Service Name: MeterReadingMainframePortBinding

Request:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:read="http://reading.metering.services.canonical.gridsmart.aep.com/" xmlns:prov="http://www.aep.com/gridsmart/canonical/buslogic/messages/metering/reading/provision" xmlns:mes="http://www.iec.ch/TC57/2008/schema/message" xmlns:get="http://iec.ch/TC57/2007/GetMeterReadings#">

<soapenv:Header/>

<soapenv: Body>

<read:provisionCanonicalMeterOnDemandRead>

<!--Optional:-->

<PCMeterOnDemandRequest QOS="false" correlator="correlator" definitive="false" onBehalfOf="david pang" origin="MACSS" sequence="1" version="1">

<RequestMessage>

<prov:Header>

<mes:Verb>REQUEST</mes:Verb>

<mes:Noun>noun</mes:Noun>

<mes:User>

<mes:UserID>s195642</mes:UserID>

<mes:Organization>AEP</mes:Organization>

</mes:User>

<mes:MessageID/>

<mes:CorrelationID/>

<mes:MessagePriority>good</mes:MessagePriority>

<mes:Comment>this is a comment</mes:Comment>

<mes:Source>MACSS</mes:Source>

</prov:Header>

<prov:Payload>

<prov:GetMeterReadings>

<get:EndDeviceAsset>

<get:mRID>774372852</get:mRID>

<get:Identifier>

<get:authority>MACSS</get:authority>

<get:content>774372852</get:content>

<get:kind>METERSERIALNUMBER</get:kind>

</get:Identifier>

</get:EndDeviceAsset>

</prov:GetMeterReadings>

</prov:Payload>

</RequestMessage>

</PCMeterOnDemandRequest>

</read:provisionCanonicalMeterOnDemandRead>

</soapenv:Body>

</soapenv:Envelope>

URL for the above SOAP request is: <http://amiws-t.aepsc.com/gsCanonical-war/MeterReadingService>

When the service start executing it is pointing to the MeterReading class which is available in the package

**com.aep.gridsmart.canonical.services.metering.reading;**

**MeterReading.java**

This class contains the CanonicalTransformSessionLocal interface object named it as xformBean and it is mentioned with the annotation @[EJB](#EJB).

And it contains the two methods. They are

* determineOpSubType(String origin).
* provisionCanonicalMeterOnDemandRead.

These are the methods and variables are present in the class MeterReading.

**Method: provisionCanonicalMeterOnDemandRead**

@WebMethod(operationName = "provisionCanonicalMeterOnDemandRead")

public @[WebResult](#Result)(name = "PCMeterOnDemandResp")

PCMeterOnDemandReadResp provisionCanonicalMeterOnDemandRead(

@WebParam(name = "PCMeterOnDemandRequest") PCMeterOnDemandReadRequest request)

The return type for this method is PCMeterOnDemandResponse. The retrieved information will be stored in the object of PCMeterOnDemandResponse.

**Steps:**

* It creates the object for PCMeterOnDemandResp using parametrized constructor and passing the PCMeterOnDemandRequest object request as the parameter.
* In that constructor it sets the Header class variables of verb as **REQUEST** to **REPLY** and noun as **NOUN** to **GET METER READINGS** and current time with milliseconds and the reply message. Then it gets the information like Correlator, onBehalfOf, origin, QOS, sequence, version and definitive. This data will store in the super class of PCMeterOnDemandResp.
* Then it checks for the Correlation Id is coming from the request object or not. If it is not getting any information about correlation id, then it will create one random value for the correlation id that value will be store in correlationId.
* Then it takes the information for reply address and stores the information in replyAddress and if that variable has the information and the length is more than 100 then it will throw an IllegalArgumentException.
* After that, it calls the method determineOpSubType and passing the origin as a parameter in that method. If the origin is **MACSS** it will return the subtype as **MR** or else if the origin is **SMT** it will return **DB.**
* Then it calls the method processJAXBMessage which is implemented in the CanonicalTransformSessionLocal class. The method is called by the object xformBean and the information coming from the method it will be store in the object of ReplyContainer and passing the parameters as JaxbObject, opType, opSubType, correlator, origin, replyAddress.
* The interface **CanonicalTransformSessionLocal** is the annotated with @[**Local**](#Local) so it is the Enterprise Java Bean(**EJB**) class the method which they’ve called it goes to override method in @**Stateless** class. The @**Stateless** class is **CanonicalTransformSession** for @**Local** interface. In this class it contains the method JaxBMessage is implemented. So, whenever the JaxBMessage method calls in any service it is directly refer to the CanonicalTransformSession class but that method uses the reference of CanonicalTransformSessionLocal class object.
* Finally, it sets the information from the ReplyContainer object to PCMeterOnDemandResp object response. The response object is passing the information to the response side of MeterReadingMainframePortBinding in SOAP UI Web service.
* If any exception raises in the application, it will set reply code as 5.2 and sets the error as Internal System Error - Exception Caught in Web Service Method.

