API

- Common Information
- aB-Agenta API Inbetriebnahme
- API Troubleshooting
- Object types, Properties and Selection lists
- Filter

Common Information

The aB-Agenta API enables third-party programs to access aB-Agenta data. To enable the API, you have to install the aB-Agenta webservice and configure the API options.How to do this and the meaning of the options you can see here.

The API offers these functions:

- load / search records
- modify records
- create new records
- delete records
- download documents
- upload new documents

The **OpenAPI** specifications can be found here: https://demoserver.artbase-software.de/api2_1/swaggerui

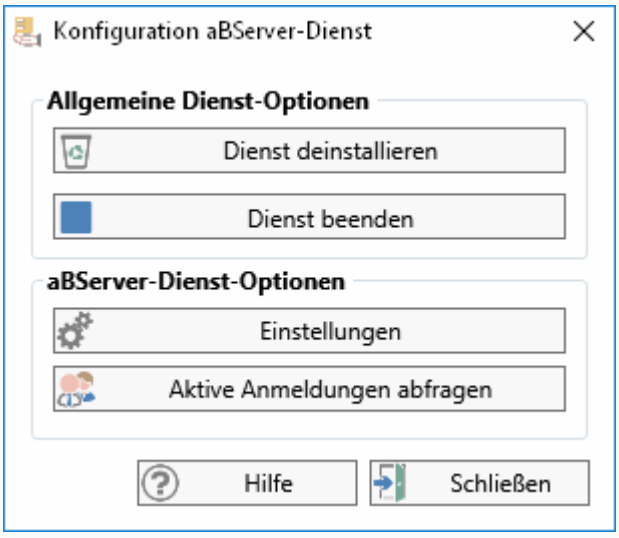
We recommend to use it on your own service with your own data and credentials to run tests that better suit your environment. (http(s)://your-server:port/api2_1/swaggerui)

Please keep in mind that **aB-Agenta is not a cloud solution**, but an offline application. The actual base url of your productive API is individual. There is no global API server. **The url above is intended for testing purposes only.**

