

System Design for Automated Benchmarking System, to link up with EPA's servers and help a utility to automate the data transactions. This system is vital for Utilities who want to participate in ABS and come up with a solution in a short timeframe. This document can form the backbone of any large/small ABS system.

The main components of ABS.

1. A Database consisting of Tables and sequences and running PLSQL (or any other Programming language).
2. Java classes to be deployed on the machine which will send the data and communicate with EPA's server.
3. Utilities own data ware house from where it will pick the billing information.
4. Register with EnergyStar as an ESP (Energy Service Provider) and formulate a strategy of what unique identifiers will be used in authenticate the online Portfolio Manager customer and at what level these unique custom level identifiers will be asked from the Customer. The less places it is asked the better for an ESP as it would reduce programming logic and code. These unique identifiers can be asked at the Customer Level, Building Level , Meter level and combination among them, again the less the simple it makes. Two identifiers including the Meter Unique Identifier and another one to authenticate a customer could be sufficient.

1 and # 3 are discussed below, for # 2 please take help from EnergyStar's ABS documentation and following it closely should be sufficient. The documentation/code available on EnergyStar's ABS website is thorough.

#4 The ESP should test it out on Portfolio Manager and decide on the Level's.

(Important Note the database design is for 2 customIDValues at each(Customer,Building and Meter) levels :CUSTOM...IDNAME1, . CUSTOM...IDVALUE1, CUSTOM...IDNAME2, CUSTOM...IDVALUE2, Though this complexity would not be needed in most cases. Most of the Utilities will ask one customid at the Customer Level and one at the Meter Level only OR just at the Meter Level, These customID's are just to authenticate a customer as an utility would somehow authenticate if the customer belongs to the utility and the utility is authorized to release data for this customer)

#1.

The Portfolio Manager/EnergyStar's customer (can be anybody within CONUS) who want to benchmark their energy use data and authorize their ESP to send their data Automatically.

Customer data goes into **PM_CUSTOMER** table. This data comprises of Customer's basic description and some info about the Individual/organization.

Customers will have many or single buildings so we need **PM_BUILDINGS** table storing the Building data.

Customer's Building/s will have Many Spaces so we need **PM_BUILDINGS_SPACE** table which will store Space Type (Office/Parking/Swimming pool etc...)

Customer's Building Spaces will have many attributes which are like number or hours the space is open, floor are covered and heated, number or personnel on the main shift etc...so we need **PM_BUILDINGS_SPACE_ATTRIBS** table

Buildings will have Meter/s associated with the Spaces in the buildings so we need **PM_BUILDINGS_METERS** table which will store the Last Meter End Date based on which the system will know that it has to send information (Energy usage and Cost data) for the next cycle whose Star Date is greater than this EndDate. Apart from other meter related information.

These tables all will have Staging/Temp Tables which will hold data before it is filtered to ESP's main ABS Benchmarking Tables listed above. The staging tables have _STG as a Suffix in this design.

The Strategy is:

(Important Note: Benchmarking System should not be a part of existing ESP's systems. It should not reside in Billing system or any such system. It will be a standalone system.)

- a. Please pick a system/machine where this ABS Database will reside it can be Windows machine / UNIX system or any machine which can hold database tables and allow the personnel to install/run a XML parser on files which comes from EPA (step b. below)
- b. See if you can communicate from this system to EPA's servers via the java classes they provide.
- c. Once this is establish, make sure that the person who will run /design this system does have the required grants / permissions to access billing data ware house tables and also select ,insert, delete, update the ABS benchmarking tables. Create sequences, views and execute PLSQL etc. from the machine. Basically an ESP DBA will grant most of these privileges.
- d. The other main steps are as follows.
 - Ping the EPA server for Pending Authorization, PENDING_AUTH_REQUEST, basically just running the Java pendAuth.bat file.
 - i. Place the obtained XML in PENDING_AUTH_RESPONSE_FOLDER(create these folders on the machine).Read this file by XML parser and pick the data which is needed and put the data in _STG tables, make sure the main Primary and Foreign keys to the tables are copied into all the staging tables (They are self explanatory by looking at the database tables, I have made them as simple to read as possible). Example PMCUSTID will be stored in 4 staging tables mentioned below. PMBLDGID will be stored in building table, space table and meter table.
 - ii. Once this file is parsed and data is in the staging tables, move this file to PENDING_AUTH_RESPONSE_HISTORY_BACKUP_FOLDER.
 - Run Programming logic to get data from the Staging tables into the Main tables. Here is what makes it simple if the ESP uses less complex level of identifiers. Create a CONFIRM_AUTH_REQUEST. XML (This can be achieved by PLSQL creating XML data and TAGS) and place it into CONFIRM_AUTH_REQUEST_FOLDER.

