## **Rapport: Funktionsbeskrivning och Teknisk Uppbyggnad av Home Assistant-integrationen "Elpris Timme"**

**Version:** Baserad på utvecklingen fram till och med 2025-05-21-rev15

### **1. Introduktion**

Integrationen "Elpris Timme" för Home Assistant har utvecklats för att ge användare en detaljerad och aktuell översikt över timvisa elpriser. Huvudsyftet är att hämta spotpriser från en extern API-tjänst, möjliggöra för användaren att konfigurera ett eget elpåslag, och sedan presentera både det rena spotpriset samt det totala priset (spotpris plus påslag) i användarvänliga format, både i öre/kWh och SEK/kWh. Integrationen skapar även dedikerade sensorer som enbart visar det konfigurerade påslaget.

Denna rapport beskriver integrationens tekniska uppbyggnad, datainsamlingsprocess, hur sensorer skapas och uppdateras, samt de konfigurationsmöjligheter som erbjuds användaren.

### **2. Integrationens Kärnkomponenter och Filstruktur**

Integrationen följer standardstrukturen för en "custom component" i Home Assistant och består av följande nyckelfiler, var och en med en specifik roll:

* **manifest.json**: Denna fil innehåller metadata om integrationen, såsom domännamn (elpris\_timme), namn, version, beroenden, och vilken IoT-klass den tillhör. Den deklarerar också att integrationen har ett konfigurationsflöde (config\_flow) och ett alternativflöde (options\_flow), vilket möjliggör installation och modifiering via Home Assistants användargränssnitt.
* **const.py**: Innehåller alla konstanter som används inom integrationen. Detta inkluderar domännamnet, plattformar som stöds (i detta fall "sensor"), standardvärden för konfiguration (t.ex. standard elområde och påslag), API-URL, nycklar för konfigurationsdata, samt namn för de attribut som sensorerna kommer att exponera.
* **config\_flow.py**: Hanterar processen när en användare lägger till integrationen via Home Assistants gränssnitt. Den definierar vilka fält som användaren ska fylla i (elområde och påslag i öre), validerar indata, och skapar en konfigurationspost (ConfigEntry) som lagrar dessa inställningar. Denna fil implementerar även en options\_flow som gör det möjligt för användaren att senare ändra det konfigurerade påslaget utan att behöva installera om hela integrationen.
* **\_\_init\_\_.py**: Är startpunkten för integrationen när den laddas av Home Assistant. Den ansvarar för att:
  + Sätta upp den grundläggande datastrukturen för integrationen inom Home Assistant.
  + Initiera ElprisDataUpdateCoordinator, som är ansvarig för att hämta och uppdatera prisdata från den externa API:n.
  + Vidarebefordra konfigurationsposten till sensorplattformen (sensor.py) för att skapa de faktiska entiteterna.
  + Registrera en "options update listener" som laddar om integrationen om användaren ändrar påslaget via optionsflödet.
* **sensor.py**: Definierar och hanterar alla sensor-entiteter som integrationen skapar. Detta inkluderar logiken för hur sensorernas tillstånd (värde) och attribut beräknas och uppdateras.

### **3. Konfiguration av Integrationen**

Användaren installerar och konfigurerar "Elpris Timme" via Home Assistants användargränssnitt:

1. Navigera till Inställningar -> Enheter & Tjänster.
2. Klicka på LÄGG TILL INTEGRERING och sök efter "Elpris Timme".
3. Ett konfigurationsformulär visas där användaren anger:
   * **Elområde:** En rullgardinsmeny med valen SE1, SE2, SE3, och SE4 (med SE4 som standard).
   * **Elpåslag (öre/kWh):** Ett numeriskt fält där användaren kan ange sitt personliga påslag, t.ex. för elhandelsavgifter, moms, eller andra tillägg. Standardvärdet är 0.0 öre.
4. Efter att ha sparat konfigurationen skapas en ConfigEntry som lagrar dessa val. Integrationen initieras sedan baserat på denna data.

Modifiering av Påslag:

Efter installationen kan användaren ändra det angivna elpåslaget genom att:

1. Navigera till den installerade "Elpris Timme"-integrationen under Enheter & Tjänster.
2. Klicka på KONFIGURERA (eller motsvarande alternativikon).
3. Detta öppnar "options flow" (definierad i config\_flow.py) där endast fältet för elpåslag visas och kan redigeras.
4. När det nya påslaget sparas, laddas integrationen automatiskt om för att de nya inställningarna ska börja gälla för de relevanta sensorerna.