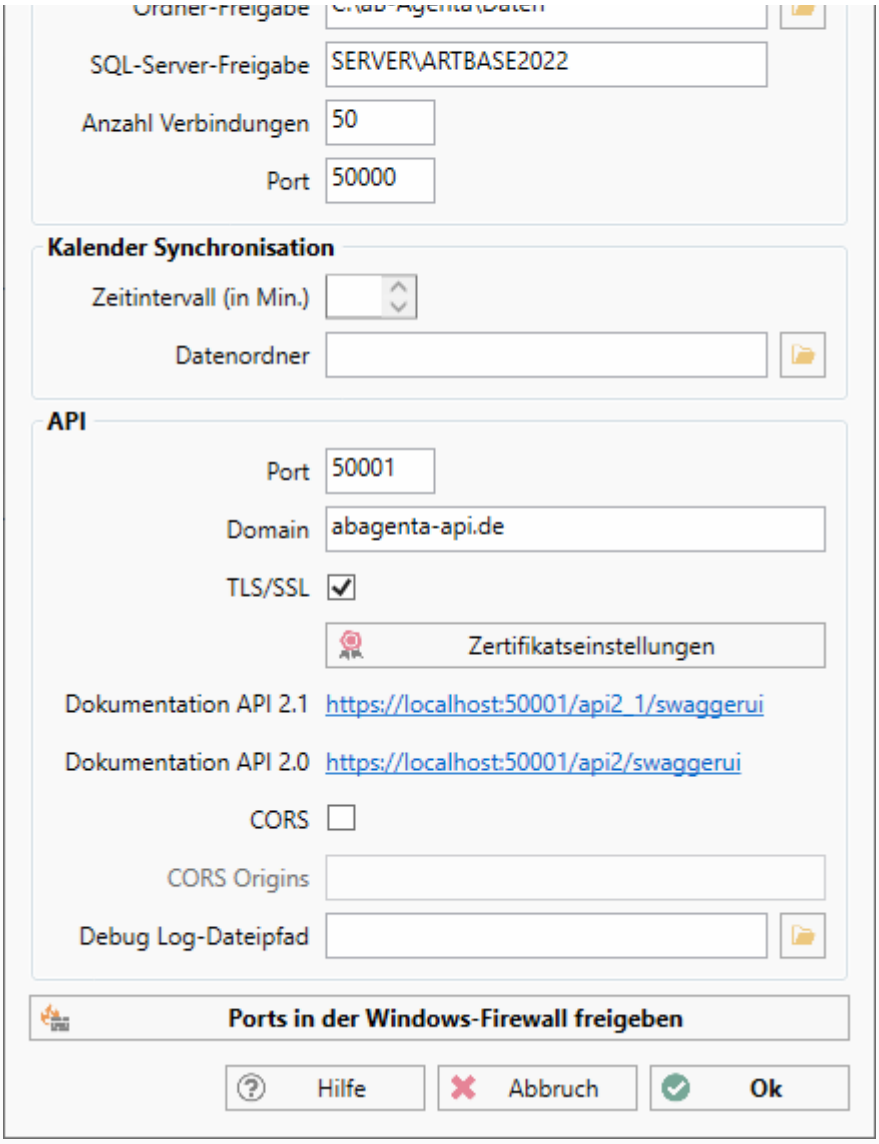
aB-Agenta API Inbetriebnahme

Öffnen Sie aB-Agenta auf dem Rechner, auf dem aB-Agenta installiert ist und die "aB-Agenta Daten" liegen.

Wechseln Sie in das Applikationsmenü und klicken links auf den Eintrag "System-Verwaltung". Rechts im Abschnitt System-Module klicken Sie auf aBServer-Dienst Konfiguration.
Es öffnet sich das Konfigurationsfenster unseres Dienstes, der für die Bereitstellung Ihrer Daten über die aB-Agenta API zuständig ist.



Je nach Status und Einrichtung des Dienstes sind die Knöpfe hier unterschiedlich benannt und aktiviert. In unserem Beispiel ist der Dienst bereits konfiguriert und aktiv.
Klicken Sie nun auf Einstellungen. Es öffnet sich der folgende Dialog.



Im Abschnitt API können Sie die Einstellungen vornehmen, damit der Zugriff von InsurMagic auf Ihre aB-Agenta Daten funktioniert.
Zuerst wählen Sie einen lokalen Port. Auf diesem Port horcht die API auf Anfragen von außerhalb.
Außerdem müssen Sie den Haken bei CORS setzen, damit Anfragen von anderen Seiten, wie z.B. InsurMagic, vom Dienst erlaubt werden.
Drücken Sie nun noch auf den Knopf Ports in der Windows-Firewall freigeben, sofern Sie die Windows-Firewall nutzen und diese eingeschaltet ist. Nutzen Sie eine andere Software-Firewall auf diesem Rechner, so müssen Sie dort den zugriff auf den oben eingestellten Port für TCP-Verbindungen erlauben.
Sollten Sie WhatsApp-Web und InsurMagic nur intern im Büro-Netzwerk nutzen, ist die Einrichtung nun schon abgeschlossen.
Sollte ein Zugriff von außerhalb, beispielsweise einem Remote-Arbeitsplatz, notwendig sein, so müssen Sie die Verbindung TSL/SSL-verschlüsseln, damit die Daten nicht im Klartext über das Internet übertragen werden.
Dazu setzen Sie den Haken bei TLS/SSL. Damit der verschlüsselte Zugriff auf die API funktioniert, wird zusätzlich eine Domain und ein Zertifikat für diese Domain benötigt. Tragen Sie also die Domain im dafür vorgesehenen Feld ein. Diese Domain muss so eingestellt sein, dass eine Anfrage über einen Webbrowser zu Ihrem Internet-Anschluss geroutet wird. Sprechen Sie Ihren EDV-Betreuer

darauf an, damit er Ihnen dieses Routing einstellt.
Wenn diese Voraussetzungen erfüllt sind, können Sie ein Zertifikat nutzen. Klicken Sie für die Einstellungen auf Zertifikatseinstellungen. Was Sie dort einstellen müssen, können Sie unter Zertifikatseinstellungen nachlesen.
Nachdem alle Einstellungen getroffen wurden, klicken Sie auf Ok, damit diese übernommen werden. Der Dienst wird nun mit den neuen Einstellungen neu gestartet, so dass diese gültig werden.

API Troubleshooting

Grundregel

- Bei Problemen zuerst sicherstellen, dass aB-Agenta auf die aktuelle Version aktualisiert ist. Viele Fehlerursachen sind bereits in neueren Versionen behoben.

