

Carbon Certificate Management System

By

Dhruvika Rajput

ITM SLS Baroda University

Bachelors in Engineering (CSE)

SAP Educate to Employ (E2E)

Software Development Track

Index

1. **Project Description**
2. **Database Schema (P)**
 - 2.1. ZCFM_SOURCE – Source Table
 - 2.2. ZCFM_FACTOR – Factor Table
 - 2.3. ZCFM_USAGE_LOG – Usage Log Table
 - 2.4. ZCFM_EMISSION – Emission Table
 - 2.5. ZCFM_CREDIT – Credit Score Table
 - 2.6. ZCFM_CERTIFICATE – Certificate Issue Table
3. **Core Functionality (ABAP Classes)**
 - 3.1. ZCL_CFM_DATA - Handle Insertion & validation
 - 3.2. ZCL_DATA_MANAGER – CRUD Operations
 - 3.3. ZCL_EMISSION_REPORT – Calculate Emission
 - 3.4. ZCL_CERTIFICATE - Generate Certificate
 - 3.5. ZCL_SUMMARY_REPORT – Dashboard
4. **SQL Usage**
 - 4.1. Join Operations
 - 4.2. Code Pushdown

RESTFUL APPLICATION

5. **Project Description**
6. **Database Schema (Persistence Layer)**
 - 6.1. ZEM_SOURCES – Source Table
 - 6.2. ZEM_FACTOR_TABLE – Factor Table
 - 6.3. ZEM_USAGE_LOG – Usage Log Table
 - 6.4. ZEM_CERTIFICATE – Certificate Issue Table
7. **CDS Views (Data Model Layer)**
 - 7.1. Data Definitions
 - 7.1.1. ZEM_USAGE_ENTITY
 - 7.1.2. ZEM_EMISSION
 - 7.1.3. ZI_CREDIT_REPORT
 - 7.1.4. ZI_ELIGCERTIFICATE
 - 7.2. *Metadata Extensions (UI)*
 - 7.2.1. ZC_EM_USAGE_LOG000
 - 7.2.2. ZI_EMISSIONREPORT_UI
 - 7.2.3. ZI_CERTIFICATE_UI
8. **Service Definition & Binding**
 - 8.1. Service Definitions
 - 8.2. Service Bindings
 - 8.2.1. USAGE_LOG_TABLE
 - 8.2.2. EMISSION REPORT
 - 8.2.3. CERTIFICATE SUMMARY DASHBOARD
9. **Behaviour Implementation Classes**
 - 9.1. ZBP_CFM_USAGE_ENTITY
 - 9.2. ZBP_CERTIFY
10. **Final Outcome**
11. **Conclusion**

1. Project Description

Overview

Carbon Certificate Management System is a SAP ABAP-based backend solution designed for enterprises to manage, monitor, and certify carbon emissions. It captures energy usage, calculates emissions using defined emission factors, assigns carbon credits, and generates official carbon certificates with automated expiry and duplicate checks. The system supports sustainability tracking and environmental compliance in alignment with ESG goals.

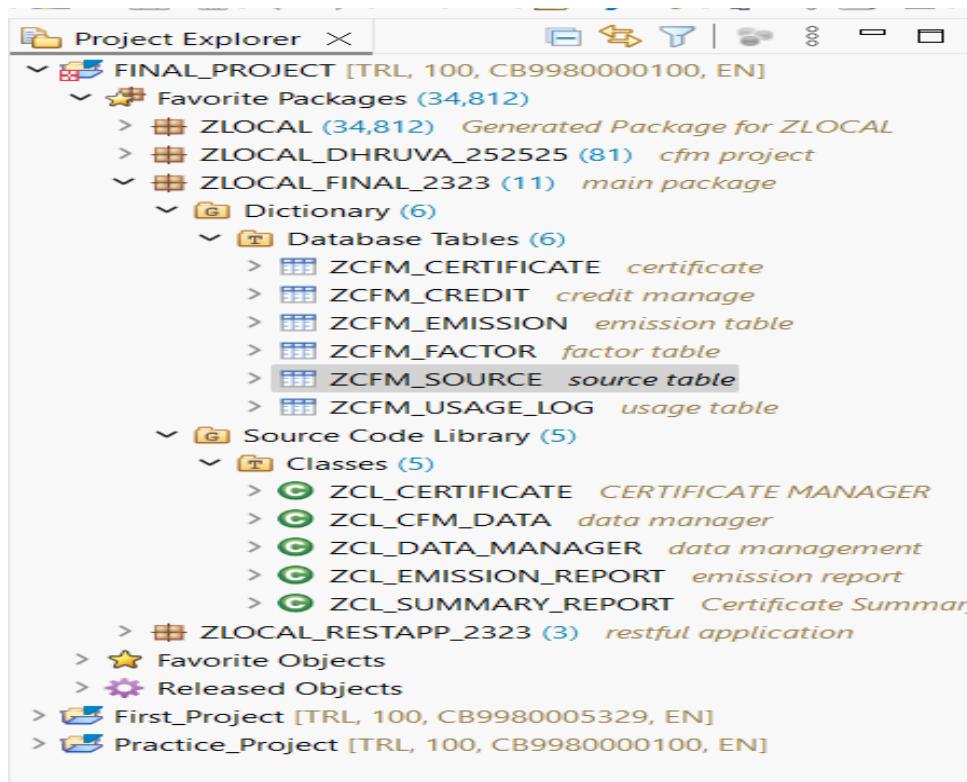
Key Features

- **Source & Factor Management:** Add, update, and validate emission sources and their factors.
- **Usage Logging:** Store validated energy usage logs with user attribution.
- **Emission Calculation:** Auto-compute CO₂ emissions using factor-based formulas.
- **Credit Assignment:** Classify emissions and assign credits dynamically.
- **Certificate Issuance:** Create carbon certificates with expiry, duplicate detection, and status control.
- **Reporting:** Generate emission summaries and certificate audit logs.

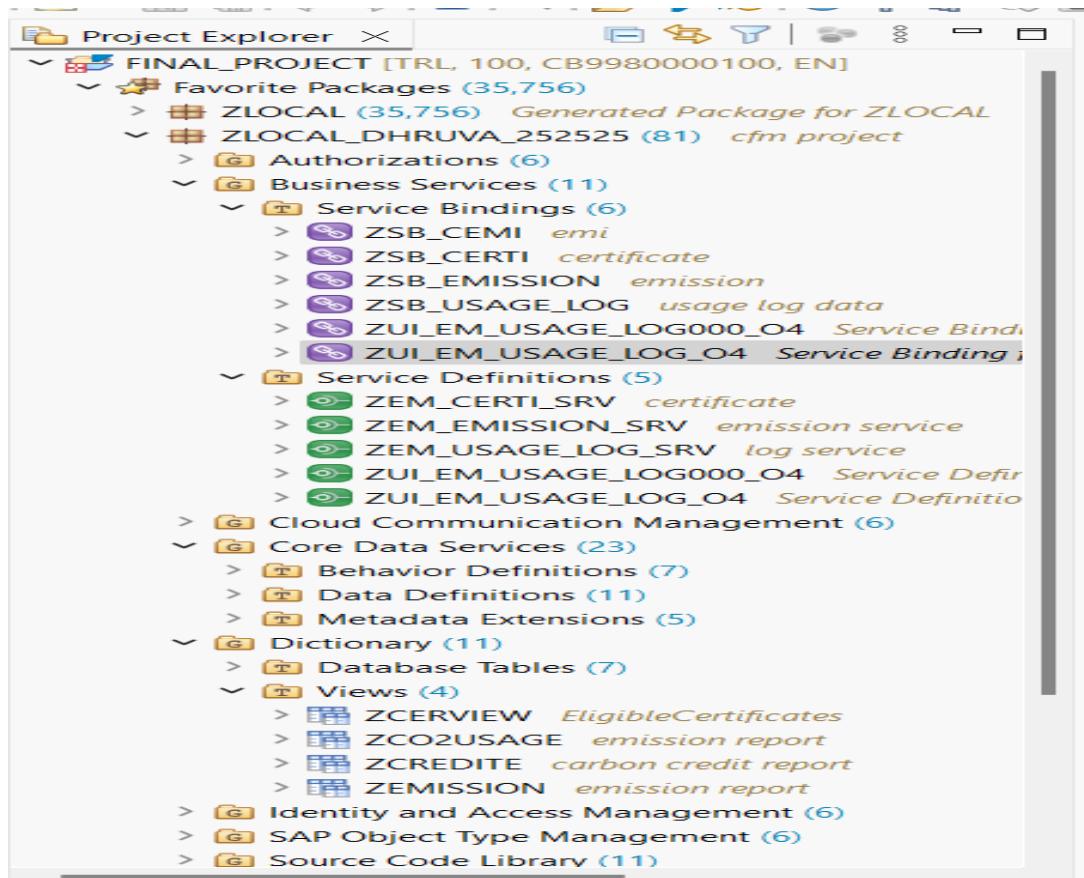
Benefits

- **Regulatory Compliance:** Helps meet carbon emission reporting standards.
- **Sustainability Focus:** Encourages use of clean energy sources.
- **Data-Driven Decisions:** Provides actionable reports for managers.
- **Integrity & Control:** Validations and structured logging ensure reliable data.
- **SAP Integration:** Built fully in ABAP, easily extendable to Fiori/UI5 or RAP.

Project Architecture :-



RAP :-



2. Database Schema

2.1 Table: ZCFM_SOURCE – Source Table

Field Name	Data Type	Description
client	abap.clnt	Client Key
source_id	abap.char(10)	Unique Source ID
source_name	abap.char(50)	Source Name
source_type	abap.char(20)	Source Type
unit_of_measure	abap.char(10)	Unit of Measure
active	abap.char(1)	Active Flag (Y/N)

eclipse - Database Table ZCFM_SOURCE [TRL,100] - active - FINAL_PROJECT [TRL, 100, CB9980000100,

```
File Edit Source Code Navigate Search Project Run Window Help
[TRL] ZBP_CEM_USAGE_ENTITY [TRL] ZCFM_SOURCE [TRL] ZCFM_FACTOR
C [TRL] ZCFM_SOURCE
1 @EndUserText.label : 'source table'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zcfm_source {
7
8     key client      : abap.clnt not null;
9     key source_id   : abap.char(10) not null;
10    source_name    : abap.char(50);
11    source_type    : abap.char(20);
12    unit_of_measure: abap.char(10);
13    active         : abap.char(1);
14
15 }
```

E [TRL] ZI_EMISS... D [TRL] ZI_ELIG... E [TRL] ZI_CERTI... S [TRL] ZSB_CERTI C [TRL] ZCL_EMIS... C [

Data Preview | find pattern | 15 rows retrieved - 25 ms

RB	CLIENT	RB SOURCE_ID	RB SOURCE_NAME	RB SOURCE_TYPE	RB UNIT_OF_MEASURE	RB ACTIVE
100		SRC001	Solar Panel	Renewable	kWh	X
100		SRC002	Wind Turbine	Renewable	kWh	X
100		SRC003	Diesel Gen	Non-Renewable	Litre	
100		SRC004	Hydro Plant	Renewable	kWh	X
100		SRC005	Wind Turbine	Renewable	kWh	X
100		SRC006	Petrol Generator	Fuel	Litre	X
100		SRC007	Biogas Unit	Renewable	m3	X
100		SRC008	Hydro Power Plant	Renewable	kWh	X
100		SRC009	Coal Furnace	Fuel	Kg	X
100		SRC010	Solar Water Heater	Renewable	Litre	X
100		SRC012	Geo-Thermal Pump	Renewable	kWh	X
100		SRC013	Biofuel Station	Fuel	Litre	X
100		SRC014	Electric Vehicle	Transport	kWh	X
100		SRC016	Hydro Power Plant	Renewable	kWh	X
100		SRC011	Updated Solar Panel	Backup	kWh	X

2.2 Table: ZCFM_FACTOR – Factor Table

Field Name	Data Type	Description
client	abap.clnt	Client Key
source_id	abap.char(10)	Related Source ID
unit	abap.char(10)	Unit of Measurement
factor_value	abap.dec(10,2)	Conversion Factor Value

eclipse - Database Table ZCFM_FACTOR [TRL,100] - active - FINAL_PROJECT [TRL, 100, CB99800]

File Edit Source Code Navigate Search Project Run Window Help

[TRL] ZBP_CEM_USAGE_ENTITY [TRL] ZCFM_SOURCE [TRL] ZCFM_FACTOR

```

1  @EndUserText.label : 'factor table'
2  @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3  @AbapCatalog.tableCategory : #TRANSPARENT
4  @AbapCatalog.deliveryClass : #A
5  @AbapCatalog.dataMaintenance : #RESTRICTED
6  define table zcfm_factor {
7
8    key client      : abap.clnt not null;
9    key source_id   : abap.char(10) not null;
10   unit          : abap.char(10);
11   factor_value  : abap.dec(10,2);
12
13 }

```

Data Preview

find pattern 14 rows retrieved - 29 ms

CLIENT	SOURCE_ID	UNIT	FACTOR_VALUE
100	SRC001		0.15
100	SRC002		0.10
100	SRC003		2.50
100	SRC005		0.12
100	SRC006		2.31
100	SRC007		0.45
100	SRC008		0.00
100	SRC009		2.42
100	SRC010		0.00
100	SRC011		0.20
100	SRC012		0.00
100	SRC013		1.80
100	SRC014		0.50
100	SRC021	kg	1.50

2.3 Table: ZCFM_USAGE_LOG – Usage Log Table

Field Name	Data Type	Description
client	abap.cln	Client Key
usage_id	abap.char(10)	Unique Usage Log ID
source_id	abap.char(10)	Source ID
usage_date	abap.dats	Date of Usage
quantity	abap.dec(10,2)	Quantity Used
created_by	syuname	Created By (User Name)
created_on	abap.dats	Created On (Date)

File Edit Source Code Navigate Search Project Run Window Help

[TRL] ZBP_CEM_USAGE_ENTITY [TRL] ZCFM_SOURCE [TRL] ZCFM_USAGE_LOG

```

1 @EndUserText.label : 'usage table'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zcfm_usage_log {
7
8   key client      : abap.cln not null;
9   key usage_id    : abap.char(10) not null;
10  key source_id   : abap.char(10) not null;
11  usage_date     : abap.dats;
12  quantity       : abap.dec(10,2);
13  created_by     : syuname;
14  created_on     : abap.dats;
15
16

```

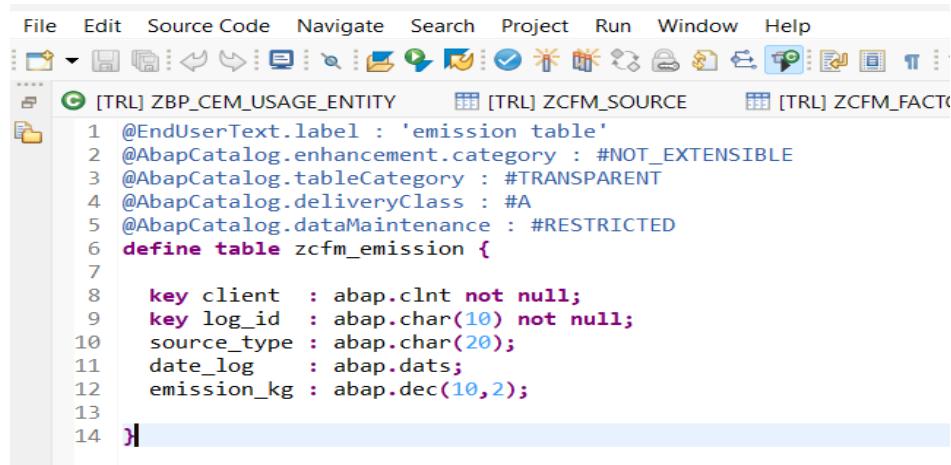
Data Preview

find pattern 15 rows retrieved - 29 ms

CLIENT	USAGE_ID	SOURCE_ID	USAGE_DATE	QUANTITY	CREATED_BY
100	USG001	SRC001	2025-07-01	120.50	ADMIN
100	USG002	SRC002	2025-07-01	75.00	ADMIN
100	USG003	SRC003	2025-07-02	500.00	DHRUVA
100	USG004	SRC004	2025-07-02	80.00	DHRUVA
100	USG005	SRC005	2025-07-03	300.00	ADMIN
100	USG006	SRC006	2025-07-03	55.00	ADMIN
100	USG007	SRC007	2025-07-04	42.00	DHRUVA
100	USG008	SRC008	2025-07-04	150.00	DHRUVA
100	USG009	SRC009	2025-07-05	100.00	ADMIN
100	USG010	SRC010	2025-07-05	90.00	ADMIN
100	USG011	SRC011	2025-07-06	50.00	ADMIN
100	USG012	SRC012	2025-07-06	60.00	DHRUVA
100	USG013	SRC013	2025-07-07	70.00	ADMIN
100	USG014	SRC014	2025-07-07	33.33	ADMIN
100	USG021	SRC021	2025-07-31	120.50	CB9980000100