### **4. Datahämtning och Koordination**

Centralt för integrationen är ElprisDataUpdateCoordinator-klassen (definierad i \_\_init\_\_.py). Dess ansvarsområden är:

* **API-kommunikation:** Använder aiohttp\_client för att asynkront göra anrop till API:et https://www.elprisetjustnu.se/api/v1/prices/{ÅR}/{MÅNAD-DAG}\_{ELOMRÅDE}.json.
* **Schemalagd Hämtning:**
  + Normalt försöker koordinatorn hämta ny data varje timme.
  + En specifik logik finns för att hämta priser för **nästa dygn**: Varje dag kl. 14:00 (lokal tid) görs ett försök. Om detta misslyckas (t.ex. om priserna inte publicerats än), övergår koordinatorn till ett kortare retry-intervall (30 minuter) och försöker igen tills data för nästa dygn har hämtats framgångsrikt. Därefter återgår den till det normala timvisa intervallet.
* **Datalagring:** Koordinatorn lagrar de hämtade priserna (som är i råformat SEK/kWh från API:et) i en dictionary, self.all\_prices, där nyckeln är ett datumobjekt och värdet är en lista med timpriser för det datumet.
* **Dataparsning och Validering (\_parse\_and\_validate\_prices):** Innan datan lagras, parsas den för att säkerställa att prisvärden (SEK\_per\_kWh) är flyttal och att tidsstämplar (time\_start, time\_end) är korrekta ISO-strängar. Den validerar också att prisposterna tillhör det förväntade datumet.
* **Notifiering av Sensorer:** När koordinatorn har slutfört en uppdateringscykel (antingen lyckad eller misslyckad), meddelar den alla anslutna sensorer att ny data (eller status om misslyckande) finns tillgänglig.
* **Tidsstämpel:** Koordinatorn håller reda på last\_api\_call\_timestamp (i UTC) för den senaste lyckade uppdateringscykeln.

### **5. Sensor-Entiteter: Funktionalitet och Uppdateringar**

Integrationen skapar sex olika sensor-entiteter, alla baserade på en gemensam BaseElprisSensor-klass (definierad i sensor.py) som ärver från CoordinatorEntity och SensorEntity. Denna basklass hanterar den grundläggande logiken för att ansluta till koordinatorn, schemalägga exakta timvisa uppdateringar, och beräkna det råa aktuella spotpriset (\_raw\_current\_spot\_price\_sek).

Gemensam Uppdateringsmekanism för Spot- och Totalprissensorer:

Dessa fyra sensorer använder en intern timer baserad på async\_track\_point\_in\_time för att trigga en uppdatering av sitt tillstånd exakt när klockan slår hel timme (HH:00:00). När timern utlöses, anropas \_update\_internal\_data som i sin tur:

1. Anropar \_calculate\_raw\_current\_spot\_price\_sek för att hämta det aktuella spotpriset i SEK/kWh från koordinatorns data.
2. Anropar \_update\_sensor\_specific\_data (som implementeras unikt i varje subklass) för att sätta sensorns native\_value och extra\_state\_attributes.
3. Skriver det nya tillståndet till Home Assistant.
4. Schemalägger nästa timvisa uppdatering.

Sensorernas attribut uppdateras också när koordinatorn signalerar att den har ny data (via \_handle\_coordinator\_data\_update\_for\_base).