Logdatei prüfen

- Im Unterordner logs liegt abserverservicefx.log. Viele Fehlerursachen können hier gefunden werden.

Service startet nicht

- Fehlkonfiguration ? Einstellungen auf Korrektheit und Vollständigkeit prüfen
- Antivirus unterbindet Ausführung des Service ? Ausnahme definieren

Service via localhost nicht erreichbar

- Port schon belegt ? anderen Port ausprobieren

Service im LAN nicht erreichbar

- Port in Windows-Firewall gesperrt ? Port öffnen via Button in Service-Einstellungen
- Port in Drittanbieter Software Firewall gesperrt ? Port manuell öffnen

Service im WAN nicht erreichbar

- Port in Hardware-Firewall gesperrt ? öffnen
- Router zwischen LAN und WAN ? Port weiterleiten
- Verweist die Domain auf die korrekte öffentliche IP-Adresse?
- Ist die öffentliche IP-Adresse erreichbar?

Let’s Encrypt Zertifikat kann nicht ausgestellt werden

- Let’s Encrypt Port in lokaler oder Hardware-Firewall gesperrt ? öffnen
- Let’s Encrypt muss im WAN auf Port 80 erreichbar sein. Ggf. entsprechende Port-Weiterleitung einrichten
- Service hat keine Schreibrechte auf Zertifikatsspeicher ? Service als Benutzer mit Admin-Rechten betreiben

Anmeldung scheitert trotz korrekten Anmeldedaten

- Datenordner wurde aus Client Perspektive angegeben ? aus Server Perspektive angeben
- Nicht ausreichende Zugriffsrechte auf Datenordner ? Rechte für den Benutzer erteilen, der den Service ausführt
- Datenordner liegt im LAN auf Netzwerkfreigabe ? der Benutzer „Lokales System“ kann nicht auf Netzwerkfreigaben zugreifen. Melden Sie den Service mit einem anderen

Benutzer mit lokalen Administrator-Rechten an.
- Datenordner liegt auf Netzlaufwerk ? Netzlaufwerke sind meistens für einzelne Benutzer definiert. Verwenden Sie stattdessen einen UNC-Pfad.
- Antivirus sperrt Zugriff auf Datenordner ? Ausnahme definieren
- Datenordner wurde nicht (korrekt) im Service freigegeben ? In Service-Einstellungen den Datenordner korrekt in **Ordner-Freigabe** eingeben. Schreibweise muss mit Datenordnereinstellung übereinstimmen. Siehe Log-Datei für erwartete Schreibweise.
- SQL-Server wurde nicht (korrekt) im Service freigegeben ? In Service-Einstellungen die SQL-Server-Instanz korrekt in **SQL-Server-Freigabe** eingeben. Schreibweise muss mit Datenordnereinstellung übereinstimmen. Siehe Log-Datei für erwartete Schreibweise.

Kein Zugriff auf Dokumente

- Nicht ausreichende Zugriffsrechte auf ggf. *abweichenden Dokumentenordner* (Arbeitsplatzeinstellungen) ? Rechte für den Benutzer erteilen, der den Service ausführt
- ggf. *abweichender Dokumentenordner* liegt im LAN auf Netzwerkfreigabe ? der Benutzer „Lokales System“ kann nicht auf Netzwerkfreigaben zugreifen. Melden Sie den Service mit einem anderen Benutzer mit lokalen Administrator-Rechten an.

Object types, Properties and Selection lists

In aB-Agenta, data is available in a specific structure. It consists of records, each of which belongs to a specific object type and has multiple properties and values.
Some object types inherit from others, resulting in a tree-like hierarchy in which each object type inherits the properties of its parent.
Keep in mind that there may be user-defined properties and object types.
The API has **operations for retrieving object types and their properties**. This allows you to determine their IDs and names.

Another way of looking up object types and properties is the Masken-Designer, a module of aB-Agenta.