2.4 Table: ZCFM_EMISSION – Emission Table

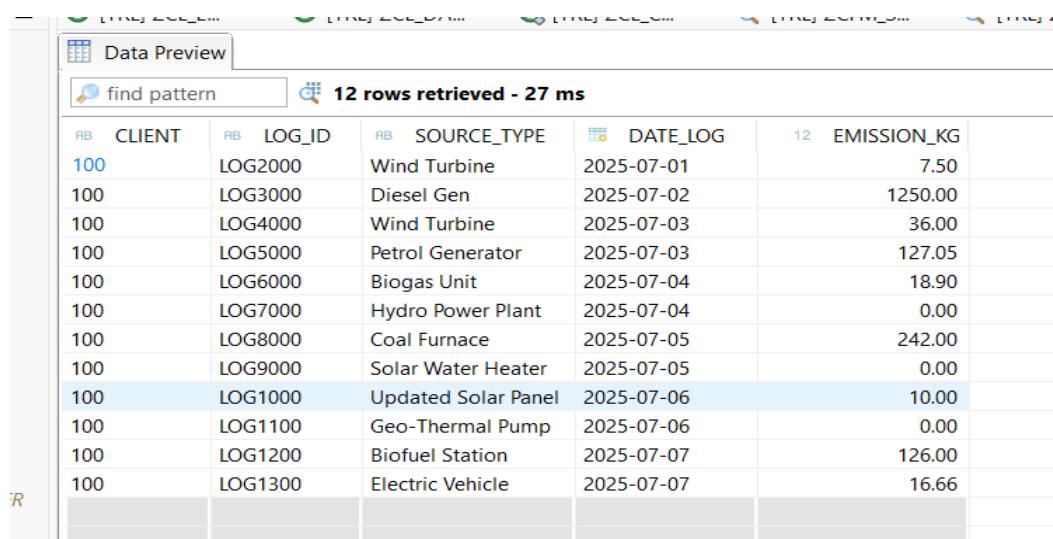
Field Name	Data Type	Description
client	abap.clnt	Client Key
log_id	abap.char(10)	Unique Emission Log ID
source_type	abap.char(20)	Type of Source
date_log	abap.dats	Log Date
emission_kg	abap.dec(10,2)	Emission Quantity (in kg)



```

File Edit Source Code Navigate Search Project Run Window Help
[TRL] ZBP_CEM_USAGE_ENTITY [TRL] ZCFM_SOURCE [TRL] ZCFM_FACTO
1 @EndUserText.label : 'emission table'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zcfm_emission {
7
8   key client : abap.clnt not null;
9   key log_id : abap.char(10) not null;
10  source_type : abap.char(20);
11  date_log : abap.dats;
12  emission_kg : abap.dec(10,2);
13
14 }

```

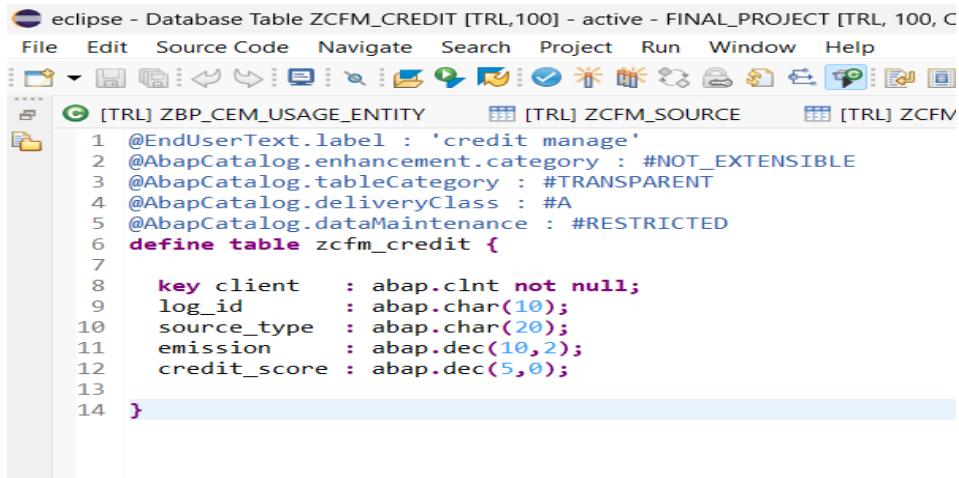


The screenshot shows the SAP Data Preview tool displaying the data from the ZCFM_EMISSION table. The table has 12 rows retrieved in 27 ms.

AB CLIENT	AB LOG_ID	AB SOURCE_TYPE	DATE_LOG	12 EMISSION_KG
100	LOG2000	Wind Turbine	2025-07-01	7.50
100	LOG3000	Diesel Gen	2025-07-02	1250.00
100	LOG4000	Wind Turbine	2025-07-03	36.00
100	LOG5000	Petrol Generator	2025-07-03	127.05
100	LOG6000	Biogas Unit	2025-07-04	18.90
100	LOG7000	Hydro Power Plant	2025-07-04	0.00
100	LOG8000	Coal Furnace	2025-07-05	242.00
100	LOG9000	Solar Water Heater	2025-07-05	0.00
100	LOG1000	Updated Solar Panel	2025-07-06	10.00
100	LOG1100	Geo-Thermal Pump	2025-07-06	0.00
100	LOG1200	Biofuel Station	2025-07-07	126.00
100	LOG1300	Electric Vehicle	2025-07-07	16.66

2.5 Table: ZCFM_CREDIT – Credit Score Table

Field Name	Data Type	Description
client	abap.clnt	Client Key
log_id	abap.char(10)	Emission Log Reference ID
source_type	abap.char(20)	Type of Source
emission	abap.dec(10,2)	Total Emission (kg)
credit_score	abap.dec(5,0)	Assigned Credit Score



The screenshot shows the Eclipse IDE interface with the title "eclipse - Database Table ZCFM_CREDIT [TRL,100] - active - FINAL_PROJECT [TRL, 100, C]". The menu bar includes File, Edit, Source Code, Navigate, Search, Project, Run, Window, and Help. Below the menu is a toolbar with various icons. The main area displays ABAP code:

```
1 @EndUserText.label : 'credit manage'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zcfm_credit {
7
8   key client    : abap.clnt not null;
9   log_id       : abap.char(10);
10  source_type  : abap.char(20);
11  emission     : abap.dec(10,2);
12  credit_score : abap.dec(5,0);
13
14 }
```

2.6 Table: ZCFM_CERTIFICATE – Certificate Issue Table

Field Name	Data Type	Description
client	abap.clnt	Client Key
cert_id	abap.char(10)	Certificate ID (Primary Key)
source_type	abap.char(10)	Type of Source
total_emission	abap.dec(10,2)	Total Emission (kg)
total_credit	int4	Total Assigned Credit Score
cert_date	abap.dats	Certificate Issue Date
expiry_date	abap.dats	Certificate Expiry Date
status	abap.char(10)	Certificate Status

eclipse - Database Table ZCFM_CERTIFICATE [TRL,100] - active - FINAL_PROJECT - Eclipse IDE

```

File Edit Source Code Navigate Search Project Run Window Help
[TRL] ZBP_CEM_USAGE_ENTITY [TRL] ZCFM_SOURCE [TRL] ZCFM_FA
1 @EndUserText.label : 'certificate'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zcfm_certificate {
7
8   key client      : abap.clnt not null;
9   key cert_id     : abap.char(10) not null;
10  source_type    : abap.char(10);
11  total_emission : abap.dec(10,2);
12  total_credit   : int4;
13  cert_date      : abap.dats;
14  expiry_date    : abap.dats;
15  status          : abap.char(10);
16
17 }
```

Data Preview

find pattern 9 rows retrieved • 18 ms SQL Console Number of Entries

AB	CLIENT	AB	CERT_ID	AB	SOURCE_TYPE	12	TOTAL_EMISSION	12	TOTAL_CREDIT	AB	CERT_DATE	AB	EXPIRY_DATE	AB	STATUS
100			CERT1000		Hydro Plan		18.00		5		2025-07-30		2026-07-30		Active
100			CERT2000		Solar Pane		15.00		5		2025-07-30		2026-07-30		Active
100			CERT3000		Wind Turbi		20.00		5		2025-07-30		2026-07-30		Active
100			CERT5000		Electric V		16.66		5		2025-07-31		2026-07-31		Active
100			CERT6000		Geo-Therma		0.00		5		2025-07-31		2026-07-31		Active
100			CERT7000		Hydro Powe		0.00		5		2025-07-31		2026-07-31		Active
100			CERT8000		Petrol Gen		127.05		3		2025-07-31		2026-07-31		Active
100			CERT9000		Solar Wate		0.00		5		2025-07-31		2026-07-31		Active
100			CERT1100		Wind Turbi		43.50		10		2025-07-31		2026-07-31		Active

2. Core Functionality (ABAP Classes)

1. CLASS :- zcl_cfm_data

- ▶ Handles insertion of core data including sources, emission factors, and usage logs with validation logic.
- ▶ Used to initialize or update master and transactional data.

2. CLASS :- zcl_data_manager

- ▶ Provides CRUD (Create, Read, Update, Delete) operations specifically for managing source records.
- ▶ Supports efficient source maintenance with ABAP methods.

3. CLASS :- zcl_emission_report

- ▶ Calculates carbon emissions by combining usage logs and emission factors.
- ▶ Stores emission results in a dedicated emission log table (zcfm_emission).

4. CLASS :- zcl_certificate

- ▶ Analyzes emissions and assigns carbon credits based on thresholds.
- ▶ Generates carbon certificates per source type with expiry control and duplicate checks.
- ▶ Logs failed certificate attempts for audit and traceability.

5. CLASS :- zcl_summary_report

- ▶ Produces a summarized report of all carbon certificates.
- ▶ Shows total issued, expired, and failed certificates in a readable console format.

Class : ZCL_CFM_DATA - Data Manager for Source, Factor, and Usage Logs

Method : insert_source - Insert new energy source details into ZCFM_SOURCE

The screenshot shows the Eclipse IDE interface with the ABAP code for the `insert_source` method. The code is part of the `ZCL_CFM_DATA` class and implements the `if_o_adt_classrun` interface. It loops through a list of sources (`lt_sources`) and inserts each one into the `ZCFM_SOURCE` table. The `ABAP Console` tab shows the successful insertion of multiple source entries, with IDs ranging from SRC001 to SRC016.

```
15  INTERFACES if_o_adt_classrun .
16  PROTECTED SECTION.
17  PRIVATE SECTION.
18  ENDCLASS.
19
20
21  CLASS zcl_cfm_data IMPLEMENTATION.
22
23  DATA lt_sources TYPE STANDARD TABLE OF zcfm_source.
24
25  DATA lv_message TYPE string.
26
27
28  DELETE FROM zcfm_source.
29
30  " Add multiple source entries
31  lt_sources = VALUE #(
32    ( source_id = 'SRC001' source_name = 'Solar Panel' source_type = 'Renewable' unit_of_measure = 'kWh' active = 'X' )
33    ( source_id = 'SRC002' source_name = 'Wind Turbine' source_type = 'Renewable' unit_of_measure = 'kWh' active = 'X' )
34    ( source_id = 'SRC003' source_name = 'Diesel Gen' source_type = 'Non-Renewable' unit_of_measure = 'Litre' active = 'X' )
35    ( source_id = 'SRC004' source_name = 'Hydro Plant' source_type = 'Renewable' unit_of_measure = 'Kwh' active = 'X' )
36    ( source_id = 'SRC005' source_name = 'Wind Turbine' source_type = 'Renewable' unit_of_measure = 'Kwh' active = 'X' )
37    ( source_id = 'SRC006' source_name = 'Petrol Generator' source_type = 'Fuel' unit_of_measure = 'Litre' active = 'X' )
38    ( source_id = 'SRC007' source_name = 'Biogas Unit' source_type = 'Renewable' unit_of_measure = 'm3' active = 'X' )
39    ( source_id = 'SRC008' source_name = 'Hydro Power Plant' source_type = 'Renewable' unit_of_measure = 'Kwh' active = 'X' )
40    ( source_id = 'SRC009' source_name = 'Coal Furnace' source_type = 'Fuel' unit_of_measure = 'Kg' active = 'X' )
41    ( source_id = 'SRC010' source_name = 'Solar Water Heater' source_type = 'Renewable' unit_of_measure = 'Litre' active = 'X' )
42    ( source_id = 'SRC011' source_name = 'Battery Storage' source_type = 'Backup' unit_of_measure = 'Kwh' active = 'X' )
43    ( source_id = 'SRC012' source_name = 'Geothermal Pump' source_type = 'Renewable' unit_of_measure = 'Kwh' active = 'X' )
44    ( source_id = 'SRC013' source_name = 'Fuel Oil Station' source_type = 'Fuel' unit_of_measure = 'Litre' active = 'X' )
45    ( source_id = 'SRC014' source_name = 'Electric Vehicle' source_type = 'Transport' unit_of_measure = 'Kwh' active = 'X' )
46    ( source_id = 'SRC015' source_name = 'Steam Generator' source_type = 'Thermal' unit_of_measure = 'Kg' active = 'X' )
47    ( source_id = 'SRC016' source_name = 'Hydro Power Plant' source_type = 'Renewable' unit_of_measure = 'kWh' active = 'X' )
48
49  ). 
50
51  " Loop and insert each one
52  LOOP AT lt_sources INTO ls_source.
53    lv_message = insert_source( ls_source ).
54    out->write( lv_message ).
55  ENDLOOP.
56
57
58
59  DATA: lt_factors TYPE STANDARD TABLE OF zcfm_factor,
60        ls_factor TYPE zcfm_factor.
61
62  DELETE FROM zcfm_factor.
63
64
65  lt_factors = VALUE #(
66    ( source_id = 'SRC001' factor_value = '0.15' )
67    ( source_id = 'SRC002' factor_value = '0.10' )
68    ( source_id = 'SRC003' factor_value = '2.50' )
69    ( source_id = 'SRC005' factor_value = '0.12' )
70    ( source_id = 'SRC006' factor_value = '0.00' )
71    ( source_id = 'SRC007' factor_value = '2.31' )
72    ( source_id = 'SRC008' factor_value = '0.00' )
73    ( source_id = 'SRC009' factor_value = '0.42' )
74    ( source_id = 'SRC010' factor_value = '10.00' )
75    ( source_id = 'SRC011' factor_value = '0.20' )
76    ( source_id = 'SRC012' factor_value = '0.00' )
77    ( source_id = 'SRC013' factor_value = '1.80' )
78    ( source_id = 'SRC014' factor_value = '0.50' )
79    ( source_id = 'SRC015' factor_value = '1.00' )
80    ( source_id = 'SRC018' factor_value = '0.00' )
81  ). 
82
83
84  LOOP AT lt_factors INTO ls_factor.
85    lv_message = insert_factor( ls_factor ). " Call your insert method
86    out->write( lv_message ).
87  ENDLOOP.
88
89
90  DATA: lt_usages TYPE STANDARD TABLE OF zcfm_usage_log,
91        ls_usage TYPE zcfm_usage_log.
```

Method : insert_factor - Insert emission factor value linked to source

The screenshot shows the Eclipse IDE interface with the ABAP code for the `insert_factor` method. The code loops through a list of factors (`lt_factors`) and inserts each one into the `ZCFM_FACTOR` table. The `ABAP Console` tab shows the successful insertion of multiple factor entries, with IDs SRC015 through SRC018. It also handles an invalid log entry for source ID SRC018.

```
53  " Loop and insert each one
54  LOOP AT lt_factors INTO ls_factor.
55    lv_message = insert_factor( ls_factor ). " Call your insert method
56    out->write( lv_message ).
57  ENDLOOP.
58
59
60  DATA: lt_usages TYPE STANDARD TABLE OF zcfm_usage_log,
61        ls_usage TYPE zcfm_usage_log.
```

Method : insert_usage_log - Insert usage entry with quantity for energy source

```
eclipse - Class ZCL_CFM_DATA [TRL100] - active - FINAL_PROJECT - Eclipse IDE
File Edit Navigate Search Project Run Window Help
[trl] ZCL_DATA_MANAGER [trl] ZCL_CFM_DATA
ZCL_CFM_DATA > IF_OO_ADT_CLASSRUN-MAIN
84
85 LOOP AT lt_factors INTO ls_factor.
86   lv_message = insert_factor( ls_factor ). " Call your insert method
87   OUT-WRITEL( lv_message ).
88 ENDLOOP.
89
90 DATA: lt_usages TYPE STANDARD TABLE OF zcfm_usage_log,
91       ls_usage TYPE zcfm_usage_log.
92 DATA(lv_today) := cl_abap_context_info->get_system_date( ).
93 DELETE FROM zcfm_usage_log.
94
95 lt_usages = VALUE #(
96   ( usage_id = 'USG001' usage_date = lv_today source_id = 'SRC001' quantity = 100 )
97   ( usage_id = 'USG002' usage_date = lv_today source_id = 'SRC002' quantity = 200 )
98   * ( usage_id = 'USG003' usage_date = lv_today source_id = 'SRC006' quantity = 75 )
99   * ( usage_id = 'USG004' usage_date = lv_today source_id = 'SRC004' quantity = 150 )
100  ( usage_id = 'USG001' source_id = 'SRC001' usage_date = '20250701' quantity = '120.50' created_by = 'ADMIN' )
101  ( usage_id = 'USG002' source_id = 'SRC002' usage_date = '20250701' quantity = '75.00' created_by = 'ADMIN' )
102  ( usage_id = 'USG003' source_id = 'SRC003' usage_date = '20250702' quantity = '500.00' created_by = 'DHRUVAA' )
103  ( usage_id = 'USG004' source_id = 'SRC004' usage_date = '20250702' quantity = '80.00' created_by = 'DHRUVAA' )
104  ( usage_id = 'USG005' source_id = 'SRC005' usage_date = '20250703' quantity = '300.00' created_by = 'ADMIN' )
105  ( usage_id = 'USG006' source_id = 'SRC006' usage_date = '20250703' quantity = '100.00' created_by = 'DHRUVAA' )
106  ( usage_id = 'USG007' source_id = 'SRC007' usage_date = '20250704' quantity = '42.00' created_by = 'DHRUVAA' )
107  ( usage_id = 'USG008' source_id = 'SRC008' usage_date = '20250704' quantity = '155.00' created_by = 'DHRUVAA' )
108  ( usage_id = 'USG009' source_id = 'SRC009' usage_date = '20250705' quantity = '190.00' created_by = 'ADMIN' )
109  ( usage_id = 'USG010' source_id = 'SRC010' usage_date = '20250705' quantity = '90.00' created_by = 'ADMIN' )
110  ( usage_id = 'USG011' source_id = 'SRC011' usage_date = '20250706' quantity = '50.00' created_by = 'ADMIN' )
111  ( usage_id = 'USG012' source_id = 'SRC012' usage_date = '20250706' quantity = '60.00' created_by = 'DHRUVAA' )
112  ( usage_id = 'USG013' source_id = 'SRC013' usage_date = '20250707' quantity = '70.00' created_by = 'ADMIN' )
113  ( usage_id = 'USG014' source_id = 'SRC014' usage_date = '20250707' quantity = '33.33' created_by = 'ADMIN' )
114  ( usage_id = 'USG015' source_id = 'SRC015' usage_date = '20250708' quantity = '21.00' created_by = 'DHRUVAA' )
115  ( usage_id = 'USG016' source_id = 'SRC016' usage_date = '20250708' quantity = '0.00' created_by = 'DHRUVAA' )
116  ( usage_id = 'USG018' source_id = 'SRC018' usage_date = '20250708' quantity = '90.00' created_by = 'ADMIN' )
117
118
119).
120
121 LOOP AT lt_usages INTO ls_usage.
122   lv_message = insert_usage_log( ls_usage ). " Call your insert method
123   OUT-WRITEL( lv_message ).
```

