klasse*
	Wert1Typ	*Bedeutung der Zahl in Wert1 z.B. „Heizwärmebedarf (HWB)" oder „Endenergiebedarf"*
	Wert2	*Zweiter Energiewert*
	Wert2Klasse	*Energiewertklasse zu Wert2*
	Wert2KlasseId	*Verwendete GrenzwertId zu der Klasse*
	Wert2Typ	*Bedeutung der Zahl in Wert2 z.B. „Gesamtenergieeffizienz (fGEE)" oder „Endenergieverbrauch Strom"*

Platzhalter (Variablen innerhalb der Xml-Antwort)

[IMMOBILIENART]

die Immobilien-Art, z.B. Wohnung

[IMMOBILIEN-KATEGORIE]

Unterkategorie der Immobilienart, bei Wohnung z.B. Loft, bei Haus z.B. Einfamilienhaus

[MIETEKAUF]

Gibt die Vertriebsart an.
Mögliche Werte:
1: Kauf
2: Miete
0: Miete oder Kauf

[ITEMID]

Zur Identifizierung der Items, dieser Wert soll nicht ausgegeben, sondern nur intern verwendet werden, so kann z.B. mit dieser ItemID per XPATH eine Teilmenge aus dem Expose selektiert werden.

Mögliche Werte der Item-IDs:

ReferenceNumber	Referenznummer
OnlineID	Online-ID mit der das Objekt auch aufgerufen werden kann.
Description	Die Überschrift des Expose
LocationStreet	Die Straße, in der sich das Objekt befindet.
LocationZipCity	PLZ / Ort
LocationZip	PLZ
LocationCity	Ort
LocationCountry	Land
Price	Preis / Kaltmiete
PriceNettoKaltmiete	
HeatingCosts	Heizkosten
AdditionalCosts	Nebenkosten
PriceWarmmiete	Warmmiete
ParkingPrice	Stellplatzpreis
CommonCharge	Hausgeld
PriceQm	Preis / m²
Kaution	Kaution
Provision	Makler-Provision
Baujahr	Baujahr der Immobilie

Bezugsfrei	Bezugsfrei
AreaLiving	Wohnfläche
AreaLand	Grundstücksfläche
Rooms	Zimmer
WindowFront	Größe des Fenster (z.B. bei Schaufenster)
XFach	X-fache Miete
Ausstattung	Ausstattungsmerkmale
Lage	
Info1	Beschreibungsfeld: Objektbeschreibung
Info2	Beschreibungsfeld: Lagebeschreibung
Info3	Beschreibungsfeld: Ausstattung
Info4	Beschreibungsfeld: Nebenkosten/Wohngeld
Info5	Beschreibungsfeld: Sonstiges
Info6	Beschreibungsfeld: AGB/Haftung
Info7	Beschreibungsfeld: Besichtigung
Info8	Beschreibungsfeld: Nutzung
Info9	Beschreibungsfeld: weiteres Feld
Info10	Beschreibungsfeld: weiteres Feld
Mindestmietdauer	Mindestmietdauer
Moeblierung	Angabe ob eine Möblierung vorhanden ist.
MaximumPersonen	Maximale Personen Anzahl
ParkingSlots	Stellplatzanzahl
Stockwerk	Stockwerk
AvailableInRegion	Verfügbar in Region für Typenhäuser
PreferredRoommate	
SeatCount	

5.4.5. Anmerkungen

Das Element **<images />** wird nur angezeigt, wenn das Expose Bilder enthält. Im ersten Kind-Element **<thumbnail>** steht die URL des Thumbnails des Startbilds. Danach kommen die Image-Elemente aller Bilder mit URL und Beschreibung. (Die URL liefert das Bild jeweils in 400er Breite, die URLs anderer Breiten sind zurzeit noch nicht enthalten.)

Der Anbieter steht im Element **<address />,** jedoch wird dort immer nur soviel Information angezeigt, wie auch im Standard-Immowelt-Expose enthalten ist. Handelt es sich z.B. um eine Chiffre-Anzeige, wird gar nichts angezeigt. In diesem Fall kann nur über den nachfolgend beschriebenen Communication-Service ein Kontakt hergestellt werden.

