



Bundesverband für Energie- und
Wasserdatenmanagement e.V.

ERP-Implementierungsleitfaden

Billing Input API v1.1

Version 1.1.0

Oktober 2026 | Zielgruppe: ERP-Anbieter / Softwareentwickler

ERP-Implementierungsleitfaden – bved Billing Input API v1.1

Version: 1.1.0

Datum: 2026-10-01

Hinweis: Dieser Leitfaden wurde mit Hilfe von KI erzeugt.

Zielgruppe: ERP-Anbieter / Softwareentwickler (Wohnungswirtschaft)

Spezifikation: bved Billing Input API v1.1.0 (OpenAPI: [bved-billing-input-1-1.openapi.yaml](#))

Inhalt

1. Einführung und Kontext
 2. Voraussetzungen
 3. Authentifizierung
 4. API-Endpunkte
 5. Vollständige Feldreferenz
 6. ERP-Datenmapping
 7. Implementierungsmuster
 8. Vollständige JSON-Beispiele
 9. Fehlerbehandlung
 10. Logging-Empfehlungen
 11. Implementierungs-Checkliste
 12. Testempfehlungen
-

1. Einführung und Kontext

Was ist die Billing Input API?

Die **bved Billing Input API v1.1** (BI) ist die REST-Schnittstelle, über die eine Hausverwaltung (Property Manager, PM) den **Abrechnungsauftrag** für einen Abrechnungszeitraum an den Messdienstleister (MSC) übermittelt. Der PM liefert dabei die abrechnungsrelevanten Eingangsdaten – Brennstoffrechnungen und Lagerbestände, Heiz- und Hausnebenkosten, CO₂-Konfiguration sowie die abrechnungsbezogenen Parameter je Empfänger (Verteilschlüssel, Vorauszahlungen, MwSt.-Ausweis, ...).

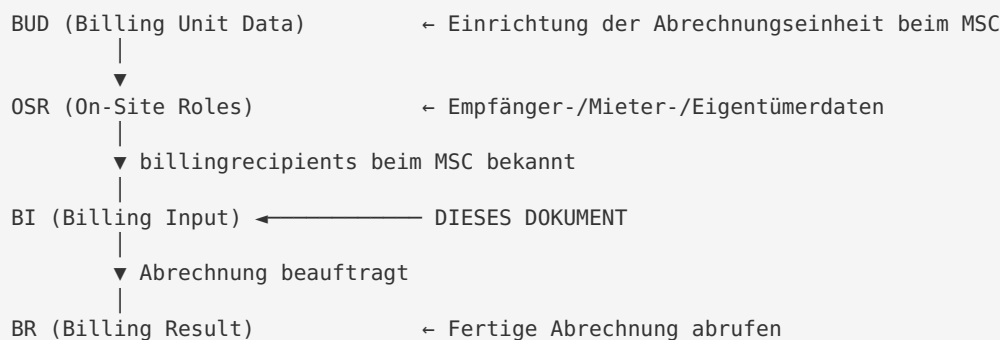
Der Ablauf ist **schablonengesteuert**: Der MSC stellt für jeden Abrechnungszeitraum eine **Abrechnungs-Schablone** (Billing Template) bereit, die die beim MSC konfigurierten

Dienstleistungen, erwarteten Kostenschlüssel und die (zuvor über OSR gemeldeten) Abrechnungsempfänger enthält. Das ERP füllt diese Schablone mit den konkreten Daten und sendet sie zurück.

Übertragen werden:

- **Abrechnungszeiträume** (Lesen) – welche Zeiträume beim MSC offen bzw. abrufbar sind, inkl. Status
- **Abrechnungs-Schablone** (Lesen) – Vorlage mit konfigurierten Diensten, erwarteten Kostenschlüsseln und Empfängern
- **Abrechnungs-Auftrag** (Schreiben) – die ausgefüllten Eingangsdaten, wahlweise nur zur Prüfung oder zur Abrechnungsauslösung

Einordnung in die bved-API-Familie



Abhängigkeit: BI setzt voraus, dass die Abrechnungseinheit und ihre Nutzeinheiten via BUD beim MSC angelegt sind **und** die Abrechnungsempfänger via OSR gemeldet wurden. Die Empfänger in der Schablone stammen aus OSR – **über BI können keine neuen Empfänger angelegt werden**, sondern nur deren abrechnungsbezogene Parameter gesetzt werden.

Kernprinzip: Schablone abrufen, ausfüllen, prüfen, senden

1. GET /billingperiods → Welche Zeiträume sind offen?
2. GET /billingperiods/{ende} → Schablone (Dienste, erwartete Kosten, Empfänger)
3. POST /billingperiods/{ende}?action=VALIDATE → Trockenlauf: nur prüfen, nichts speichern
4. POST /billingperiods/{ende}?action=SEND → Auftrag senden, Abrechnung startet

Zwei zentrale Besonderheiten

1. **Vollständige Ersetzung:** Ein gesendeter Abrechnungsauftrag **überschreibt alle bisherigen Daten desselben Typs** für diesen Abrechnungszeitraum beim MSC. Es werden keine Deltas gesendet, sondern immer der vollständige Stand.
2. **Einmalige Auslösung:** Nach erfolgreichem **SEND / SEND_AND_IGNORE_WARNINGS** **startet die Abrechnung sofort** und der Zeitraum nimmt keine weiteren Eingaben mehr an – ein erneutes

Senden wird mit `409` abgelehnt. `VALIDATE` speichert dagegen nichts und ist beliebig oft wiederholbar.

2. Voraussetzungen

Voraussetzung	Details
OAuth 2.0 Credentials	Client ID und Client Secret vom MSC – nötig für Token-Abruf
<code>billingunitMscnumber</code>	9-stellige MSC-Nummer der AE (aus BUD-Setup)
OSR abgeschlossen	Abrechnungsempfänger müssen vorab über die OSR-API gemeldet sein – sie erscheinen in der Schablone
Kostenschlüssel-Kennntnis	Mapping der ERP-Kostenarten auf bved-Kostenschlüssel (siehe Kostenschlüssel-Spezifikation)
HTTPS/TLS	Alle API-Aufrufe ausschließlich über verschlüsselte Verbindung
Basis-URL	Wird vom jeweiligen MSC bereitgestellt – kein gemeinsamer Endpunkt, jeder MSC hat seine eigene URL

3. Authentifizierung

Die API ist mit **OAuth 2.0 Client Credentials Flow** gesichert.