Object types

Every record has a specific object type. A record is uniquely identifiable by its ID **AND** its object type.
Object types are entities (like customers, contracts etc.) and have properties.
An object type may belong to an inheritance hierarchy, where it inherits the properties of its parent object type. The parent object type is referenced in the field "*basicidobject*".
An object type that has children is a "group". If you use a group while loading records you can retrieve records of different object types at once.
If you load records by a group you can only load properties that belong to that group.
The object type of a record can never be a group, it must be an end node that has no children.
If you try to save a record with a group as object type the operation will fail.
You can change a record's object type only if it inherits from one of the following groups:

- "Standardadressen" - ID: "-6"
- "Agenturen / Mitarbeiter" - ID: "-62"

Properties

Properties belong to specific object types and define the fields a record may have.
A property has a name and an internal name ("*bound_on*"). The name can be user-defined in aB-Agenta so whenever you reference a property (e.g. accessing fields of a record or setting up a query-filter) you must use the internal name.
A property has a data type ("*datatype_user*"):

- 1=text
- 2=integer
- 3=decimal
- 4=date
- 5=boolean
- 6=memo (long text)
- 7=time
- 8=currency (decimal with only two decimal places)
- 9=date-time
- 11=true-false-unknown (tristate type: true=1, false=2, unknown=0)

A property has a property type ("*type*") (don't mistake with data type):

- 1=standard

- 2=reference (to another record by its id, like a foreign key)
- 3=encoded (the value is always an integer that has a bigger meaning; encoded properties always have an encoded selection list thats values are used)
- 8=image (references an image document -which is a normal record too- so it is a variant of the type 2)

A property can have a selection list ("*idlist*") that contains the valid values that a record can have for that property.

Selection lists

A selection list is referenced by one or more properties and contains values that a record can have for those properties.
A selection list also has a data type ("*datatype*") that matches the ones of its properties.
There are two types of selection list: Encoded and Non-Encoded lists ("*encoded*")

- ## encoded
  : records **MUST** store a value of one of the list entries (field "*value*" in list-entries)

- ## non-encoded
  : records may save any value. The list entries are merely suggestions. (field "*name*" in list-entries)

Filter

The filter parameter in the
loadrecords
and
loadobjecttypes
functions is a complex JSON object representing the query criteria.
If the filter is omitted, no filter is applied.
A filter object can have different characteristics and also contain other filter objects. Multiple nested filters can therefore be defined.

Filter classes

Comparer

This filter performs one or more comparison operations on a field.
The properties of this filter have the name of a field and as value another object that describes the comparison operation and the comparison values:
`{field name: {operator: value}}`
Some comparison operations expect 2 or more comparison values. These are put into an array:
`{field name: {operator: [value1, value2, ...]}}`
Multiple comparison operations can also be specified for a single field by placing them in an array:
(The comparison operations are ANDed, so they must all apply)
`{field name: [{operator1: value1},{operator2: value2},...]}`
Example:
`{$system_id: {$equals, "123456"}}  ≙ select * from tab where system_id='123456'`

comparison operations

Operator	Value (data type)	Description	Example	corresponds to this SQL query
$isEmpty	bool	true only; field has empty value; identical to $isNull	{system_id: {$isEmpty: true}}	system_id IS NULL
$notEmpty	bool	true only; field has no empty value	{system_id: {$notEmpty: true}}	system_id<>''
$isNull	bool	true only; field value is NULL	{system_id: {$isNull: true}}	system_id IS NULL
$notNull	bool	true only; field value is not NULL	{system_id: {$notNull: true}}	system_id IS NOT NULL
$equals	any	field value equals comparison value	{system_id: {$equals: "7"}} **shorthand**: {system_id: "7"}	system_id='7'
$notEquals	any	field value not equal to comparison value	{system_id: {$notEquals: "7"}}	system_id<>'7'