- i. Send this CONFIRM_AUTH_REQUEST.XML by confirmAuth.bat java file and after sending this file move it to CONFIRM_AUTH_REQUEST_HISTORY_BACKUP_FOLDER
 - ii. CONFIRM_AUTH_RESPONSE.XML is received as soon as a CONFIRM_AUTH_REQUEST.XML is sent, Place this RESPONSE file in CONFIRM_AUTH_RESPONSE_FOLDER.
- Run XML parser on the CONFIRM_AUTH_RESPONSE.XML file and extract data in the _STG tables.
 - Run Programming logic and get the data from the _STG tables and place it in the MAIN Tables (note at this step the mapping indicators come back so you might just want to update the main tables with the data.) Set flags in PM_BUILDINGS and PM_BUILDINGS_METERS tables indicating that the METER is ready to be UPLOADED. Send the meter information to the next step that prepares data for that particular meter id. I mean the ESP's billing unit will be familiar with this identifier and the billing system will know what to look for (Discussed at the end of the document)
 - Run the Programming logic which will interact with the billing data warehouse tables and get the data for the unique meter identifiers. (use example ABS_BENCHMARKING_BILLING_VIEW mentioned below)
 - Prepare a MANAGE_METER_REQUEST.XML from the data in the previous step and place this file in MANAGE_METER_REQUEST_FOLDER.
 - Send this file by manageMeter.bat file, IF you receive a Transaction ID back then the transfer is successful else try it again, or reset the indicators so that you can run the creation of the file again and try to send it again.
 - Move the MANAGE_METER_REQUEST.XML from MANAGER_METER_REQUEST folder to MANAGE_METER_REQUEST_HISTORY_BACKUP_FOLDER after the file is sent.
 - Build BUILDING_DETAILS_REQUEST.XML by programming logic and place this file in BUILDING_DETAILS_REQUEST_FOLDER This file has just the ESPBLDGID's tags which will tell the EPA's server that it has to send back details for the ESPBLDGIDs in this file.
 - Send this file by using buildingDetails.bat java file
 - A Transaction ID comes back, store it in the PM_TRANSACTION_ID Table.
 - Send this Transaction ID using getTxnStatus.bat java file
 - A soon as you send this file a BUILDING_DETAILS_RESPONSE.XML comes back
 - Place this into BUILDING_DETAIL_RESPONSE_FOLDER
 - Run XML parser on this file and place data into _STG tables, move the file to BUILDING_RESPONSE_HISTORY_BACKUP_FOLDER
 - Run Programming logic to obtain data from the staging tables and place it into the main tables, (Note at this step you will get a rating back along with other calculated scores).

These should cover the ABS system vital steps, you have successfully sent a request and worked on it and send the data back and received a rating back.