Validation for Energy Source, Factor, and Usage Logs in ABAP

ZCL_CFM_DATA ► INSERT_USAGE_LOG

```
137@ IF sy-subrc = 0.
138   rv_message = |Source ID already exists: { is_source-source_id }| .
139   RETURN.
140 ENDIF.
141
142 INSERT zcfm_source FROM @is_source.
143@ IF sy-subrc = 0.
144   rv_message = |Source inserted: { is_source-source_id }| .
145 ELSE.
146   rv_message = |Failed to insert source: { is_source-source_id }|.
147 ENDIF.
148 ENDMETHOD.
149
150 METHOD insert_factor.
151   " Additional check: Ensure source exists
152   SELECT SINGLE source_id
153     FROM zcfm_source
154    WHERE source_id = @is_factor-source_id
155    INTO @DATA(lv_source).
156@ IF sy-subrc <> 0.
157   rv_message = |Invalid source ID for factor: { is_factor-source_id }| .
158   RETURN.
159 ENDIF.
160
161 INSERT zcfm_factor FROM @is_factor.
162@ IF sy-subrc = 0.
163   rv_message = |Factor inserted for source: { is_factor-source_id }| .
164 ELSE.
165   rv_message = |Failed to insert factor for source: { is_factor-source_id }| .
166 ENDIF.
167 ENDMETHOD.
168
169 METHOD insert_usage_log.
170   " Validate source exists
171   SELECT SINGLE source_id
172     FROM zcfm_source
173    WHERE source_id = @is_usage-source_id
174    INTO @DATA(lv_valid_source).
175@ IF sv-subrc <> 0.
```

Source inserted: SRC016
Source ID already exists: SRC015
Invalid source ID for factor: SRC018
Quantity must be greater than 0 for usage ID: USG016
Invalid source ID in usage log:
SRC018

Ln 6, Col 1 187 charact Plain tx 100% Windows UTF-8

*Global Class Class-relevant Local Types Local Types Test Classes Macros

Writable Smart Insert 170 : 7 : 8204 TRL_100, https://tab63f61-d9c7-4b-ana.ondemand.com, CB9980000100, EN

9 31°C Mostly cloudy

Windows Search ?

ENG IN WiFi 15:39 31-07-2025

```

158     RETURN.
159   ENDIF.
160
161   INSERT zcfm_factor FROM @is_factor.
162@   IF sy-subrc = 0.
163     rv_message = |Factor inserted for source: { is_factor-source_id }| .
164   ELSE.
165     rv_message = |Failed to insert factor for source: { is_factor-source_id }| .
166   ENDIF.
167   ENDMETHOD.
168
169@ METHOD insert_usage_log.
170   " Validate source exists
171   SELECT SINGLE source_id
172     FROM zcfm_source
173     WHERE source_id = @is_usage-source_id
174     INTO @DATA(lv_valid_source).
175@   IF sy-subrc < 0.
176     rv_message = |Invalid source ID in usage log: { is_usage-source_id }| .
177   RETURN.
178 ENDIF.
179
180@   IF is_usage-quantity <= 0.
181     rv_message = |Quantity must be greater than 0 for usage ID: { is_usage-usage_id }| .
182   RETURN.
183 ENDIF.
184
185   INSERT zcfm_usage_log FROM @is_usage.
186@   IF sy-subrc = 0.
187     rv_message = |Usage log inserted: { is_usage-usage_id }| .
188   ELSE.
189     rv_message = |Failed to insert usage log: { is_usage-usage_id }| .
190   ENDIF.
191
192 ENDMETHOD.
193
194
195
196
197 ENDMETHOD.
198

```

Source inserted: SRC016
Source ID already exists: SRC015
Invalid source ID for factor: SRC018
Quantity must be greater than 0 for usage ID: USG016
Invalid source ID in usage log: SRC018

Ln 6, Col 1 187 charact Plain tr 100% Windows UTF-8

Class: ZCL_DATA_MANAGER – CRUD Operations for Source, Factor & Usage Logs

Source Table CRUD Operations :-

```

23
24@ **** SOURCE TABLE CRUD *
25
26 ****
27
28
29   " GET / READ
30   DATA ls_read_source TYPE zcfm_source.
31   SELECT SINGLE * FROM zcfm_source
32     WHERE source_id = 'SRC011'
33     INTO @ls_read_source.
34
35@   IF sy-subrc = 0.
36     out->writeln( |Read -> { ls_read_source-source_id }, Name: { ls_read_source-source_name }| ).
37   ELSE.
38     out->writeln( |Read Failed: SRC011 not found| ).
39   ENDIF.
40
41   " UPDATE
42   ls_read_source-source_name = 'Updated Solar Panel'.
43   UPDATE zcfm_source SET source_name = @ls_read_source-source_name
44     WHERE source_id = @ls_read_source-source_id.
45@   IF sy-subrc = 0.
46     out->writeln( |Updated: { ls_read_source-source_id }| ).
47   ELSE.
48     out->writeln( |Update Failed: { ls_read_source-source_id }| ).
49   ENDIF.
50
51   " DELETE
52   DELETE FROM zcfm_source WHERE source_id = 'SRC015'.
53@   IF sy-subrc = 0.
54     out->writeln( |Deleted: SRC015| ).
55   ELSE.
56     out->writeln( |Delete Failed: SRC015 not found| ).
57   ENDIF.
58
59   out->writeln( ****).
60
61@ **** FARTIR TARI F CINIY *
62

```

ABAP Console
Usage Updated: USG021
Usage Deleted: USG015

Read -> SRC011, Name: Updated Solar Panel
Updated: SRC011
Delete Failed: SRC015 not found

Factor Insert Failed: SRC021
Factor Read -> SRC021, Unit: kg
Factor Updated: SRC021
Factor Delete Failed: SRC015 not found

Usage Insert Failed: USG021
Usage Read -> ID: USG021, Qty: 120.50
Usage Updated: USG021
Usage Delete Failed: USG015 not found

Emission Factor Management :-

The screenshot shows the Eclipse IDE interface with an ABAP project named ZCL_DATA_MANAGER. The code in the editor handles the creation, update, and deletion of emission factors. The ABAP Console on the right displays the results of the operations, including successful reads, updates, and deletes, as well as failed attempts due to missing data.

```

610 "*****  
611 " FACTOR TABLE CRUD *  
612 "*****  
613  
614 " CREATE  
615 DATA(ls_factor) = VALUE zcfm_factor(  
616   client      = sy-mandt  
617   source_id   = 'SRC021'  
618   unit        = 'kg'  
619   factor_value = '1.25' ).  
620  
621 INSERT zcfm_factor FROM @ls_factor.  
622 IF sy-subrc = 0.  
623   out->write( |Factor Inserted: { ls_factor-source_id }| ).  
624 ELSE.  
625   out->write( |Factor Insert Failed: { ls_factor-source_id }| ).  
626 ENDIF.  
627  
628 " READ  
629 DATA ls_read_factor TYPE zcfm_factor.  
630 SELECT SINGLE * FROM zcfm_factor WHERE source_id = 'SRC021' INTO @ls_read_factor.  
631 IF sy-subrc = 0.  
632   out->write( |Factor Read -> { ls_read_factor-source_id }, Unit: { ls_read_factor-unit }| ).  
633 ELSE.  
634   out->write( |Factor Read Failed: SRC021 not found| ).  
635ENDIF.  
636  
637 " UPDATE  
638 ls_read_factor-factor_value = '1.50'.  
639 UPDATE zcfm_factor SET factor_value = @ls_read_factor-factor_value  
640   WHERE source_id = @ls_read_factor-source_id.  
641 IF sy-subrc = 0.  
642   out->write( |Factor Updated: { ls_read_factor-source_id }| ).  
643 ELSE.  
644   out->write( |Factor Update Failed: { ls_read_factor-source_id }| ).  
645ENDIF.  
646  
647 " DELETE  
648 DELETE FROM zcfm_factor WHERE source_id = 'SRC015'.  
649 IF sy-subrc = 0.  
650   out->write( |Factor Deleted: SRC015 not found| ).  
651ENDIF.

```

Usage Log Handling :-

The screenshot shows the Eclipse IDE interface with an ABAP project named ZCL_DATA_MANAGER. The code in the editor handles the creation, update, and deletion of usage logs. The ABAP Console on the right displays the results of the operations, including successful reads, updates, and deletes, as well as failed attempts due to missing data.

```

109@ "*****  
110 " USAGE LOG TABLE CRUD *  
111 "*****  
112  
113 " CREATE  
114 DATA(ls_usage) = VALUE zcfm_usage_log(  
115   client      = sy-mandt  
116   usage_id    = 'USG021'  
117   source_id   = 'SRC021'  
118   usage_date  = cl_abap_context_info->get_system_date( )  
119   quantity     = '100.00'  
120   created_by  = sy-uname  
121   created_on   = cl_abap_context_info->get_system_date( ).  
122  
123 INSERT zcfm_usage_log FROM @ls_usage.  
124 IF sy-subrc = 0.  
125   out->write( |Usage Inserted: { ls_usage-usage_id }| ).  
126 ELSE.  
127   out->write( |Usage Insert Failed: { ls_usage-usage_id }| ).  
128ENDIF.  
129  
130 " READ  
131 DATA ls_read_usage TYPE zcfm_usage_log.  
132 SELECT SINGLE * FROM zcfm_usage_log WHERE usage_id = 'USG021' INTO @ls_read_usage.  
133 IF sy-subrc = 0.  
134   out->write( |Usage Read -> ID: { ls_read_usage-usage_id }, Qty: { ls_read_usage-quantity }| ).  
135 ELSE.  
136   out->write( |Usage Read Failed: USG021 not found| ).  
137ENDIF.  
138  
139 " UPDATE  
140 ls_read_usage-quantity = '120.50'.  
141 UPDATE zcfm_usage_log SET quantity = @ls_read_usage-quantity  
142   WHERE usage_id = @ls_read_usage-usage_id.  
143 IF sy-subrc = 0.  
144   out->write( |Usage Updated: { ls_read_usage-usage_id }| ).  
145 ELSE.  
146   out->write( |Usage Update Failed: { ls_read_usage-usage_id }| ).  
147ENDIF.

```

Class : ZCL_EMISSION_REPORT - Emission Report Generator

The screenshot shows the Eclipse IDE interface with the ZCL_EMISSION_REPORT class selected. The code editor displays the implementation of the `if_oo_adt_classrun-main` method. The ABAP Console window shows the execution results, including a list of usage records and a summary of total emission records inserted.

```

111 METHOD if_oo_adt_classrun-main.
112
113   SELECT a-usage_id,
114         a-usage_date,
115         b-source_name,
116         a-quantity,
117         b-unit-of-measure,
118         c-factor-value,
119         ( a-quantity * c-factor_value ) AS emission_result
120   FROM zcfm_usage_log AS a
121   INNER JOIN zcfm_source AS b ON a-source_id = b-source_id
122   INNER JOIN zcfm_factor AS c ON a-source_id = c-source_id
123   INTO TABLE @lt_report.
124
125   IF lt_report IS INITIAL.
126     out->write( 'No data found in emission report.' ).
127   RETURN.
128   ENDIF.
129
130   " OUTPUT
131   LOOP AT lt_report INTO DATA(ls_report).
132     out->write( |Usage ID: { ls_report-source_id }, Source: { ls_report-source_name }, Emission: { ls_report-emission_result }| ).
133
134   CLEAR ls_emission.
135   ls_emission-log_id = |LOG{ sy-table WIDTH = 4 PAD = '0' }|.
136   ls_emission-source_type = ls_report-source_name.
137   ls_emission-date_log = ls_report-usage_date.
138   ls_emission-emission_kg = ls_report-emission_result.
139
140   " Insert into ZEM_EMISSTION_LOG
141   MODIFY zcfm_emission FROM @ls_emission.
142
143 ENDOOP.
144   out->write( |Total emission records inserted: { lines( lt_report ) }| ).
145
146
147 ENDMETHOD.
148

```

ABAP Console Output:

```

ABAP Console
Usage Delete Failed: USG015 not found

Usage ID: USG001, Source: Solar Panel, Emission: 18.07
Usage ID: USG002, Source: Wind Turbine, Emission: 7.50
Usage ID: USG003, Source: Diesel Gen, Emission: 1250.00
Usage ID: USG005, Source: Wind Turbine, Emission: 36.00
Usage ID: USG006, Source: Petro Generator, Emission: 127.05
Usage ID: USG007, Source: Coal Furnace, Emission: 18.00
Usage ID: USG009, Source: Hydro Power Plant, Emission: 0.00
Usage ID: USG009, Source: Coal Furnace, Emission: 342.00
Usage ID: USG010, Source: Solar Water Heater, Emission: 0.00
Usage ID: USG011, Source: Updated Solar Panel, Emission: 10.00
Usage ID: USG012, Source: Geo-Thermal Pump, Emission: 0.00
Usage ID: USG013, Source: Biofuel Station, Emission: 126.00
Usage ID: USG014, Source: Electric Vehicle, Emission: 16.66
Total emission records inserted: 13

```

Class : ZCL_CERTIFICATE - Carbon Certificate Issuer

The screenshot shows the Eclipse IDE interface with the ZCL_CERTIFICATE class selected. The code editor displays the implementation of the `if_oo_adt_classrun-main` method. The ABAP Console window shows the execution results, including a summary of certificate issuance and expiration.

```

101
102
103   " Step 5: Insert Certificate if valid
104   IF ls_cert-total_credit > 0 AND ls_cert-total_credit <= 1000.
105     INSERT zcfm_certificate FROM @ls_cert.
106   IF sy-subrc = 0.
107     out->write( | Certificate ( lv_cert_id ) issued for source { ls_cert-source_type }: { ls_cert-total_credit } |
108   ELSE.
109     APPEND VALUE #( source_type = ls_cert-source_type
110       reason = 'Insert failed for unknown reason'
111       ) TO lt_error_log.
112   ENDIF.
113 ELSE.
114   APPEND VALUE #( source_type = ls_cert-source_type
115     reason = [Invalid credit score: { ls_cert-total_credit } ]
116     ) TO lt_error_log.
117 ENDIF.
118
119 ENDOOP.
120
121   " Step 6: Commit once
122 COMMIT WORK.
123
124   " Step 7: Expire old certificates
125   SELECT * FROM zcfm_certificate
126   WHERE status = 'Active'
127   INTO TABLE @lt_cert_update.
128
129   LOOP AT lt_cert_update INTO DATA(ls_old_cert).
130     IF ls_old_cert-expiry_date < cl_abap_context_info>get_system_date( ).
131     ls_old_cert-status = 'Expired'.
132     UPDATE zcfm_certificate FROM @ls_old_cert.
133   IF sy-subrc = 0.
134     lv_updated += 1.
135   ENDIF.
136   ENDOOP.
137
138   out->write( |@ Updated { lv_updated } certificate(s) to status 'Expired'.| ).
139

```

ABAP Console Output:

```

ABAP Console
Certificate CERT5000 issued for source Electric V: 5 credits on 20250731.
Certificate CERT6000 issued for source Geo-Thermal: 5 credits on 20250731.
Certificate CERT7000 issued for source Hydro Powe: 5 credits on 20250731.
Certificate CERT8000 issued for source Petrol Gen: 3 credits on 20250731.
Certificate CERT9000 issued for source Coal Furne: 5 credits on 20250731.
Certificate CERT1100 issued for source Wind Turbi: 10 credits on 20250731.
Updated 0 certificate(s) to status 'Expired'.
Skipped Certificate Entries Summary:
- Source: Biofuel St, Reason: Insert failed for unknown reason
- Source: Biogas Uni, Reason: Insert failed for unknown reason
- Source: Coal Furna, Reason: Insert failed for unknown reason
- Source: Diesel Gen, Reason: Invalid credit score: 0
- Source: Updated So, Reason: Insert failed for unknown reason

