Smart Meter Texas In-Home Device Application Programming Interface (API)

Table of Contents

Figures

Tables

1. Introduction

This document covers the current release of the Smart Meter Texas (SMT) Application Programming Interfaces (APIs) to support In-Home Devices. This API enables:

- Provisioning In-Home Devices
- De-provisioning In-Home Devices
- Send Zigbee Smart Energy Profile 1.0 (SEP 1.0 or SEP) messages to In-Home Devices

Third-Party Service Providers (Third-Party) are the primary users of the In-Home Device messaging functionality. Third-Parties will require an active In-Home Device Agreement in order to provision, de-provision an In-Home Device with a customer’s Smart Meter, and an active In-Home Device Services Agreement to send SEP message to a customer’s In-Home Devices.

The interfaces described in this document have been defined based on Service Oriented Architecture (SOA). The interfaces have been specified in the Web Service Description Language (WSDL). Web Services between Third-Partand SMT will be secured using mutual authentication implemented with Secure Socket Layer (SSL).

2. Related Information

2.1 Glossary of Terms

Term	Definition
API	Application Programming Interface that allows one program to talk to another
Current Release	Rel 2 Summer Release of SMT Portal
DUNS	Number assigned to a business entity by Dunn and Bradstreet. REPS and Third-Party Service Providers can establish multiple businesses within their parent company using multiple
ESB	Enterprise Server Bus
ESIID	Electric Service Identifier, a unique identifier for the point of delivery
HAN	Home Area Network - – this is the network between the advanced meter and the home device as mentioned in the AMS rules
REP	Retail Electric Provider
ROR	REP of Record
SMT	Smart Meter Texas
SOA	Service Oriented Architecture
SSL	Secure Socket Layer is a set of cryptographic protocols that provide security and data integrity for communications over networks such as the Internet
SEP	Smart Energy Profile
TDSP	Transmission and Distribution Service Provider
UEG	Update Utility Enrollment Group
WSDL	Web Service Description Language - XML-based language for describing Web services
XML	Extensible Markup Language
ZigBee	Specification of a suite of high level communication protocols using small low-power radios

Table 1: Glossary of Terms

2.2 Relevant Documents

Reference Description / Name	Document Number / Location / Links
Zigbee SEP 1.0 Document	http://zigbee.org/Markets/ZigBeeSmartEnergy/PublicApplicationProfile.aspx

Table 2: Relevant Documents

2.3 Soap Fault Codes

Soap Fault Code	Code Description
2100	Invalid Message Signature failure
2200	SMT System Error
2300	Authentication Failure Invalid Credentials
2400	SMT Internal System Error
2500	SMT internal Fault
2600	SOAP Fault Message Rejected by SMT

Table 3: Soap Fault Code Description

3. In-Home Device System Overview

Figure 1 presents the system context diagram for SMT with In-Home Devices functionality highlighted.