Token-Abruf

```
POST {tokenUrl}
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&client_id={client_id}
&client_secret={client_secret}
```

Die `tokenUrl` wird vom MSC mitgeteilt.

Token-Verwendung

```
GET /billinginput/v1/billingunits/123456789/billingperiods
Authorization: Bearer {access_token}
```

Implementierungshinweise

- Access Token vor Ablauf erneuern (TTL ist MSC-spezifisch – typisch 3600 Sekunden)
- Token serverseitig cachen, nicht per Request neu abrufen

- Bei **401** -Antwort: Token erneuern und Request einmalig wiederholen
- Credentials nicht im Quellcode oder in Logs speichern
- Der gleiche Token kann für alle bved-APIs desselben MSC wiederverwendet werden, sofern der MSC das so vorsieht

4. API-Endpunkte

Basis-URL: `{msc-base-url}/billinginput/v1`

Alle Antworten enthalten den Response-Header **X-API-VERSION** mit der vom MSC implementierten Version.

4.1 GET – Abrechnungszeiträume lesen

```
GET /billingunits/{billingunit}/billingperiods
GET /billingunits/{billingunit}/billingperiods?status=OPEN|ALL
```

Liefert die verfügbaren Abrechnungszeiträume einer AE inkl. Status.

Query-Parameter:

Parameter	Typ	Pflicht	Beschreibung
status	enum	Nein	OPEN (Default) = nur nicht abgeschlossene Zeiträume; ALL = alle Zeiträume, also auch RELEASED

Antwort bei Erfolg: **200 OK**

```
{
  "periods": [
    { "from": "2023-01-01", "to": "2023-12-31", "status": "READY" }
  ]
}
```

Status eines Abrechnungszeitraums (**BillingPeriodStatus):**

Status	Bedeutung	POST möglich?	Bei status=OPEN enthalten?
DRAFT	Zeitraum ist zur Datenübermittlung geöffnet, die benötigten Messwerte liegen beim MSC aber noch nicht vor	Ja	Ja

Status	Bedeutung	POST möglich?	Bei status=OPEN enthalten?
READY	Zeitraum ist zur Datenübermittlung geöffnet und abrechnungsreif – die Messwerte liegen beim MSC vor	Ja	Ja
CORRECTION	Zeitraum wurde für Korrekturen wieder geöffnet	Ja (überschreibt den bisherigen Stand, siehe §7.5)	Ja
RELEASED	Zeitraum ist freigegeben, die Abrechnung läuft beim MSC	Nein → 409	Nein, nur bei status=ALL

READY ist der eigentliche Startschuss: Bei DRAFT kann das ERP zwar bereits senden, dem MSC fehlen aber noch Messwerte. Wer den Auftrag erst bei READY sendet, vermeidet unnötige Wartezeiten und Rückfragen. Das ERP muss READY in jedem Fall verarbeiten können – es ist ein regulärer Rückgabewert.

4.2 GET – Abrechnungs-Schablone lesen

```
GET /billingunits/{billingunit}/billingperiods/{billing-period-to}
```

Liefert die Schablone für den durch sein **Enddatum** identifizierten Abrechnungszeitraum: beim MSC konfigurierte Dienste, erwartete Kostenschlüssel (mit Markierung REQUIRED / PROPOSED) und die via OSR gemeldeten Abrechnungsempfänger.

Antwort bei Erfolg: 200 OK – Schema BillingTemplate (siehe §5.7 und Beispiel §8.2).

4.3 POST – Abrechnungs-Auftrag senden

```
POST /billingunits/{billingunit}/billingperiods/{billing-period-to}?action=VALIDATE|SEND|SEND_AND_IGNORE_WARNINGS
Content-Type: application/json
```

Sendet den ausgefüllten Abrechnungsauftrag. Der Pflicht-Query-Parameter **action** steuert das Verhalten:

action	Verhalten	Erfolgs-Antwort
VALIDATE	Nur plausibilisieren – es werden keine Daten beim MSC gespeichert	204 No Content
SEND	Senden, aber Abbruch bei Warnungen (und Fehlern)	200 OK + transactionid

action	Verhalten	Erfolgs-Antwort
SEND_AND_IGNORE_WARNINGS	Senden und Warnungen ignorieren – Abrechnung startet trotz Warnungen	200 OK + transactionid

Antwort bei **SEND** / **SEND_AND_IGNORE_WARNINGS** erfolgreich: **200 OK**

```
{ "transactionid": "T-0000042" }
```

Nach **200** **startet die Abrechnung sofort**; für diesen Zeitraum werden keine weiteren Eingaben mehr angenommen.

Antwort bei **VALIDATE** erfolgreich: **204 No Content** – keine Nutzlast, keine Speicherung.

Antwort bei Validierungsfehlern: **400 Bad Request** – strukturiertes **Validation**-Objekt mit Meldungen auf Auftrags- und Positionsebene (siehe §5.8 und Beispiel §8.5).

Antwort bei bereits abgeschlossenem Zeitraum: **409 Conflict** – der Abrechnungsauftrag wurde bereits entgegengenommen; keine neue Eingabe möglich.

4.4 Path-Parameter

Parameter	Feld	Typ	Länge / Format	Beispiel
billingunit	billingunitMscnumber	string	9 Stellen	123456789
billing-period-to	Enddatum des Zeitraums	date	YYYY-MM-DD	2023-12-31

Der Abrechnungszeitraum wird **über sein Enddatum** identifiziert (**billing-period-to**) – dasselbe Datum für **GET**-Schablone und **POST**-Auftrag verwenden.