```

eclipse - Class ZCL_CERTIFICATE [TRL100] - active - FINAL_PROJECT - Eclipse IDE

```

File Edit Navigate Search Project Run Window Help
[trl] ZCL_DATA_MANAGER [trl] ZCL_CFM_DATA [trl] ZCL_SUMMARY_REPORT [trl] ZCL_EMISSION_REPORT [trl] ZCL_CERTIFI Prop... Tem... Book... Feed... Trans... Cons... Dictio... Quick... ABAP Console

144  APPEND VALUE #1 source_type = ls_cert-source_type
145    reason = [Invalid credit score: { ls_cert-total_credit }]
146    ) TO lt_error_log.
147  ENDIF.
148
149  ENDDO.
150
151  " Step 6: Commit once
152  COMMIT WORK.
153
154  " Step 7: Expire old certificates
155  SELECT * FROM zcfm_certificate
156  WHERE status = 'Active'
157  INTO TABLE @lt_cert_update.
158
159  LOOP AT lt_cert_update INTO DATA(ls_old_cert).
160  IF ls_old_cert-expiry_date < cl_abap_context_info>get_system_date( ).
161    ls_old_cert-status = 'Expired'.
162    UPDATE zcfm_certificate FROM @ls_old_cert.
163    IF sy-subrc = 0.
164      lv_updated ++ 1.
165    ENDIF.
166  ENDLOOP.
167
168  out->write( |@ Updated { lv_updated } certificate(s) to status 'Expired'.| ).
169
170  " Step 8: Print Summary of Skipped Entries
171  IF lt_error_log IS NOT INITIAL.
172    out->write( |@ Skipped Certificate Entries Summary:| ).
173    LOOP AT lt_error_log INTO DATA(ls_log).
174      out->write( | { ls_log-source_type }, Reason: { ls_log-reason } | ).
175    ENDLOOP.
176
177  ENDIF.
178
179  ENDMETHOD.
180
181  ENDMETHOD.
182
183  ENCLASS.
184
185 4
Global Class Class-relevant Local Types Local Types Test Classes Macros

```

Upcoming Earnings Search 16:23 ENG IN 31-07-2025

Class : ZCL_SUMMARY_REPORT - Carbon Credit Report Dashboard

eclipse - Class ZCL_SUMMARY_REPORT [TRL100] - active - FINAL_PROJECT - Eclipse IDE

```

File Edit Navigate Search Project Run Window Help
[trl] ZCL_DATA_MANAGER [trl] ZCL_CFM_DATA [trl] ZCL_SUMMARY_REPORT [trl] ZCL_EMISSION_REPORT [trl] ZCL_CERTIFI Prop... Tem... Book... Feed... Trans... Cons... Dictio... Quick... ABAP Console

23  DATA: lt_cert          TYPE TABLE OF zcfm_certificate,
24    ls_cert           TYPE zcfm_certificate,
25    lv_line           TYPE string,
26    lv_issued         TYPE i VALUE 0,
27    lv_failed         TYPE i VALUE 0,
28    lv_expired        TYPE i VALUE 0.
29
30  " Step 1: Fetch all certificates
31  SELECT * FROM zcfm_certificate ORDER BY cert_date DESCENDING INTO TABLE @lt_cert.
32  lv_issued = lines( lt_cert ).
33
34  " Step 2: Count expired certificates
35  LOOP AT lt_cert INTO ls_cert WHERE status = 'Expired'.
36  lv_expired ++ 1.
37  ENDOLOOP.
38
39  " Step 3: Fetch failed logs (optional table zcfm_error_log)
40
41
42  " Step 4: Print Summary Header
43  out->write( |@ Carbon Certificate Summary Report:| ).
44  out->write( |=====| ).
45  out->write( |@ Total Issued : { lv_issued } |).
46  out->write( |@ Total Failed : { lv_failed } |).
47  out->write( |@ Total Expired : { lv_expired } |).
48  out->write( |=====| ).
49
50  " Step 5: Detailed List
51  LOOP AT lt_cert INTO ls_cert.
52  lv_line = [certificate ID : { ls_cert-cert_id }|| & cl_abap_char_utilities>newline &
53    |Source Type : { ls_cert-source_type }|| & cl_abap_char_utilities>newline &
54    |Total Emission : { ls_cert-total_emission } kg|| & cl_abap_char_utilities>newline &
55    |Total Credit : { ls_cert-total_credit }|| & cl_abap_char_utilities>newline &
56    |Issue Date : { ls_cert-cert_date }|| & cl_abap_char_utilities>newline &
57    |Expiry Date : { ls_cert-expiry_date }|| & cl_abap_char_utilities>newline &
58    |-----|].
59
60  out->write( lv_line ).
```

Carbon Certificate Summary Report:

Total Issued :	9
Total Failed :	0
Total Expired :	0

Certificate ID : CERT5000
Source Type : Electric V
Total Emission : 16.66 kg
Total Credit : 5
Status : Active
Issue Date : 20250731
Expiry Date : 20260731

Certificate ID : CERT6000
Source Type : Geo-Therma
Total Emission : 0.00 kg
Total Credit : 5
Status : Active
Issue Date : 20250731
Expiry Date : 20260731

Certificate ID : CERT7000
Source Type : Hydro Powe
Total Emission : 0.00 kg
Total Credit : 5
Status : Active
Issue Date : 20250731
Expiry Date : 20260731

Certificate ID : CERT8000
Source Type : Petrol Gen
Total Emission : 127.05 kg
Total Credit : 5
Status : Active
Issue Date : 20250731

Upcoming Earnings Search 16:25 ENG IN 31-07-2025

CODE :-

CLASS :- ZCL_CFM_DATA

```
CLASS zcl_cfm_data DEFINITION
PUBLIC
FINAL
CREATE PUBLIC .

PUBLIC SECTION.
METHODS:
    insert_source IMPORTING is_source TYPE zcfm_source
                  RETURNING VALUE(rv_message) TYPE string,
    insert_factor IMPORTING is_factor TYPE zcfm_factor
                  RETURNING VALUE(rv_message) TYPE string,

    insert_usage_log IMPORTING is_usage TYPE zcfm_usage_log
                      RETURNING VALUE(rv_message) TYPE string.
INTERFACES if_oo_adt_classrun .
PROTECTED SECTION.
PRIVATE SECTION.
ENDCLASS.
```

CLASS zcl_cfm_data IMPLEMENTATION.

```
METHOD if_oo_adt_classrun~main.
```

```
DATA lt_sources TYPE STANDARD TABLE OF zcfm_source.
DATA lv_message TYPE string.
```

```
DELETE FROM zcfm_source.
" Add multiple source entries
lt_sources = VALUE #(
( source_id = 'SRC001' source_name = 'Solar Panel' source_type = 'Renewable'
unit_of_measure = 'kWh' active = 'X' )
( source_id = 'SRC002' source_name = 'Wind Turbine' source_type = 'Renewable'
unit_of_measure = 'kWh' active = 'X' )
( source_id = 'SRC003' source_name = 'Diesel Gen' source_type = 'Non-Renewable'
unit_of_measure = 'Litre' active = '' )

).
```

```
" Loop and insert each one
LOOP AT lt_sources INTO DATA(ls_source).
lv_message = insert_source( ls_source ).
out->write( lv_message ).
ENDLOOP.
```

```
DATA: lt_factors TYPE STANDARD TABLE OF zcfm_factor,
ls_factor TYPE zcfm_factor.
```

```
DELETE FROM zcfm_factor.

lt_factors = VALUE #(

```

```
( source_id = 'SRC001' factor_value = '0.15' )
( source_id = 'SRC002' factor_value = '0.10' )
( source_id = 'SRC003' factor_value = '2.50' )
).
```

```
LOOP AT lt_factors INTO ls_factor.
lv_message = insert_factor(ls_factor). " Call your insert method
out->write(lv_message).
ENDLOOP.
```

```
DATA: lt_usages TYPE STANDARD TABLE OF zcfm_usage_log,
ls_usage TYPE zcfm_usage_log.
DATA(lv_today) = cl_abap_context_info=>get_system_date().
DELETE FROM zcfm_usage_log.
```

```
lt_usages = VALUE #(
( usage_id = 'USG001' source_id = 'SRC001' usage_date = '20250701' quantity = '120.50'
created_by = 'ADMIN' )
( usage_id = 'USG002' source_id = 'SRC002' usage_date = '20250701' quantity = '75.00'
created_by = 'ADMIN' )
( usage_id = 'USG003' source_id = 'SRC003' usage_date = '20250702' quantity = '500.00'
created_by = 'DHRUVA' )
).
```

```
LOOP AT lt_usages INTO ls_usage.
lv_message = insert_usage_log(ls_usage). " Call your insert method
out->write(lv_message).
ENDLOOP.
```

```
ENDMETHOD.
```

```
METHOD insert_source.
" Basic check for existing source
DATA: lv_exists TYPE zcfm_source-source_id.
```

```
SELECT SINGLE source_id
FROM zcfm_source
WHERE source_id = @is_source-source_id
INTO @lv_exists.
```

```
IF sy-subrc = 0.
rv_message = |Source ID already exists: { is_source-source_id }|.
RETURN.
ENDIF.
```

```
INSERT zcfm_source FROM @is_source.
IF sy-subrc = 0.
rv_message = |Source inserted: { is_source-source_id }|.
ELSE.
rv_message = |Failed to insert source: { is_source-source_id }|.
ENDIF.
ENDMETHOD.
```

```

METHOD insert_factor.
" Optional check: Ensure source exists
SELECT SINGLE source_id
  FROM zcfm_source
 WHERE source_id = @is_factor-source_id
  INTO @DATA(lv_source).
IF sy-subrc <> 0.
  rv_message = |Invalid source ID for factor: { is_factor-source_id }|.
  RETURN.
ENDIF.

INSERT zcfm_factor FROM @is_factor.
IF sy-subrc = 0.
  rv_message = |Factor inserted for source: { is_factor-source_id }|.
ELSE.
  rv_message = |Failed to insert factor for source: { is_factor-source_id }|.
ENDIF.
ENDMETHOD.

METHOD insert_usage_log.
" Validate source exists
SELECT SINGLE source_id
  FROM zcfm_source
 WHERE source_id = @is_usage-source_id
  INTO @DATA(lv_valid_source).
IF sy-subrc <> 0.
  rv_message = |Invalid source ID in usage log: { is_usage-source_id }|.
  RETURN.
ENDIF.

IF is_usage-quantity <= 0.
  rv_message = |Quantity must be greater than 0 for usage ID: { is_usage-usage_id }|.
  RETURN.
ENDIF.

INSERT zcfm_usage_log FROM @is_usage.
IF sy-subrc = 0.
  rv_message = |Usage log inserted: { is_usage-usage_id }|.
ELSE.
  rv_message = |Failed to insert usage log: { is_usage-usage_id }|.
ENDIF.

ENDMETHOD.

ENDCLASS.

```

CLASS :- ZCL_DATA_MANAGER

```
CLASS zcl_data_manager DEFINITION
PUBLIC
FINAL
CREATE PUBLIC .
PUBLIC SECTION.
INTERFACES if_oo_adt_classrun .
PROTECTED SECTION.
PRIVATE SECTION.
ENDCLASS.
```

```
CLASS zcl_data_manager IMPLEMENTATION.
```

```
METHOD if_oo_adt_classrun~main.
```

```
DATA lv_message TYPE string.
```

```
"*****"
** SOURCE TABLE CRUD *
"*****"

" GET / READ
DATA ls_read_source TYPE zcfm_source.
SELECT SINGLE * FROM zcfm_source
WHERE source_id = 'SRC011'
INTO @ls_read_source.

IF sy-subrc = 0.
  out->write( |Read -> { ls_read_source-source_id }, Name: { ls_read_source-source_name }| ).
ELSE.
  out->write( |Read Failed: SRC011 not found| ).
ENDIF.

" UPDATE
ls_read_source-source_name = 'Updated Solar Panel'.
UPDATE zcfm_source SET source_name = @ls_read_source-source_name
WHERE source_id = @ls_read_source-source_id.
IF sy-subrc = 0.
  out->write( |Updated: { ls_read_source-source_id }| ).
ELSE.
  out->write( |Update Failed: { ls_read_source-source_id }| ).
ENDIF.

" DELETE
DELETE FROM zcfm_source WHERE source_id = 'SRC015'.
IF sy-subrc = 0.
  out->write( |Deleted: SRC015| ).
ELSE.
  out->write( |Delete Failed: SRC015 not found| ).
ENDIF.
```

```

        out->write(
'=====').

"*****"
"** FACTOR TABLE CRUD  *"
"*****"

" CREATE
DATA(ls_factor)=VALUE zcfm_factor(
    client      = sy-mandt
    source_id   = 'SRC021'
    unit        = 'kg'
    factor_value = '1.25' ).

INSERT zcfm_factor FROM @ls_factor.
IF sy-subrc = 0.
    out->write( |Factor Inserted: { ls_factor-source_id }| ).
ELSE.
    out->write( |Factor Insert Failed: { ls_factor-source_id }| ).
ENDIF.

" READ
DATA ls_read_factor TYPE zcfm_factor.
SELECT SINGLE * FROM zcfm_factor WHERE source_id = 'SRC021' INTO
@ls_read_factor.
IF sy-subrc = 0.
    out->write( |Factor Read -> { ls_read_factor-source_id }, Unit: { ls_read_factor-unit }| ).
ELSE.
    out->write( |Factor Read Failed: SRC021 not found| ).
ENDIF.

" UPDATE
ls_read_factor-factor_value = '1.50'.
UPDATE zcfm_factor SET factor_value = @ls_read_factor-factor_value
    WHERE source_id = @ls_read_factor-source_id.
IF sy-subrc = 0.
    out->write( |Factor Updated: { ls_read_factor-source_id }| ).
ELSE.
    out->write( |Factor Update Failed: { ls_read_factor-source_id }| ).
ENDIF.

" DELETE
DELETE FROM zcfm_factor WHERE source_id = 'SRC015'.
IF sy-subrc = 0.
    out->write( |Factor Deleted: SRC015| ).
ELSE.
    out->write( |Factor Delete Failed: SRC015 not found| ).
ENDIF.

        out->write(
'=====').

"*****"
"** USAGE LOG TABLE CRUD *"
"*****"

```

```

" CREATE
DATA(ls_usage) = VALUE zcfm_usage_log(
    client      = sy-mandt
    usage_id    = 'USG021'
    source_id   = 'SRC021'
    usage_date  = cl_abap_context_info=>get_system_date( )
    quantity    = '100.00'
    created_by  = sy-uname
    created_on  = cl_abap_context_info=>get_system_date( ) ).

INSERT zcfm_usage_log FROM @ls_usage.
IF sy-subrc = 0.
    out->write( |Usage Inserted: { ls_usage-usage_id }| ).
ELSE.
    out->write( |Usage Insert Failed: { ls_usage-usage_id }| ).
ENDIF.

" READ
DATA ls_read_usage TYPE zcfm_usage_log.
SELECT SINGLE * FROM zcfm_usage_log WHERE usage_id = 'USG021' INTO
@ls_read_usage.
IF sy-subrc = 0.
    out->write( |Usage Read -> ID: { ls_read_usage-usage_id }, Qty: { ls_read_usage-
quantity }| ).
ELSE.
    out->write( |Usage Read Failed: USG021 not found| ).
ENDIF.

" UPDATE
ls_read_usage-quantity = '120.50'.
UPDATE zcfm_usage_log SET quantity = @ls_read_usage-quantity
    WHERE usage_id = @ls_read_usage-usage_id.
IF sy-subrc = 0.
    out->write( |Usage Updated: { ls_read_usage-usage_id }| ).
ELSE.
    out->write( |Usage Update Failed: { ls_read_usage-usage_id }| ).
ENDIF.

" DELETE
DELETE FROM zcfm_usage_log WHERE usage_id = 'USG015'.
IF sy-subrc = 0.
    out->write( |Usage Deleted: USG015| ).
ELSE.
    out->write( |Usage Delete Failed: USG015 not found| ).
ENDIF.

ENDMETHOD.

ENDCLASS.

```

```

CLASS :- ZCL_EMISSION_REPORT
CLASS zcl_emission_report DEFINITION
PUBLIC
FINAL
CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES if_oo_adt_classrun .

PROTECTED SECTION.
PRIVATE SECTION.

TYPES: BEGIN OF ty_emission,
    usage_id      TYPE zcfm_usage_log-usage_id,
    usage_date    TYPE zcfm_usage_log-usage_date,
    source_name   TYPE zcfm_source-source_name,
    quantity      TYPE zcfm_usage_log-quantity,
    unit          TYPE zcfm_source-unit_of_measure,
    factor_value  TYPE zcfm_factor-factor_value,
    emission_result TYPE p LENGTH 10 DECIMALS 2,
END OF ty_emission.

DATA: lt_report TYPE TABLE OF ty_emission,
      ls_emission TYPE zcfm_emission.

ENDCLASS.

CLASS zcl_emission_report IMPLEMENTATION.

METHOD if_oo_adt_classrun~main.

SELECT a~usage_id,
      a~usage_date,
      b~source_name,
      a~quantity,
      b~unit_of_measure,
      c~factor_value,
      ( a~quantity * c~factor_value ) AS emission_result
FROM zcfm_usage_log AS a
INNER JOIN zcfm_source AS b ON a~source_id = b~source_id
INNER JOIN zcfm_factor AS c ON a~source_id = c~source_id
INTO TABLE @lt_report.

IF lt_report IS INITIAL.
  out->write( 'No data found in emission report.' ).
  RETURN.
ENDIF.

" OUTPUT
LOOP AT lt_report INTO DATA(ls_report).
  out->write( |Usage ID: { ls_report-usage_id }, Source: { ls_report-source_name },
Emission: { ls_report-emission_result }| ).

CLEAR ls_emission.

```

```

ls_emission-log_id      = |LOG{ sy-tabix WIDTH = 4 PAD = '0' }|.
ls_emission-source_type = ls_report-source_name.
ls_emission-date_log   = ls_report-usage_date.
ls_emission-emission_kg = ls_report-emission_result.

" Insert into ZEM_EMISSION_LOG
MODIFY zcfm_emission FROM @ls_emission.

ENDLOOP.
out->write( |Total emission records inserted: { lines( lt_report ) }| ).

ENDMETHOD.
ENDCLASS.