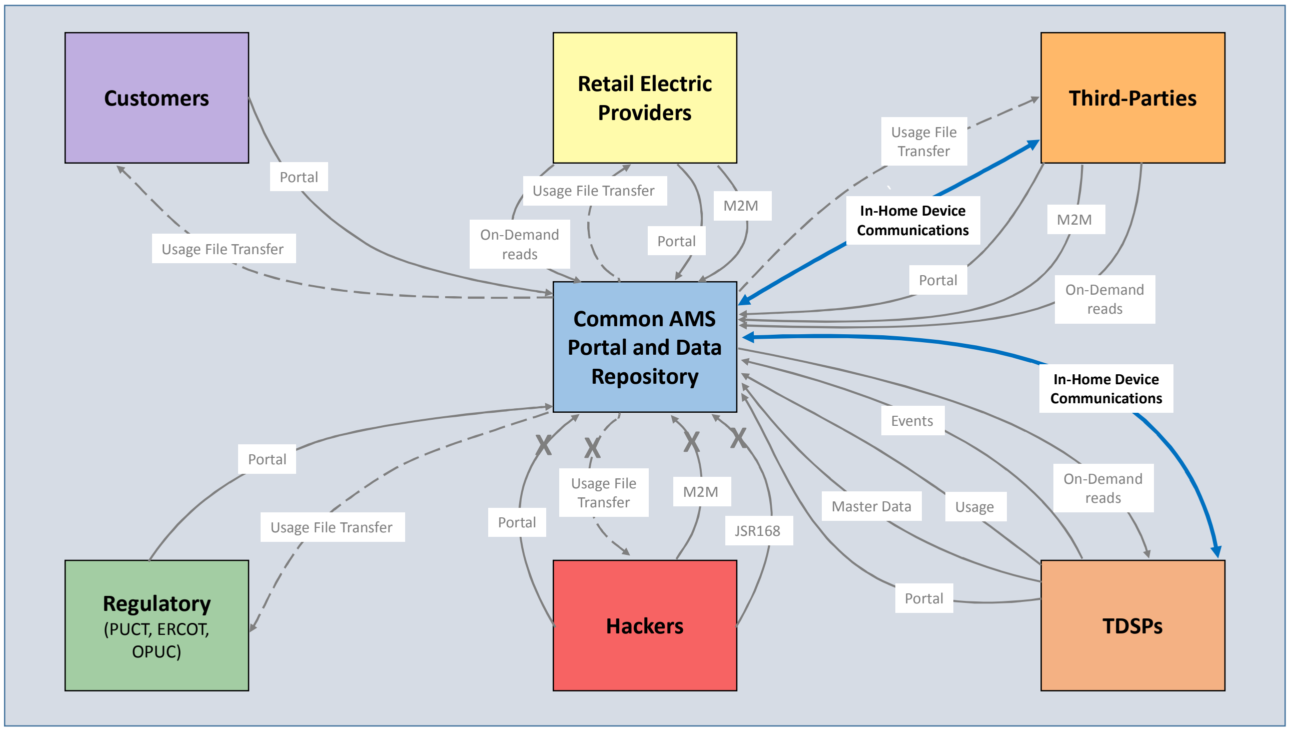


Figure 1: System context diagram with Current release In-Home Device communications

SMT provides In-Home Device provisioning, de-provisioning and Smart Energy messaging functions via Application Programming Interfaces. SMT will perform validations of the requests and act as a pass-through to TDSP systems that implement In-Home Device functionality.

These interfaces (as well as front-end interfaces) may be used by the following types of requesters as of Release 4.3 of the project:

- Third-Parties
- REPs acting as a Third-Party

3.1 Interface Overview

Table 4 shows the set of In-Home Device interface requests, acknowledgements, status updates and responses. An interface number is associated with each type of message.

Interface Message Description	Response	Invocation Type
Provisioning HAN Device Request	Provision Request Acknowledgement	Synchronous
Deprovisioning HAN Device Request	Provision Request Acknowledgement	Synchronous
Simple Messaging Request	Messaging Request Acknowledgement	Synchronous
Cancel Simple Message Request	Messaging Request Acknowledgement	Synchronous
Load Control Messaging Request	Messaging Request Acknowledgement	Synchronous
Cancel Load Control Message Request	Messaging Request Acknowledgement	Synchronous

Interface Message Description	Response	Invocation Type
Cancel All Load Control Message Request	Messaging Request Acknowledgement	Synchronous
Price Signal Request	Messaging Request Acknowledgement	Synchronous
Update Enrollment Group Request	Messaging Request Acknowledgement	Synchronous

Table 4: SMT In-Home Device Interfaces

These interfaces (as well as front-end interfaces) may be used by Third-Party Service Providers as presented in Figure 2 below.