5. Vollständige Feldreferenz

5.1 BillingInput (POST Request Body)

Feld	Typ	Pflicht	Beschreibung
currency	string (1–3)	Ja	Währung der Abrechnung (z.B. EUR)
expectedvat	enum	Ja	MwSt.-Art: GROSS / NET / VAT
energysources	array	Nein	Brennstoff-/Energiequellen mit Rechnungen und Beständen

Feld	Typ	Pflicht	Beschreibung
co2configuration	object	Nein	CO ₂ -Konfiguration auf AE-Ebene (deutsche Gesetzgebung)
heatingsysteminvoices	array	Nein	Rechnungen zu Heiznebenkosten
ancillaryinvoices	array	Nein	Rechnungen zu Hausnebenkosten
billingrecipients	array	Nein	Abrechnungsbezogene Parameter je Empfänger (Empfänger stammen aus OSR)
riskallocationdefaultpercentage	number (0–9.99)	Nein	Standard-Prozentsatz UAW für die AE

Auf Schema-Ebene sind nur `currency` und `expectedvat` Pflicht. **Welche Kostenpositionen tatsächlich erforderlich sind, gibt die Schablone vor** (Markierung `REQUIRED`, siehe §5.7). Das Senden überschreibt alle bisherigen Daten desselben Typs für den Zeitraum.

5.2 EnergySource (`energysources[]`)

Feld	Typ	Pflicht	Beschreibung
energysourcetype	enum	Ja	Energieträgertyp (siehe unten)
begin / end	date	Nein	Liefer-/Versorgungszeitraum
caloricvalue	number (0–1000)	Nein	Brennwert/Heizwert — erwartet bei Brennstoffrechnung
performancefactor	number (≥1.0)	Nein	Jahresarbeitszahl / Nutzungsfaktor (Wärmepumpe)
totalheatoutput	number (≥1.0)	Nein	Gesamtwärmeleistung (Wärmepumpe)
primaryenergyfactor	number (0–9.99)	Nein	Primärenergiefaktor (Fernwärme)
energysourceinvoices	array	Nein	Brennstoffrechnungen
stock	array	Nein	Lagerbestände (Anfangs-/Restbestand)

energysourcetype — gültige Werte: `LIGHT_FUEL_OIL`, `HEAVY_FUEL_OIL`, `COKE`, `WOOD_PELLETS`, `LIGHT_NATURAL_GAS`, `HEAVY_NATURAL_GAS`, `BIO_GAS`, `LOCAL_HEAT`, `DISTRICT_HEATING`, `LIQUID_GAS`, `ELECTRICITY`, `WOOD`, `WOOD_AIRDRY`, `WOOD_CHIPS`, `GEOTHERMAL`, `LIGNITE`, `HARD_COAL`, `BROWN_COAL`, `ELECTRICITY_FOR_HEAT_PUMP`, `SOLAR_ENERGY`, `NUCLEAR_ENERGY`, `RENEWABLE`, `OTHER`.

energysourceinvoices[] – EnergySourceInvoice:

Feld	Typ	Pflicht	Beschreibung
cost	EnergySourceCost	Ja	Kostenschlüssel (Gruppe 4.0 / 5.0 / 6.0)
quantity	number	Ja	Menge laut Rechnung
amounts	Amount	Ja	Beträge (brutto/netto/MwSt.)
unit	string (2-3)	Ja	Einheit (z.B. LTR , KWH)
invoicedate	date	Nein	Rechnungsdatum
co2emission	CO2Emission	Nein	CO ₂ -Emissionen dieser Rechnung

stock[] – EnergySourceStock:

Feld	Typ	Pflicht	Beschreibung
cost	EnergySourceCost	Ja	Kostenschlüssel
type	enum	Ja	INITIAL (Anfangs-) / REMAINING (Restbestand) – seit v1.1.0 Pflichtfeld
quantity	number	Ja	Bestandsmenge zum Stichtag
unit	string (2-3)	Ja	Einheit
date	date	Nein	Stichtag des Bestands
amounts	Amount	Nein	Beträge
co2emission	CO2Emission	Nein	CO ₂ -Emissionen

5.3 HeatingSystemInvoice (heatingsysteminvoices[])

Feld	Typ	Pflicht	Beschreibung
cost	HeatingSystemCost	Ja	Kostenschlüssel (Gruppe 4.1 / 4.2 / 5.1 / 5.2 / 6.2)
amounts	Amount	Ja	Beträge
invoicedate	date	Nein	Rechnungsdatum
wagecosts	WageCosts	Nein	Lohnkostenanteil (§35a EStG)

5.4 AncillaryInvoice (ancillaryinvoices[])

Feld	Typ	Pflicht	Beschreibung
cost	AncillaryCost	Ja	Kostenschlüssel (nicht Gruppe 4/5/6)
allocationkey	string	Ja	Verteilschlüssel (Muster Vnn.nn[.nn]) – seit v1.0.3 Pflicht
amounts	Amount	Ja	Beträge – seit v1.0.3 Pflicht
invoicedate	date	Nein	Rechnungsdatum
wagecosts	WageCosts	Nein	Lohnkostenanteil

5.5 BillingRecipientInput (billingrecipients[])

Bezieht sich auf **bereits über OSR gemeldete** Empfänger. Hier werden nur deren abrechnungsbezogene Parameter gesetzt – es können **keine neuen Empfänger** angelegt werden.

Feld	Typ	Pflicht	Beschreibung
residentialunitMscnumber	string (4)	Ja	MSC-Nummer der NE
partnerPmnumber	string (1-20)	Ja	Referenz auf den OSR-Partner
validfrom	date	Ja	Gültig-ab im Kontext des Zeitraums
residentialunitPmnumber	string (1-20)	Nein	ERP-Nummer der NE
validto	date	Nein	Gültig-bis im Kontext des Zeitraums
allocations	array	Nein	Verteilschlüssel-Werte des Empfängers
riskallocation	boolean	Nein	UAW für diesen Empfänger relevant
vatdisplay	enum	Nein	MwSt.-Ausweis: NO_VAT_DISPLAY / COMMERCIAL_RENTAL / COMMERCIAL_OWNER / ON_BALANCE
allocatebillingrecipientchange fee	boolean	Nein	Nutzerwechselgebühr diesem Empfänger zuordnen
exceptedcosts	array	Nein	Ausgenommene Kosten (nur cost)

Feld	Typ	Pflicht	Beschreibung
<code>directinvoices</code>	array	Nein	Direktrechnungen des Empfängers
<code>budgetpayments</code>	array	Nein	Vorauszahlungen des Empfängers

`allocations[]` — BillingRecipientAllocation:

Feld	Typ	Pflicht	Beschreibung
<code>key</code>	string	Ja	Verteilschlüssel (Muster <code>Vnn.nn[.nn]</code>)
<code>value</code>	number (0–99999999.99)	Ja	Wert des Verteilschlüssels
<code>description</code>	string	Nein	Beschreibung
<code>validfrom</code> / <code>validto</code>	date	Nein	Nur wenn sich der Schlüssel innerhalb des Zeitraums ändert

`budgetpayments[]` — BillingRecipientBudgetPayment:

Feld	Typ	Pflicht	Beschreibung
<code>type</code>	enum	Ja	<code>OVERALL</code> / <code>HEATING_AND_HOT_WATER</code> / <code>HEATING</code> / <code>HOT_WATER</code> / <code>ANCILLARY</code>
<code>amounts</code>	Amount	Ja	Betrag der Vorauszahlung

`directinvoices[]` — DirectInvoice: `cost` (Ja), `amounts` (Ja), `invoicedate`, `wagecosts`, `text` (jeweils Nein). **`exceptedcosts[]` — ExceptedCost:** `cost` (Ja).

5.6 CO2Configuration (`co2configuration`) und CO2Emission

`co2configuration` (alle Felder optional):

Feld	Typ	Beschreibung
<code>condominium</code>	boolean	Wohnungseigentum (WEG)
<code>totalfloorspace</code>	number (0–99999.99)	Gesamtfläche AE, falls abweichend von Summe der Empfänger
<code>nonresidential</code>	boolean	>50 % Nichtwohngebäude (§8 CO2KostAufG)
<code>energyimprovementnotpossible</code>	boolean	Energet. Verbesserung öffentl.-rechtl. ausgeschlossen (§9/§172)

Feld	Typ	Beschreibung
<code>heatandhotwaterimprovementnotpossible</code>	boolean	Wärme-/Warmwasser-Verbesserung ausgeschlossen
<code>co2sharelandlord</code>	integer (0–100)	Vermieteranteil CO ₂ , falls abweichend von MSC-Standard
<code>districtheatingconnection</code>	date	Datum Fernwärmeanschluss (Stichtag 1.1.2023)

co2emission – CO2Emission: `quantity` (Ja, kg), `cost` (Ja, Amount), `factor` (Nein, kg/kWh), `energymix` (Nein, max. 6 Einträge). Seit v1.0.3 ist der Gesamt- `factor` auf AE-Ebene **kein** Pflichtfeld mehr. Seit v1.1.0 sind für `quantity` auch **negative Werte zulässig** (Wertebereich –999999.999 bis 999999.999) – für Korrekturbuchungen; ebenso ist `energymix` seither auch als **leere Liste** erlaubt.

energymix[] – CO2EnergyMix: `energysourcetype` (Ja), `share` (Ja, 1–100 %), `emissionfactor` (Ja, kg/kWh).

5.7 Schablonen-Schemas (Antwort von GET)

BillingTemplate – Pflicht: `currency`, `expectedvat`, `energysources`, `billingrecipients`.

Feld	Beschreibung
<code>currency</code> , <code>expectedvat</code>	Wie beim MSC konfiguriert
<code>energysources[]</code>	Gültige Energieträger mit erwarteten Kostenschlüsseln, Beständen, Brennwert usw.
<code>heatingsystemcosts[]</code> / <code>ancillarycosts[]</code>	Erwartete Kostenschlüssel für Heiz- bzw. Hausnebenkosten
<code>co2configuration</code>	CO ₂ -Konfiguration
<code>billingrecipients[]</code>	Empfänger wie über OSR gemeldet (inkl. <code>name</code> , <code>vacancy</code> , <code>allocations</code>)
<code>riskallocationdefaultpercentage</code>	Vorhandensein zeigt an, dass der UAW-Dienst für den Zeitraum aktiv ist

expected -Markierung je Kostenschlüssel (`BillingTemplateExpectedCosts`):

Wert	Bedeutung	ERP-Verhalten
REQUIRED	Kosten sind erforderlich; Fehlen führt zu Validierungsfehler	ERP muss diese Kosten sicherstellen

Wert	Bedeutung	ERP-Verhalten
PROPOSED	Kosten vorgeschlagen (Typ oder Vorperioden)	ERP sollte sie dem Nutzer anbieten
(ohne)	Kann je nach Kontext / MSC-Absprache gesendet werden	optional

5.8 Validation (Antwort bei 400)

Das `Validation`-Objekt liefert Meldungen **strukturiert entlang des Auftrags**, sodass das ERP Fehler exakt zuordnen kann:

Feld	Beschreibung
<code>messages</code>	Pflicht – Meldungen auf Auftragsebene
<code>energysources[]</code>	Meldungen je Energiequelle inkl. <code>energysourceinvoices[]</code> und <code>stock[]</code>
<code>co2configuration</code>	Meldungen zur CO ₂ -Konfiguration
<code>heatingsysteminvoices[]</code> / <code>ancillaryinvoices[]</code>	Meldungen je Rechnung (mit <code>cost</code> -Referenz)
<code>billingrecipients[]</code>	Meldungen je Empfänger inkl. <code>allocations[]</code> , <code>exceptedcosts[]</code> , <code>directinvoices[]</code> , <code>budgetpayments[]</code>

Jede `messages`-Liste enthält `ValidationMessage`-Einträge (`type`: `ERROR` / `WARNING`, `shortmessage`, `message`, `language`).

5.9 Gemeinsame Bausteine

amounts – **Amount-Objekt**: `grossamount`, `vatamount`, `netamount` (jeweils -9.999.999,99 ... 9.999.999,99), `vatrate` (0–100), `text`. Dezimaltrennzeichen ist der Punkt.

Kostenschlüssel (`cost.key`) nach bved-Kostenschlüssel-Spezifikation:

Kontext	Erlaubte Gruppen	Muster
Energieträger (<code>EnergySourceCost</code>)	4.0 / 5.0 / 6.0	<code>^[456]\.0\.\d{1,3}\$</code>
Heiznebenkosten (<code>HeatingSystemCost</code>)	4.1 / 4.2 / 5.1 / 5.2 / 6.2	<code>^[456]\.[12]\.\d{1,3}\$</code>
Hausnebenkosten (<code>AncillaryCost</code>)	nicht 4/5/6	<code>^[123789]\d?\.\d{1,3}?\.\d{1,3}\$</code>

Jeder `cost` trägt zusätzlich `accountinggroup` (integer; relevant bei mehrfach vorkommenden Schlüsseln) und optional `text` (1–25 Zeichen, abweichende Bezeichnung).