(Programming logic is straight forward. Here are some details. You create cursors for customer, building, space and attributes' tables, call the customer cursor first and loop, call the building cursor and loop, call the meter cursor and loop, call the space cursor and loop, call the attribute cursor and loop, close the loops in the same order by having customer loop at last). For each kind of XML file generate d or validated set indicators and use these to control the overall logic. Like when you get and store the data via XML parser into the staging tables set a flag like PROCESSED = 'N' so that the next step when you want to bring this data into the main tables you check for this flag and after you have move data into the main tables you can reset the flag or delete the row. If CUSTOMLEVEL identifiers are used pass them to the level on which it is desired to validate: example if CUSTCUSTOMIDVALUE is asked at the customer level then just pass it to the customer table. If the same id is asked for each building and at the customer level then passes it to the customer table cursor as well as to the building table cursor and so on. The custom ID identifiers are **ONLY** needed if you (ESP / MUNI etc) is going to ask them and "at what level" to Authenticate if the customer is genuine, else **ONLY** one METERCUSTOMIDVALUE is needed and you ask it every time the customer enters new meter in the portfolio manager. In case you are doing bulk upload strategy (Template upload) and going to sign this customer for ABS then you/customer would have to pick the "meter ID" from the meter page in Portfolio manager and copy it into the METERCUSTOMIDVALUE field when the Automated Benchmarking Meter Custom page (the one which the ESP requires open up). If it is an automated approach remember the EPA's server will only send the METERCUSTOMIDVALUE back to the ESP and there should be a value in this field if the ESP wants to Automate in the simplest fashion)

PM_CUSTOMER

Name	Null?	Type
------	-------	------

ESPCUSTID		VARCHAR2(80) // this will be same as pmcustid or some sequence if you want to enter data from your own system, I mean in case you have customers joining Benchmarking first coming to ESP instead of PM, so you will need to create and store a sequence ID for the customer, in our discussed approach the customer goes to Portfolio manager first so ESPID can remain same and you just store the same in your system as it comes along).
PMCUSTOMID		VARCHAR2(80)
CUSTOMER_DESCRIPTION		VARCHAR2(255)
CUSTCUSTOMIDVALUE1		VARCHAR2(80)
CUSTCUSTOMIDVALUE2		VARCHAR2(80)

CUSTMAPPEDYN	CHAR(1)
ENTRY_DATE	DATE
UPDATE_DATE	DATE
ALERT_CODE	VARCHAR2(10)
ALERT_STRING	VARCHAR2(60)
RELEASE_CUSTOMER_DATA	CHAR(1)
CUSTCUSTOMIDNAME1	VARCHAR2(60)
CUSTCUSTOMIDNAME2	VARCHAR2(60)

PM_BUILDINGS

Name	Null?	Type
PMCUSTID		VARCHAR2(80)
ESPCUSTID		VARCHAR2(80)
PMBLDGID		VARCHAR2(80)
ESPBLDGID		VARCHAR2(80)
BLDG_NAME		VARCHAR2(80)
BLDG_ADDRESS		VARCHAR2(64)
BLDG_CITY		VARCHAR2(30)
BLDG_STATE		VARCHAR2(6)
BLDG_ZIP_CODE		VARCHAR2(9)
RELEASE_IMMEDIATE		CHAR(1)
LAST_EXPORT_DATE		DATE
YEAR_BUILT		NUMBER(4)
BLDG_DELETE_INDICATOR		CHAR(1)
RATING_END_MONTH		NUMBER(2)
RATING_END_YEAR		NUMBER(4)
ENTRY_DATE		DATE
UPDATE_DATE		DATE
ALERT_CODE		VARCHAR2(10)
ALERT_STRING		VARCHAR2(255)
RATING		NUMBER(3)
BLDGMAPPEDYN		CHAR(1)
BLDGCUSTOMIDVALUE1		VARCHAR2(80)
BLDGCUSTOMIDVALUE2		VARCHAR2(80)
SITEKBTUTOTALWN		NUMBER(9,2)
SITEKBTUTOTALNWN		NUMBER(9,2)
SITEKBTUSFWN		NUMBER(9,2)
SITEKBTUSFNWN		NUMBER(9,2)
SOURCEKBTUTOTALWN		NUMBER(9,2)
SOURCEKBTUTOTALNWN		NUMBER(9,2)
SOURCEKBTUSFWN		NUMBER(9,2)

SOURCEKBTUSFNWN	NUMBER(9,2)
BLDGGROSSFLOORAREA	NUMBER(10)
BLDGCUSTOMIDNAME1	VARCHAR2(60)
BLDGCUSTOMIDNAME2	VARCHAR2(60)