```

CLASS :- ZCL_CERTIFICATE

```

CLASS zcl_certificate DEFINITION
PUBLIC
FINAL
CREATE PUBLIC.

PUBLIC SECTION.
INTERFACES if_oo_adt_classrun.
PROTECTED SECTION.
PRIVATE SECTION.
ENDCLASS.

CLASS zcl_certificate IMPLEMENTATION.

METHOD if_oo_adt_classrun~main.

DATA: lt_emission    TYPE TABLE OF zcfm_emission,
      ls_emission    TYPE zcfm_emission,
      ls_credit      TYPE zcfm_credit,
      lv_index       TYPE i VALUE 1,
      lv_cert_id    TYPE c LENGTH 10,
      ls_cert        TYPE zcfm_certificate,
      lt_credit      TYPE TABLE OF zcfm_credit,
      lt_cert_update TYPE TABLE OF zcfm_certificate,
      lv_updated     TYPE i VALUE 0.

" For error logging
TYPES: BEGIN OF ty_error_log,
      source_type TYPE zcfm_certificate-source_type,
      reason      TYPE string,
      timestamp   TYPE string,
      END OF ty_error_log.

DATA: lt_error_log TYPE TABLE OF ty_error_log.

" Step 1: Read Emission Log
SELECT * FROM zcfm_emission INTO TABLE @lt_emission.

LOOP AT lt_emission INTO ls_emission.

```

```

CLEAR ls_credit.
ls_credit-log_id = ls_emission-log_id.
ls_credit-source_type = ls_emission-source_type.
ls_credit-emission = ls_emission-emission_kg.

" Step 2: Assign carbon credit
IF ls_emission-emission_kg <= 100.
  ls_credit-credit_score = 5.
ELSEIF ls_emission-emission_kg <= 500.
  ls_credit-credit_score = 3.
ELSEIF ls_emission-emission_kg <= 1000.
  ls_credit-credit_score = 1.
ELSE.
  ls_credit-credit_score = 0.
ENDIF.

" Step 3: Insert credit entry
APPEND ls_credit TO lt_credit.
INSERT zcfm_credit FROM @ls_credit.
ENDLOOP.

SORT lt_credit BY source_type.

" Step 4: Group by source_type to generate certificates
LOOP AT lt_credit INTO DATA(ls_grouped) GROUP BY ( source_type = ls_grouped-
source_type ) INTO DATA(group).

CLEAR ls_cert.
lv_cert_id = |CERT{ lv_index WIDTH = 4 PAD = '0' }|.
lv_index += 1.

ls_cert-cert_id = lv_cert_id.
ls_cert-source_type = group-source_type.
ls_cert-cert_date = cl_abap_context_info=>get_system_date( ).
ls_cert-expiry_date = cl_abap_context_info=>get_system_date( ) + 365.
IF ls_cert-expiry_date < cl_abap_context_info=>get_system_date( ) .
  ls_cert-status = 'Expired'.
ELSE.
  ls_cert-status = 'Active'.
ENDIF.

LOOP AT GROUP group ASSIGNING FIELD-SYMBOL(<credit_entry>).
  ls_cert-total_emission += <credit_entry>-emission.
  ls_cert-total_credit += <credit_entry>-credit_score.
ENDLOOP.

" Step 4.1: Check if certificate already exists for today
SELECT SINGLE cert_id FROM zcfm_certificate
  WHERE source_type = @ls_cert-source_type
    AND cert_date = @ls_cert-cert_date
  INTO @DATA(lv_existing_cert_id).

IF sy-subrc = 0.
  APPEND VALUE #( source_type = ls_cert-source_type

```

```

        reason = 'Duplicate certificate already exists for today.'
        ) TO lt_error_log.
CONTINUE.
ENDIF.

" Step 5: Insert Certificate if valid
IF ls_cert-total_credit > 0 AND ls_cert-total_credit <= 1000.
  INSERT zcfm_certificate FROM @ls_cert.
  IF sy-subrc = 0.
    out->write( |☒ Certificate { lv_cert_id } issued for source { ls_cert-source_type }: { ls_cert-total_credit } credits on { ls_cert-cert_date }.| ).
  ELSE.
    APPEND VALUE #( source_type = ls_cert-source_type
      reason = 'Insert failed for unknown reason'
      ) TO lt_error_log.
  ENDIF.
ELSE.
  APPEND VALUE #( source_type = ls_cert-source_type
    reason = |Invalid credit score: { ls_cert-total_credit }|
    ) TO lt_error_log.
ENDIF.

ENDLOOP.

" Step 6: Commit once
COMMIT WORK.

" Step 7: Expire old certificates
SELECT * FROM zcfm_certificate
  WHERE status = 'Active'
  INTO TABLE @lt_cert_update.

LOOP AT lt_cert_update INTO DATA(ls_old_cert).
  IF ls_old_cert-expiry_date < cl_abap_context_info->get_system_date( ).
    ls_old_cert-status = 'Expired'.
    UPDATE zcfm_certificate FROM @ls_old_cert.
    IF sy-subrc = 0.
      lv_updated += 1.
    ENDIF.
  ENDIF.
ENDLOOP.
out->write( |☒ Updated { lv_updated } certificate(s) to status 'Expired'.| ).

" Step 8: Print Summary of Skipped Entries
IF lt_error_log IS NOT INITIAL.
  out->write( |⚠ Skipped Certificate Entries Summary:| ).
  LOOP AT lt_error_log INTO DATA(ls_log).
    out->write( | - Source: { ls_log-source_type }, Reason: { ls_log-reason }| ).
  ENDLOOP.
ENDIF.

ENDMETHOD.

ENDCLASS.

```

CLASS :- ZCL_SUMMARY_REPORT

```
CLASS zcl_summary_report DEFINITION
PUBLIC
FINAL
CREATE PUBLIC .
```

PUBLIC SECTION.

```
INTERFACES if_oo_adt_classrun .
PROTECTED SECTION.
PRIVATE SECTION.
ENDCLASS.
```

```
CLASS zcl_summary_report IMPLEMENTATION.
```

```
METHOD if_oo_adt_classrun~main.
```

```
DATA: lt_cert      TYPE TABLE OF zcfm_certificate,
      ls_cert      TYPE zcfm_certificate,
      lv_line      TYPE string,
      lvIssued     TYPE i VALUE 0,
      lvFailed     TYPE i VALUE 0,
      lvExpired    TYPE i VALUE 0.
```

```
" Step 1: Fetch all certificates
```

```
SELECT * FROM zcfm_certificate ORDER BY cert_date DESCENDING INTO TABLE
@lt_cert.
lvIssued = lines( lt_cert ).
```

```
" Step 2: Count expired certificates
```

```
LOOP AT lt_cert INTO ls_cert WHERE status = 'Expired'.
  lv_expired += 1.
ENDLOOP.
```

```
" Step 4: Print Summary Header
```

```
out->write( 'Carbon Certificate Summary Report:' ).
out->write(
'-----').
out->write( || Total Issued : { lvIssued }| ).
out->write( || Total Failed : { lvFailed }| ).
out->write( || Total Expired : { lvExpired }| ).
```

```

" Step 5: Detailed List
LOOP AT lt_cert INTO ls_cert.
  lv_line = |Certificate ID : { ls_cert-cert_id }| && cl_abap_char_utilities=>newline &&
             |Source Type   : { ls_cert-source_type }| && cl_abap_char_utilities=>newline &&
             |Total Emission : { ls_cert-total_emission } kg| &&
  cl_abap_char_utilities=>newline &&
             |Total Credit   : { ls_cert-total_credit }| && cl_abap_char_utilities=>newline &&
             |Status        : { ls_cert-status }| && cl_abap_char_utilities=>newline &&
             |Issue Date    : { ls_cert-cert_date }| && cl_abap_char_utilities=>newline &&
             |Expiry Date   : { ls_cert-expiry_date }| && cl_abap_char_utilities=>newline &&
             '-----'.
  out->write( lv_line ).
ENDLOOP.

ENDMETHOD.
ENDCLASS.

```

RESTFUL APPLICATION (RAP)

5. Project Description

Overview

The **Carbon Emission Management System (CEMS)** is a backend solution developed using the **RESTful ABAP Programming Model (RAP)**. It enables organizations to monitor their energy usage, calculate carbon emissions, and generate carbon credit certificates automatically based on usage data. The system is fully built on CDS views, RAP behavior classes, and ABAP logic to ensure automation, scalability, and data integrity.

It allows users to manage energy sources, define emission factors, log usage, compute emissions, and certify low-carbon activities with credit scores — supporting sustainable operations and environmental compliance.

Key Features

- **Source Management:** Master data of energy sources (like solar, fuel, grid) stored in ZEM_SOURCES
- **Factor Table:** CO₂ emission factors per unit defined in ZEM_FACTOR_TABLE.
- **Usage Logging:** All source usage data is captured in ZEM_USAGE_LOG.
- **Emission Calculation:** ZEM_EMISsion CDS view calculates total emission as quantity × co2_factor.
- **Credit Calculation:** ZI_CREDIT_REPORT computes credit scores for emissions below 1000.
- **Certificate Generation:** Certificates are stored in ZEM_CERTIFICATE with status (Active/Expired).
- **Eligibility View:** ZI_ELIGCERTIFICATE shows all issued certificates.

Benefits

- **Automated Carbon Tracking:** No manual calculations—everything is computed via RAP and CDS logic
- **Promotes Sustainability:** Rewards lower emissions with higher credit scores.
- **Validated Entries:** Behaviour classes ensure only valid data is inserted.
- **RAP-Based Architecture:** Clean, modern ABAP code with service binding and OData-readiness.

- **Integration Ready:** Can be extended to Fiori apps or third-party dashboards.

7. CDS Views (Data Model Layer)

7.1. Data Definitions

7.1.1. **ZEM_USAGE_ENTITY** : Root entity view for energy usage logs. Stores usage data with quantity, date, and source reference.

7.1.2. **ZEM_EMISSION** : Calculates total carbon emissions by joining usage data with source and emission factor tables. Computes $\text{total_emission} = \text{quantity} \times \text{co2_factor}$.

7.1.3. **ZI_CREDIT_REPORT** : Computes carbon credit scores based on emission levels. Credits are given when $\text{total_emission} \leq 1000$, calculated dynamically.

7.1.4. **ZI_ELIGCERTIFICATE** : Lists all eligible carbon credit certificates issued, including emissions, scores, and validity dates.

7.2. Metadata Extensions (UI)

7.2.1. **ZC_EM_USAGE_LOG000** : UI projection of usage log data with fields like UsageId, SourceId, UsageDate, and Quantity, including audit fields for tracking changes.

7.2.2. **ZI_EMISSIONREPORT_UI** : UI annotation for **ZEM_EMISSION** showing usage ID, source details, CO2 factor, and calculated total emissions.

7.2.3. **ZI_CERTIFICATE_UI** : UI annotation for **ZI_ELIGCERTIFICATE**, displaying certificate ID, emissions, credit score, and certificate status with issue and expiry dates.

8. Service Definition & Binding

8.1. Service Definitions

8.2. Service Bindings

8.2.1. **USAGE_LOG_TABLE**

8.2.2. **EMISSION REPORT**

8.2.3. **CERTIFICATE SUMMARY DASHBOARD**

9. Behaviour Implementation Classes

9.1. **ZBP_CFM_USAGE_ENTITY** – Handles insert operations for source, factor, and usage logs in the Carbon Emission Management system.

9.2. **ZBP_CERTIFY** – Generates and inserts carbon emission certificates from credit reports with UUID-based IDs, setting status and expiry logic.

Carbon Usage Tracker

The screenshot shows a Fiori Elements App interface titled "Preview for Fiori Elements App". The main area displays a table of usage records with columns: Usageld, Sourceld, UsageDate, Quantity, User Name, Time Stamp, Changed On, and Changed On. The table contains 11 rows of data. At the top of the table, there are filter fields for Usageld, Sourceld, UsageDate, Quantity, User Name, Time Stamp, and Changed On. Below the table are buttons for Create, Delete, and other actions. The status bar at the bottom shows weather information (30°C, Mostly cloudy), system icons, and the date/time (31-07-2025, 17:53).

Usageld	Sourceld	UsageDate	Quantity	User Name	Time Stamp	Changed On	Changed On
USG001	SRC001	Jul 1, 2025	120.50	ADMIN	Jul 1, 2025, 4:00:00 PM	Jul 1, 2025, 4:00:00 PM	Jul 1, 2025, 4:00:00 PM
USG002	SRC002	Jul 1, 2025	75.00	ADMIN	Jul 1, 2025, 4:05:00 PM	Jul 1, 2025, 4:05:00 PM	Jul 1, 2025, 4:05:00 PM
USG003	SRC003	Jul 2, 2025	500.00	DHRUVA	Jul 2, 2025, 3:30:00 PM	Jul 2, 2025, 3:30:00 PM	Jul 2, 2025, 3:30:00 PM
USG004	SRC004	Jul 2, 2025	80.00	DHRUVA	Jul 2, 2025, 3:45:00 PM	Jul 2, 2025, 3:45:00 PM	Jul 2, 2025, 3:45:00 PM
USG005	SRC005	Jul 3, 2025	300.00	ADMIN	Jul 3, 2025, 5:00:00 PM	Jul 3, 2025, 5:00:00 PM	Jul 3, 2025, 5:00:00 PM
USG006	SRC006	Jul 3, 2025	55.00	ADMIN	Jul 3, 2025, 5:15:00 PM	Jul 3, 2025, 5:15:00 PM	Jul 3, 2025, 5:15:00 PM
USG007	SRC007	Jul 4, 2025	42.00	DHRUVA	Jul 4, 2025, 3:40:00 PM	Jul 4, 2025, 3:40:00 PM	Jul 4, 2025, 3:40:00 PM
USG008	SRC008	Jul 4, 2025	150.00	DHRUVA	Jul 4, 2025, 3:55:00 PM	Jul 4, 2025, 3:55:00 PM	Jul 4, 2025, 3:55:00 PM
USG009	SRC009	Jul 5, 2025	100.00	ADMIN	Jul 5, 2025, 5:30:00 PM	Jul 5, 2025, 5:30:00 PM	Jul 5, 2025, 5:30:00 PM
USG010	SRC010	Jul 5, 2025	90.00	ADMIN	Jul 5, 2025, 5:45:00 PM	Jul 5, 2025, 5:45:00 PM	Jul 5, 2025, 5:45:00 PM
USG011	SRC011	Jul 6, 2025	50.00	ADMIN	Jul 6, 2025, 3:30:00 PM	Jul 6, 2025, 3:30:00 PM	Jul 6, 2025, 3:30:00 PM

Create :-

The screenshot shows a Fiori Elements App interface titled "Preview for Fiori Elements App". The main area displays a form for creating a new usage record. The form has sections for General Information and Advanced Information. In the General Information section, fields include Usageld (USG023), UsageDate (Jul 16, 2025), User Name (ADMIN), and Changed On (Jul 31, 2025, 5:57:03 PM). The Advanced Information section includes Sourceld (SRC023) and Quantity (90.00). At the bottom right, there are buttons for "Draft updated", "Create", and "Discard Draft". The status bar at the bottom shows weather information (30°C, Mostly cloudy), system icons, and the date/time (31-07-2025, 17:57).

Edit :-

The screenshot shows an SAP Fiori Elements App interface. At the top, there are three tabs: 'ADT Logon', 'Preview for Fiori Elements App', and '##GENERATED Core Data Service Entity'. The main content area is titled '##GENERATED Core Data Service Entity' and displays a record for 'USG023'. The record details are as follows:

General Information			
UsageId: USG023	UsageDate: Jul 16, 2025	User Name: ADMIN	Changed On: Jul 31, 2025, 5:58:13 PM
Sourceld: SRC023	Quantity: 90.00	Time Stamp: Jul 16, 2025, 5:57:31 PM	Changed On: Jul 31, 2025, 5:58:13 PM

At the bottom right of the screen, there are buttons for 'Edit', 'Delete', and a refresh icon. The system status bar at the bottom shows '30°C Mostly cloudy', 'Search', and various system icons.