Die beiden Elemente **<laengengrad />** und **<breitengrad />** sind beide nur gefüllt, falls die GeoID >= 11 Stellen hat. Falls dem Objekt /Expose eine GeoID

kleiner 11 Stellen zugeordnet ist, ist in der Response der Bereich **<GeoData>…</GeoData> nicht enthalten**! Falls zu dem entsprechenden Objekt keine Koordinaten vorliegen, werden Laengengrad und breitengrad mit -1 belegt.

Das Element <youtubelinks> ist nur vorhanden, falls das entsprechende Expose auch Youtube-Links beinhaltet.

5.5. Methode GetImpressumByAdressGuid

Mit der *AdrGuid* können Sie das Impressum des Anbieters abfragen. (Die AdrGuid wird mit dem ExposeServiceResponse.XmlExpose im Tag address als XmlAttribut mitgeliefert.)

```
ImpressumResponse GetImpressumByAdressGuid(string AdrGuid)
```

5.5.1. Parameter

AdrGuid [string]	Die eindeutige Guid eines Anbieters.

5.5.2. Rückgabewert

ImpressumResponse

Status [enum]	Ok, wenn kein Fehler aufgetreten ist. Invalid, wenn die AdrGuid nicht mehr gültig ist. Empty, wenn das Impressum keine Daten enthält. (z.B. falls der Anbieter kein Impressum hinterlegt hat.)
LogoUrl [string]	Die Url zum Anbieterlogo.
DistrictCourt [string]	Enthält die Daten von: Berufskammer
TradeRegister [string]	Enthält die Daten von: Handelsregisternummer
UID [string]	Enthält die Daten von: Umsatzsteueridentifikationsnummer
Representant [string]	Enthält die Daten von: Vertretungsberechtigter
RepName_Company [string]	Enthält die Daten von: Firmenname
RepName_Street [string]	Enthält die Daten von: Straße der Firmenanschrift
RepName_PLZ [string]	Enthält die Daten von: PLZ der Firmenanschrift
RepName_City [string]	Enthält die Daten von: Ort der Firmenanschrift
RepName_Phone [string]	Enthält die Daten von: Telefonnummer
RepName_Fax	Enthält die Daten von: Telefaxnummer

[string]	
RepName_Email [string]	Enthält die Daten von: E-Mail
RepName_AufsBeh [string]	Enthält die Daten von: Berufsaufsichtsbehörde
RepName_GEW_Stadt [string]	Enthält die Daten von: Handelsregister
RepName_HomepageLink [string]	Enthält die Daten von: Link zur Homepage
RepName_Bemerkung [string]	Enthält die Daten von: Bemerkung
AGB [string]	Enthält die Daten von: Link zu AGBs
CustomData [string]	Bei manchen Anbietern gibt es zusätzliche Daten, die angezeigt werden sollen. Diese befinden sich innerhalb dieses Datenfeldes als unformatierter Text.

5.5.3. Exceptions

UnauthorizedAccessException	Fehlerhafte Authentifizierung, oder keine Zugriffsrechte.
ArgumentException	Der Übergabe-Parameter entspricht nicht einer Korrekten AdrGuid.

6. CommunicationService – Versenden von Kontaktanfragen

6.1. URLS

Web-Service
http://api.immowelt.de/WebServices/CommunicationService.asmx

WSDL
http://api.immowelt.de/WebServices/CommunicationService.asmx?WSDL

6.2. Übersicht

Der CommunicationService enthält folgende Funktionen:

- *SendContactMail*
- *SendRecommendationMail*

6.3. Methode SendContactMail

```
bool SendContactMail(ContactMailRequest request)
```

6.3.1. Parameter

`request [ContactMailRequest]`	Datenstruktur mit Informationen zum Versenden einer Anfrage E-Mail an den Immobilien Anbieter ▪ LinkToDetail ▪ MailTemplateEnquirerUrl (muss nicht gesetzt werden) ▪ MailTemplateProviderUrl (muss nicht gesetzt werden) ▪ EstateGuid ▪ Enquirer ▪ Message ▪ Domain ▪ SubjectType

- LinkToDetail:
  Setzt oder gibt die Url zurück, unter der das Expose abgerufen werden kann (die ID des Exposes muss dabei nicht übergeben werden).

