BILINFO CASE PLUGIN

Third party integration document

Abstract

Bilinfo Case Plugin enables Dealers to manage financing and insurance offers on behalf of the Customer within Bilinfo.nets “Sager” system. This document describes the prerequisites and details for building a Case Plugin integration into Bilinfo.

BilInfo

bilinfo@bilinfo.dk

Version history

Version	Date	Authors	Comments
0.1.0	12/12/2016	Cosmin Constantin Lazar Kim Jørgensen	First draft version
0.1.2	09/01/2017	Lisbeth Storgaard	Added case flow
0.1.3	16/01/2017	Cosmin Lazar	Use test client in OpenID examples
0.1.4	27/02/2017	Ole Munk Lauritsen	Updated input models
0.1.5	28/02/2017	Ole Munk Lauritsen	Updated input models
0.1.6	14/03/2017	Cosmin Constantin Lazar	Describe application status reporting api, brands and products api, Oauth2 appendix, update plugin messages
0.1.7	16/03/2017	Ole Munk Lauritsen	Added deliveryCost, licensePlate and color to Car model
0.1.8	24/03/2017	Cosmin Constantin Lazar	Add phone to PrivateCustomer, CoApplicant, and BusinessCustomer
0.1.9	27/03/2017	Cosmin Constantin Lazar	Add `vin` and `kmWarranty` to Car
0.1.10	10/04/2017	Cosmin Constantin Lazar	Add description of finance specification message
0.1.11	25/04/2017	Cosmin Constantin Lazar	Add description for Finance Offer on Marketplaces
0.1.12	23/05/2017	Cosmin Constantin Lazar	Add api description for accessing the sales contract
0.1.13	08/06/2017	Cosmin Constantin Lazar	Add specification for `discount` and `discountVat`
0.1.14	10/07/2017	Cosmin Constantin Lazar	Add specification for `driverLicenseNumber`, `timingBeltReplaced`, and `serviceBook`
0.1.15	14/07/2017	Cosmin Constantin Lazar	Add specification for `decisionMaker` and `businessUser1`
0.1.16	27/07/2017	Cosmin Constantin Lazar	Add interaction diagrams and restructure sub-chapters
0.1.17	28/07/2017	Cosmin Constantin Lazar	Update disclaimer and remove watermark
0.1.18	05/09/2017	Cosmin Constantin Lazar	Update QA base path for Bilinfo service
0.1.19	06/09/2017	Cosmin Constantin Lazar	Specify verb and format for Calculation Matrix
0.1.20	06/09/2017	Cosmin Constantin Lazar	Rename ‘case.data’ message to ‘data.case’
0.1.21	06/09/2017	Cosmin Constantin Lazar	Add `toldAndSkatVariant` to car data
1.22.0	06/09/2017	Cosmin Constantin Lazar	Add code field to dealer mounted extra equipment.
1.22.1	15/09/2017	Cosmin Constantin Lazar	Remove companyName and carPrice from Matrix calculation input
1.23.0	21/09/2017	Cosmin Constantin Lazar	Add `taxableAmount` to car sales price
1.24.0	01/11/2017	Jacob Emborg Sønderskov Lisbeth Storgaard	Reworked document structure

1.25.0	07/11/2017	Jacob Emborg Sønderskov	Added `ownerTaxPerYear` to Car data
2.0.0	15/11/2017	Henrik Thomsen	Changed `finance.update` to `case.update`
2.1.0	24/11/2017	Henrik Thomsen	Changed get brands and products to include services provided
2.1.1	29/11/2017	Henrik Thomsen	Changed this document from only supporting finance, to being a finance and insurance document (text change)
2.1.2	29/11/2017	Henrik Thomsen	Add `modelCatalogueId` to Car
2.2.0	29/11/2017	Henrik Thomsen	Changed Bilinfo Services
2.2.1	30/11/2017	Lisbeth Storgaard	Updated user journey
2.2.2	8/1/2018	Jacob Emborg Sønderskov	Move FOOP into Plugin Server-Side as optional step.
2.2.3	17/1/2018	Jacob Emborg Sønderskov	Added description to `downPayment` being with VAT.
2.2.4	24/1/2018	Jacob Emborg Sønderskov	Add Versioning and Deprecation Policy chapter
2.2.5	31/01/2018	Henrik Thomsen	Added 4 extra Cartypes
2.2.6	13/02/2018	Jacob Emborg Sønderskov	Update Disclaimer and Versioning and Deprecation chapters.
2.2.7	07/03/2018	Henrik Thomsen	Disallow multiple servicetypes in the same brand
2.2.8	12/04/2018	Jacob Emborg Sønderskov	Change `logoUrl` to 133x35
2.2.9	12/04/2018	Jacob Emborg Sønderskov	Add “Migrating from Case Plugin Architecture v1 to v2” chapter
2.2.10	25/04/2018	Jacob Emborg Sønderskov	Add `timingBeltReplacedAtMileage` and `timingBeltReplacedAtDate` in TradeInCar type.
2.2.11	04/05/2018	Jacob Emborg Sønderskov	Add `companyLogoUrl` field to calculation matrix endpoint.
2.2.12	18/05/2018	Jacob Emborg Sønderskov	Amended Versioning and Deprecation Policy chapter
2.2.13	31/05/2018	Dianna Kristensen	Add `effect` field to Car
2.3.0	31/05/2018	Dianna Kristensen	Change QA urls from QA1 to QA01 ahead of fall 2018 update. NB: Will not impact production.
2.3.1	31/05/2018	Jacob Emborg Sønderskov	Add change policy associated with GDPR to Versioning and Deprecation Policy chapter
2.3.2	20/06/2018	Lisbeth Storgaard Jacob Emborg Sønderskov	Add Finance Offer on Platforms user journey
2.3.3	14/08/2018	Jacob Emborg Sønderskov	Add Unknown value to the FuelType enum
2.3.4	07/09/2018	Jacob Emborg Sønderskov	Add Content-Security-Policy and X-FRAME-OPTIONS recommendations

2.4.0	10/09/2018	Jacob Emborg Sønderskov	Split User Journeys, Finance Offer On Platforms, Bilinfo Shared Services and Bilinfo Auth Services into separate documents
2.5.0	07/11/2018	Jacob Emborg Sønderskov	Add CashPriceInclVat Case Specification Message (Car)
2.6.0	12/11/2018	Henrik Thomsen	Add new fields to Car
2.7.0	12/12/2018	Jacob Emborg Sønderskov	Add extended service book field, ServiceBookEnum to Car and TradeInCar
2.7.1	29/03/2019	Lars Tabro Sørensen	Add PreApproved status to StatusApi states
3.0.0	17/12/2025	Wojciech Janas	Rework of business users structure – a new BusinessUsers object is now related with the Case object, not Customer

Contents

A. Disclaimer

Information presented here might be altered by the Bilinfo team from time to time. Inconsistencies across the document are to be expected and they will be addressed in updates.

Any update will be specified in Version history.

GDPR compliance

Note, that the Plugin produced using this document may be subject to review by Bilinfo before it will be made available via Bilinfo.net to ensure proper security measures have been employed.

Data and system integrity

Abuse of the system is forbidden in any regard. If you find a security issue or exploitation outside the original intent of the system, you are expected to report the exploit or bug to the Bilinfo team.

B. Versioning and Deprecation Policy

Versioning in Bilinfo Services is essential to achieving our vision behind Partner integrations in Bilinfo. Using the versioning principles described below will allow for your Bilinfo integrations to remain stable and fully functional as the Bilinfo business continues to evolve and mature.

New Versions of the Bilinfo Services

The versioning principles employed in Bilinfo Services largely follow that of the *Semantic Versioning Specification*[1]. The Semantic Versioning Specification, in short, specifies a version increment based on the backwards compatibility of the API or Web Service. A summary of the specification can be seen in the following Listing B-1:

> Given a version number `MAJOR.MINOR.PATCH`, increment the:
>
> - `MAJOR` version when you make incompatible API changes,
> - `MINOR` version when you add functionality in a backwards-compatible manner, and
> - `PATCH` version when you make backwards-compatible bug fixes.
>
> Additional labels for pre-release and build metadata are available as extensions to the `MAJOR.MINOR.PATCH` format.

Listing B-1: Semantic Versioning Specification 2.0.0 summary

The types of changes that are minor version changes and backward compatible are:

- Adding a new method (`GET`, `POST` etc.) to an API
- Adding a new property to the method response payload
- Adding a new *non-personal data*[2] property to an `iframe` communication

The types of changes that are major version changes and not backward compatible are:

- Removing existing method (`GET`, `POST` etc.) from an API
- Renaming existing method path
- Changing request body or query string for existing method
- Changing method response structure and/or property names
- Removing a property from an `iframe` communication
- Renaming a property from an `iframe` communication
- Renaming a message in an `iframe` communication
- Adding a new *personal data*[2] property to an `iframe` communication

[1] https://semver.org

[2] Personal data as defined in Regulation (EU) 2016/679 of 27 April 2016 (GDPR) and the Danish Data Protection Act.

In general, new major versions of Bilinfo Services will only be introduced, when existing interfaces do not allow for further evolution and improving of our Partner integrations without modification. Due to the need for Partner action, major versions are used as a last resort and are as such very rare. Minor version updates will require no Partner action.

Updating your Bilinfo integration

Updating your Bilinfo integration to support a new major version is non-optional as the existing integration paradigm is fundamentally changed. It is as such not possible to opt out without risking major problems with your Bilinfo integration. Minor versions, however, are fully optional, but may contain new fields, which may enrich the experience and value of your Bilinfo integration.

To assist Partners in upgrading their Bilinfo integration with minimal efforts, each major version will be associated with *migration chapters* added to this document. Minor version changes are specified primarily in the Version History and are subject to the reader to adhere to the changes.

Deprecation Policy and Supported Versions

Bilinfo Services will support older versions for a grace period appropriate to the contractual obligations. After that time, integrations based on older versions may no longer work or experience severe operational issues.

C. Migrating from Case Plugin Architecture v1 to v2

The following guide describes the main changes going from version 1 to version 2 of the Case Plugin Architecture and how to migrate your version 1 Case Plugin to work in the new architecture.

Keep in mind that the Case Plugin Architecture is a dynamic solution, which – based on our Partners need – will be updated continuously with new fields and data from Bilinfo.net, but rarely require specific Partner actions.

Version 2 of the Case Plugin Architecture enables the inclusion of additional services in your Case Plugin integration which previously has been focused on Financing, i.e. Loan and Leasing. With version 2 it is possible to define the *delivered service*, such as loan, leasing, insurance and – in time – more, that the Partner enables the dealer and end customer to consume via the Bilinfo.net platform.

Major version changes for version 2 are found in:

- Brands & Products API
    - Add delivered services to Product. See Bilinfo Brands and Products (Integration) document and more specifically the Product type.

- Application Status
    - Change in reporting scheme to a dictionary of delivered services. See the Request section.

- Plugin Client-Side
    - Change name of `finance.update` message to `case.update`. See Data exchange interaction section.
    - Change version from "1.x.x" to "2.x.x." for all application messages. See Messaging and more specifically Header.

Additional minor version changes for version 2 can be found in:

- Plugin Client-Side
    - Addition of `modelCatalogueId` to Car type. This requires access to Bilinfo's *ModelCatalogue service* which is outside the scope of this document.
    - Addition of new types to CarType type.

D. Migrating from Case Plugin Architecture v2 to v3

Major version changes for version 3 are found in the Case specification message.

Change version from "2.x.x" to "3.x.x." for all application messages. See Messaging and more specifically Header.