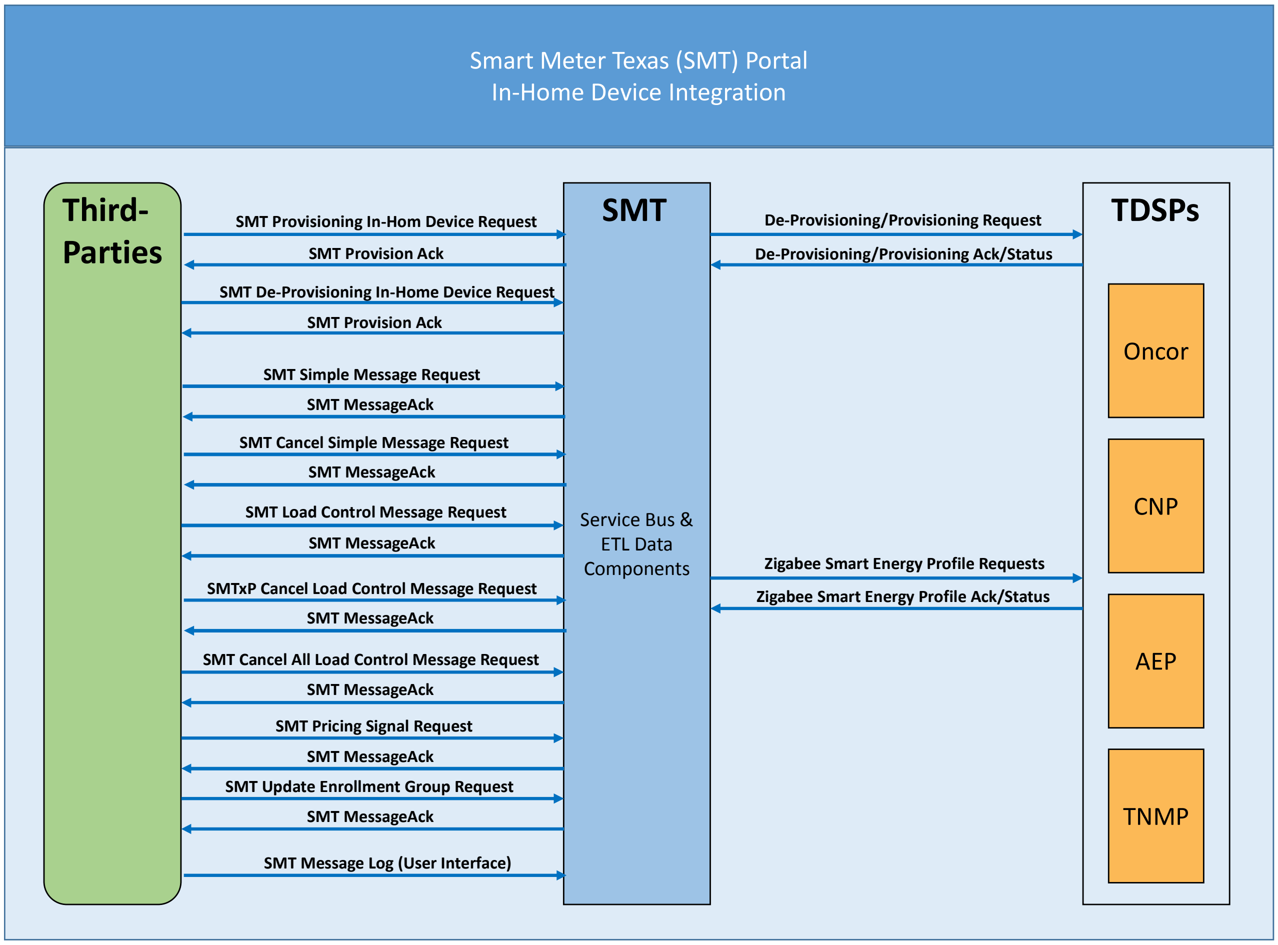

Smart Meter Texas (SMT) Portal
In-Home Device Integration
Third-Parties
SMT
TDSPs
SMT Provisioning In-Hom Device Request
SMT Provision Ack
SMT De-Provisioning In-Home Device Request
SMT Provision Ack
SMT Simple Message Request
SMT MessageAck
SMT Cancel Simple Message Request
SMT MessageAck
SMT Load Control Message Request
SMT MessageAck
SMTxP Cancel Load Control Message Request
SMT MessageAck
SMT Cancel All Load Control Message Request
SMT MessageAck
SMT Pricing Signal Request
SMT MessageAck
SMT Update Enrollment Group Request
SMT MessageAck
SMT Message Log (User Interface)
Service Bus & ETL Data Components
De-Provisioning/Provisioning Request
De-Provisioning/Provisioning Ack/Status
Zigabee Smart Energy Profile Requests
Zigabee Smart Energy Profile Ack/Status
Oncor
CNP
AEP
TNMP

Figure 2: System interface view of all Current release interfaces

3.2 Interface Security

Requesters who communicate with SMT using the In-Home Device web service will be on 2-way SSL, the interface must support mutual authentication over SSL. To support authentication, SMT requires user credentials to be passed in Username Token that is part of SOAP header. Token data is used to map the sender of a request to a system account.

When a Username token is sent, the User Name element identifies the system account. The SMT security infrastructure will validate the request sender by verifying the WS-Security signature using the signer certificate from the SMT certificate store. It will also validate that the system account is authorized to access data associated with the DUNs number provided in the Requester Type element of the SMT Header. If the digital signature and DUNs number pass validations and the user can be authenticated, the web service request is passed to the ESB. Otherwise, a SOAP fault is issued.

Tokens are placed into the soap Header. A sample soap Header containing a User token appears below:

```
<soapenv:Header>
  <wsse:Security xmlns:wsse=
  "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
       <wsse:UsernameToken>
    <wsse:Username>Entity_Sytem_Account</wsse:Username>
       </wsse:UsernameToken>
  </wsse:Security>
</soapenv:Header>
```

Note that examples found in subsequent sections will use an empty soap Header in lieu of this. Instructions for the setup of accounts and security credentials will be made available through the SMT help desk. The SMT help desk can be reached on 1- 888-616-5859.