**De sex sensorerna är:**

1. **Spotpris i öre/kWh**
   * **Namn (Friendly Name):** "Spotpris i öre/kWh"
   * **Typiskt Entitets-ID:** sensor.spotpris\_i\_ore\_kwh (eller sensor.timelpris\_{elområde}\_ore\_spot beroende på HA:s slugifiering)
   * **Tillstånd:** Aktuellt timspotpris, omvandlat till öre/kWh och avrundat till 2 decimaler.
   * **Nyckelattribut:**
     + raw\_today: Lista med dagens timpriser. Varje objekt i listan innehåller { "SEK\_per\_kWh": råvärde, "time\_start": "...", "time\_end": "..." }.
     + tomorrow\_hourly\_prices\_ore: Lista med morgondagens timpriser (rådata från API, men kan formateras om till öre vid behov för min/max).
     + min\_price\_today\_ore, max\_price\_today\_ore: Lägsta/högsta spotpris i öre för idag.
     + min\_price\_tomorrow\_ore, max\_price\_tomorrow\_ore: Lägsta/högsta spotpris i öre för imorgon.
     + last\_api\_data\_update: Senaste lyckade API-anrop (läsbar tid).
     + price\_area: Konfigurerat elområde.
2. **Spotpris + påslag i öre/kWh**
   * **Namn:** "Spotpris + påslag i öre/kWh"
   * **Typiskt Entitets-ID:** sensor.spotpris\_paslag\_i\_ore\_kwh (eller sensor.timelpris\_{elområde}\_ore\_total)
   * **Tillstånd:** Aktuellt timspotpris (i öre) + konfigurerat påslag (i öre), avrundat till 2 decimaler.
   * **Nyckelattribut:**
     + raw\_today: Lista med dagens totala timpriser (spot+påslag) i öre. Varje objekt: { "ore\_per\_kWh": totalpris\_öre, "time\_start": "...", "time\_end": "..." }.
     + spot\_price\_ore: Det aktuella spotpriset i öre som användes i beräkningen.
     + surcharge\_applied\_ore: Det påslag i öre som användes.
     + last\_api\_data\_update, price\_area.
3. **Spotpris i SEK/kWh**
   * **Namn:** "Spotpris i SEK/kWh"
   * **Typiskt Entitets-ID:** sensor.spotpris\_i\_sek\_kwh (eller sensor.timelpris\_{elområde}\_sek\_spot)
   * **Tillstånd:** Aktuellt timspotpris i SEK/kWh, avrundat till 4 decimaler.
   * **Nyckelattribut:**
     + raw\_today: Lista med dagens timpriser. Varje objekt: { "SEK\_per\_kWh": råvärde, "time\_start": "...", "time\_end": "..." }.
     + tomorrow\_hourly\_prices\_sek: Lista med morgondagens timpriser (rådata från API).
     + min\_price\_today\_sek, max\_price\_today\_sek: Lägsta/högsta spotpris i SEK för idag.
     + min\_price\_tomorrow\_sek, max\_price\_tomorrow\_sek: Lägsta/högsta spotpris i SEK för imorgon.
     + last\_api\_data\_update, price\_area.
4. **Spotpris + påslag i SEK/kWh**
   * **Namn:** "Spotpris + påslag i SEK/kWh"
   * **Typiskt Entitets-ID:** sensor.spotpris\_paslag\_i\_sek\_kwh (eller sensor.timelpris\_{elområde}\_sek\_total)
   * **Tillstånd:** Aktuellt timspotpris (i SEK) + konfigurerat påslag (omvandlat till SEK), avrundat till 4 decimaler.
   * **Nyckelattribut:**
     + raw\_today: Lista med dagens totala timpriser (spot+påslag) i SEK. Varje objekt: { "SEK\_per\_kWh": totalpris\_sek, "time\_start": "...", "time\_end": "..." }.
     + spot\_price\_sek: Det aktuella spotpriset i SEK som användes.
     + surcharge\_applied\_sek: Det påslag i SEK som användes.
     + last\_api\_data\_update, price\_area.
5. **Påslag Öre**
   * **Namn:** "Påslag Öre"
   * **Typiskt Entitets-ID:** sensor.paslag\_ore (eller sensor.elpris\_paslag\_ore\_{elområde})
   * **Tillstånd:** Det konfigurerade elpåslaget, direkt från integrationens inställningar, visat i öre/kWh.
   * **Uppdatering:** Uppdateras när integrationen laddas om (t.ex. efter att påslaget ändrats i "options").
   * **Attribut:** Minimala, primärt för att visa värdet.
6. **Påslag SEK**
   * **Namn:** "Påslag SEK"
   * **Typiskt Entitets-ID:** sensor.paslag\_sek (eller sensor.elpris\_paslag\_sek\_{elområde})
   * **Tillstånd:** Det konfigurerade elpåslaget (som är i öre), omvandlat till SEK/kWh och avrundat.
   * **Uppdatering:** Samma som "Påslag Öre".
   * **Attribut:** Minimala.

### **6. Felhantering och Loggning**

* **API-fel:** Om API-anrop misslyckas (t.ex. timeout, HTTP-fel) loggar koordinatorn ett varnings- eller felmeddelande. Sensorerna kommer då inte att få ny data och behåller sina tidigare värden tills en lyckad uppdatering sker. Om \_raw\_current\_spot\_price\_sek inte kan beräknas, sätts sensorernas tillstånd till None (visas som "unknown" eller "-" i UI).
* **Dataparsningsfel:** Felaktig data från API:et (t.ex. ogiltigt JSON, saknade fält) hanteras med varningar i loggen, och de felaktiga posterna hoppas över.
* **DEBUG-loggning:** Omfattande DEBUG-loggning har implementerats i alla delar av integrationen för att underlätta felsökning av datflöden, sensorinitialisering och uppdateringar.

### **7. Nyckelfunktioner och Användarfördelar**

* **Exakt Timvis Pris:** Sensorerna för spotpris och totalpris uppdateras exakt vid varje hel timme.
* **Flexibelt Påslag:** Användaren kan enkelt konfigurera och ändra sitt elpåslag.
* **Dubbla Valutor:** Priser presenteras både i öre/kWh (för tydlighet vid låga priser) och SEK/kWh (standardenhet).
* **Detaljerad Data:** Attributen för timpriser (raw\_today, tomorrow\_hourly\_prices\_...) ger tillgång till komplett prisdata för både innevarande och kommande dygn, vilket kan användas i avancerade automationer, skript eller anpassade Lovelace-kort.
* **Min/Max Priser:** Snabbt överblick över dagens och morgondagens lägsta och högsta priser.
* **Tydlig Påslagsvisning:** Dedikerade sensorer visar det aktuella påslaget.

### **8. Sammanfattning**

"Elpris Timme"-integrationen tillhandahåller en robust och flexibel lösning för att övervaka timvisa elpriser i Home Assistant. Genom en kombination av en datakoordinator för API-anrop och sensorer med precisa timers för uppdatering, får användaren tillgång till aktuell och relevant prisinformation. Konfigurationsmöjligheterna för elområde och påslag, tillsammans med presentationen i både öre och SEK, gör integrationen anpassningsbar för olika behov. De detaljerade attributen öppnar upp för vidare anpassning och användning i hemautomationssystemet.