A new field `businessUsers` was added to the message, which reflects the change in objects relation – a business user ("Bruger") is related with the case, not with the customer. It has been therefore removed from the `customer` object, and now will contain both business users ("Bruger 1" & "Bruger 2").

The field `businessUsers` will remain `null` for private customers.

1. Introduction

1.1 Purpose and Scope

This document describes the technical solution for building a Case Plugin integration into Bilinfo.net via the Case Plugin Architecture.

For Finance Offer On Platforms please refer to the Bilinfo Finance Offer On Platforms (integration) document.

1.2 References

Documents relevant to the reading of this document are listed here. Links and other external resources accessible via the internet are referenced via footnotes relative to the term or technology. You should have access to every document mentioned in this list. If that is not the case, contact Bilinfo.

Table 1-1: Document references

Document name	Description
Bilinfo Case Plugin (Integration)	Bilinfo Case Plugin enables Dealers to manage financing and insurance offers on behalf of the Customer within Bilinfo.nets "Sager" system. This document describes the prerequisites and details for building a Case Plugin integration into Bilinfo.
Bilinfo Case Plugin (User journey)	Bilinfo Case Plugin enables Dealers to manage financing and insurance offers on behalf of the Customer within Bilinfo.nets "Sager" system. This document describes the User Journey of the product result of a Bilinfo Case Plugin integration.
Bilinfo Auth Services (Integration)	Bilinfo Auth Services encompass Single-Sign On and OAuth 2.0 mechanisms that must be used when integrating Bilinfo. This document describes the prerequisites and details of integrating into Bilinfo Auth Services.
Bilinfo Shared Services (Integration)	Bilinfo Shared Services encompass Dealer Lookup and User Lookup Services that may be used in building a Bilinfo integration. This document describes the prerequisites and details of integrating into Bilinfo Shared Services.
Bilinfo Brands and Products (Integration)	Brands and Products are a required element of both a Case Plugin and Finance Offer On Platforms integration. This document describes the prerequisites and details for building a Brands and Products API for integrating into Bilinfo.
Bilinfo Finance Offer On Platforms (User journey)	Bilinfo Finance Offer On Platforms enables Dealers to select Finance Offers for Cars to be shown on Bilbasen, DBA and Dealer CMS sites. This document describes the User Journey of the product result of a Bilinfo Finance Offer On Platforms integration.
Bilinfo Finance Offer On Platforms (Integration)	Bilinfo Finance Offer On Platforms enables Dealers to select Finance Offers for Cars to be shown on Bilbasen, DBA and Dealer CMS sites. This document describes the prerequisites and details for building a Finance Offer On Platforms integration into Bilinfo.

1.3 Definitions and acronyms

The definitions and acronyms defined in Table 1-2 cover frequently used concepts, terms and acronyms used throughout this document. It is suggested that the reader acquaints him- or herself with the key concepts and refer to this list, when in doubt.

Table 1-2: Definitions and acronyms

Term/acronym	**Definition**
Bilinfo Services	Includes – but not limited to – Bilinfo Auth Services, Bilinfo Shared Services, Bilinfo Finance Offer On Platform integrations and Bilinfo Case Plugin integrations.
Partner	Synonym for the integrating party.
Case Plugin Architecture	System enabling the integration of external third party Case Plugins in the Bilinfo.net context.
Bilinfo Shared Services	Aggregate term for API and Data Services provided by Bilinfo to external Partners and integrations.
FOOP	Acronym for Finance Offer On Platforms.
Finance Offer On Platforms	Feature which enables displaying Finance Offers on Bilbasen through Bilinfo.
Platforms	Shorthand of the platforms with support for FOOP, i.e. DBA, Bilbasen and CMS services.
Plugin (server-side/client-side)	Third party system developed for viewing within the context of Bilinfo. Consists of a server-side and a client-side, respectively referred to as a *Plugin server-side* and *Plugin client-side*.
SSO	Single Sign-On. Authentication protocol employed by Bilinfo Host. Must be adhered to by the Plugin.
Case	A Case in Bilinfo consists minimum of a Customer and a car, but can also include extra equipment and have a trade-in car added. On each case you also find a calculation of the car price. The case is the collection of data involved in selling a car in Bilinfo.
Brand	Name of partner or an OEM brand.
Product	Campaigns, loan-, leasing-, and insurance products, bundled or All inclusive products.
Plugin Implementer	A Finance or Insurance company or any other partner that is implementing a plugin for Bilinfo.net.

2. System overview

The following chapter will introduce the Plugin context and the main actors within the Case Plugin Architecture with the intent of aiding the reader in understanding the implementation specifics to come in the subsequent chapters.

2.1 System context

A *Case Plugin* (or simply a *Plugin*) is an extension point of Bilinfo.net that allows third parties to seamlessly integrate new functionality. The interaction between the *Host* (Bilinfo.net) and the *Plugin* (third party), as seen in Figure 2-1, is designed to offer the best experience to the end-user via real-time updates and ease of use.

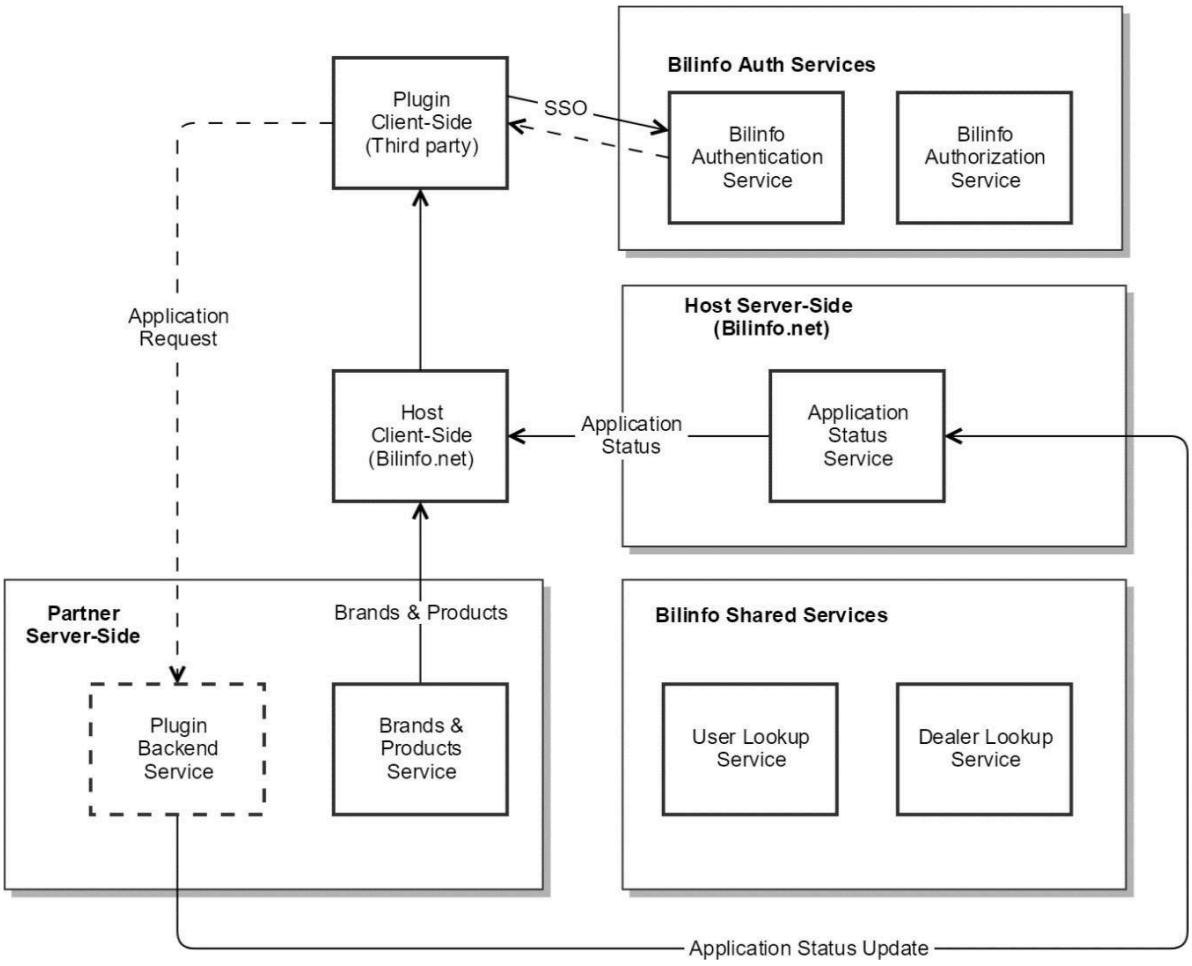

Figure 2-1: Case Plugin Architecture context diagram

There are several actors that tie together to make the integration work. These are largely split into Client-Side and Server-Side components within different contexts as seen in Figure 2-1. Note that the dotted boxes and interactions, seen in the Figure 2-1 context diagram, concern communications outside of the scope of this document as this involves the internal processing done by the integrating party.

2.1.1 Plugin Server-Side

The *Plugin Server-Side* represents a set of services needed by the Host Server-Side in order to load the proper third party Plugin Client-Side for a specific integration need. By querying said services, Bilinfo.net will be able to easily integrate different Brands and Products into the Case Plugin context.

2.1.2 Plugin Backend Service

The *Plugin Backend* has a number of functions within the Plugin context, but is otherwise undefined in terms of its implementation as it is outside the scope of this document. The functions the Plugin Backend must serve is:

1. Provide the JavaScript, CSS and HTML files constituting the Plugin Client-Side.
2. Handle internal processing of an Application based on business needs.
3. Report status updates to the Host Server-Side

Note, the Plugin Web Server and Plugin Server-Side are intentionally split into two distinct parts, but may exist within the same deployment-wise context as shown by the "Partner Server-Side" grouping in Figure 2-1.

2.1.3 Plugin Client-Side

The *Plugin Client-Side* represents the UI with which the Dealer interacts when preparing an Application for a specific Brand and Product. It communicates both with Bilinfo via the Host Client-Side and the Plugin Implementer's backend via the Plugin Backend. Once an Application is ready for processing it is sent directly to the Plugin Backend, processed internally by the Plugin Implementer with an Application Status being forwarded to the Host Server-Side, and ultimately updated within the Plugin Client-Side via a message from the Host Client-Side.

2.1.4 Host Server-Side

The *Host Server-Side* constitutes a set of services used by the Plugin Server-Side and Plugin Web Server. The available services are described in further detail in Chapter 6.

In addition to providing services and API's it also serves up the Host Client-Side.

2.1.5 Host Client-Side

The *Host Client-Side* constitutes the Bilinfo.net Case Plugin Architecture, which loads the Plugin Client-Side into an iframe from an URL provided by the Plugin Server-Side. The resource pointed to the URL is not (necessarily) served up by the Partner Server-Side, but is rather served by a Plugin Backend.

Additionally, the Host Client-Side negotiates the communication protocol to be used between it and the Plugin Client-Side. The communication specifics regarding the *handshake* and client-side communication is described in further detail in Chapter 5.

2.1.6 Bilinfo Shared Services

The *Bilinfo Shared Services* are described in the *Bilinfo Shared Services* document. See 1.2 References.

2.2 System introduction

The Plugin is hosted within the Bilinfo.net Case feature as seen in Figure 2-2. The plugins therefore become an extension of the Case with bi-directional data exchange capabilities with Bilinfo.net.

Additional information regarding the flow and user interactions can be found in the *Bilinfo Case Plugin (User journey)* document. See 1.2 References.