**Method:determineOpSubType**

**Steps:**

* This method is calling from the above method to append the sub type for the service. It is getting the origin information as parameter.
* If the origin is **SMT** it will return the subtype as **DB.**
* Or else if the origin is **MACSS** it will return the subtype as **MR.**
* This is the functionality of the determineOpSubType method. If the origin is **NULL** it will return the subtype as **EMPTY** string.

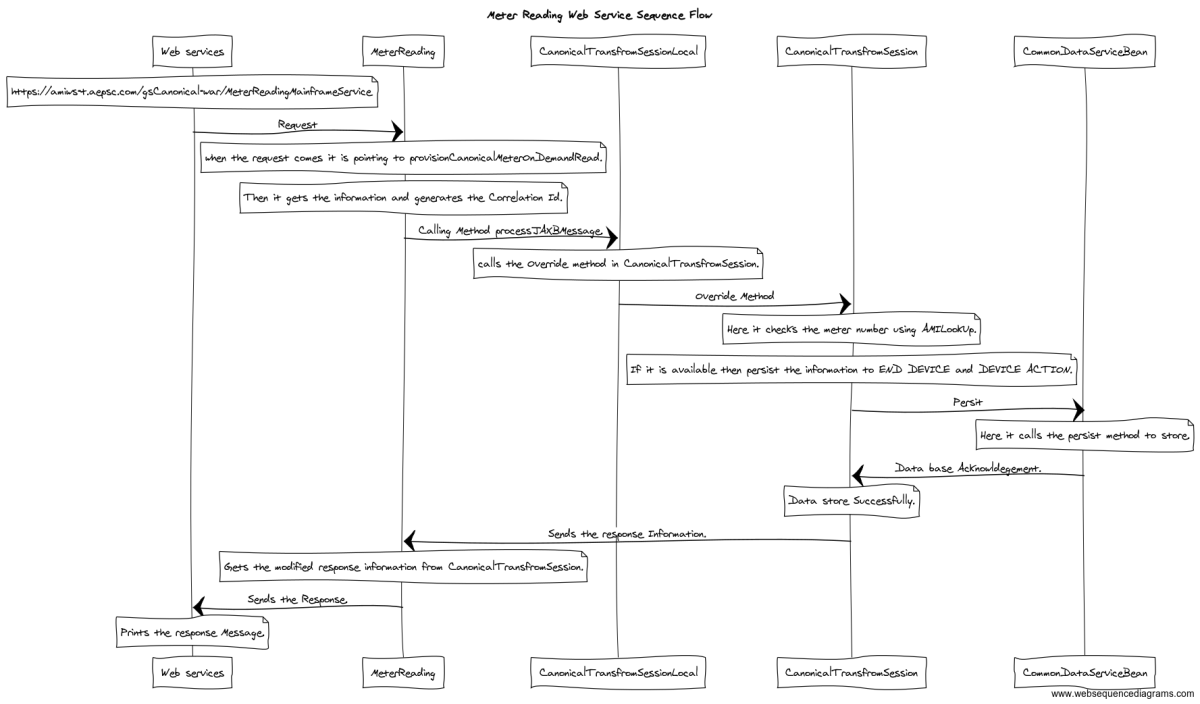
**Conclusion:**

For all non-main frame services, the code flow is same. It repeats the same procedure for all the normal services. The difference in the services are OpTypes and OpSubTypes. For Example, MeterReading – MR, MeterStatus – MS etc. The Procedure is same for all other OPCO’s. The difference is while looking for meter serial number using AMILookUp method. In that method by meter serial number it identifies the OPCO, System ID and System Number.

**Note:**

* @**Local**: This annotation is applies only to session beans and their interfaces. When used on an interface, designates that interface as a local business interface.
* @**EJB**: It indicates the dependency on local or remote view of an Enterprise Java Bean.
* Enterprise Java Bean(EJB): EJB is the server side software component, it encapsulates the business logic of an application.
* @**Stateless**: This annotation used in the class CanonicalTransformSession. EJB container automatically creates the relevant configurations or interfaces required by reading this annotation during deployment.
* @**WebResult**: It customizes the mapping of the return value to a WSDL part and XML element.
* @WebParam: It indicates the object is passing as the parameter to a WSDL part and XML element.
* @**WebMethod**: It indicates the whenever the SOAP request raises it is pointing to this method.
* An **ObjectFactory** allows you to programmatically construct the new instances of the java representation for XML content. The java representation of XML content can consist of Schema derived interfaces and classes representing the binding of schema type definitions, element declarations and model groups.

**Sequence Diagram:**

****