Verteilschlüssel (`allocationkey` / `key`): Muster `^\d{2}\.\d{2}(?:\.\d{2})?$` (z.B. `V02.01`). Seit v1.0.3 existiert zusätzlich die Gruppe `V05`.

wagecosts – **WageCosts:** `wagetype` (Ja: `PAR35A_ABS1` / `PAR35A_ABS2` / `PAR35A_ABS3`), `amounts` (Ja).

6. ERP-Datenmapping

6.1 Abrechnungszeitraum

ERP-Feld	API	Hinweis
Abrechnungsjahr/-zeitraum	<code>billing-period-to</code> (Path)	Enddatum als Identifikator; via <code>GET / billingperiods</code> ermitteln
Offen/abrechenbar	<code>status</code>	Nur offene Zeiträume beauftragen (<code>DRAFT</code> / <code>READY</code> / <code>CORRECTION</code>), nicht <code>RELEASED</code>

6.2 Brennstoff-/Energiedaten → `energysources[]`

ERP-Feld	API-Feld	Besonderheit
Brennstoffart	<code>energysourcetype</code>	Auf Enum mappen
Lieferzeitraum	<code>begin</code> / <code>end</code>	
Brennstoffrechnung	<code>energysourceinvoices[]</code>	<code>cost</code> , <code>quantity</code> , <code>amounts</code> , <code>unit</code> sind Pflicht
Brennwert / JAZ / PEF	<code>caloricvalue</code> / <code>performancefactor</code> / <code>primaryenergyfactor</code>	Je nach Anlagentyp
Anfangs-/ Restbestand	<code>stock[]</code> (<code>type</code> <code>INITIAL</code> / <code>REMAINING</code>)	Restbestand oft negativ (Verbrauchslogik)
CO ₂ -Emissionen	<code>co2emission</code>	<code>quantity</code> + <code>cost</code> Pflicht, <code>factor</code> / <code>energymix</code> optional

6.3 Nebenkosten → `heatingsysteminvoices[]` / `ancillaryinvoices[]`

ERP-Feld	API-Feld	Besonderheit
Heiznebenkosten-Rechnung	<code>heatingsysteminvoices[]</code>	<code>cost</code> + <code>amounts</code> Pflicht

ERP-Feld	API-Feld	Besonderheit
Hausnebenkosten-Rechnung	<code>ancillaryinvoices[]</code>	<code>cost + allocationkey + amounts</code> Pflicht
Lohnkostenanteil (§35a)	<code>wagecosts</code>	<code>wagetype + amounts</code>
ERP-Kostenart	<code>cost.key (+ accountinggroup)</code>	Auf bved-Kostenschlüssel mappen; Gruppe je Kontext beachten

6.4 Empfänger-Parameter → `billingrecipients[]`

ERP-Feld	API-Feld	Besonderheit
NE / Partner	<code>residentialunitMscnumber / partnerPmnumber</code>	Muss zur OSR-Meldung passen
Gültigkeit im Zeitraum	<code>validfrom / validto</code>	<code>validfrom</code> Pflicht
Verteilschlüssel-Werte	<code>allocations[] (key , value)</code>	Flächen/Einheiten je Schlüssel
Vorauszahlungen	<code>budgetpayments[] (type , amounts)</code>	Nach Kostenblock getrennt
MwSt.-Ausweis	<code>vatdisplay</code>	Bei gewerblichen Empfängern relevant
UAW relevant	<code>riskallocation</code>	
Direktkosten	<code>directinvoices[]</code>	Empfängerindividuelle Rechnungen
Ausgenommene Kosten	<code>exceptedcosts[]</code>	Kosten, die diesen Empfänger nicht treffen

Empfänger stammen aus OSR: Die zu befüllenden Empfänger liefert die Schablone (`GET`). Enthält die Schablone einen Empfänger nicht, der abgerechnet werden soll, ist das ein OSR-Thema — **nicht** über BI lösbar.

7. Implementierungsmuster

7.1 Grundregel: Schablone ist die Vorlage

Vor jedem Abrechnungsauftrag die **aktuelle Schablone abrufen** (`GET`) und die eigenen Daten daran ausrichten: Es sind die vom MSC konfigurierten Energieträger und Kostenschlüssel zu verwenden,

REQUIRED -Kosten müssen befüllt werden, und die Empfängerliste stammt verbindlich aus der Schablone.

Zusätzliche Kostenschlüssel sind zulässig, wenn sie mit dem MSC vereinbart wurden. Die Schablone gibt die vom MSC erwarteten Kostenschlüssel vor (**REQUIRED** / **PROPOSED**), schließt darüber hinausgehende aber nicht grundsätzlich aus — siehe die **expected** -Markierung in §5.7. Solche Schlüssel sind vorab bilateral mit dem MSC abzustimmen; ob sie ohne Absprache akzeptiert oder mit **NOT_EXPECTED** abgelehnt werden, ist MSC-abhängig.

7.2 Muster 1: Regulärer Abrechnungsauftrag

1. GET /billingperiods?status=OPEN
→ offener Zeitraum, z.B. to = 2023-12-31, status = DRAFT/READY
2. GET /billingperiods/2023-12-31
→ Schablone: Energieträger, erwartete Kostenschlüssel, Empfänger
3. Auftrag im ERP aufbauen (Schablone ausfüllen)
4. POST /billingperiods/2023-12-31?action=VALIDATE
→ 204 = fehlerfrei | 400 = Validation-Objekt auswerten, korrigieren, erneut VALIDATE
5. POST /billingperiods/2023-12-31?action=SEND
→ 200 + transactionid → Abrechnung startet

7.3 Muster 2: Validieren vor dem Senden (dringend empfohlen)

VALIDATE speichert nichts und ist beliebig oft wiederholbar. Empfohlener Zyklus: **so oft VALIDATE**, **bis fehlerfrei**, erst dann **SEND**. So wird die einmalige, nicht zurücknehmbare Auslösung nicht mit vermeidbaren Fehlern „verbraucht“.

```
do {
  POST ...?action=VALIDATE
} while (Antwort == 400) // Fehler beheben, erneut prüfen
POST ...?action=SEND    // erst bei 204 senden
```

7.4 Muster 3: Umgang mit Warnungen

SEND bricht sowohl bei Fehlern als auch bei **Warnungen** ab (**400**). Sind die Warnungen fachlich vertretbar und bewusst akzeptiert, mit **SEND_AND_IGNORE_WARNINGS** senden. Fehler (**type: ERROR**) können **nie** ignoriert werden — sie führen immer zu **400** .