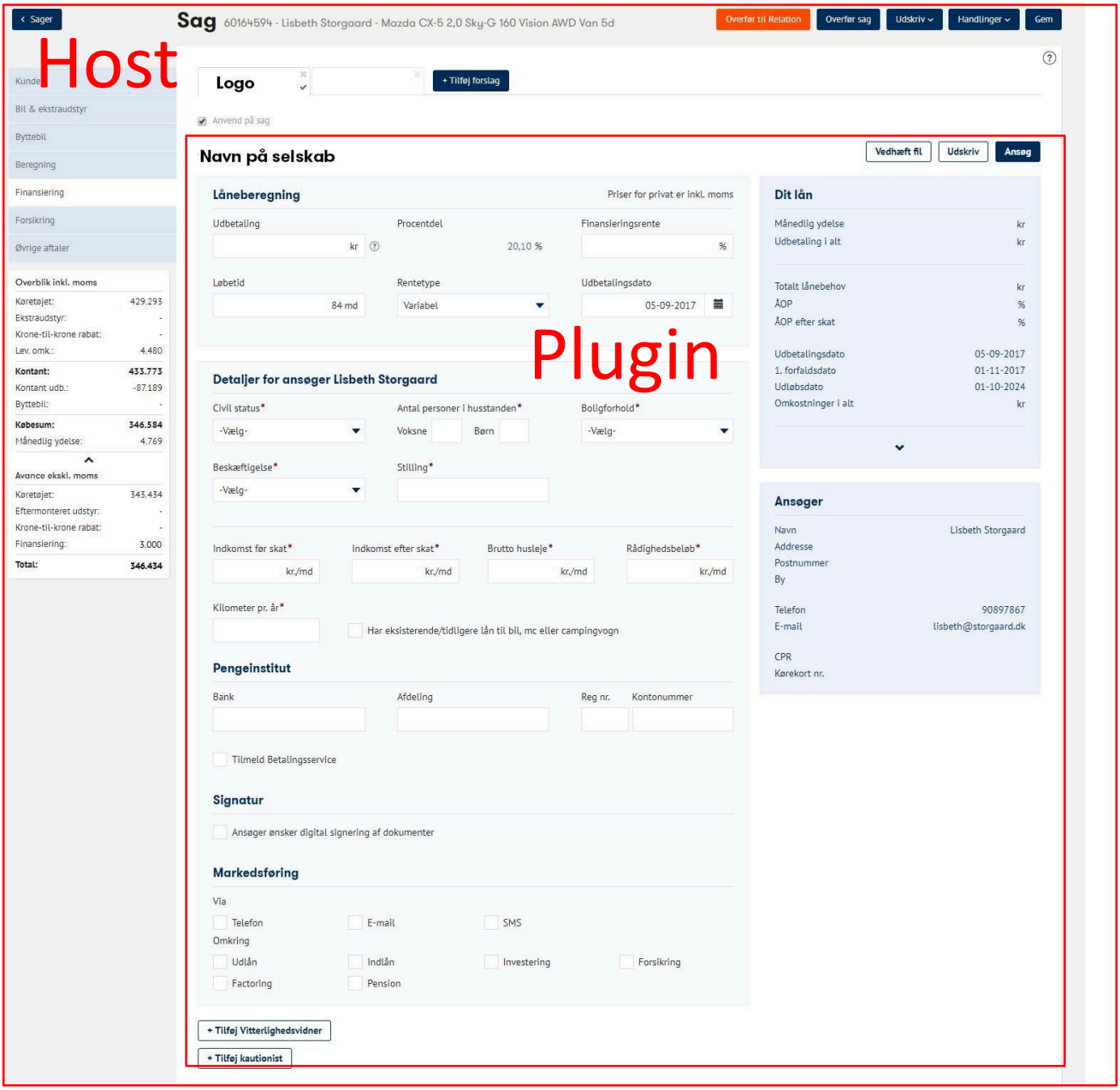

Figure 2-2: Case plugin

The system is intended to support entry of Finance and Insurance details pertaining to the specific Application (Brand and Product) the Dealer has selected and work within the confines of Bilinfo.net. As such, any necessary information needed by the Plugin Implementer, must be defined and communicated conforming to internal information requirements.

It is recommended that the implementing party incorporates a user experience similar to that of Bilinfo.net in general. However, understandably, the visual feel in terms of applied color scheme may be different to that of Bilinfo.net to better indicate the Plugin Implementer with which the Application is made.

3. Prerequisites

This chapter describes the preliminary information and tasks you should get under way before starting any development on the Plugin Server-Side and the Plugin as a whole. These tasks are dependent on external parties and may take some time to complete.

3.1 Security

All Bilinfo Shared Services use SSL/TLS (HTTPS) and OAuth 2.0 for Authorization. Additionally, the Case Plugin must authenticate itself with Bilinfo.net via Single Sign-On.

3.2 Separate environments

Two instances of the Plugin Server-Side should be available, one for Production and one for QA.

3.3 Onboarding

The integrating party must contact Bilinfo.net to start the *onboarding process*. In onboarding, infrastructural information pertaining to the integration is exchanged.

For the Case Plugin integration a number of elements have to be aligned. The onboarding activities include:

- Registration of SSO and OAuth 2.0 integrations covered in the Bilinfo Auth Services (Integration) document.
- Exchange URI's and paths for Partner Brands & Products API in Bilinfo.net
- Retrieve `plugin_implementer_identifier` URI for Application Status API callback

4. Plugin Server-Side

This chapter will cover the Plugin Server-Side components that must be implemented in order to enable integration of a third party Plugin into Bilinfo.net.

4.1 Brands & Products API

The *Brands & Products API* is the primary component of the Plugin Server-Side and assists the Bilinfo.net site in loading the Plugin correctly. The API is used to present the User with a list of Brands and Products he can use to finance or insure the current vehicle on the Case. Once the User chooses a Brand and Product combination, a *Plugin URL* is used to load the Plugin Client-Side within an `iframe`.

The initial create new offer interaction described above can be seen in Figure 4-1. Once created Bilinfo.net saves the Plugin URL of the offer and will reuse it for future reloads of it. This subsequent load can be seen in Figure 4-2.

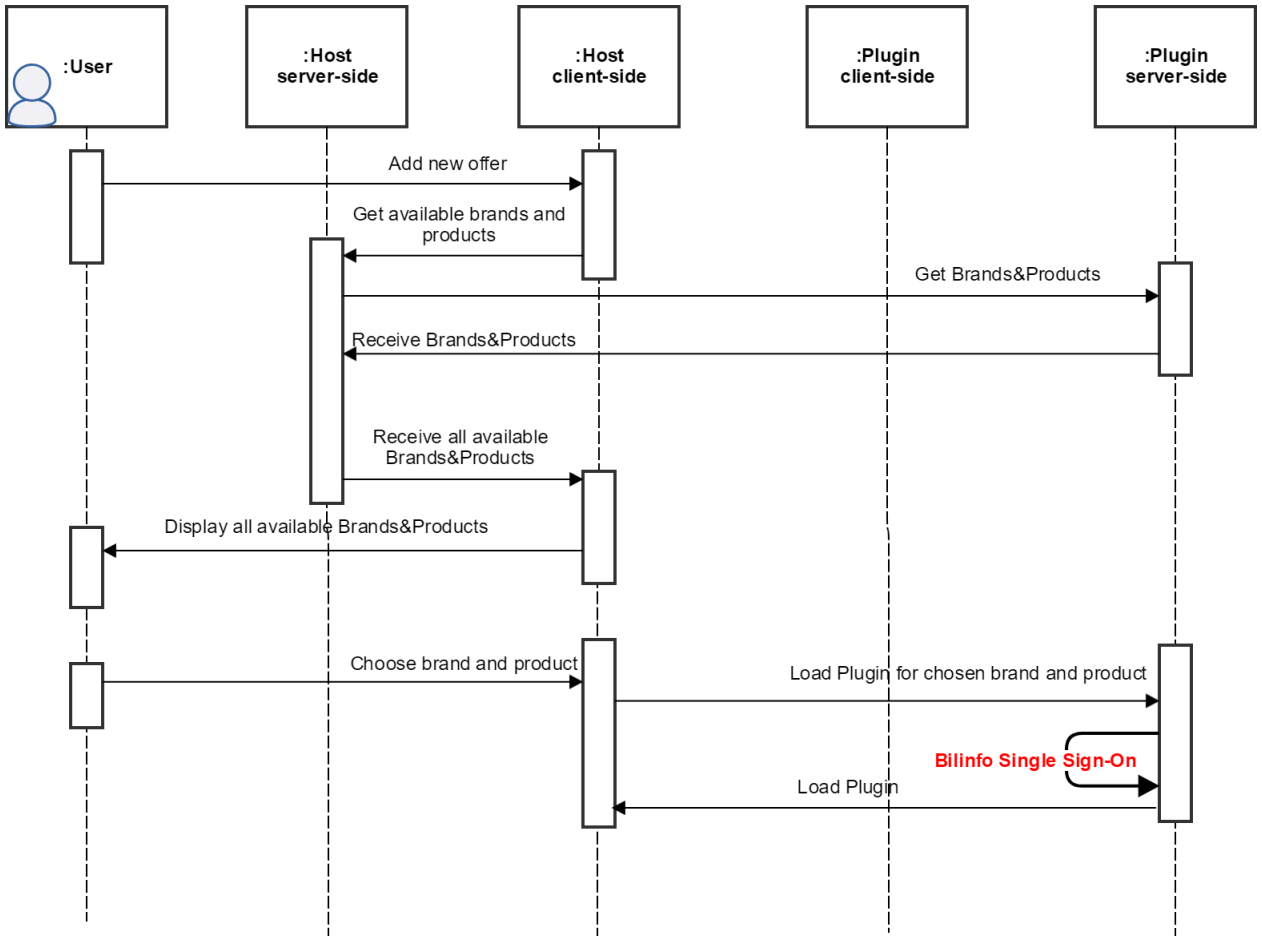

Figure 4-1: Create new offer interaction

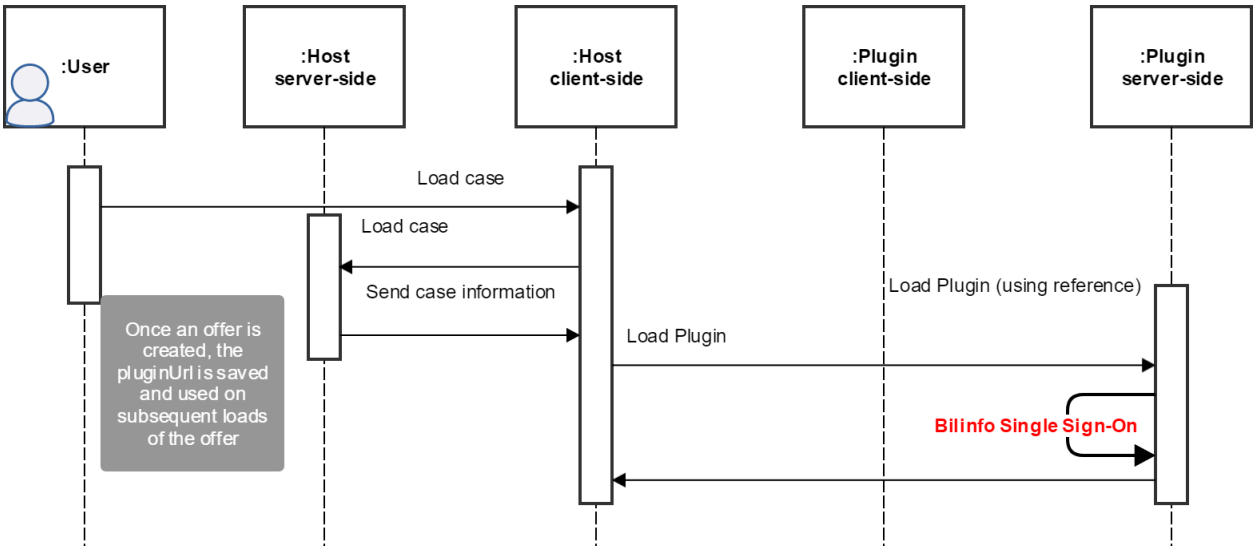

Figure 4-2: Load existing offer interaction

Note, that in both Figure 4-1 and Figure 4-2 a Single Sign-On (SSO) is negotiated with the Bilinfo Authentication Server. This step is described in further detail in chapter 5.