$endswith	string	field value ends with comparison value	{system_id: {$endswith: "7"}}	system_id LIKE '%7'
$startswith	string	field value starts with comparison value	{system_id: {$startswith: "7"}}	system_id LIKE '7%'
$contains	string	fiel value contains comparison value	{system_id: {$contains: "7"}}	system_id LIKE '%7%'
$like	string	field value matches the pattern	{system_id: {$like: "%7_7%"}}	system_id LIKE '%7_7%'
$in	any[]	field value equals one of the comparison values	{system_id: {$in: ["7","8","9"]}}	system_id IN ('7','8','9')
$notIn	any[]	field value not equal to any of the comparison values	{system_id: {$notIn: ["7","8","9"]}}	system_id NOT IN ('7','8','9')
$between	any[2]	field value lies between the two comparison values	{system_id: {$between: ["7","9"]}}	system_id BETWEEN '7' AND '9'
$notBetween	any[2]	field value not lies between the two comparison values	{system_id: {$notBetween: ["7","9"]}}	system_id NOT BETWEEN '7' AND '9'
$gt	any	field value is greater than comparison value	{system_id: {$gt: "7"}}	system_id > '7'
$gte	any	field value is greater than or equal to comparison value	{system_id: {$gte: "7"}}	system_id >= '7'
$lt	any	field value is less than comparison value	{system_id: {$lt: "7"}}	system_id < '7'
$lte	any	field value is less than or equal to comparison value	{system_id: {$lte: "7"}}	system_id <= '7'
$equals_field	string	field value is equal to specified other field	{system_last_change_at: {$equals_field: "system_created_at"} }	system_last_change_at = system_created_at
$notEquals_field	string	field value not equal to specified other field	{system_last_change_at: {$notEquals_field: "system_created_at"} }	system_last_change_at <> system_created_at
$gt_field	string	field value is greater than specified other field	{system_last_change_at: {$gt_field: "system_created_at"} }	system_last_change_at > system_created_at
$gte_field	string	field value is greater than or equal to specified other field	{system_last_change_at: {$gte_field: "system_created_at"} }	system_last_change_at >= system_created_at
$lt_field	string	field value is less than specified other field	{system_last_change_at: {$lt_field: "system_created_at"} }	system_last_change_at < system_created_at
$lte_field	string	field value is less than or equal to specified other field	{system_last_change_at: {$lte_field: "system_created_at"} }	system_last_change_at <= system_created_at

$between_field	string[2]	field value lies between the values of two other specified fields	{system_last_change_at: {$between_field: ["system_created_at", "system_delete_at"]}}	system_last_change_at BETWEEN system_created_at AND system_delete_at
$notBetween_field	string[2]	field value not lies between the values of two other specified fields	{system_last_change_at: {$notBetween_field: ["system_created_at", "system_delete_at"]}}	system_last_change_at NOT BETWEEN system_created_at AND system_delete_at
$date	int	Date/Time only: day of month equals comparison value	{system_created_at: {$date: 1}}	DAY(system_created_at) = 1
$day	int	Date/Time only: weekday equals comparison value	{system_created_at: {$day: 1}}	DATEPART(dw, system_created_at) = 1
$year	int	Date/Time only: year equals comparison value	{system_created_at: {$year: 2000}}	YEAR(system_created_at)=2000
$month	int	Date/Time only: month equals comparison value	{system_created_at: {$month: 12}}	MONTH(system_created_at)=12
$minutes	int	Date/Time only: minute equals comparison value	{system_created_at: {$minutes: 59}}	DATEPART(mi, system_created_at) = 59
$hours	int	Date/Time only: hour equals comparison value	{system_created_at: {$hours: 18}}	DATEPART(hh, system_created_at) = 18

ALL

Default filter. This filter loads all records without restrictions.
Example:
`{$all: true}` ? `select * from tab where 1=1`
only true is allowed

EMPTY

This filter creates an empty query. Usually not required.
Example:
`{$empty: true}` ? `select * from tab where 1=0`
only true is allowed

NOT

This filter negates the included filter
Example:
`{$not: {system_id: "7"}}` ? `select * from tab where NOT (sytem_id='7')`

AND

This filter connects the contained filters to a logical conjunction (AND)

Example:
`{$and: [{vertragsnummer: {$contains: {"123"}}, {vertragsnummer: {$notEquals: "100123100"}}, ...]}` ? `select * from tab where vertragsnummer like '%123%' AND vertragsnummer <> '100123100' AND ...`
shorthand: `{vertragsnummer: {$contains: {"123"}, vertragsnummer: {$notEquals: "100123100"}, ...}`

OR

This filter connects the contained filters to a logical disjunction (OR)
Example:
`{$or: [{vertragsnummer: {$contains: {"123"}}, {vertragsnummer: {$notEquals: "100123100"}}, ...]}` ? `select * from tab where vertragsnummer like '%123%' OR vertragsnummer <> '100123100' OR ...`

Combination of Shorthands

There are shorthands (see above) for both the $and filter and the $equals comparator, which can be combined.
Example:
`{name: "Albrecht", vorname: "Klaus"}` ? `select * from tab where name='Albrecht' AND vorname='Klaus'`

Nesting

$and, $or and $not filters can be nested in any way.
Example:
`{$or: [{$and: [filter1, filter2], $not: {$or: [filter3, filter4]}]}` ? `select * from tab where (filter1 AND filter2) OR NOT(filter3 OR filter4)`