Read :-

The screenshot shows an SAP Fiori Elements App interface. At the top, there are three tabs: 'ADT Logon', 'Preview for Fiori Elements App', and 'Preview for Fiori Elements App'. The main content area is titled 'Standard' and shows a list of records. A filter message '1 filter active: Editing Status' is displayed above the table. The table has the following columns: UsageId, Sourceld, UsageDate, Quantity, User Name, Time Stamp, Changed On, and another 'Changed On' column. The data rows are as follows:

UsageId	Sourceld	UsageDate	Quantity	User Name	Time Stamp	Changed On	Changed On
USG008	SRC008	Jul 4, 2025	150.00	DHRUVA	Jul 4, 2025, 3:55:00 PM	Jul 4, 2025, 3:55:00 PM	Jul 4, 2025, 3:55:00 PM
USG009	SRC009	Jul 5, 2025	100.00	ADMIN	Jul 5, 2025, 5:30:00 PM	Jul 5, 2025, 5:30:00 PM	Jul 5, 2025, 5:30:00 PM
USG010	SRC010	Jul 5, 2025	90.00	ADMIN	Jul 5, 2025, 5:45:00 PM	Jul 5, 2025, 5:45:00 PM	Jul 5, 2025, 5:45:00 PM
USG011	SRC011	Jul 6, 2025	50.00	ADMIN	Jul 6, 2025, 3:30:00 PM	Jul 6, 2025, 3:30:00 PM	Jul 6, 2025, 3:30:00 PM
USG012	SRC012	Jul 6, 2025	60.00	DHRUVA	Jul 6, 2025, 4:00:00 PM	Jul 6, 2025, 4:00:00 PM	Jul 6, 2025, 4:00:00 PM
USG013	SRC013	Jul 7, 2025	70.00	ADMIN	Jul 7, 2025, 5:00:00 PM	Jul 7, 2025, 5:00:00 PM	Jul 7, 2025, 5:00:00 PM
USG014	SRC014	Jul 7, 2025	33.33	ADMIN	Jul 7, 2025, 5:15:00 PM	Jul 7, 2025, 5:15:00 PM	Jul 7, 2025, 5:15:00 PM
USG015	SRC015	Jul 8, 2025	21.00	DHRUVA	Jul 8, 2025, 3:30:00 PM	Jul 8, 2025, 3:30:00 PM	Jul 8, 2025, 3:30:00 PM
USG016	SRC016	Jul 8, 2025	95.00	DHRUVA	Jul 8, 2025, 4:00:00 PM	Jul 8, 2025, 4:00:00 PM	Jul 8, 2025, 4:00:00 PM
USG017	SRC017	Jul 9, 2025	61.00	ADMIN	Jul 9, 2025, 5:00:00 PM	Jul 9, 2025, 5:00:00 PM	Jul 9, 2025, 5:00:00 PM
USG018	SRC018	Jul 9, 2025	12.00	ADMIN	Jul 9, 2025, 5:30:00 PM	Jul 9, 2025, 5:30:00 PM	Jul 9, 2025, 5:30:00 PM
USG019	SRC019	Jul 10, 2025	44.00	DHRUVA	Jul 10, 2025, 3:30:00 PM	Jul 10, 2025, 3:30:00 PM	Jul 10, 2025, 3:30:00 PM
USG020	SRC020	Jul 10, 2025	66.00	DHRUVA	Jul 10, 2025, 4:00:00 PM	Jul 10, 2025, 4:00:00 PM	Jul 10, 2025, 4:00:00 PM
USG021	SRC001	Jul 23, 2025	100.00	ADMIN	Jul 23, 2025, 5:55:50 PM	Jul 31, 2025, 5:56:13 PM	Jul 31, 2025, 5:56:13 PM
USG023	SRC023	Jul 16, 2025	90.00	ADMIN	Jul 16, 2025, 5:57:31 PM	Jul 31, 2025, 5:58:13 PM	Jul 31, 2025, 5:58:13 PM

At the bottom right of the screen, there are buttons for 'Create', 'Delete', and a refresh icon. The system status bar at the bottom shows '30°C Mostly cloudy', 'Search', and various system icons.

Delete :-

The screenshot shows a SAP Fiori application interface. A modal dialog box is centered over a table of data. The dialog has a yellow warning icon and the word "Delete". Below it, the text "Delete object USG021?" is displayed. At the bottom of the dialog are two buttons: "Delete" (highlighted in blue) and "Cancel". The background table lists various entries with columns: UsageId, SourceId, UsageDate, Quantity, User Name, Time Stamp, Changed On, and a right-pointing arrow. One row, corresponding to USG021, has a checked checkbox in the first column. The status bar at the bottom shows the date as 31-07-2025 and the time as 17:59.

Emission Report

The screenshot shows a SAP Fiori application interface displaying an "Emission Report". The table has the following columns: usage_id, source_id, source_name, quantity, co2_factor, total_emission, and a right-pointing arrow. The data rows are as follows:

usage_id	source_id	source_name	quantity	co2_factor	total_emission	
USG001	SRC001	Solar Panel	120.50	0.00	0.00	>
USG002	SRC002	Diesel Generator	75.00	2.68	201.00	>
USG003	SRC003	Electric Grid	500.00	0.85	425.00	>
USG004	SRC004	Natural Gas Boiler	80.00	2.05	164.00	>
USG005	SRC005	Wind Turbine	300.00	0.00	0.00	>
USG006	SRC006	Petrol Generator	55.00	2.31	127.05	>
USG007	SRC007	Biogas Unit	42.00	0.45	18.90	>
USG008	SRC008	Hydro Power Plant	150.00	0.00	0.00	>
USG009	SRC009	Coal Furnace	100.00	2.42	242.00	>
USG010	SRC010	Solar Water Heater	90.00	0.00	0.00	>
USG011	SRC011	Battery Storage	50.00	0.20	10.00	>
USG012	SRC012	Geo-Thermal Pump	60.00	0.00	0.00	>
USG013	SRC013	Biofuel Station	70.00	1.80	126.00	>
USG014	SRC014	Electric Vehicle	33.33	0.50	16.66	>
USG015	SRC015	Steam Generator	21.00	1.00	21.00	>
USG016	SRC016	Furnace Oil Heater	95.00	2.95	280.25	>

The status bar at the bottom shows the date as 31-07-2025 and the time as 18:01.

Certificate Summary Dashboard

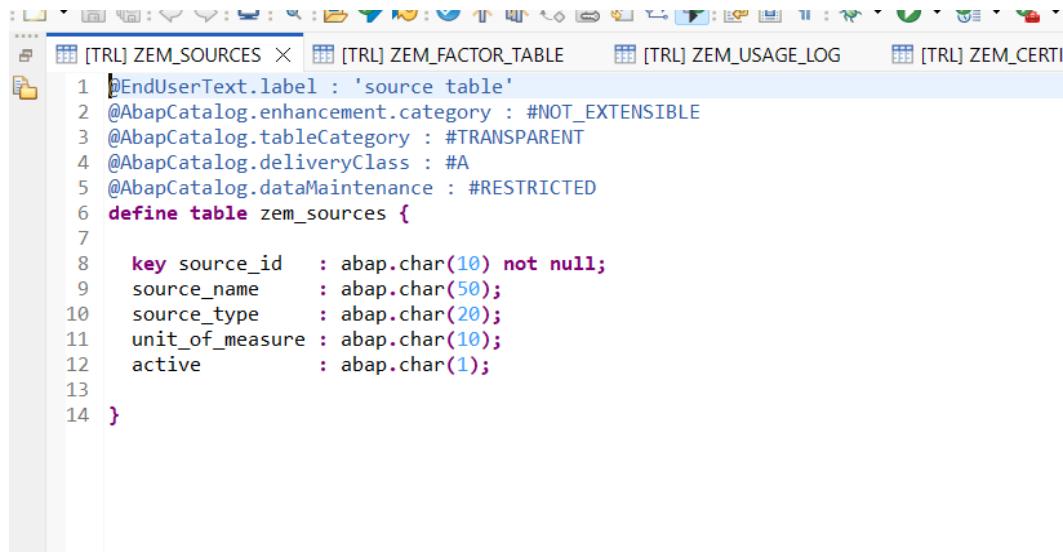
The screenshot shows a SAP Fiori Elements App interface titled "Preview for Fiori Elements App". The page is titled "Standard" and displays a table of certificate data. The columns include cert_id, source_name, total_emission, credit_score, issue_date, expiry_date, and status. The data is as follows:

cert_id	source_name	total_emission	credit_score	issue_date	expiry_date	status
6EFFC2	Diesel Gen	201.00	80	Aug 4, 2025	Aug 4, 2026	Active
6EFFC4	Electric G	425.00	58	Aug 4, 2025	Aug 4, 2026	Active
6EFFC6	Natural Ga	164.00	84	Aug 4, 2025	Aug 4, 2026	Active
6EFFC8	Wind Turbi	0.00	100	Aug 4, 2025	Aug 4, 2026	Active
6EFFCA	Petrol Gen	127.05	87	Aug 4, 2025	Aug 4, 2026	Active
6EFFCC	Biogas Uni	18.90	98	Aug 4, 2025	Aug 4, 2026	Active
6EFFCE	Hydro Powe	0.00	100	Aug 4, 2025	Aug 4, 2026	Active
6EFFD0	Coal Furna	242.00	76	Aug 4, 2025	Aug 4, 2026	Active
6EFFD2	Solar Wate	0.00	100	Aug 4, 2025	Aug 4, 2026	Active
6EFFD4	Battery St	10.00	99	Aug 4, 2025	Aug 4, 2026	Active
6EFFD6	Geo-Therma	0.00	100	Aug 4, 2025	Aug 4, 2026	Active
6EFFD8	Biofuel St	126.00	87	Aug 4, 2025	Aug 4, 2026	Active
6EFFDA	Electric V	16.66	98	Aug 4, 2025	Aug 4, 2026	Active
6EFFDC	Steam Gene	21.00	98	Aug 4, 2025	Aug 4, 2026	Active
6EFFDE	Furnace Oi	280.25	72	Aug 4, 2025	Aug 4, 2026	Active
6EFFE0	Compressed	6.10	00	Aug 4, 2025	Aug 4, 2026	Active

The interface includes a top navigation bar with "ADT Logon" and "Preview for Fiori Elements App", and a bottom taskbar with various icons and system status.

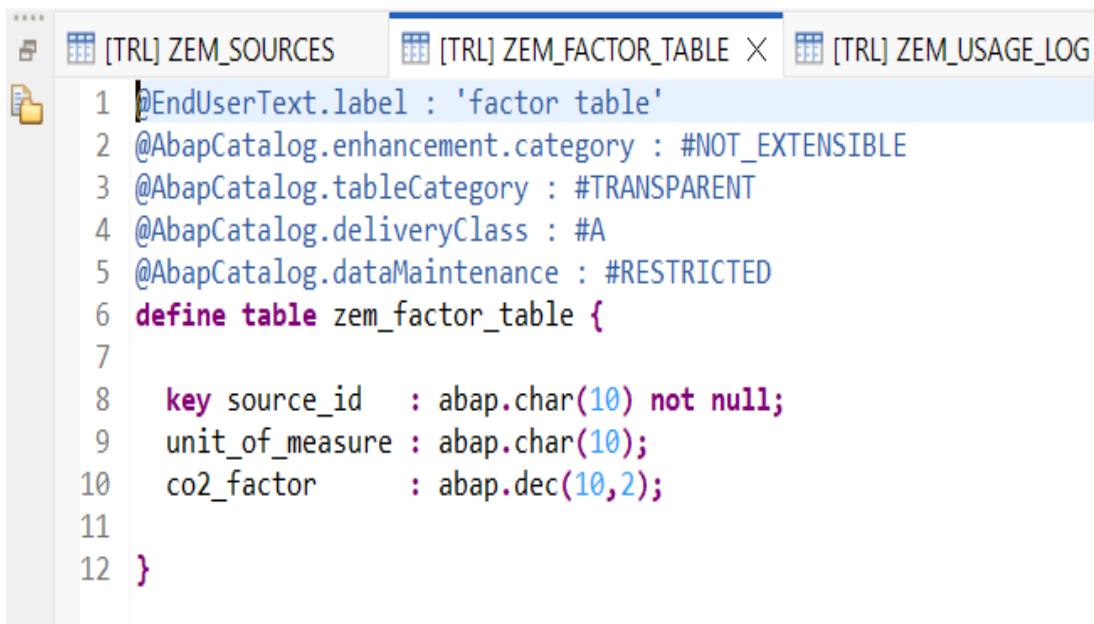
Database Tables :-

1. ZEM_SOURCES – Source Table



```
1 @EndUserText.label : 'source table'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zem_sources {
7
8   key source_id    : abap.char(10) not null;
9   source_name      : abap.char(50);
10  source_type      : abap.char(20);
11  unit_of_measure : abap.char(10);
12  active          : abap.char(1);
13
14 }
```

2. ZEM_FACTOR_TABLE – Factor Table



```
1 @EndUserText.label : 'factor table'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zem_factor_table {
7
8   key source_id    : abap.char(10) not null;
9   unit_of_measure : abap.char(10);
10  co2_factor      : abap.dec(10,2);
11
12 }
```

3. ZEM_USAGE_LOG – Usage Log Table

The screenshot shows a SAP ABAP code editor window. The title bar has tabs for [TRL] ZEM_SOURCES, [TRL] ZEM_FACTOR_TABLE, [TRL] ZEM_USAGE_LOG (which is the active tab), and [TRL] ZEM_CREDIT. The code itself defines a table named zem_usage_log with various fields:

```
1 @EndUserText.label : 'log table'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zem_usage_log {
7
8   key client      : abap.clnt not null;
9   key usage_id    : abap.char(10) not null;
10  source_id      : abap.char(10);
11  usage_date     : abap.dats;
12  quantity       : abap.dec(10,2);
13  created_by     : syuname;
14  created_at     : timestamp;
15  loal_last_changed : abp_locinst_lastchange_tstmp;
16  last_changed   : abp_lastchange_tstmp;
17
18 }
```

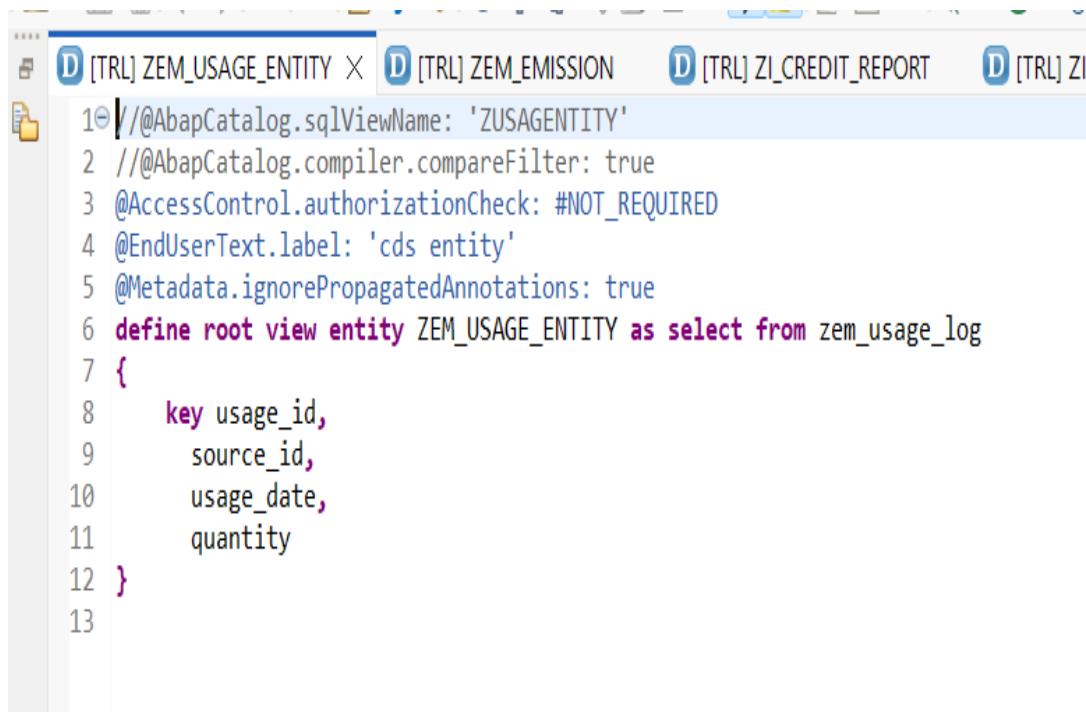
4. ZEM_CERTIFICATE – Certificate Table

The screenshot shows a SAP ABAP code editor window. The title bar has tabs for [TRL] ZEM_CERTIFICATE (active), [TRL] ZBP_CERTIFY, and [TRL] ZCL_CERTIFICATE. The code defines a table named zem_certificate with various fields:

```
1 @EndUserText.label : 'certificate manage'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zem_certificate {
7
8   key client      : abap.clnt not null;
9   key cert_id     : abap.char(32) not null;
10  usage_id       : abap.char(10);
11  source_name    : abap.char(10);
12  total_emission : abap.dec(10,2);
13  credit_score   : abap.int4;
14  issue_date     : abap.dats;
15  expiry_date    : abap.dats;
16  status         : abap.char(10);
17
18 }
```