Further information regarding the Brands & Products specification can be found in the Bilinfo Brands and Products (Integration) found in 1.2 References.

5. Plugin Client-Side

The *Plugin Client-Side* refers to the web-based User Interface (UI) produced by the HTML, JavaScript and CSS elements provided by the Plugin Backend. It is displayed on the Bilinfo.net site via an iframe.

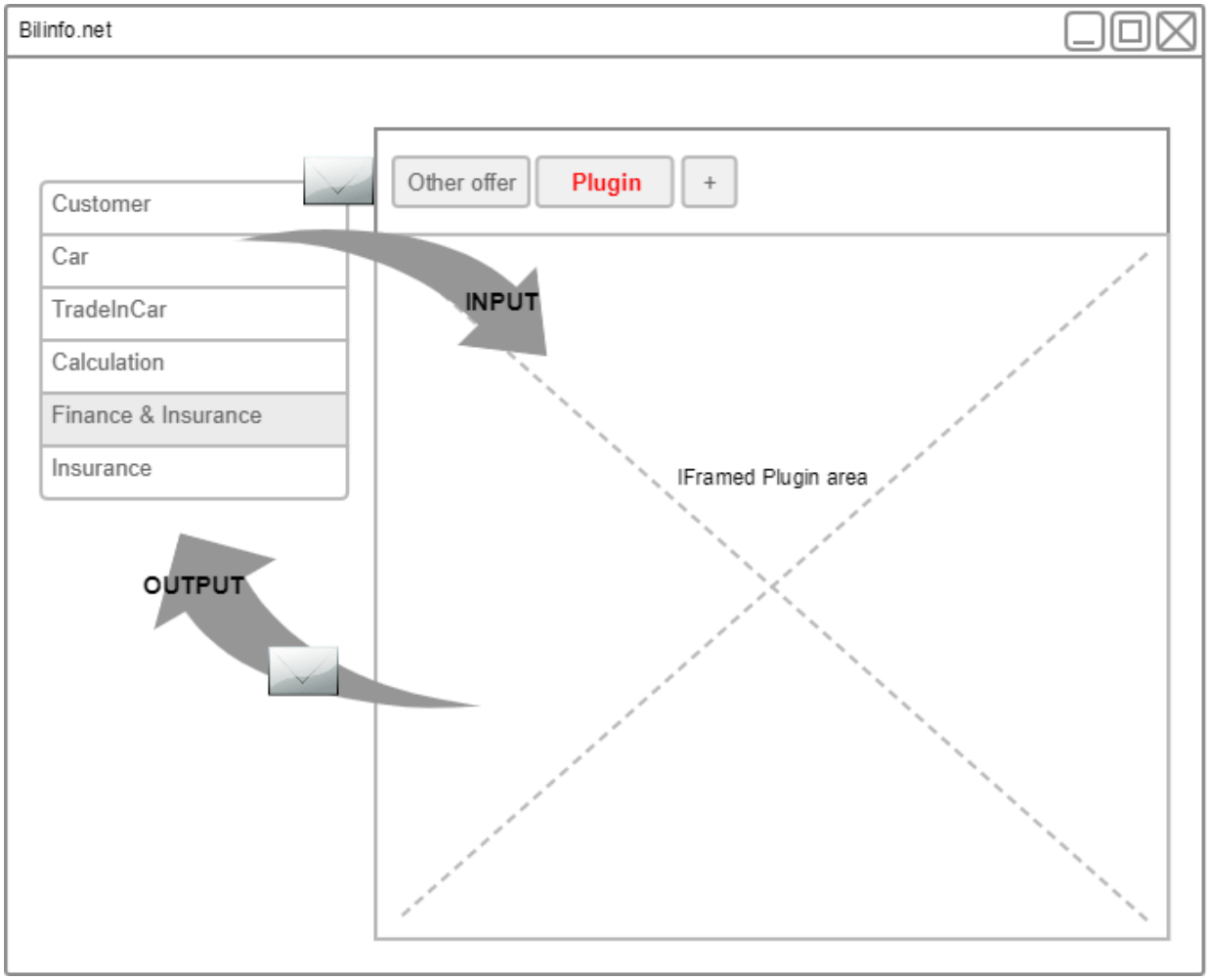

Figure 5-1: Case Plugin

This chapter covers the interactions available to the Plugin Client-Side within the Host Client-Side context. Each section found in this chapter pertains to a portion of the implementation needed to interact with Bilinfo. Any interaction with the Plugin Backend is out of scope of this document.

5.1 Security

Before the Plugin is shown to the user, the Plugin must negotiate Single Sign-On with the Bilinfo Authentication Server. The Plugin must not prompt the user for signing in, it should instead trust Bilinfo to confirm the logged user identity. The user identity is established using the OpenID Connect protocol. Therefore, upon starting up, the Plugin should only grant a session after it has successfully established the user identity.

Figure 5-2 shows how the authentication using SSO specified in Bilinfo Auth Services (Integration) document is performed within the context of Bilinfo. Out of the box, the Bilinfo Authentication Service supports Authorization Code and Implicit flow. The Plugin developer can choose the flow it desires to implement; however, since it offers a higher degree of protection, Authorization Code is preferred.

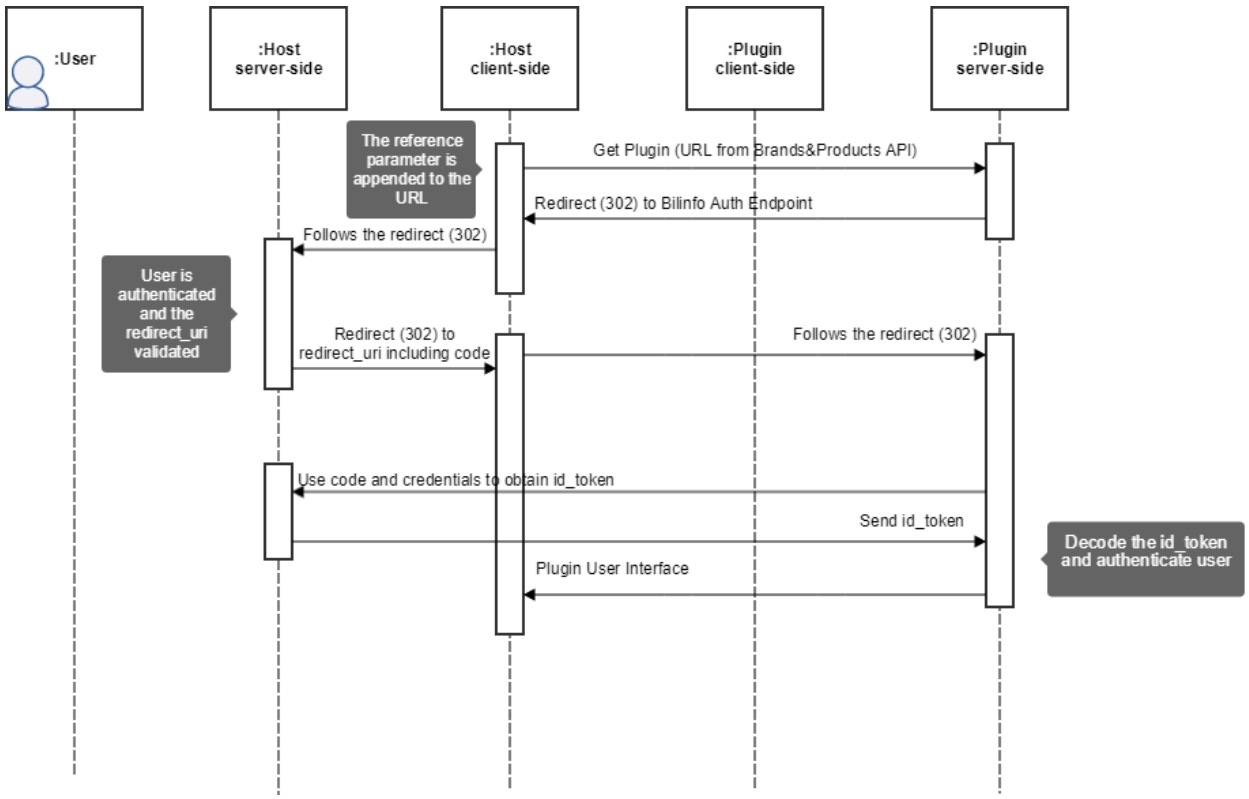

Figure 5-2: Single Sign-On Authorization Code flow

Explaining the OpenID Connect protocol is outside the bounds of this document. However, meaningful information about the supported flows and test credentials may be found in the Bilinfo Auth Services (Integration) document.

The protocol revolves around a third party obtaining an `id_token` and optionally an `access_token`. The `id_token` is a self-standing JSON Web Token describing the identity of a user (claims). Furthermore, the `access_token` can be used later on to obtain more detailed information about the user.

The `id_token` will contain claims about the current user in accordance with the scopes requested by the client. Table 5-1 shows a list of accessible scopes along with their associated claims.

Table 5-1: id_token claims

Scope	Claims	Included in id_token	Description
`openid`	`sub`	yes	User identifier (e.g.: clazar@ebay.com)
`profile`	`name`	yes	User full name (e.g.: Cosmin Constantin Lazar)
	`userId`	yes	User id in Bilinfo.net (e.g.: 1253)
`dealer`	`dealerId`	yes	Dealer Id in Bilinfo.net (e.g.: 42304136-FDCA-DF11-9E98-0025B3E6B7D8)
	`bilinfoDealerId`	yes	Dealer Id in Bilinfo Program (e.g.: 100)
`email`	`email`	yes	User email (e.g.: clazar@ebay.com)

Communication with the Bilinfo Authentication and Authorization Server, Bilinfo Shared Services, or between the Plugin and its server-side counterparts shall always take place over a secure connection (TLS).

Please note that most programming languages have libraries that you can use to access and validate tokens. A non-exhaustive list of libraries for various programming languages may be found at the OpenID Connect developers site[3]. Furthermore, if the library supports automatic configuration, you can use the OpenID configuration endpoint https://www.bilinfo.net/oauth/.well-known/openid-configuration.

5.1.1 Content Security Policy (CSP)

The Plugin Client-Side is loaded within an iframe context. A Content-Security-Policy (CSP) header is set up in Bilinfo.net, which will reject all non-https communication. As such, HTTPS is mandatory when providing Case Plugin static files and API calls.

It is strongly recommended that the Case Plugin uses CSP headers in turn to best protect their iframe solution from being used in unexpected ways.

The Content-Security-Policy **frame-ancestors**[4] directive may be used to assert access only from

- https://www.bilinfo.net
- https://www.qa01.bilinfo.net

```
  Content-Security-Policy: frame-ancestors https://www.bilinfo.net https://
www.qa01.bilinfo.net;
```

5.1.2 X-Frame-Options

Given that the Content-Security-Policy header is primarily supported in modern versions of major browsers, we recommend also using the **X-Frame-Options**[5] header to enable controlling requests on older browsers.