PM_BUILDINGS_SPACE

Name	Null?	Type
PMCUSTID		VARCHAR2(80)
ESPCUSTID		VARCHAR2(80)
PMBLDGID		VARCHAR2(80)
ESPBLDGID		VARCHAR2(80)
PMSPACEID		VARCHAR2(80)
ESPSPACEID		VARCHAR2(80)
SPACE_NAME		VARCHAR2(80)
SPACE_TYPE		VARCHAR2(40)
ENTRY_DATE		DATE
UPDATE_DATE		DATE
ALTER_CODE		VARCHAR2(25)
ALERT_STRING		VARCHAR2(255)
ATTRIB		VARCHAR2(60)
ATTRIB_EFFECTIVE_DATE		DATE
RELEASE_IMMEDIATE		CHAR(1)

PM_BUILDINGS_SPACE_ATTRIBS

Name	Null?	Type
PMCUSTID		VARCHAR2(80)
ESPCUSTID		VARCHAR2(80)
PMBLDGID		VARCHAR2(80)
ESPBLDGID		VARCHAR2(80)
PMSPACEID		VARCHAR2(80)
ESPSPACEID		VARCHAR2(80)
ATTRIB		VARCHAR2(60)
ATTRIB_VALUE		VARCHAR2(60)
ATTRIB_UC_IND		CHAR(1)
ATTRIB_EFF_DATE		DATE

Staging tables

PM_CUSTOMER_STG

Name	Null?	Type
------	-------	------

ESPCUSTID	VARCHAR2(80)
PMCUSTID	VARCHAR2(80)
CUSTOMER_DESCRIPTION	VARCHAR2(255)
CUSTCUSTOMIDVALUE1	VARCHAR2(80)
CUSTCUSTOMIDVALUE2	VARCHAR2(80)
CUSTMAPPEDYN	CHAR(1)
ENTRY_DATE	DATE
UPDATE_DATE	DATE
ALERT_CODE	VARCHAR2(10)
ALERT_STRING	VARCHAR2(60)
RELEASE_CUSTOMER_DATA	CHAR(1)
CUSTCUSTOMIDNAME1	VARCHAR2(60)
CUSTCUSTOMIDNAME2	VARCHAR2(60)
PROCESSED	CHAR(1)

PM_BUILDINGS_STG

Name	Null?	Type
PMCUSTID		VARCHAR2(80)
ESPCUSTID		VARCHAR2(80)
PMBLDGID		VARCHAR2(80)
ESPBLDGID		VARCHAR2(80)
BLDG_NAME		VARCHAR2(80)
BLDG_ADDRESS		VARCHAR2(64)
BLDG_CITY		VARCHAR2(30)
BLDG_STATE		VARCHAR2(6)
BLDG_ZIP_CODE		VARCHAR2(9)
RELEASE_IMMEDIATE		CHAR(1)
LAST_EXPORT_DATE		DATE
YEAR_BUILT		NUMBER(4)
BLDG_DELETE_INDICATOR		CHAR(1)
RATING_END_MONTH		NUMBER(2)
RATING_END_YEAR		NUMBER(4)
ENTRY_DATE		DATE
UPDATE_DATE		DATE
ALERT_CODE		VARCHAR2(10)
ALERT_STRING		VARCHAR2(255)
RATING		NUMBER(3)
BLDGMAPPEDYN		CHAR(1)
BLDGCUSTOMIDVALUE1		VARCHAR2(80)
BLDGCUSTOMIDVALUE2		VARCHAR2(80)

SITEKBTUTOTALWN	NUMBER(9,2)
SITEKBTUTOTALNWN	NUMBER(9,2)
SITEKBTUSFWN	NUMBER(9,2)
SITEKBTUSFNWN	NUMBER(9,2)
SOURCEKBTUTOTALWN	NUMBER(9,2)
SOURCEKBTUTOTALNWN	NUMBER(9,2)
SOURCEKBTUSFWN	NUMBER(9,2)
SOURCEKBTUSFNWN	NUMBER(9,2)
BLDGGROSSFLOORAREA	NUMBER(10)
BLDGCUSTOMIDNAME1	VARCHAR2(60)
BLDGCUSTOMIDNAME2	VARCHAR2(60)
PROCESSED	CHAR(1)

