

Fältprotokoll

Fältprotokoll — en timme vid skarpa ERP:et

Skriv ut detta eller ha det på en andra skärm. Allt du behöver. Kommandon i ordning, med plats för att skriva ned svar.

Länkar - Analysverktyget: <https://svenskfast.azzar.org/analysverktyg> - Scriptet: <https://svenskfast.azzar.org/web/content/1128/sf-inspect.js> - Tillägget (gamla versionen, för jämförelse): <https://svenskfast.azzar.org/installera-tillagg>

FÖRE — 5 minuter

- ☐ **Ladda ner sf-inspect.js** till ett USB-minne eller till datorn. Gör detta **innan** du sitter hos dem — då är du oberoende av deras nät.
 - ☐ Öppna analysverktyget i din egen webbläsare och läs igenom kommandona
 - ☐ Kontrollera att du kan logga in i deras CRM
-

STEG 0 — Kom igång (2 min)

1. Öppna CRM:et i Chrome eller Edge. **Logga in.**
2. Tryck **F12** → fliken **Console**
3. Om Console är tom eller klagar på “paste” — skriv `allow pasting` och tryck Enter
4. Klistra in **hela sf-inspect.js** → Enter

Förväntat:

SF – analysverktyg för Svenskfast-CRM

✗ Får du felmeddelande om CSP? Då blockerar deras system Console-inmatning. Kör istället: DevTools → **Sources** → **Snippets** → New snippet → klistra in → Ctrl+Enter. Dokumentera att det hände.

- ☐ Scriptet laddades utan fel
-

STEG 1 — Kartlägg sidan (10 min)

`sfInspect()`

Läs resultatet. **Fyll i:**

Ramverk:

(React / Vue / Angular / jQuery / ExtJS / okänt)

Iframes: _____ st ← om >0, se varning nedan

Shadow DOM: _____ rötter ← om >0, se varning nedan

Antal fält: _____

Antal knappar: _____

! Om iframes > 0

Innehållet ligger i en ram. Då måste tillägget injiceras i ramen. Dokumentera:

! Om shadow DOM > 0

Vanliga selektorer når **inte** in. Detta kräver kodändring. Dokumentera antal och var:

! Om ramverket är React/Vue

Naiv värdesättning (`el.value = x`) **fungerar inte**. Metod B krävs. Detta är redan byggt — men verifiera i steg 3.

☐ Steg 1 klart

STEG 2 — Se hur sidan reagerar (15 min)

Detta är det du inte kan lista ut i förväg. **Kör och klicka.**

`sfWatch(90000)`

Klicka nu **manuellt** i 90 sekunder:

- ☐ Skriv ett **telefonnummer** i sökrutan → sök
- ☐ Se vad som händer i resultatlistan
- ☐ Klicka på en träff (om det finns någon)
- ☐ Klicka “ny kontakt” / motsvarande
- ☐ Fyll i ett par fält
- ☐ Spara

Konsolen loggar varje element som dyker upp (+ input) och försvinner (– table).

Anteckna flödet

Vid SÖKNING dyker upp:

Vid TRÄFF – öppnas kontakten automatiskt?

[] Ja, automatiskt [] Nej, måste klicka på raden

Vid NY KONTAKT – hur öppnas formuläret?

Spara-knappens text: _____

☐ Steg 2 klart

STEG 3 — Verifiera selektorer (20 min) ★

Detta är det viktigaste steget.

`sfSuggest()`

Ger förslag. **Lita inte på dem** — verifiera varje.

För varje selektor, kör:

`sfProbe('<selektor>')`

Fyll i

Roll	Selektor	Metod (A/B/C)
Sökruta	_____	_____
Sök-knapp	_____	_____
Resultatrad	_____	_____
Ny-knapp	_____	_____
Namnfält	_____	_____
Telefonfält	_____	_____
Adressfält	_____	_____
E-postfält	_____	_____
Anteckningsfält ★	_____	_____
Spara-knapp	_____	_____
Avbryt-knapp	_____	_____

★ **Anteckningsfältet är viktigast.** Det är dit kampanjtexten ska. Verifiera extra noga att `sfProbe` ger recommended: 'B' eller 'C'.