[3] http://openid.net/developers/libraries/
[4] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/frame-ancestors
[5] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

Again, allow access from

- https://www.bilinfo.net
- https://www.qa01.bilinfo.net

```
X-Frame-Options: allow-from https://www.bilinfo.net https://www.qa01.bilinfo.net;
```

Please avoid using the meta tag when applying this, but rather put it in the HTTP response header.

5.2 Handling the `reference`

When loading the Plugin, the Host will generate a unique identifier and pass it to the Plugin by appending a "`reference`" query string parameter to the plugin load URL, e.g. https://example.com/plugin?reference=1234abcd).

The Plugin should save the provided reference and supply it in the header section of all the messages it sends to the Host. The Host will also supply the plugin reference in the header of all the messages it sends. It is recommended – though not necessary – that the client validates against it. How the reference is passed to the Host is described in further detail in the Messaging section.

The Host will reuse the reference on each subsequent re-initialization of the Plugin. This means that the identifier may be used for saving or reloading instance-specific user data or provide other meaningful instance-specific information back to the Plugin Backend. This is not mandatory, but may enrich the data available to the Plugin Implementer and improve the overall user experience using the Plugin.

Note: During the load sequence described in the Handshake section, the Plugin needs to authenticate the current user by using the Bilinfo Single Sign-On sequence. Performing this authentication will result in a series of redirects, through which it is important that the Plugin keep a hold of the reference that was supplied during these redirects.

5.3 User Interface

This section describes certain guidelines, which should be considered, when developing a Case Plugin to be hosted within the Bilinfo user interface.

The following guidelines may be considered:

- Screen size
    - Height is unbound
    - Width
        - When in full screen: 1184, afterwards 900
        - iPad landscape: 733
- Modal and errors: The Plugin is encouraged to use its own screen space to provide meaningful errors
- Printing
- Bookmarks

- Top navigation – will be disallowed – the plugin can only control its own iframe and will not be able to set top window navigation (aka will not be able to change the address of the current page, opening links in new tabs/windows will probably be allowed)

5.4 Messaging

The communication between the Host Client-Side and Plugin Client-Side is done using the HTML5 `Window.postMessage()` infrastructure as it allows safe cross-origin communication. You can read more about sending and receiving messages at Mozilla Developer Network (MDN) web docs[6]. The communication protocol will be versioned according to Semantic Versioning rules[7].

The two parties (Host and Plugin) will exchange data by posting messages to each other. The posted messages are required to follow the structure defined in the ApplicationMessage section. After sending an `ApplicationMessage`, the browser will wrap the message into a `MessageEvent` and deliver it to the intended destination.

5.4.1 MessageEvent

To enable safe data transfer from one frame to another, browsers automatically encapsulate all sent messages in a `MessageEvent`.

For the purpose of this integration, only the properties seen in Table 5-2 are required. An example can be seen in Listing 5-1.

Table 5-2: MessageEvent properties

Property name	Description
`Origin`	The scheme, hostname and port of the document sending the message
`Data`	The payload of the message, which in our case, is an `ApplicationMessage`

```
{
	"origin": "https://www.bilinfo.net/",
	"data": { ...ApplicationMessage... }
}
```

Listing 5-1: MessageEvent example

One should think of `MessageEvent` as a Transport Layer, enabling security validation and payload carry. The security part is described in further detail in the Security section.

5.4.2 ApplicationMessage

An *ApplicationMessage* contains the protocol specific data exchanged between the Host and Plugin. The message is composed of two mandatory parts: a *Header* and a Body.

[6] https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage
[7] http://semver.org/

Header

The *Header* is a mandatory part of a message containing general-purpose information. Table 5-3 shows the mandatory properties for an `ApplicationMessage`.

Table 5-3: ApplicationMessage header properties

Property name	Description
`Type`	A protocol-specific string identifying the data type contained in the body
`Version`	The protocol version the Plugin desires to use throughout the communication
`reference`	The reference query string parameter specified by the Host when loading the Plugin

Body

The Body is a mandatory part of a message containing actual application data, according to the type specified in the header. Listing 5-2 shows an example of a handshake request enveloped in a `MessageEvent`.

```
{
	"origin": "https://www.bilinfo.net/",
	"data": {
		"header":{
			"type": "handshake.request",
			"version": "1.0.0",
			"reference": "1234"
		},
		"body":{ }
	}
}
```

Listing 5-2: ApplicationMessage embedded in MessageEvent

5.5 Security

Security procedures need to be employed by the Host and Plugin when sending and receiving data, according to the following. This section will cover the security measures that must be adhered to when sending and receiving data to and from the Host.

5.5.1 Sending data

Sending data is achieved through a `Window.postMessage` call as seen in Listing 5-3. The properties can be seen in Table 5-4.

```
window.postMessage(message, targetOrigin, [transfer]);
```

Listing 5-3: Sending data example

Table 5-4: window.postMessage parameters

`window`	The window the message should be dispatched to. From the Host perspective, this is the Plugin iframe. From the Plugin perspective, this is the Host window (which can be obtained through `window.parent` property)
`message`	An `ApplicationMessage` – the browser will automatically wrap it in a `MessageEvent`
`targetOrigin`	To whom this message should be sent

	From the Host perspective, this is the URL used to load the Plugin From the Plugin perspective, this should always be https://www.bilinfo.net
`transfer`	Optional parameter, not used

It is paramount always to specify the `targetOrigin`, when posting a message, as this is the only way to authenticate the receiving frame. Specifying "*" as the `targetOrigin` in production is forbidden, as it will cause the browser to post messages to any frame, regardless of origin.

5.5.2 Receiving data

Receiving data is achieved by subscribing to the "`message`" event. An example can be seen in Listing 5-4.

```
window.addEventListener("message", receiveMessage, false);

function receiveMessage(event)
{
      //perform security checks
}
```

Listing 5-4: Receiving data example

The `receiveMessage` event handler will be called with a `MessageEvent` parameter. Upon receiving a message, the receiving party should *always*:

- Check the `origin` attribute of the event to authenticate the sender
- Perform input validation on the data attribute to ensure it's in the desired format, i.e. `ApplicationMessage`.
- Check the origin properly exactly to match the FQDN(s) you expect. Note that code such as that found in Listing 5-5 leads to very imprecise matching and may result in insecure data reception. Consider that an origin "www.owasp.org.attacker.com" would result in a positive match.

```
if (message.orgin.indexOf(".owasp.org")!=-1){
      /* ... */
}
```

Listing 5-5: Insecure Fully Qualified Domain Name (FQDN) check example

Additionally, even after receiving a message from an authenticated origin, it is of good practice to never evaluate the received messages as code by using e.g. `eval()` or DOM insertion via `innerHTML`. Doing so, will enable Cross-site scripting (XSS), which may have a severe impact on both Bilinfo.net and the Plugin Backend and any related subsystems.

You can read more about security recommendations when doing web messaging at the *Open Web Application Security Project* (OWASP) website[8].

[8] https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet#Web_Messaging

5.6 Handshake

The *handshake* procedure – seen in Figure 5-3 – ensures that both parties are fully loaded and agree upon what protocol specification version they are going to use. The Plugin should wait for the completion of the handshake procedure before becoming available for user interaction. A loading indicator or spinner should be displayed until completion.

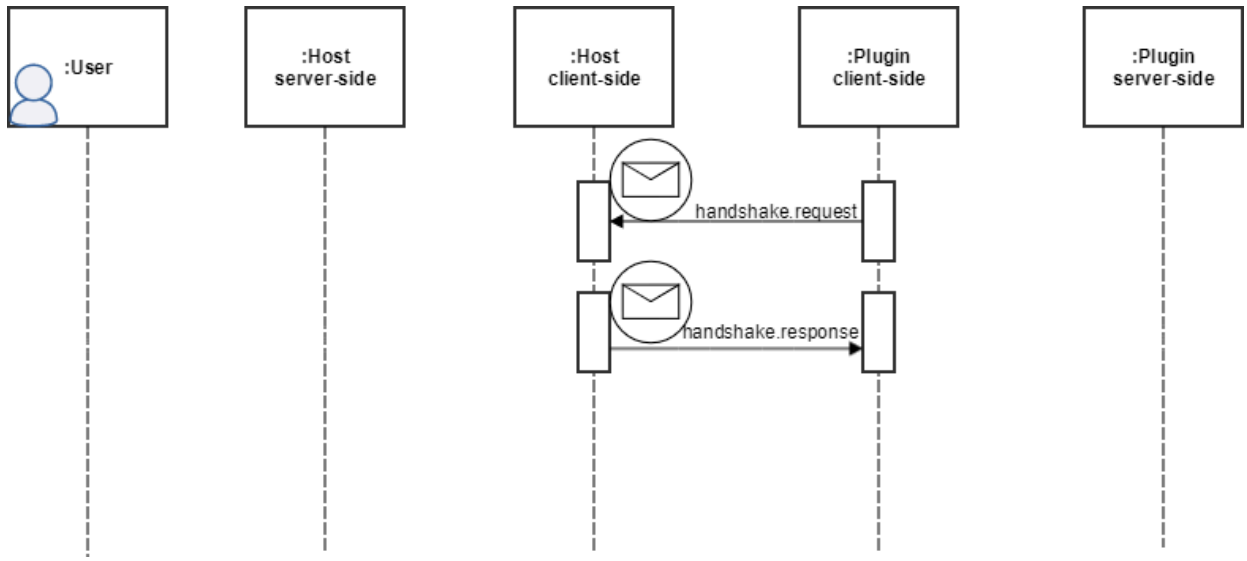

Figure 5-3: Handshake interaction

The Plugin will initiate the handshake procedure, specifying the highest protocol version it supports. The Host will then accept or reject the handshake request. For instance, a handshake request can be rejected if the version specified by the Plugin is not supported by the Host

The protocol is versioned according to Semantic Versioning guidelines. This means that only changes to the major component of the version (e.g. going from 1.0.0 to 2.0.0) will not be backwards compatible. Changes to the minor (e.g. 1.0.0 to 1.1.0) and patch (e.g. 1.0.0 to 1.0.1) component will always be backwards compatible. Consequently, during the handshake protocol, the two parties should agree on the major version component. As an example, it is valid for the Host to use version 1.2.3 while the Plugin uses 1.0.0.

The Plugin should always specify the version using all three components and not use any pre-release specifiers. Please refer to the guidelines specified in Table 5-5.

Table 5-5: Plugin handshake version request guidelines

Version	Is valid
1.0.0	Yes
1	No
1.0	No
1.0.0-beta	No

The handshake sequence follows the steps illustrated below:

1. The Host will load the Plugin specifying the `reference` query string parameter
2. The Plugin will send a *Handshake Request Message*
3. The Host will send a *Handshake Response Message*

Host will tolerate multiple handshake requests from the same Plugin, successfully accepting all of them if they are valid. Therefore, when implementing the handshake sequence, the Plugin developer can use the `setInterval` function to request a handshake until it receives an acceptance from the Host. Please remember to cancel the interval once you received a handshake acceptance/rejection.

5.6.1 Handshake Request Message

The *Handshake Request Message* specification and example can be seen in Table 5-6 and Listing 5-6 respectively.

Table 5-6: Handshake request message specification

Location	Name	Description
	`type`	Hardcoded to "`handshake.request`"
	`version`	The protocol version used by the Plugin
	`reference`	The reference query string parameter specified by the Host when loading the Plugin
body	*This message has an empty body*	

Example

```
{
	"header":{
		"type": "handshake.request",
		"version": "1.0.0",
		"reference": "1234abcd"
	},
	"body":{ }
}
```

Listing 5-6: Handshake request message example

5.6.2 Handshake Response Message

The *Handshake Response Message* specification can be seen in Table 5-7. Example responses can be seen in Listing 5-7 and Listing 5-8 for successful and failed handshakes, respectively.