PM_BUILDINGS_SPACE_STG

Name	Null?	Type
PMCUSTID		VARCHAR2(80)
ESPCUSTID		VARCHAR2(80)
PMBLDGID		VARCHAR2(80)
ESPBLDGID		VARCHAR2(80)
PMSPACEID		VARCHAR2(80)
ESPSPACEID		VARCHAR2(80)
SPACE_NAME		VARCHAR2(80)
SPACE_TYPE		VARCHAR2(40)
ENTRY_DATE		DATE
UPDATE_DATE		DATE
ALTER_CODE		VARCHAR2(25)
ALERT_STRING		VARCHAR2(255)
ATTRIB		VARCHAR2(60)
ATTRIB_EFFECTIVE_DATE		DATE
RELEASE_IMMEDIATE		CHAR(1)
PROCESSED		CHAR(1)

PM_BUILDINGS_SPACE_ATTRIBS_STG

Name	Null?	Type
PMCUSTID		VARCHAR2(80)
ESPCUSTID		VARCHAR2(80)
PMBLDGID		VARCHAR2(80)

ESPBLDGID	VARCHAR2(80)
PMSPACEID	VARCHAR2(80)
ESPSPACEID	VARCHAR2(80)
ATTRIB	VARCHAR2(60)
ATTRIB_VALUE	VARCHAR2(60)
ATTRIB_UC_IND	CHAR(1)
ATTRIB_EFF_DATE	DATE
PROCESSED	CHAR(1)

PM_BUILDINGS_METERS

Name	Null?	Type
PMCUSTID		VARCHAR2(80)
ESPCUSTID		VARCHAR2(80)
PMBLDGID		VARCHAR2(80)
ESPBLDGID		VARCHAR2(80)
PMMETERID		VARCHAR2(9)
ESPMETERID		VARCHAR2(30)
DELETE_INDICATOR		CHAR(1)
ENTRY_DATE		DATE
UPDATE_DATE		DATE
METER_ID_VALIDATED		CHAR(1)
METER_ID_AUTHORIZED		CHAR(1)
START_DATE		DATE
END_DATE		DATE
CUST_RES_NONRES		CHAR(1)
METERMAPPEDYN		CHAR(1)
ALERT_CODE		VARCHAR2(20)
ALERT_STRING		VARCHAR2(255)
METERCUSTOMIDVALUE1		VARCHAR2(80)
METERCUSTOMIDVALUE2		VARCHAR2(80)
RELEASE_IMMEDIATE		CHAR(1)
METERCUSTOMIDNAME1		VARCHAR2(30)
METERCUSTOMIDNAME2		VARCHAR2(30)
METERACTIVEINDICATOR		CHAR(1)
METERADDTOTOTAL		CHAR(1)
METERNAME		VARCHAR2(100)

PM_BUILDINGS_METERS_STG

Name	Null?	Type
PMCUSTID		VARCHAR2(80)
ESPCUSTID		VARCHAR2(80)

(write the relationship among tables if your data comes from many tables)

Write flags here (example METER_STATUS = 'Y' etc.) AND

YYYYMM >= 200301

GROUP By meter_id, yyyyymm, PRIOR_READ_DATE, CURRENT_READ_DATE, electricales, gassales, electricrevenue, gasrevenue, residential_or_nonres;

(Note use UNION ALL and join with this view if your billing data is spread on various tables), example

If data for suppose 2005 and beyond is in table X and the rest is in table y then

It will be something like

VIEW 1

For date <= 2005

UNION ALL

VIEW2

For date >= 2005

The strategy not discussed is DEAUTHORIZATION of a Customer etc., which can be achieved once the main objective is realized.