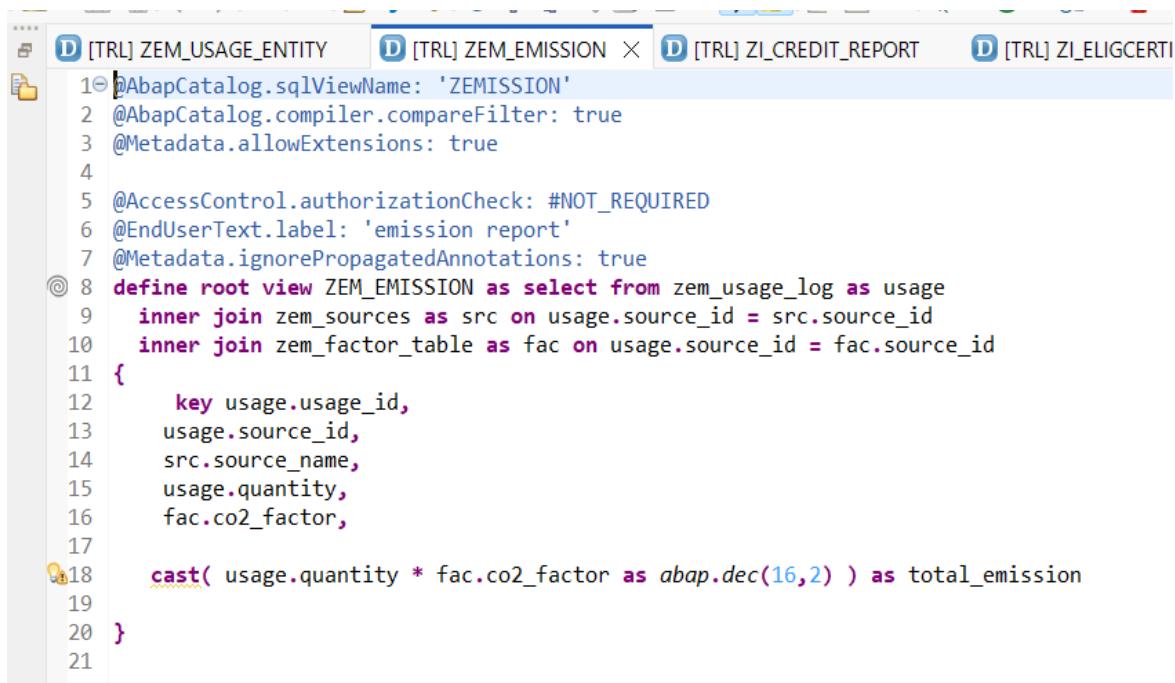
7. CDS VIEWS (DATA DEFINITION) :-

7.1.1 ZEM_USAGE_ENTITY



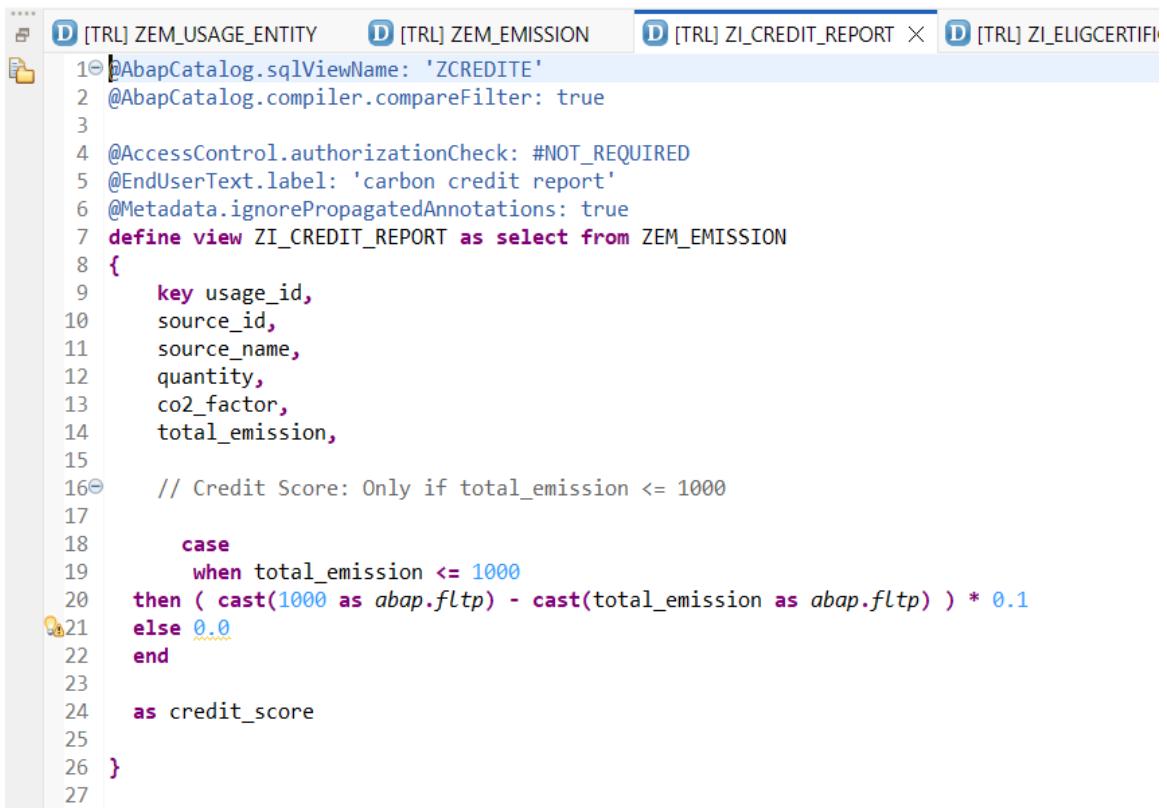
```
1 //@AbapCatalog.sqlViewName: 'ZUSAGENTITY'
2 //@AbapCatalog.compiler.compareFilter: true
3 @AccessControl.authorizationCheck: #NOT_REQUIRED
4 @EndUserText.label: 'cds entity'
5 @Metadata.ignorePropagatedAnnotations: true
6 define root view entity ZEM_USAGE_ENTITY as select from zem_usage_log
7 {
8     key usage_id,
9     source_id,
10    usage_date,
11    quantity
12 }
13
```

7.1.2 ZEM_EMISSION



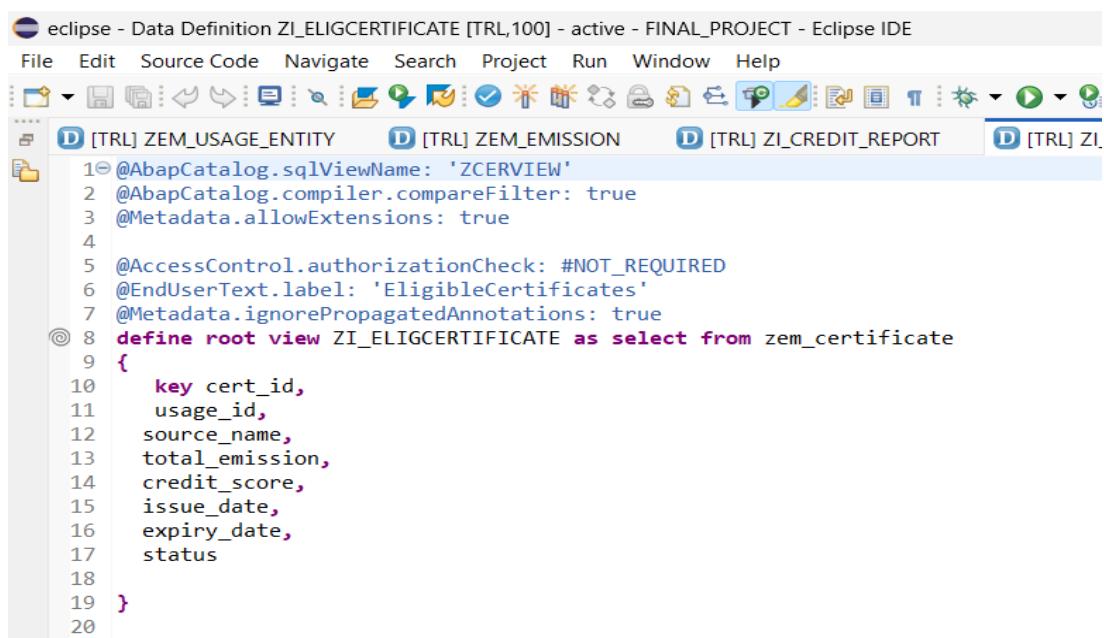
```
1 //@AbapCatalog.sqlViewName: 'ZEMISSION'
2 //@AbapCatalog.compiler.compareFilter: true
3 @Metadata.allowExtensions: true
4
5 @AccessControl.authorizationCheck: #NOT_REQUIRED
6 @EndUserText.label: 'emission report'
7 @Metadata.ignorePropagatedAnnotations: true
8 define root view ZEM_EMISSION as select from zem_usage_log as usage
9   inner join zem_sources as src on usage.source_id = src.source_id
10  inner join zem_factor_table as fac on usage.source_id = fac.source_id
11 {
12     key usage.usage_id,
13     usage.source_id,
14     src.source_name,
15     usage.quantity,
16     fac.co2_factor,
17
18     cast( usage.quantity * fac.co2_factor as abap.dec(16,2) ) as total_emission
19
20 }
21
```

7.1.3 ZI_CREDIT_REPORT



```
1 @AbapCatalog.sqlViewName: 'ZCREDITE'
2 @AbapCatalog.compiler.compareFilter: true
3
4 @AccessControl.authorizationCheck: #NOT_REQUIRED
5 @EndUserText.label: 'carbon credit report'
6 @Metadata.ignorePropagatedAnnotations: true
7 define view ZI_CREDIT_REPORT as select from ZEM_EMIS
8 {
9     key usage_id,
10    source_id,
11    source_name,
12    quantity,
13    co2_factor,
14    total_emission,
15
16    // Credit Score: Only if total_emission <= 1000
17
18    case
19        when total_emission <= 1000
20    then ( cast(1000 as abap.fltp) - cast(total_emission as abap.fltp) ) * 0.1
21 else 0.0
22 end
23
24 as credit_score
25
26 }
27
```

7.1.4 ZI_ELIGCERTIFICATE



```
1 @AbapCatalog.sqlViewName: 'ZCERVIEW'
2 @AbapCatalog.compiler.compareFilter: true
3 @Metadata.allowExtensions: true
4
5 @AccessControl.authorizationCheck: #NOT_REQUIRED
6 @EndUserText.label: 'EligibleCertificates'
7 @Metadata.ignorePropagatedAnnotations: true
8 define root view ZI_ELIGCERTIFICATE as select from zem_certificate
9 {
10    key cert_id,
11    usage_id,
12    source_name,
13    total_emission,
14    credit_score,
15    issue_date,
16    expiry_date,
17    status
18
19 }
```

Metadata Extension :-

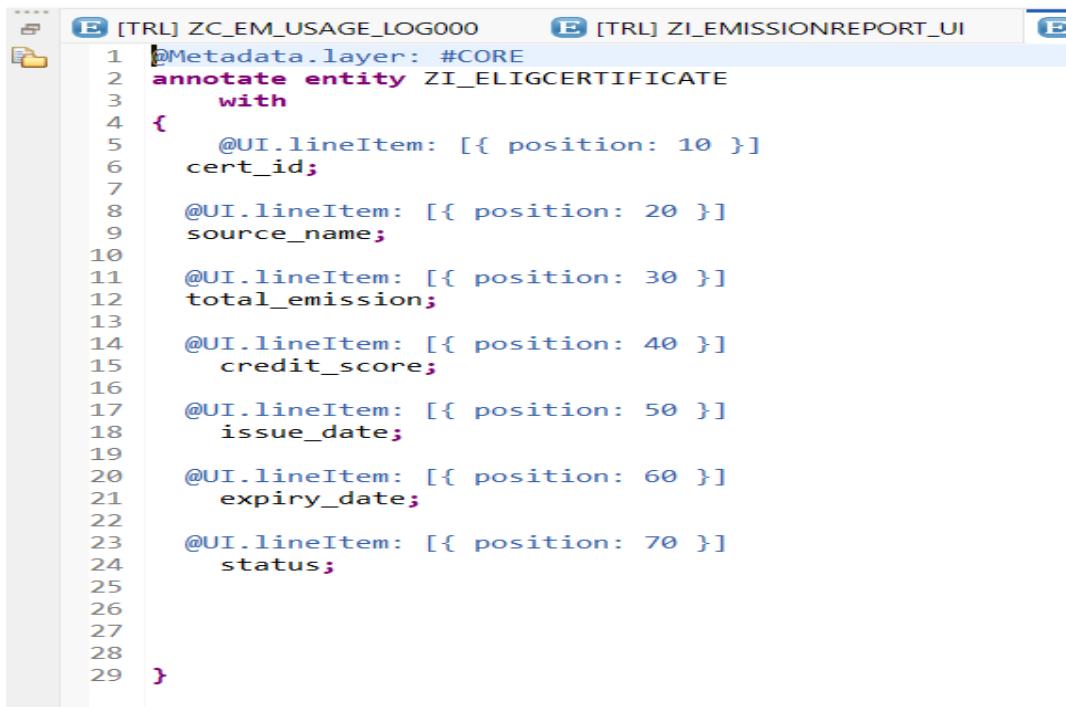
7.2.1 ZC_EM_USAGE_LOG000

```
1 [TRL] ZC_EM_USAGE_LOG000 × 2 [TRL] ZI_EMISSIONREPORT_UI 3 [TRL] ZI_CER
1@Metadata.layer: #CORE
2 @UI.headerInfo.title.type: #STANDARD
3 @UI.headerInfo.title.value: 'UsageId'
4 @UI.headerInfo.description.type: #STANDARD
5 @UI.headerInfo.description.value: 'UsageId'
6 annotate view ZC_EM_USAGE_LOG000 with
7 {
8+ @EndUserText.label: 'UsageId' UsageId;
27
28
29+ @EndUserText.label: 'SourceId' SourceId;
41
42
43+ @EndUserText.label: 'UsageDate' UsageDate;
55
56
57+ @EndUserText.label: 'Quantity' Quantity;
69
70
71+ @UI.identification: [ {} CreatedBy;
80
81
82+ @UI.identification: [ {} CreatedAt;
91
92
93+ @UI.identification: [ {} LoalLastChanged;
102
103
104+ @UI.identification: [ {} LastChanged;
113
114 }
```

7.2.1 ZI_EMISSIONREPORT_UI

```
1 [TRL] ZC_EM_USAGE_LOG000 × 2 [TRL] ZI_EMISSIONREPORT_UI >
1@Metadata.layer: #CORE
2 annotate entity ZEM_EMISIION
3   with {
4     @UI.lineItem: [{ position: 10 }]
5     usage_id;
6
7     @UI.lineItem: [{ position: 20 }]
8     source_id;
9
10    @UI.lineItem: [{ position: 30 }]
11    source_name;
12
13    @UI.lineItem: [{ position: 40 }]
14    quantity;
15
16    @UI.lineItem: [{ position: 50 }]
17    co2_factor;
18
19    @UI.lineItem: [{ position: 60 }]
20    total_emission;
21
22
23
24
25 }
```

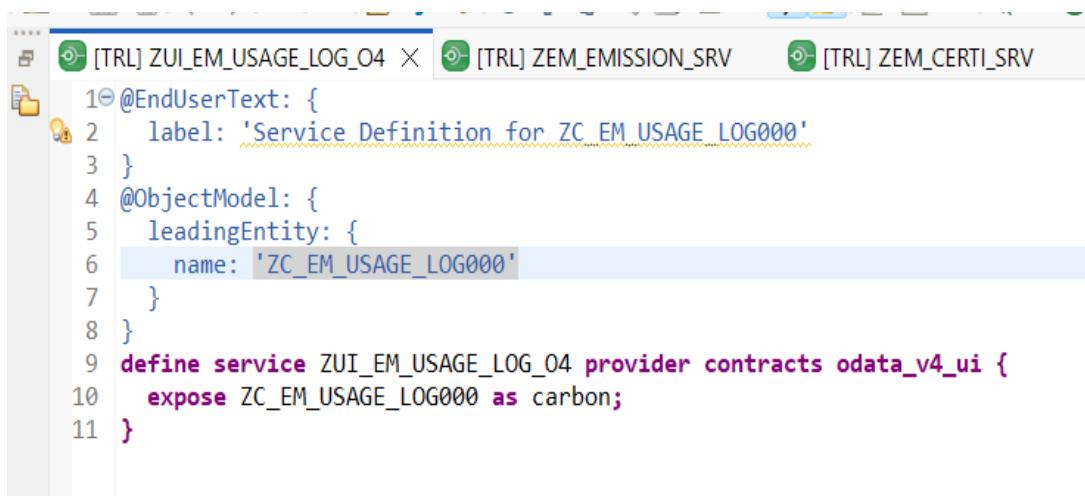
7.2.3 ZI_CERTIFICATE_UI



```
1 @Metadata.layer: #CORE
2 annotate entity ZI_ELGCERTIFICATE
3   with
4   {
5     @UI.lineItem: [{ position: 10 }]
6     cert_id;
7
8     @UI.lineItem: [{ position: 20 }]
9     source_name;
10
11    @UI.lineItem: [{ position: 30 }]
12    total_emission;
13
14    @UI.lineItem: [{ position: 40 }]
15    credit_score;
16
17    @UI.lineItem: [{ position: 50 }]
18    issue_date;
19
20    @UI.lineItem: [{ position: 60 }]
21    expiry_date;
22
23    @UI.lineItem: [{ position: 70 }]
24    status;
25
26
27
28
29 }
```

8. Service Definition :-

ZUI_EM_USAGE_LOG_04



```
1 @EndUserText: {
2   label: 'Service Definition for ZC_EM_USAGE_LOG000'
3 }
4 @ObjectModel: {
5   leadingEntity: {
6     name: 'ZC_EM_USAGE_LOG000'
7   }
8 }
9 define service ZUI_EM_USAGE_LOG_04 provider contracts odata_v4_ui {
10   expose ZC_EM_USAGE_LOG000 as carbon;
11 }
```

ZEM_EMISsION_SRV

The screenshot shows a code editor window with three tabs at the top: [TRL] ZUI_EM_USAGE_LOG_04, [TRL] ZEM_EMISsION_SRV (which is the active tab), and [TRL] ZEM_CERTI_SRv. The code in the editor is:

```
1 @EndUserText.label: 'emission service'
2 define service ZEM_EMISsION_SRV {
3   expose ZEM_EMISsION;
4 }
```