Table 5-7: Handshake response message specification

Location	Name	Description
header	`type`	Hardcoded to "`handshake.response`"
	`version`	The protocol version used by the Host
	`reference`	The reference query string parameter specified by the Host when loading the Plugin
body	`status`	The result of the handshake attempt. Possible values are: - "`ok`" – successful handshake - "`rejected`" – failed handshake
	`reason`	In the case of rejected handshakes

Successful handshake

```
{
	"header":{
		"type": "handshake.response",
		"version": "1.2.3",
		"reference": "1234abcd"
	},
	"body":{
		"status": "ok"
	}
}
```

Listing 5-7: Successful handshake response message

Failed handshake

```
{
	"header":{
		"type": "handshake.response",
		"version": "9.2.3",
		"reference": "1234abcd"
	},
	"body":{
		"status": "rejected"
		"reason": "version not specified or not supported"
	}
}
```

Listing 5-8: Failed handshake response message

5.7 Data exchange interaction

This section will cover the data and message exchanges between the Host and Plugin – after a successful Handshake has been performed. Figure 5-4 shows the typical data exchange flow between the Host and Plugin.

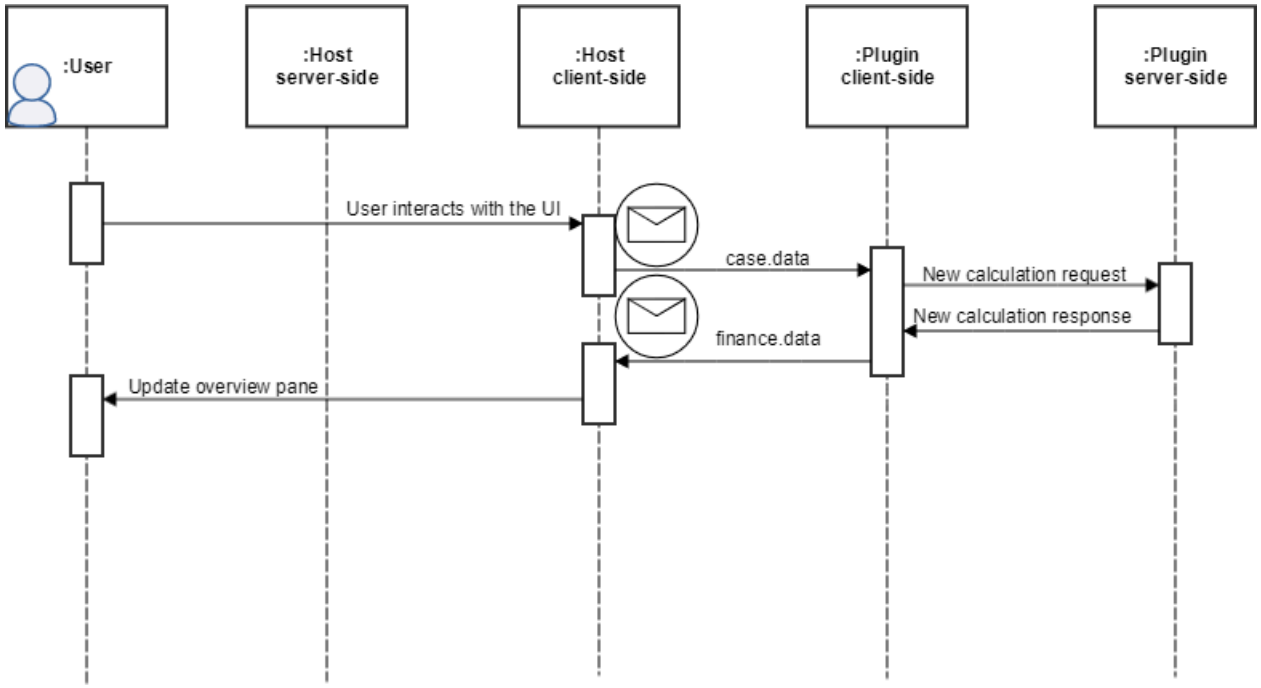

Figure 5-4: Data exchange interaction

The Host will send updates to the Plugin in real-time and the Plugin via the `data.case` message. The Plugin is expected to report information back to the Host in real-time via the `finance.data` message. The message `finance.data` is only for finance and leasing, other services like insurance are not expected to send data back.

5.7.1 Input (Host-to-Plugin)

Input defines the data the Host will send to the Plugin. The Plugin does not have VETO power[9] towards accepting or denying the data sent by the host. However, in the case of invalid data, the Plugin may choose to display an error message to the user via its own user interface.

All the Case data will be sent to the Plugin via a single message containing information about the Customer (PrivateCustomer or BusinessCustomer), Car and TradeInCar.

Subsequent updates to the Case – performed either by the user via updates or automatically via recalculations – will result in the Plugin receiving a new message containing information about all the entities, not just the modified ones. This means that the Plugin can always consider the last received message as the “source of truth”.

[9] Power or ability to unilaterally stop an official action (la. "I forbid")

There are no guarantees about the minimum amount of data the Host will send to the Plugin. This is because data is being sent to the Plugin as the user enters it. Therefore, the Plugin should be able to deal with missing fields, or fields initialized with null or empty string.

Case specification message

Table 5-8 shows the specification for the `data.case` message sent from the Host to the Plugin. In essence, this specification body may be read as the root type, with each subtype – and their respective subtypes – specified in the following subsections.

Table 5-8: Case data message specification

Location	Name	Description
header	`type`	Hardcoded to "`data.case`"
	`version`	The protocol version used by the Host
	`reference`	The reference query string parameter specified by the Host when loading the Plugin
body	`caseId`	Uniquely identifies the case this Plugin is part of
	`viewUrl`	A url that can be used to navigate to the case hosting the Plugin
	`isActive`	A boolean describing whether the Plugin is marked as active by the user (Anvend på sag)
	`customer`	An object representing a PrivateCustomer or a BusinessCustomer
	`car`	An object representing a Car
	`tradeInCar`	An object representing a TradeInCar
	`businessUsers`	An object representing BusinessUsers (only if customer is business)

PrivateCustomer

Table 5-9 shows the specification for the PrivateCustomer variant of the `customer` field of the Case specification message. An example can be seen in Listing 5-9 on Page 31.

Table 5-9: PrivateCustomer data specification

Field	Type	Description
`type`	`string`	Hardcoded to "`private`" for private customers
`info`	`object`	Object of type PrivateCustomer
`coApplicant`	`object`	Object of type CoApplicant

PrivateCustomerInfo

Table 5-10 shows the specification for the PrivateCustomerInfo type, member of the PrivateCustomer type.

Table 5-10: PrivateCustomerInfo data specification

Field	Type	Description	Example
`firstName`	`string`		"John"
`lastName`	`string`		"Doe"
`street`	`string`		"Axel Kiers Vej"
`streetNumber`	`string`		"11"
`zipCode`	`number`		8270
`city`	`string`		"Højbjerg"
`country`	`string`		"Denmark"
`email`	`string`		"private@email.com"
`cellphone`	`string`	Mobile phone	"86112233"
`coOrAtt`	`string`		
`cpr`	`string`		"0101870006"
`phone`	`string`	Landline phone	"86112233"
`driverLicenseNumber`	`string`	Driver license number	"12345678"

CoApplicant

Table 5-11 shows the specification for the CoApplicant type with example values, member of the PrivateCustomer type.

Table 5-11: CoApplicant data specification

Field	Type	Description	Example
`firstName`	`string`		"John"
`lastName`	`string`		"Doe"
`street`	`string`		"Axel Kiers Vej"
`streetNumber`	`string`		"11"
`zipCode`	`number`		8270

city	string		“Højbjerg”
country	string		“Denmark”
email	string		“coapplicant@email.com”
cellphone	string	Mobile phone	“86112233”
coOrAtt	string		
cpr	string		“0101870006”
phone	string	Landline phone	“86112233”

Example

Listing 5-9 below shows an example of the PrivateCustomer variant of the customer type. This is a part of the overall Case specification message received by the Plugin from the Host.

```
"customer": {
    "type": "private",
    "info": {
        "firstName": "Cosmin",
        "lastName": "Lazar",
        "cpr": "2510871212",
        "street": "gade",
        "streetNumber": "1",
        "city": "Aarhus N",
        "zipCode": 8200,
        "country": "denmark",
        "cellphone": "21212121",
        "coOrAtt": null,
        "phone": "21212121"
    },
    "coApplicant": {
        "firstName": "Cosmin2”,
        "lastName": "Lazar2",
        "cpr": "2510871212",
        "street": "gade",
        "streetNumber": "1",
        "city": "Aarhus N",
        "zipCode": 8200,
        "country": "denmark",
        "cellphone": "21212121",
        "coOrAtt": null,
        "phone": "21212121"
    }
}
```

Listing 5-9: Case data message example

BusinessCustomer

Table 5-12 shows the BusinessCustomer variant of the `customer` field of the Case specification message.

Table 5-12: BusinessCustomer data specification

Field	Type	Description
`type`	`string`	Hardcoded to "`business`" for business customers
`info`	`object`	Object of type BusinessCustomerInfo
`decisionMaker`	`object`	Object of type DecisionMaker

BusinessCustomerInfo

Table 5-13 shows the specification for the BusinessCustomerInfo type, member of the BusinessCustomer type.

Table 5-13: BusinessCustomerInfo data specification

Field	Type	Description	Example
`companyName`	`string`		"Ebay Classifieds"
`companyType`	`number`	Value from CompanyType	1
`street`	`string`		"Axel Kiers Vej"
`streetNumber`	`string`		"11"
`zipCode`	`number`		8270
`city`	`string`		"Højbjerg"
`country`	`string`		"Denmark"
`email`	`string`		"business@business.com"
`phone`	`string`	Landline phone	"86112233"
`cellphone`	`string`	Mobile phone	"86112233"
`coOrAtt`	`string`		
`cvr`	`string`		"20618175"

CompanyType

Table 5-14 shows the specification for of the CompanyType identifier, member of the BusinessCustomerInfo type.

Table 5-14: CompanyType data specification

Value	Type	Description
`0`	`number`	Unknown
`1`	`number`	A/S
`2`	`number`	ApS
`3`	`number`	Enkeltmandsvirksomhed
`4`	`number`	I/S

DecisionMaker

Table 5-15 shows the specification of the DecisionMaker type, member of the BusinessCustomer type.

Table 5-15: Decision maker data specification

Field	Type	Description	Example
`firstName`	`string`		
`lastName`	`string`		
`title`	`string`		
`phone`	`string`		
`cellphone`	`string`		
`email`	`string`		

Car

Table 5-17 shows the specification of the Car type found in the `car` field of the Case specification message.