Situation	Aktion
Nur Fehler (ERROR)	Korrigieren; SEND_AND_IGNORE_WARNINGS hilft nicht
Nur Warnungen (WARNING), fachlich ok	SEND_AND_IGNORE_WARNINGS
Warnungen unklar	Klären, ggf. korrigieren, dann SEND

7.5 Muster 4: Korrektur eines Zeitraums (CORRECTION)

Ein bereits abgerechneter Zeitraum kann vom MSC in den Status **CORRECTION** zurückversetzt werden. Erst dann nimmt **POST** wieder Eingaben an. Der erneute Auftrag **überschreibt** den bisherigen Stand vollständig — also stets den kompletten, korrigierten Datensatz senden.

7.6 Retry-Verhalten (wichtiger Unterschied zu BUD/OSR)

BI ist **nicht** frei idempotent: Ein erfolgreicher **SEND** ist einmalig.

- **VALIDATE** ist gefahrlos wiederholbar (keine Speicherung).
- **SEND** bei **Netzwerkfehler/Timeout**: identischen Body erneut senden. Kommt daraufhin ein **409**, war die ursprüngliche Übermittlung bereits erfolgreich → als Erfolg werten (ggf. **transactionid** fehlt dann, per Support/Logging klären).
- **Kein automatischer Retry** bei **400** — erst Fehler beheben.

8. Vollständige JSON-Beispiele

Alle Beispiele verwenden AE **123456789**, Zeitraum **2023-01-01 – 2023-12-31**, Energieträger Heizöl.

8.1 GET /billingperiods?status=ALL — Antwort

```
{
  "periods": [
    { "from": "2023-01-01", "to": "2023-12-31", "status": "READY" },
    { "from": "2022-01-01", "to": "2022-12-31", "status": "RELEASED" },
    { "from": "2021-01-01", "to": "2021-12-31", "status": "CORRECTION" }
  ]
}
```

Mit dem Default **status=OPEN** würde derselbe Aufruf nur die Zeiträume **2023 (READY)** und **2021 (CORRECTION)** liefern — **RELEASED** wird ausgefiltert.

8.2 GET /billingperiods/2023-12-31 — Schablone (Auszug)

```
{
  "currency": "EUR",
  "expectedvat": "GROSS",
  "energysources": [
    {
      "type": "LIGHT_FUEL_OIL",
      "costs": [
        { "cost": { "key": "6.0.1", "accountinggroup": 1, "text": "Energiekosten" },
          "unit": "LTR", "expected": "REQUIRED" }
      ],
      "caloricvalue": 10,
      "performancefactor": 1,
      "primaryenergyfactor": 2.4
    }
  ],
  "ancillarycosts": [
    { "cost": { "key": "2.0.3", "accountinggroup": 1, "text": "Kaltwasser" },
      "allocationkey": "V02.01", "expected": "REQUIRED" }
  ],
  "billingrecipients": [
    {
      "residentialunitMscnumber": "0001",
      "partnerPmnumber": "117-312-B0010",
      "validfrom": "2023-01-01",
      "validto": "2023-12-31",
      "vacancy": false,
      "name": "Metzgerei Rinderer",
      "allocations": [
        { "key": "V02.01", "value": 250, "description": "Wohnfläche gesamt" },
        { "key": "V02.02", "value": 200, "description": "Wohnfläche beheizt" }
      ]
    }
  ],
  "riskallocationdefaultpercentage": 1.8
}
```

8.3 POST /billingperiods/2023-12-31?action=SEND – Auftrag (Auszug)

```

{
  "currency": "EUR",
  "expectedvat": "GROSS",
  "energysources": [
    {
      "begin": "2023-01-01",
      "end": "2023-12-31",
      "energysourcetype": "LIGHT_FUEL_OIL",
      "caloricvalue": 10,
      "primaryenergyfactor": 2.4,
      "energysourceinvoices": [
        {
          "cost": { "key": "4.0.1", "accountinggroup": 1, "text": "Energiekosten" },
          "quantity": 2000.0,
          "invoicedate": "2023-03-01",
          "amounts": { "grossamount": 1900.00, "vatamount": 303.36, "netamount": 1596.64,
            "vatrate": 16, "text": "Heizöl" },
          "unit": "LTR",
          "co2emission": {
            "quantity": 5345.0,
            "cost": { "grossamount": 190.80, "text": "CO2" },
            "energymix": [ { "energysourcetype": "LIGHT_FUEL_OIL", "share": 100,
              "emissionfactor": 3.1 } ]
          }
        }
      ],
      "stock": [
        { "cost": { "key": "4.0.4", "accountinggroup": 1, "text": "Heizöl" }, "type": "INITIAL", "date": "2023-01-01", "quantity": 2200.0, "unit": "LTR" },
        { "cost": { "key": "4.0.4", "accountinggroup": 1, "text": "Heizöl" }, "type": "REMAINING", "date": "2023-12-31", "quantity": -500.0, "unit": "LTR" }
      ]
    }
  ],
  "co2configuration": {
    "condominium": false,
    "totalfloorspace": 525.0,
    "nonresidential": false,
    "co2sharelandlord": 30
  },
  "ancillaryinvoices": [
    {
      "cost": { "key": "2.0.3", "accountinggroup": 1, "text": "Kaltwasser" },
      "allocationkey": "V02.01",
      "invoicedate": "2023-12-31",
      "amounts": { "grossamount": 1029.0, "vatamount": 67.32, "netamount": 961.68,
        "vatrate": 7, "text": "Kaltwasser" }
    }
  ],
  "billingrecipients": [
    {
      "residentialunitMscnumber": "0001",
      "residentialunitPmnumber": "117-312-R001",
      "partnerPmnumber": "117-312-B0010",
      "validfrom": "2023-01-01",
      "validto": "2023-12-31",
      "allocations": [
        { "key": "V02.01", "value": 250.0, "description": "Wohnfläche gesamt" },
        { "key": "V02.02", "value": 200.0, "description": "Wohnfläche beheizt" }
      ],
      "riskallocation": false,
      "vatdisplay": "ON_BALANCE",
      "allocatebillingrecipientchange": true,
      "budgetpayments": [