Om recommended: 'INGEN'

Fältet är inte ett vanligt input. Kan vara: - contenteditable → _____ - i en iframe → _____ - i shadow DOM → _____ - en canvas/ritad kontroll → _____

Dokumentera **exakt** vad `document.querySelector('din-selektor')` returnerar:

☐ Steg 3 klart

STEG 4 — Finns ett API? (10 min)

`sfNetwork()`

Sök sedan på ett telefonnummer manuellt.

Tittar du på konsolen: visas XHR GET /api/... eller fetch ...?

☐ Ja, det finns ett API → **STORT fynd.** Dokumentera:

Endpoint: _____
Metod: _____
Svar: _____

Om ja — då kan hela klick-strategin för sökningen ersättas. Det är en bättre lösning. Säg till omgående.

☐ Nej, ingen API-trafik → sidan renderas på servern. Klick-strategin gäller.

STEG 5 — Skicka resultatet (5 min)

Allt du kört har **kopierats automatiskt till urklipp** som JSON.

Klistra in i chatten / Pi, i denna ordning:

1. `sfInspect()` → (kopierat automatiskt)
2. `sfSuggest()` → (kopierat automatiskt)
3. `sfProbe()` → kör för varje fält, klistra in alla
4. Tabellen ovan med dina verifierade selektorer
5. Flödesanteckningarna från steg 2

Klistra bara in det sista om urklippet skrivits över — kör om kommandot.

☐ Steg 5 klart

EFTER — vad som händer

Med ditt resultat:

1. Jag fyller i `extension/adapters.js` med de verifierade selektorerna
2. Ändrar `manifest.json` om iframes kräver `all_frames`
3. Ändrar `content.js` om shadow DOM kräver traversering
4. **Bumpar versionen** (obligatoriskt — annars laddas inte tillägget om)
5. Bygger ZIP, laddar upp till `svenskfast.azzar.org`
6. Ger dig ny nedladdningslänk

Sedan på deras dator:

7. Ladda ner nya versionen, läs in tillägget på nytt
8. **Testa med EN kontakt** — kontrollera i ERP:et
9. Kör samma import igen → allt ska bli `ø` överhoppade (idempotens)
10. Öka till 10 kontakter

SNABBFELSÖKNING

Problem	Åtgärd
Console vägrar ta emot inklistring	skriv <code>allow pasting</code> + Enter
CSP blockerar Console	DevTools → Sources → Snippets
SF is not defined	scriptet kördes inte — kör om hela filen
sfInspect ger tomt	sidan är en SPA som inte hunnit ladda — vänta, kör igen
Allt ligger i iframe	välj ramen i DevTools console-väljaren (dropdown högst upp) och kör där
sfProbe ger INGEN	dokumentera — kräver annan strategi, säg till
Sidan laddar om vid sökning	då är <code>waitFor</code> i <code>content.js</code> rätt lösning — dokumentera

OM DU BARA HINNER EN SAK

Steg 3 — verifiera anteckningsfältets selektor och ifyllningsmetod.

Utan den vet vi inte om tillägget kan skriva i deras system. Allt annat kan lösas i efterhand.

VAD DU INTE SKA GÖRA

-  Kör **inte** importen mot skarpa systemet med hundratals kontakter

- ✗ Ändra **inte** i deras system manuellt “för att testa”
 - ✗ Spara **inte** något du inte kan ångra
 - ✗ Kör **inte** `sfWatch()` utan att stoppa den — den tuggar minne
-

Kontakt: om något blockerar — skriv i chatten. Säg exakt vilket steg, vad du såg, och vad du försökte. Det går snabbare än att gissa.