Table 5-17: Car data specification

Field	Type	Description	Example
`isFactoryNew`	`boolean`		
`dbiId`	`number`		
`modelCatalogueId`	`string`	Modelcatalogue identifier	893757B3-E30B-CE26-681C-08CF06294F34
`type`	`number`	Value from CarType	1
`make`	`string`		
`model`	`string`		
`variant`	`string`		
`modelYear`	`number`		
`numberOfDoors`	`number`		
`mileage`	`number`		
`motor`	`string`		
`color`	`string`		Black
`fuelType`	`number`	Value from FuelType	2
`kmPerLiter`	`number`		
`registrationNumber`	`string`		
`registrationDate`	`string`	String representing an ISO 8601 date	2015-08-31T22:00:00.000Z
`ownWeight`	`number`		1600
`deliveryCost`	`number`	Delivery cost ex. vat	2080
`deliveryCostVat`	`number`	Vat of delivery cost	520
`licensePlateCost`	`number`	Price for the license plate	1180

dealerEquipments	array	Array of DealerEquipment	
factoryEquipments	array	Array of FactoryEquipment	
listPrice	object	Object of type Price	List price from DBI
salesPrice	object	Object of type type Price	Sales price on the case
vin	string		
kmWaranty	boolean	Relevant for used cars, a warranty that the car's mileage hasn't been tampered with	
discount	number	Discount ex. vat	8000 (for a total discount of 10000)
discountVat	number	Discount vat	2000 (for a total discount of 10000)
timingBeltReplaced	number	Value from TimingBeltReplaced	2
serviceBook	boolean	Relevant for used cars, null otherwise	True
serviceBookEnum	number	Value from ServiceBook	1
toldAndSkatVariant	string	The ts_variant information – as supplied by DBI – only applies to new cars	
ownerTaxPerYear	number	Represents the value of Weight tax, Green tax or CO2 tax	3200
effect	number	Horse power	200
cashPriceInclVat	number	Car total price (discount detracted) including dealer- and factory equipment, delivery cost and license plate cost, and registration fee. Always including VAT	
numberOfAirbags	number	Number of Airbags	2
numberOfIntegratedChildSeat	number	Number of ChildSeats	2
numberOfSeatBeltAlarms	number	Number of Seatbelt alarms	2
euroNcapTopRated	boolean	5 stars in EuroNCap (toprated)	true

totalWeight	number	Total weight of the car in kg	1500
hasRadio	boolean	Does the vehicle have radio	true
hasABS	boolean	Does the vehicle have abs brakes	true
hasESP	boolean	Does the vehicle have ESP	false
isVatPaid	boolean	Has the vat on the vehicle been paid	true
isTaxPaid	boolean	Has the registration tax been paid	false
approvedKmPerLiterMeasurementType	string	Which method has been used for measuring kmPerLiter WLTP, NEDC1, NEDC2	NEDC1
approvedKmPerLiter	number	WLTP or NEDC value for KmPerLiter	

CarType

Table 5-18 shows the specification of the CarType type, member of the Car and TradeInCar types.

Table 5-18: CarType data specification

Value	Type	Description
1	number	PrivateCar
2	number	VanExVat
3	number	VanInclVat
4	number	Bus
5	number	Camping Bus
6	number	Caravan (No engine)
7	number	Motorcycle

FuelType

Table 5-19 shows the specification for the FuelType identifier, member of the Car and TradeInCar types.

Table 5-19: FuelType data specification

Value	Type	Description
0	number	Unknown
1	number	Gasoline
2	number	Diesel
3	number	Electric

Price

Table 5-20 shows the specification for the Price type, member of the Car and FactoryEquipment types.

Table 5-20: Price data specification

Field	Type	Description	Example
costPrice	number	Cost price ex. vat, profit and registration fee	80499
vat	number		22137
profit	number	Dealer profit ex. vat	8050
registrationFee	number		82382
taxableAmount	number		

DealerEquipment

Table 5-21 shows the specification for the DealerEquipment type, member of the Car type.

Table 5-21: DealerEquipment data specification

Field	Type	Description	Example
description	string		Telefonholder
vat	number		500
costPrice	number	Value ex. profit and ex. vat	1000
profit	number	Value ex. vat	1000
code	string	Dealer assigned identifier	RS01

FactoryEquipment

Table 5-22 shows the specification for the FactoryEquipment type, member of the Car type.

Table 5-22: FactoryEquipment data specification

Field	Type	Description	Example
description	string		Fartpilot
code	string	Dbi code	RS02
listPrice	Price		
salesPrice	Price		

TimingBeltReplaced

Table 5-23 shows the specification for the TimingBeltReplaced type, member of the Car and TradeInCar types.

Table 5-23: TimingBeltReplaced data specification

Value	Type	Description
0	number	Unknown
1	number	Yes
2	number	No
3	number	NoTimingBelt

ServiceBook

Table 5-24 shows the specification for the ServiceBook type, member of the Car and TradeInCar types.

Table 5-24: ServiceBook data specification

Value	Type	Description
0	number	NotFilledOut
1	number	FilledOut
2	number	PartiallyFilledOut

TradeInCar

Table 5-25 shows the specification of the TraceInCar type found in the `tradeInCar` field of the Case specification message.

Table 5-25: TradeInCar data specification

Field	Type	Description	Example
registrationDate	string	String representing an ISO 8601 date	2015-08-31T22:00:00.000Z
type	number	Value from CarType	2
make	string		Citroën
model	string		C1
variant	string		Sport
color	string		Black
kmWaranty	boolean		true
timingBeltReplaced	number	Value from TimingBeltReplaced	2
timingBeltReplacedAtMileage	number		25000
timingBeltReplaced	string	String representing an ISO 8601 date	2015-08-31T22:00:00.000Z
serviceBook	boolean		true

serviceBookEnum	number	Value from ServiceBook	1
majorDamages	number	Value from MajorDamages	0
registrationNumber	string		AB22333
vin	string		LKFG3423432345674
modelYear	number		2010
fuelType	number	Value from FuelType	3
latestMotDate	string	String representing an ISO 8601 date	2015-08-31T22:00:00.000Z
mileage	number		50000
costPrice	number		20000
vat	number		0
remainingDebt	number		5000
financedAt	string		Jyffi
isVatFree	boolean		False
comment	string		Comment about the car

MajorDamages

Table 5-26 shows the specification for the MajorDamages identifier, member of the TradeInCar type.

Table 5-26: MajorDamages data specification

Value	Type	Description
0	number	No
1	number	Yes
2	number	Unknown

BusinessUsers

Table 5-27 shows the specification of the `businessUsers` field of the Case specification message.

Table 5-27: BusinessUsers data specification

Field	Type	Description	Example
businessUser1	BusinessUser	Object of type BusinessUser	
businessUser2	BusinessUser	Object of type BusinessUser	

BusinessUser

Table 5-28 shows the specification for the BusinessUser type.

Table 5-28: Business user data specification

Field	Type	Description	Example
`firstName`	`string`		"John"
`lastName`	`string`		"Doe"
`cpr`	`string`		"0101870006"
`driversLicenseNumber`	`string`		"10557254"
`street`	`string`		"Axel Kiers Vej"
`streetNumber`	`string`		"11"
`zipCode`	`number`		8270
`phone`	`string`	Landline phone	"86112233"
`mobilePhone`	`string`	Mobile phone	"86112233"
`email`	`string`		"bruger@email.dk"
`city`	`string`		"Højbjerg"
`country`	`string`		"Denmark"
`gender`	`number`	Value from Gender	1

Gender

Table 5-29 shows the specification for the Gender identifier, member of the BusinessUser type.

Table 5-29: Gender data specification

Value	Type	Description
`0`	`number`	Unknown
`1`	`number`	Male
`2`	`number`	Female

5.7.2 Output (Plugin-to-Host)

Output defines the data Host will receive from Plugin.

Finance specification message

In case of loan offers, the Plugin should publish a loan specification message after each calculation. Table 5-30 and Table 5-31 show the specification for the Finance specification message. This message is only for plugins that offer either finance or leasing.

Table 5-30: Finance specification message

Location	Name	Description
header	`type`	Hardcoded to "`finance.data`"
	`version`	The protocol version used by the Host
	`reference`	The reference query string parameter specified by the Host when loading the Plugin
body	Details description of the fields can be found Table 5-28.	

Table 5-31: Finance specification data message fields

Field	Type	Description	Example
`downPayment`	`number`	Including VAT	
`profit`	`number`	Dealer profit	
`monthlyPayment`	`number`	Loan and leasing.	
`endDate`	`string`	String representing an ISO 8601 date loan and leasing	2015-08-31T22:00:00.000Z
`residualValue`	`number`	leasing	
`interest`	`number`	loan (rente)	
`anualExpensesPercentage`	`number`	(ÅOP) loan	
`financeExpenses`	`number`	(finansieringsomkostninger) loan	

The host will send a protocol error message if the Plugin submits an invalid Finance specification message

Case update message

The Plugin sends the `case.update` message to inform the Host about Server-Side changes to the application. Receiving a message of this type will cause the Host to do a server-side call to reload the offer/application details. This message is for all types for services, finance, leasing, insurance etc.

Table 5-32 shows the specification for the Case update message.

Table 5-32: Case update message

Location	Name	Description
header	`type`	Hardcoded to "`case.update`"
	`version`	The protocol version used by the Host
	`reference`	The reference query string parameter specified by the Host when loading the Plugin
body	Empty body	

Please note that this is a *crucial* part to integration between Bilinfo and the Plugin, which enables Bilinfo to display up-to-date information regarding the status of the Application. This interaction is described in further detail in the *Application interaction* section.

5.8 *Save* interaction

The *Save* interaction may be triggered at any time from Bilinfo to save information to a Case. In order to save any information pertaining to the Plugin, a "`save.request`" message is sent to the Plugin by the Host in the event of a user action. Figure 5-5 shows the *Save* interaction.

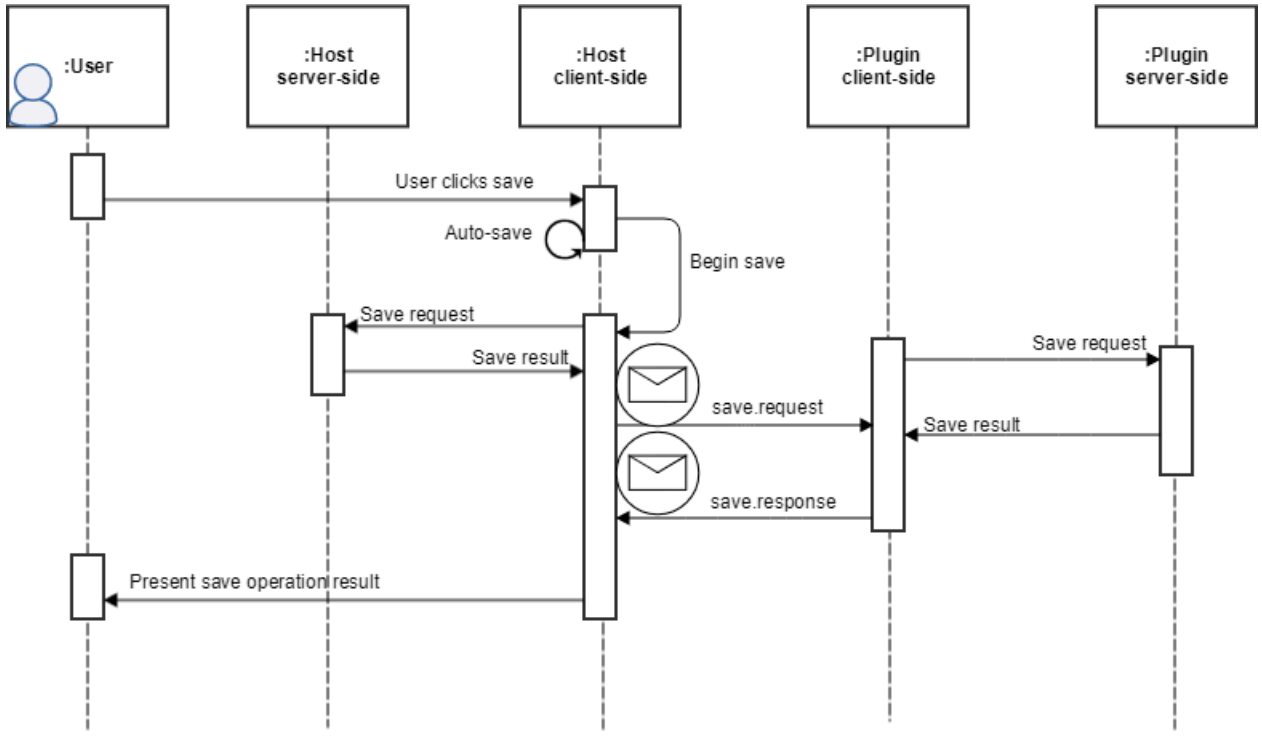

Figure 5-5: Save interaction diagram

Save request message

The Bilinfo Case contains transient information until the user decides to persist it by clicking the save button. An automatic save can also occur when a set of preconditions are met. It is therefore important for the Plugin to follow the lifecycle of the Case and save/discard changes at the same time.