```

```

    { "type": "OVERALL", "amounts": { "grossamount": 118.76, "vatamount": 22.57,
      "netamount": 96.20, "vatrate": 19, "text": "Gesamtkosten" } }
  ]
},
"riskallocationdefaultpercentage": 1.8
}

```

8.4 POST erfolgreich (SEND) — Antwort 200

```
{ "transactionid": "T-0000042" }
```

8.5 POST mit Validierungsfehler — Antwort 400

```

{
  "messages": [
    { "type": "WARNING", "shortmessage": "IMPLAUSIBLE", "message": "Restbestand höher als
      Anfangsbestand + Zukauf", "language": "de" }
  ],
  "ancillaryinvoices": [
    {
      "cost": { "key": "2.0.3", "accountinggroup": 1, "text": "Kaltwasser" },
      "messages": [
        { "type": "ERROR", "shortmessage": "INCOMPLETE", "message": "allocationkey
          fehlt", "language": "de" }
      ]
    }
  ],
  "billingrecipients": [
    {
      "residentialunitMscnumber": "0001",
      "partnerPmnumber": "117-312-B0010",
      "validfrom": "2023-01-01",
      "messages": [
        { "type": "ERROR", "shortmessage": "MISSING", "message":
          "Pflicht-Verteilschlüssel V02.01 fehlt", "language": "de" }
      ]
    }
  ]
}

```

8.6 POST action=VALIDATE erfolgreich — Antwort 204

Kein Body. Bedeutet: fehlerfrei, aber **nicht** gesendet — für die Auslösung folgt ein **SEND**.

9. Fehlerbehandlung

9.1 HTTP-Statuscodes

Status	Bedeutung	ERP-Maßnahme
200	<code>SEND</code> / <code>SEND_AND_IGNORE_WARNINGS</code> erfolgreich – Abrechnung startet	<code>transactionid</code> protokollieren, Zeitraum als beauftragt markieren
204	<code>VALIDATE</code> erfolgreich (keine Speicherung)	Auftrag ist versandbereit – mit <code>SEND</code> auslösen
400	Validierungsfehler (<code>Validation</code> -Objekt)	Meldungen strukturiert auswerten (Auftrags-/ Positionsebene), Sachbearbeiter anzeigen
401	Token ungültig/abgelaufen	Token neu abrufen, Request einmalig wiederholen
403	Keine Berechtigung	Vertragliche Klärung mit MSC; keine automatische Wiederholung
404	AE oder Abrechnungszeitraum unbekannt	<code>billingunitMscnumber</code> / <code>billing-period-to</code> prüfen
409	Zeitraum bereits abgerechnet – keine neue Eingabe	Kein Retry; ggf. beim MSC <code>CORRECTION</code> anstoßen
500	Technischer Fehler beim MSC	Exponentiellen Backoff; MSC mit <code>transactionid</code> (falls vorhanden) kontaktieren

9.2 ValidationMessage – shortmessage Routing

shortmessage	Typische Ursache bei BI	Lösungsansatz
<code>INCOMPLETE</code>	Pflichtfeld fehlt (z.B. <code>allocationkey</code> , <code>amounts</code> , <code>currency</code>)	Datenqualität prüfen; seit v1.0.3 sind <code>allocationkey</code> und <code>amounts</code> Pflicht
<code>MISSING</code>	Erwarteter (<code>REQUIRED</code>) Kostenschlüssel/ Verteilschlüssel fehlt	Schablone abgleichen, fehlende Position ergänzen
<code>NOT_EXPECTED</code>	Kostenschlüssel/Energieträger nicht in der Schablone konfiguriert. Ob dies zur Ablehnung führt, ist MSC-abhängig – ein Teil der Messdienste akzeptiert zusätzliche Schlüssel	Zusätzliche Positionen mit dem MSC abstimmen (§7.1); andernfalls auf konfigurierte Positionen beschränken
<code>IMPLAUSIBLE</code>	Werte-Konflikte (Bestände, Beträge, Datums Grenzen, Faktoren außerhalb Bereich)	Werte korrigieren

`message` ist MSC-spezifisch und nicht standardisiert. `shortmessage` ist der verlässliche Routing-Key. Die **Position** der Meldung im **Validation**-Baum (z.B. `billingrecipients[].allocations[]`) zeigt zusätzlich, **wo** der Fehler sitzt.

9.3 Retry-Strategie

Netzwerkfehler / Timeout bei **VALIDATE**:
 → gefahrlos identisch wiederholen (keine Speicherung)

Netzwerkfehler / Timeout bei **SEND**:
 → identischen Body erneut senden
 → Antwort 409 = ursprüngliches **SEND** war erfolgreich → als Erfolg werten, `transactionid` per Logging/Support klären

HTTP 400 (Validierung):
 → KEIN automatischer Retry – Meldungen auswerten, korrigieren, erneut **VALIDATE**

HTTP 401:
 → Token erneuern, einmaliger Retry

HTTP 403 / 404 / 409:
 → KEIN automatischer Retry – manuelle Klärung

HTTP 500:
 → Exponentieller Backoff: 1 Min, 5 Min, 15 Min, 1 Std → dann Eskalation