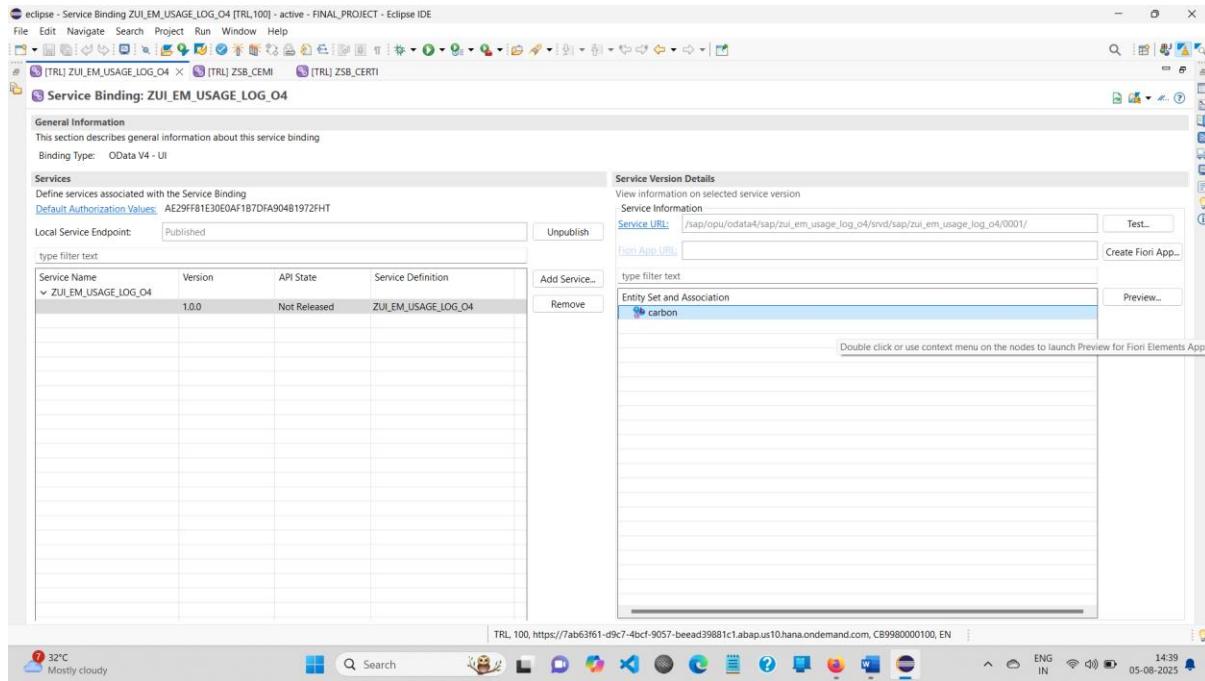
ZEM_CERTI_SRv

The screenshot shows a code editor window with three tabs at the top: [TRL] ZUI_EM_USAGE_LOG_04, [TRL] ZEM_EMISsION_SRV, and [TRL] ZEM_CERTI_SRv (which is the active tab). The code in the editor is:

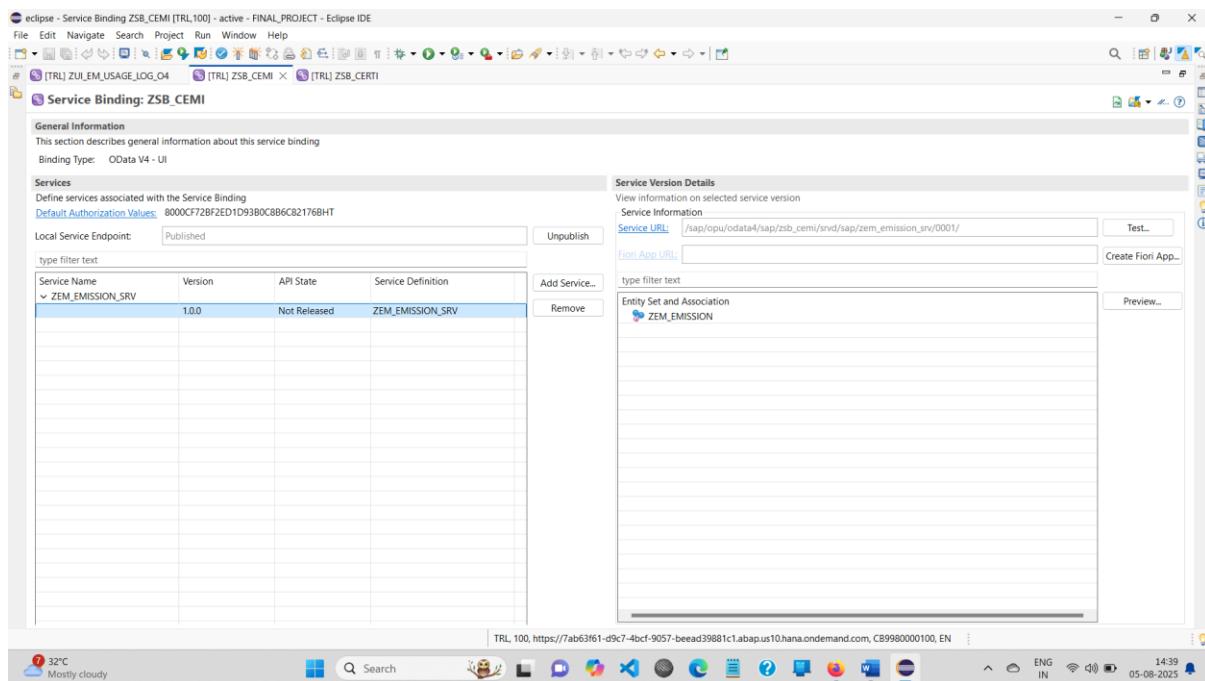
```
1 @EndUserText.label: 'certificate'
2 define service ZEM_CERTI_SRv {
3   expose ZI_ELIGCERTIFICATE;
4 }
```

Service Binding :-

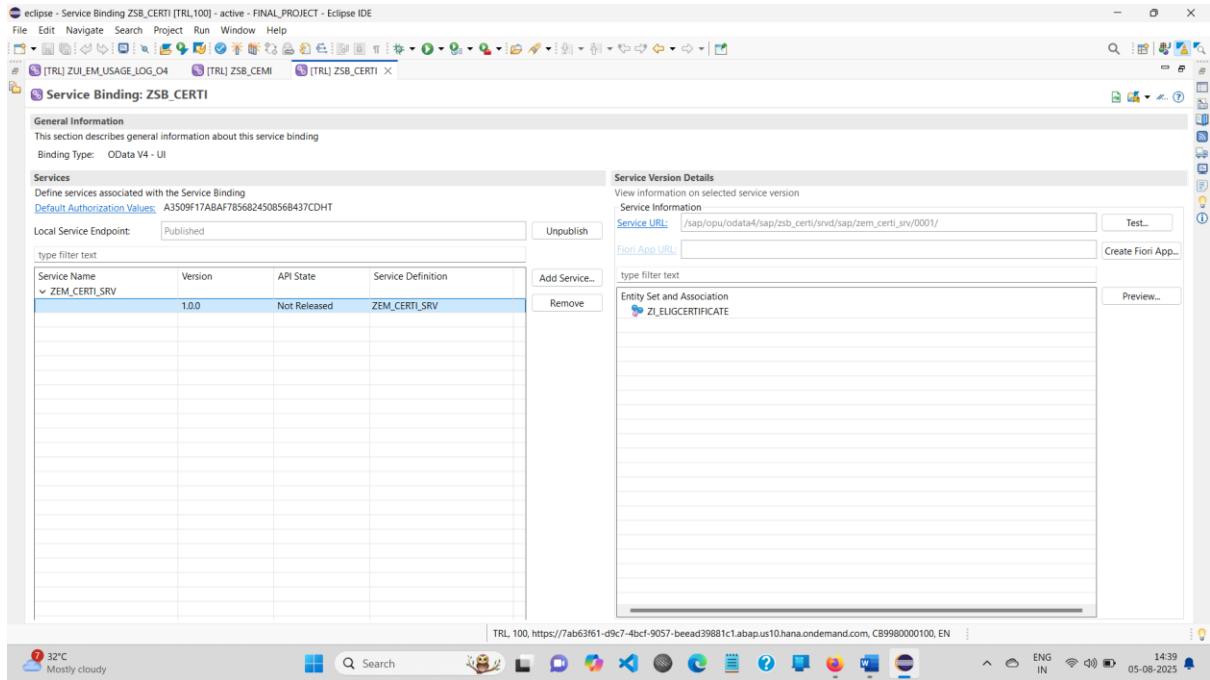
USAGE_LOG_TABLE



EMISSION_REPORT



CERTIFICATE_DASHBOARD



9. Behaviour Implementation Classes :-

CLASS : ZBP_CEM_USAGE_ENTITY

```
CLASS zbp_cem_usage_entity DEFINITION
PUBLIC
FINAL
CREATE PUBLIC .
```

PUBLIC SECTION.

METHODS:

```
    insert_source IMPORTING is_source TYPE zem_sources
                  RETURNING VALUE(rv_message) TYPE string,
    insert_factor IMPORTING is_factor TYPE zem_factor_table
                  RETURNING VALUE(rv_message) TYPE string,
```

```
    insert_usage_log IMPORTING is_usage TYPE zem_usage_log
                      RETURNING VALUE(rv_message) TYPE string.
```

INTERFACES if_oo_adt_classrun .

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS zbp_cem_usage_entity IMPLEMENTATION.

METHOD if_oo_adt_classrun~main.

DATA lt_sources TYPE STANDARD TABLE OF zem_sources.

DATA lv_message TYPE string.

" Add multiple source entries

lt_sources = VALUE # (

(source_id = 'SRC001' source_name = 'Solar Panel' source_type = 'Renewable'
unit_of_measure = 'kWh' active = 'X')
(source_id = 'SRC002' source_name = 'Diesel Generator' source_type = 'Fuel'
unit_of_measure = 'Litre' active = 'X')
(source_id = 'SRC003' source_name = 'Electric Grid' source_type = 'Power'
unit_of_measure = 'kWh' active = 'X')
(source_id = 'SRC004' source_name = 'Natural Gas Boiler' source_type = 'Fuel'
unit_of_measure = 'm3' active = 'X')
(source_id = 'SRC005' source_name = 'Wind Turbine' source_type = 'Renewable'
unit_of_measure = 'kWh' active = 'X')

).

" Loop and insert each one

LOOP AT lt_sources INTO DATA(ls_source).

lv_message = insert_source(ls_source).

out->write(lv_message).

ENDLOOP.

DATA: lt_factors TYPE STANDARD TABLE OF zem_factor_table,
ls_factor TYPE zem_factor_table.

lt_factors = VALUE # (

(source_id = 'SRC001' unit_of_measure = 'kWh' co2_factor = '0.00') " Solar Panel
(source_id = 'SRC002' unit_of_measure = 'Litre' co2_factor = '2.68') " Diesel
(source_id = 'SRC003' unit_of_measure = 'kWh' co2_factor = '0.85') " Electric Grid
(source_id = 'SRC004' unit_of_measure = 'm3' co2_factor = '2.05') " Natural Gas
(source_id = 'SRC005' unit_of_measure = 'kWh' co2_factor = '0.00') " Wind

).

LOOP AT lt_factors INTO ls_factor.

lv_message = insert_factor(ls_factor). " Call your insert method

out->write(lv_message).

ENDLOOP.

DATA: lt_usages TYPE STANDARD TABLE OF zem_usage_log,

ls_usage TYPE zem_usage_log.

DATA(lv_today) = cl_abap_context_info=>get_system_date().

lt_usages = VALUE # (

(client = '100' usage_id = 'USG001' source_id = 'SRC001' usage_date = '20250701' quantity = '120.50' created_by = 'ADMIN' created_at = '20250701103000' last_changed = '20250701103000' last_changed = '20250701103000')

```

( client = '100' usage_id = 'USG002' source_id = 'SRC002' usage_date = '20250701' quantity
= '75.00' created_by = 'ADMIN' created_at = '20250701103500' last_changed =
'20250701103500' last_changed = '20250701103500' )
( client = '100' usage_id = 'USG003' source_id = 'SRC003' usage_date = '20250702' quantity
= '500.00' created_by = 'DHRUVA' created_at = '20250702100000' last_changed =
'20250702100000' last_changed = '20250702100000' )
( client = '100' usage_id = 'USG004' source_id = 'SRC004' usage_date = '20250702' quantity
= '80.00' created_by = 'DHRUVA' created_at = '20250702101500' last_changed =
'20250702101500' last_changed = '20250702101500' )
( client = '100' usage_id = 'USG005' source_id = 'SRC005' usage_date = '20250703' quantity
= '300.00' created_by = 'ADMIN' created_at = '20250703113000' last_changed =
'20250703113000' last_changed = '20250703113000' )
).

```

LOOP AT lt_usages INTO ls_usage.

```

lv_message = insert_usage_log( ls_usage ). " Call your insert method
out->write( lv_message ).
```

ENDLOOP.

ENDMETHOD.

METHOD insert_source.

" Basic check for existing source

```

DATA: lv_exists TYPE zem_sources-source_id.
```

```

SELECT SINGLE source_id
FROM zem_sources
WHERE source_id = @is_source-source_id
INTO @lv_exists.
```

IF sy-subrc = 0.

```

rv_message = |Source ID already exists: { is_source-source_id }|.
RETURN.
```

ENDIF.

INSERT zem_sources FROM @is_source.

IF sy-subrc = 0.

```

rv_message = |Source inserted: { is_source-source_id }|.
```

ELSE.

```

rv_message = |Failed to insert source: { is_source-source_id }|.
```

ENDIF.

ENDMETHOD.

METHOD insert_factor.

" Optional check: Ensure source exists

```

SELECT SINGLE source_id
FROM zem_sources
WHERE source_id = @is_factor-source_id
INTO @DATA(lv_source).
```

IF sy-subrc <> 0.

```

rv_message = |Invalid source ID for factor: { is_factor-source_id }|.
RETURN.
```

ENDIF.

INSERT zem_factor_table FROM @is_factor.

```

IF sy-subrc = 0.
  rv_message = |Factor inserted for source: { is_factor-source_id }| .
ELSE.
  rv_message = |Failed to insert factor for source: { is_factor-source_id }| .
ENDIF.
ENDMETHOD.

METHOD insert_usage_log.
  " Optional: Validate source exists
  SELECT SINGLE source_id
    FROM zem_sources
   WHERE source_id = @is_usage-source_id
   INTO @DATA(lv_valid_source).
  IF sy-subrc <> 0.
    rv_message = |Invalid source ID in usage log: { is_usage-source_id }| .
    RETURN.
  ENDIF.

  " Validate: Quantity must be greater than 0
  IF is_usage-quantity <= 0.
    rv_message = |Quantity must be greater than 0 for usage ID: { is_usage-usage_id }| .
    RETURN.
  ENDIF.

  INSERT zem_usage_log FROM @is_usage.
  IF sy-subrc = 0.
    rv_message = |Usage log inserted: { is_usage-usage_id }| .
  ELSE.
    rv_message = |Failed to insert usage log: { is_usage-usage_id }| .
  ENDIF.
ENDMETHOD.

ENDCLASS.

```

CLASS : ZBP_CERTIFY

```

CLASS zbp_certify DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
    PROTECTED SECTION.
    PRIVATE SECTION.
ENDCLASS.

```

```

CLASS zbp_certify IMPLEMENTATION.

METHOD if_oo_adt_classrun~main.
  DATA: lt_report TYPE TABLE OF zi_credit_report,
        ls_cert  TYPE zem_certificate,

```

```

lv_count TYPE i VALUE 0,
lv_uuid_string TYPE string,
lv_uuid_full TYPE string,
lv_short_uuid TYPE string.

" 1. Fetch data from CDS View
SELECT * FROM zi_credit_report INTO TABLE @lt_report.

" 2. Loop through each and insert into ZEM_CERTIFICATE
LOOP AT lt_report INTO DATA(ls_report).

CLEAR : ls_cert,lv_uuid_string.

" Populate certificate structure
TRY.
  lv_uuid_full = cl_system_uuid->if_system_uuid_static~create_uuid_c32().
  lv_short_uuid = lv_uuid_full+0(3) && lv_uuid_full+26(3).

  ls_cert-cert_id = lv_short_uuid. " Use full UUID (char32)

  CATCH cx_uuid_error INTO DATA(lx_uuid).
    out->write( |△ UUID generation failed: { lx_uuid->get_text() }| ).
    CONTINUE.
  ENDTRY.

  ls_cert-usage_id = ls_report-usage_id.
  ls_cert-source_name = ls_report-source_name.
  ls_cert-total_emission = ls_report-total_emission.
  ls_cert-credit_score = ls_report-credit_score.
  ls_cert-issue_date = cl_abap_context_info->get_system_date().
  ls_cert-expiry_date = cl_abap_context_info->get_system_date() + 365.
  IF ls_cert-expiry_date < cl_abap_context_info->get_system_date() .
    ls_cert-status = 'Expired'.
  ELSE.
    ls_cert-status = 'Active'.
  ENDIF.

  " Insert into database table
  INSERT zem_certificate FROM @ls_cert.
  IF sy-subrc = 0.
    lv_count = lv_count + 1.
  ENDIF.

ENDLOOP.

out->write( |Inserted { lv_count } certificate(s) into ZEM_CERTIFICATE.| ).

ENDMETHOD.
ENDCLASS.

```

- **Final Outcome**

This RAP-based carbon certification system enables automated credit score assignment, certificate generation, and expiry validation using CDS views, ABAP classes, and behavior definitions. It ensures clean-core architecture and integrates seamlessly with Fiori Elements for a user-friendly dashboard and real-time tracking of emission activities.

- **Conclusion**

The Carbon Certificate Management System, built using the ABAP RESTful Application Programming Model (RAP), delivers a scalable, maintainable, and cloud-ready solution for managing carbon emissions and credit certification processes. Leveraging CDS views, behavior definitions, and service binding with OData V4, the system ensures seamless backend integration and a clean-core architecture aligned with SAP's modern development principles.

Key functionalities such as emission tracking, credit score calculation, certificate generation, expiry validation, and duplicate prevention are efficiently handled through encapsulated ABAP logic and behavior implementation. The system also features a real-time summary dashboard using Fiori Elements, enabling organizations to monitor emission trends and take proactive sustainability actions.

With its modular design and enterprise-grade extensibility, this RAP-based solution offers a future-proof framework for environmental compliance and corporate responsibility initiatives. It meets industry needs while maintaining compatibility with SAP BTP and Fiori UX standards.