- MailTemplateEnquirerUrl:
  Setzt oder gibt die URL zum E-Mail-Template für die E-Mail zurück, die an den Interessenten geht. Wenn dieses Property nicht gesetzt wird, wird das Standard Immowelt-Template verwendet.
  (Default-Value:
  http://www.immowelt.de/EMailTemplates/ObjektAnfrageUser.txt)

- MailTemplateProviderUrl:
  Setzt oder gibt die URL zum E-Mail-Template für die E-Mail zurück, die an den Anbieter geht. Wenn dieses Property nicht gesetzt wird, wird das Standard Immowelt-Template verwendet.
  (Default-Value:
  http://www.immowelt.de/EMailTemplates/ObjektAnfrageProvider.txt)

- EstateGuid:
  Die EstateGuid zu der die Anfrage gesendet wird.

- Enquirer:
  Kontaktdaten des Interessenten. Diese werden in Form eines Contact-Objektes übergeben:

  Contact besteht aus:
  - Name (String) – sollte auch als Pflichtfeld gesetzt sein!
  - EMail (String) - muss gesetzt sein!
  - Phone (String)
  - Street (String)
  - Zip (String)
  - City (String)
  - Mobile (String)
  - Fax (String)

- Message:
  Nachricht des Interessenten an den Anbieter

- Domain:
  Setzt oder gibt die URL des Auftritts zurück, von der die Anfrage stammt (wird in der Betreffzeile der E-Mail angezeigt)

- SubjectType:
  Hier steht der Betreff des Anfragenden, z.B. „Bitte um Rückruf".

Beispiel:

```
public bool TestSendAnfrageMail()
{
   ContactMailRequest request = new ContactMailRequest();

   request.Domain = "localhost";
   request.LinkToDetail =
      "http://www.immowelt.de/Immobilien/ImmoDetail.aspx";
   Contact contact = new Contact();
   contact.Name = "Hans Mustermann";
   contact.EMail = "test@immowelt.de";
   contact.Phone = "0911/5202520";

   request.Enquirer = contact;
   request.EstateGuid = "430E845F597843B389765BBAE8FA19F4";

   request.Message = "Testanfrage";
```

```
    request.SubjectType = "Bitte um Rückruf";

    if (!communication.SendContactMail(request))
        return false;

    return true;
}
```

6.4. Methode SendRecommendationMail

```
bool SendRecommendationMail (RecommendationMailRequest request)
```

6.4.1. Parameter

`request [RecommendationMailRequest]`	Datenstruktur mit Informationen zum Versenden einer Empfehlungs-E-Mail. • LinkToSite • Objid • EstateGuid • MailTemplateRecipient • MailTemplateSender • NameRecipient • EmailRecipient • NameSender • EmailSender • Message • AdditionalInfo

- LinkToSite:
  Verweist defaultmäßig auf folgende Seite:
  http://www.immowelt.de/Immobilien/ImmoDetail.aspx.
  Muss nur gesetzt werden, wenn auf eine Nicht-Expose-Seite verwiesen werden soll.

- Objid:
  Die Objekt-ID des empfohlenen Immobilien-Objektes.

- EstateGuid:
  Die Guid des empfohlenen Immobilien-Objektes.

  Wenn ein Immobilienobjekt weiter empfohlen werden soll, muss entweder die Objid oder die EstateGuid des entsprechenden Objektes gesetzt sein.

- MailTemplateRecipient:
  Setzt oder gibt die URL zum E-Mail-Template für die E-Mail zurück, die an den Empfänger geht. Wenn dieses Property nicht gesetzt wird, wird das Standard Immowelt-Template verwendet.

(Default-Value:
http://www.immowelt.de/EMailTemplates/Empfehlung_IW_Empfaenger.txt)

- MailTemplateSender**:**
  Setzt oder gibt die URL zum E-Mail-Template für die E-Mail zurück, die an den Absender zur Bestätigung geht. Wenn dieses Property nicht gesetzt wird, wird das Standard Immowelt-Template verwendet.
  (Default-Value:
  http://www.immowelt.de/EMailTemplates/Empfehlung_IW_Sender.txt)

- NameRecipient:
  Der Name des Empfängers.

- EmailRecipient**:**
  Die E-Mail-Adresse des Empfängers.

- NameSender:
  Der Name des Absenders.

- EmailSender:
  Die E-Mail-Adresse des Absenders.

- Message:
  Die Nachricht die in der E-Mail übermittelt werden soll.

- AdditionalInfo:
  Hier können zusätzliche Informationen übermittelt werden.