After issuing a save request, the Host will wait for the Plugin's acknowledgement for a maximum of **3 seconds**.

Table 5-33 shows the specification for the *Save request message*.

Table 5-33: Save request message

Location	Name	Description
header	`type`	Hardcoded to "`save.request`"
	`version`	The protocol version used by the Host
	`reference`	The reference query string parameter specified by the Host when loading the Plugin

body	The save command has an empty body

Save response message

If the Plugin replies within the allotted timeout, the Host will notify the user upon the success/failure of the save. Alternatively, if the Plugin does not reply within the allotted timeout, the Host will proceed as if it received a successful acknowledgement.

Table 5-34 and Table 5-35 respectively show a successful or failed save response from the Plugin to the Host.

Successful save

Table 5-34: Successful save response message

Location	Name	Description
header	`type`	Hardcoded to "`save.response`"
	`version`	The protocol version used by the Host
	`reference`	The reference query string parameter specified by the Host when loading the Plugin
body	`status`	"ok"

Failed save

Table 5-35: Failed save response message

Location	Name	Description
header	`type`	Hardcoded to "`save.response`"
	`version`	The protocol version used by the Host
	`reference`	The reference query string parameter specified by the Host when loading the Plugin
body	`status`	"fail"
	`reason`	Reason for failure

5.9 Application interaction

It is important for both systems to be kept in sync when the user decides to make an Application. Therefore, an Application should not be considered submitted until the Host has been notified via a Server-Side call. Once the application is submitted, the Plugin is expected to send a `case.update` message as described in section 5.7.2.

Figure 5-6 shows the Application interaction from initial User (i.e. Dealer) application through to the intercommunication between the Plugin Server-Side and Host Server-Side and subsequent Plugin Client-Side to Host Client-side `case.update` message.

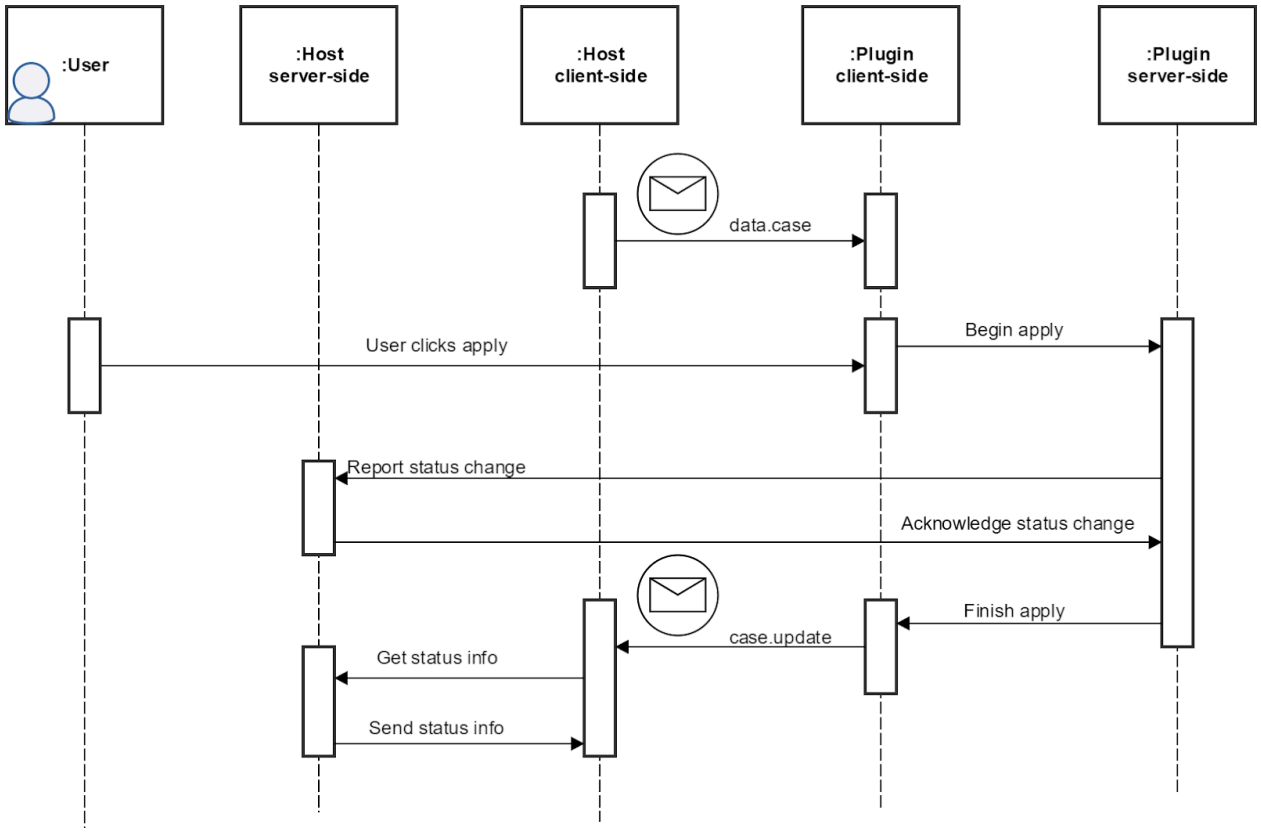

Figure 5-6: Application interaction

Please note that the Plugin should only allow the user to apply when the `isActive` field, received in the `data.case` message, is `true`.

For more information regarding the Application Status changes, the reader is referred to the Application Status API section.

5.10 Other Messages

This section covers the messages that are not associated with any given interaction.

5.10.1 Protocol Error message

The Protocol Error message is sent by the Host if the Plugin does not adhere to the protocol specification. Table 5-36 shows the specification of the Protocol Error.

Table 5-36: Protocol Error Message specification

Location	Name	Description
header	`type`	Hardcoded to "`protocol.error`"
	`version`	The protocol version used by the Host
	`reference`	The reference query string parameter specified by the Host when loading the Plugin
body	`message`	A string describing the error

Listing 5-10 shows an example Protocol Error message.

```
{
	"header": {
		"type": "protocol.error",
		"version": "1.0.0",
		"reference": "1237"
	},
	"body": {
		"message": "invalid reference, expecting reference 1237"
	}
}
```

Listing 5-10: Protocol Error message example

Errors that may be conveyed through this message include:

- Plugin sends a malformed message (missing header or body)
- Plugin specifies an invalid reference

6. Host Server-Side Services

This chapter will cover the Case Plugin Architecture related services that should be used to complete the Case Plugin integration loop by e.g. reporting Application Status back to Bilinfo from their Plugin Backend.

6.1 Application Status API

To complete the feedback loop between the Plugin and Bilinfo backend, application status reports must be supplied to the Application Status API. This section describes the API interface and specification.

6.1.1 Base path

Each Plugin Implementer is assigned a dedicated endpoint with a dedicated set of claims required for calling the API. Please note that the examples seen in Table 6-1 include a "`{plugin_implementer_identifier}`" placeholder in various locations. You must replace this placeholder with the identifier allocated upon the initial registration process.

Table 6-1: Bilinfo service base path

Environment	Endpoint
QA	`https://gw.qa01.bilinfo.net/{plugin_implementer_identifier}/api/v2`
Production	`https://gw.bilinfo.net/{plugin_implementer_identifier}/api/v2`

6.1.2 Endpoints

The following section iterates through the endpoints available in the Application Status API. Each endpoint has a specific path, which must be appended to the base path, and must be requested using a scope Bearer token as specified in Section 3.1.

ApplicationStatus

The ApplicationStatus endpoint is used to return the Application Status to Bilinfo.

- Path: `/applicationstatus/{reference}`
- Verb: `PUT`
- Authorization scope: `https://{plugin_implementer_identifier}.bilinfo.net/api/applicationstatus/write`

Request

Table 6-2 describes the request specification for ApplicationStatus endpoint. Respectively Table 6-3 and Table 6-4 show the subtypes of that specification.

Table 6-2: ApplicationStatus endpoint request specification

Location	Name	Description
path	`reference`	The `reference` parameter supplied to the plugin. Refer to Section 5.2 for further information
body	`object`	The body represent a mapping of the status for the different Brand & Product ServiceTypes applied for in the Plugin.

Listing 6-1 shows an example of the status update for a bundled application that contains both loan and insurance. Please note that loan and leasing have their own status values, as defined in Table 6-3, whereas the status for all the other ServiceTypes are defined in Table 6-4

```
{
  "loan": {
    "providerName": "Finance company Loan",
    "status": 1,
    "statusText": "Under processing"
  },
  "insurance": {
    "providerName": "Insurance company",
    "status": 1,
    "statusText": "Sent"
  }
}
```

Listing 6-1: Status update for bundled Application

Loan/Leasing ApplicationStatus

Table 6-3: ApplicationStatus identifier specification for loan or leasing

Value	Description
0	
1	Under processing
2	Effectuated
3	Approved
4	Denied
5	Cancelled
6	PreApproved

Generic ApplicationStatus

Table 6-4: Generic ApplicationStatus Identifier specification

Value	Description
0	Not Sent
1	Sent

Response

Table 6-5 shows the possible responses provided by the ApplicationStatus endpoint.

Table 6-5: ApplicationStatus endpoint response specification

Status code	Description
200	Successful response

400	When one of the required parameters is missing or invalid
404	When no offer can be found using the supplied `reference` parameter
500	Internal server error during request processing

Example

An example request for the ApplicationStatus endpoint with applied reference and Bearer token can be seen in Listing 6-2.

```
PUT /{plugin_implementer_identifier}/api/v2/applicationstatus/abc123 HTTP/1.1
Host: gw.bilinfo.net
Authorization: Bearer eyJ0eXA…PL4UPsfp
```

```
{
  "loan": {
    "providerName": "Finance company Loan",
    "status": 1,
    "statusText": "Under processing"
  }
}
```

Listing 6-2: ApplicationStatus endpoint request example

SalesContract

The SalesContract endpoint is used for acquiring the sales contract pdf file.

- Path: `/documents/salescontract/{reference}`
- Verb: `GET`
- Authorization scope:
  `https://{plugin_implementer_identifier}.bilinfo.net/api/salescontract/read`

Response

Table 6-6 shows the possible responses provided by the SalesContract endpoint.

Table 6-6: SalesContract endpoint response specification

Status code	Description
200	Successful response containing the byte array representing the pdf
400	When one of the required parameters is invalid
401	When Authorization header is not specified or invalid
403	When the sales contract cannot be generated due to: - the application is not marked as active - the user has not yet applied for financing - the application has been deleted
404	When no offer can be found using the supplied `reference` parameter, or the parameter is not supplied at all
500	Internal server error during request processing