9.4 Häufige Fehler und ihre Ursachen

Symptom	Wahrscheinliche Ursache	Lösung
400 INCOMPLETE an Rechnungen	<code>amounts</code> oder <code>allocationkey</code> fehlt (seit v1.0.3 Pflicht)	Pflichtfelder ergänzen
400 NOT_EXPECTED an Kostenschlüssel	Schlüssel nicht in der Schablone und nicht mit dem MSC vereinbart – MSC-abhängig , viele MSC akzeptieren zusätzliche Schlüssel	Zusätzliche Schlüssel vorab bilateral mit dem MSC abstimmen; im Zweifel auf Schablonen-Schlüssel beschränken (§7.1)
400 MISSING an Empfänger	REQUIRED -Position/Verteilschlüssel fehlt	Schablone gegenprüfen
409 bei SEND	Zeitraum bereits abgerechnet	MSC um CORRECTION bitten
404 bei POST	Falsches <code>billing-period-to</code> oder AE	Datum aus <code>GET /billingperiods</code> verwenden
Kostenschlüssel abgelehnt	Falsche Gruppe für den Kontext	Gruppe je Kontext beachten (§5.9)
Empfänger nicht abrechenbar	Empfänger nicht via OSR gemeldet	OSR nachziehen – nicht über BI lösbar

10. Logging-Empfehlungen

Feld	Pflicht	Zweck
Zeitstempel (ISO 8601)	Ja	Reihenfolge der Auftragsversionen rekonstruieren
HTTP-Methode + Pfad	Ja	GET (Perioden/Schablone) vs. POST unterscheiden
<code>billingunitMscnumber</code>	Ja	Nachvollziehbarkeit, MSC-Support
<code>billing-period-to</code>	Ja	Zeitraum-Zuordnung
<code>action</code> (VALIDATE/SEND/...)	Ja	Trockenlauf vs. Auslösung unterscheiden
HTTP-Status	Ja	Fehleranalyse
<code>transactionid</code> aus Response	Ja (bei 200)	MSC-Support-Anfragen identifizieren
Anzahl <code>energysources</code> / <code>ancillaryinvoices</code> / <code>billingrecipients</code>	Empfohlen	Vollständigkeitsprüfung
Validierungsmeldungen bei 400	Ja	Fehlerverfolgung (inkl. Position im Baum)
Benutzer / Trigger-Quelle	Empfohlen	Wer/was hat den Auftrag ausgelöst

11. Implementierungs-Checkliste

Setup

- ☐ OAuth 2.0 Client Credentials vom MSC erhalten
- ☐ `tokenUrl` und Basis-URL des MSC konfiguriert
- ☐ `billingunitMscnumber` aus BUD verfügbar
- ☐ OSR-Empfänger für die AE gemeldet
- ☐ Mapping ERP-Kostenarten → bved-Kostenschlüssel vorbereitet

Ablauf

- ☐ `GET /billingperiods` implementiert (Filter `OPEN` / `ALL`)
- ☐ Alle vier Werte von `BillingPeriodStatus` verarbeitet (`DRAFT` , `READY` , `RELEASED` , `CORRECTION`) – unbekannter Status bricht die Verarbeitung nicht ab
- ☐ Auftrag bevorzugt bei `READY` gesendet (Messwerte liegen beim MSC vor)
- ☐ `GET /billingperiods/{ende}` (Schablone) implementiert
- ☐ Auftrag aus Schablone aufgebaut (`REQUIRED` -Kosten sichergestellt)
- ☐ `POST ...?action=VALIDATE` als Trockenlauf implementiert
- ☐ `POST ...?action=SEND` erst nach fehlerfreier Validierung
- ☐ `SEND_AND_IGNORE_WARNINGS` nur bei bewusst akzeptierten Warnungen

Datenmapping

- ☐ `currency` und `expectedvat` gesetzt
- ☐ Energieträger auf `energysourcetype` -Enum gemappt
- ☐ Rechnungen mit `amounts` (Pflicht) und – bei Nebenkosten – `allocationkey` (Pflicht)
- ☐ Kostenschlüssel-Gruppen je Kontext korrekt (`4.0/5.0/6.0` , `4.1/4.2/...` , sonst)
- ☐ Bestände `INITIAL` / `REMAINING` gepflegt
- ☐ CO₂: `co2emission.quantity` + `cost` gesetzt (Gesamt- `factor` optional)
- ☐ Empfänger aus Schablone übernommen, `allocations` / `budgetpayments` / `vatdisplay` befüllt
- ☐ Beträge numerisch mit Punkt-Dezimaltrennzeichen

Fehlerbehandlung

- ☐ `204` (VALIDATE) → versandbereit
- ☐ `200` (SEND) → `transactionid` protokolliert, Zeitraum als beauftragt markiert
- ☐ `400` → `Validation` -Baum ausgewertet, `shortmessage` + Position angezeigt
- ☐ `401` → Token-Refresh + einmaliger Retry

- ☐ 403 / 404 / 409 → kein Auto-Retry, manuelle Klärung
- ☐ 500 → Exponentieller Backoff
- ☐ SEND -Timeout → identischer Retry; 409 als Erfolg werten

12. Testempfehlungen

Testumgebung

Für Integrationstests steht die **bved Testplattform** zur Verfügung. Sie bildet das Verhalten eines echten MSC nach und antwortet auf BI-Anfragen mit realistischen Mock-Daten.

- **Testplattform / Swagger UI:** <https://bved-api.de/testplatform/swagger-ui/index.html>

Empfohlene Testszenarien

Test	Erwartetes Ergebnis
GET /billingperiods ohne Filter	200 mit offenen Zeiträumen (DRAFT , READY , CORRECTION)
GET /billingperiods?status=ALL	200 inkl. RELEASED
Zeitraum in READY	ERP erkennt ihn als abrechnungsreif und sendet den Auftrag
GET /billingperiods/{ende}	200 mit Schablone (Dienste, erwartete Kosten, Empfänger)
POST ...?action=VALIDATE mit gültigem Auftrag	204 (keine Speicherung)
POST ...?action=VALIDATE mit fehlendem allocationkey	400 + INCOMPLETE an der Rechnung
POST ...?action=VALIDATE mit nicht konfiguriertem Kostenschlüssel	400 + NOT_EXPECTED
POST ...?action=SEND mit Warnungen	400 (Abbruch bei Warnung)
POST ...?action=SEND_AND_IGNORE_WARNINGS mit Warnungen	200 + transactionid
POST ...?action=SEND gültig	200 + transactionid, Abrechnung startet
POST ...?action=SEND auf bereits abgerechneten Zeitraum	409

ERP-Implementierungsleitfaden – bved Billing Input API v1.1

Test	Erwartetes Ergebnis
POST mit unbekanntem <code>billing-period-to</code>	404
Token-Ablauf während Request	401 , nach Token-Refresh erfolgreich