Beispiel:

```
public bool TestSendRecommendationMail()
{
  RecommendationMailRequest request = new RecommendationMailRequest();

  request.EmailRecipient = "mustermann@test.de";
  request.EmailSender = "meier@test.de";
  request.NameRecipient = "Hans Mustermann";
  request.NameSender = "Herbert Meier";
  request.Objid = "15549972";

  request.Message = "Hier steht eine kurze Nachricht für den Empfänger";

  if (!communication.SendRecommendationMail(request))
      return false;

  return true;
}
```

7. Interpretation der GeoID

1 2 3	4 5	6 7 8	9 10 11	12 13 14
108	09	564	000	001

Dreistellige GeoID: z.B. 108
Die ersten drei Stellen stehen für das **Land**: z.B. 108 für Deutschland

Fünfstellige GeoID: z.B. 10809
Die Stellen 4 und 5 geben das **Bundesland** an: z.B. 10809 steht für Bayern

Sechsstellige GeoID: z.B. 108095
Die sechste Stelle steht für den **Regierungsbezirk**: z.B. 108095 steht für RB Mittelfranken

Achtstellige GeoID: z.B. 10809564
Die Stellen 7 und 8 stehen für den **Land- oder Stadtkreis**: z.B. 10809564 steht für den Stadtkreis Nürnberg

11-stellige GeoID: z.B. 10809564000
Die Stellen 9 bis 11 stehen für die jeweilige Stadt oder den jeweiligen Ort: z.B. 10809564000 steht für die Stadt Nürnberg

14-stellige GeoID: z.B. 10809564000001
Die Stellen 12 bis 14 stehen meistens für den Stadtteil einer Stadt: z.B. 10809564000001 steht für den Stadtteil Nürnberg-Almoshof
14-stellige GeoIDs können aber auch bei Ortschaften vorkommen.

Bei einer 14-stelligen GeoID erkennt man an den Stellen 12 bis 14 auch, ob es sich um einen Stadtteil handelt oder um einen Ort.
Bei Orten beginnt die Stelle 12 in der GeoID mit einer 5. Bei Stadtteilen einer Stadt werden die Stellen 12 bis 14 mit 001 beginnend hochgezählt (z.B. 10809564000001, 10809564000002, 10809564000003, …, 10809564000124).

2 Sonderfälle: Berlin und Hamburg
Hier steht die 5/6-stellige GeoID bereits für die gesamte Stadt
Die 8-stellige GeoID steht hier für Stadtbezirke (z.B. 10811009 Berlin Treptow-Köpenick).
Die 14-stellige GeoID steht hier auch wieder für die Stadtteile.

8. TextSearchService

8.1. URLS

Web-Service
http://api.immowelt.de/WebServices/TextSearchService.asmx

WSDL
http://api.immowelt.de/WebServices/TextSearchService.asmx?WSDL

8.2. Methode GetTextSearchListCount

Diese Methode liefert als Rückgabewert ein TextSearchResponse-Objekt. In diesem Objekt werden die Anzahl der gefundenen Treffer und eine URL zu der jeweiligen Liste zurückgeliefert.
Die URL verweist je nach angegebener MedienID auf die Ergebnisliste des jeweiligen zu dieser MedienID gehörenden Medien-Portals.
Neben der Anzahl der Treffer und der URL enthält das TextSearchResponse-Objekt auch eine Property „Status". Mittels dieser Property kann man überprüfen, ob die Anfrage erfolgreich verlief.

```
TextSearchResponse GetTextSearchListCount(string valueWas, string valueWo,
string mediaID)
```

8.2.1. Parameter

`valueWas [String]`	Übergabe der Suchparameter nach "Was" man sucht, z.B.: „2 Zimmer Wohnung Miete bis 500€"
`valueWo [String]`	Übergabe der Suchparameter bezüglich dem „Wo" man sucht, z.B.: „Nürnberg" oder „90411" oder „M" für München
`mediaID [String]`	Übergabeparameter für die MedienID, mittels der die Immowelt die verschiedenen Medienauftritte verwaltet.

8.2.2. Rückgabewert

`TextSearchResponse`

`Status [enum]`	▪ OK (gültiges Suchergebnis) ▪ Unknown
`Url`	verweist je nach angegebener MedienID auf die Ergebnisliste des jeweiligen zu dieser MedienID gehörenden Medien-Portals
`Count [string]`	Anzahl der Treffer