3.3 Assumption of Central Time

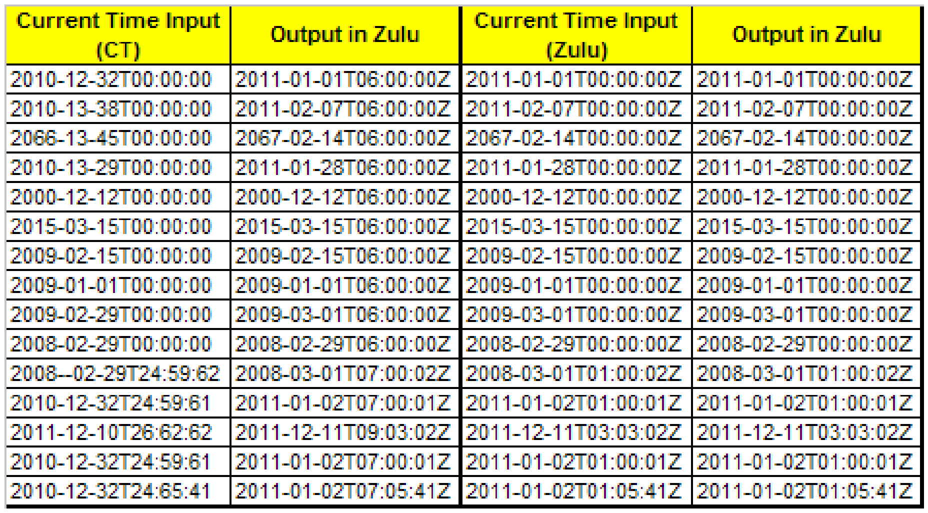
Requesters have the ability to specify the current time and/or start times for display of messages, load control events and price signals. SMT assumes that all time values input through the API are in Central Standard or Central Daylight Time and converts it to UTC (Coordinated Universal Time).

An example for how Central time values should be formatted appears below:

```
<StartTime>2009-12-01T16:00:00</StartTime>
```

UTC values should be formatted as follows:

```
<StartTime>2009-12-01T16:00:00Z</StartTime>
```



Current Time Input (CT)	Output in Zulu	Current Time Input (Zulu)	Output in Zulu
2010-12-32T00:00:00	2011-01-01T06:00:00Z	2011-01-01T00:00:00Z	2011-01-01T00:00:00Z
2010-13-38T00:00:00	2011-02-07T06:00:00Z	2011-02-07T00:00:00Z	2011-02-07T00:00:00Z
2066-13-45T00:00:00	2067-02-14T06:00:00Z	2067-02-14T00:00:00Z	2067-02-14T00:00:00Z
2010-13-29T00:00:00	2011-01-28T06:00:00Z	2011-01-28T00:00:00Z	2011-01-28T00:00:00Z
2000-12-12T00:00:00	2000-12-12T06:00:00Z	2000-12-12T00:00:00Z	2000-12-12T00:00:00Z
2015-03-15T00:00:00	2015-03-15T06:00:00Z	2015-03-15T00:00:00Z	2015-03-15T00:00:00Z
2009-02-15T00:00:00	2009-02-15T06:00:00Z	2009-02-15T00:00:00Z	2009-02-15T00:00:00Z
2009-01-01T00:00:00	2009-01-01T06:00:00Z	2009-01-01T00:00:00Z	2009-01-01T00:00:00Z
2009-02-29T00:00:00	2009-03-01T06:00:00Z	2009-03-01T00:00:00Z	2009-03-01T00:00:00Z
2008-02-29T00:00:00	2008-02-29T06:00:00Z	2008-02-29T00:00:00Z	2008-02-29T00:00:00Z
2008--02-29T24:59:62	2008-03-01T07:00:02Z	2008-03-01T01:00:02Z	2008-03-01T01:00:02Z
2010-12-32T24:59:61	2011-01-02T07:00:01Z	2011-01-02T01:00:01Z	2011-01-02T01:00:01Z
2011-12-10T26:62:62	2011-12-11T09:03:02Z	2011-12-11T03:03:02Z	2011-12-11T03:03:02Z
2010-12-32T24:59:61	2011-01-02T07:00:01Z	2011-01-02T01:00:01Z	2011-01-02T01:00:01Z
2010-12-32T24:65:41	2011-01-02T07:05:41Z	2011-01-02T01:05:41Z	2011-01-02T01:05:41Z

Figure 3: Example of the Output when Current tem is entered as CT or UTC

3.4 Meter Serial Numbers

All SMT interfaces require a MeterSerialNumber element. SMT will accept the manufacturer's serial number or a fully-qualified TDSP meter serial number in this element (if you are an authorized TDSP user), or all zeros (if you are a Third-Party Service Provider or REP acting as a Third-Party).

An example of a manufacturer's serial number appears below:

```
<MeterSerialNumber>6039657245</MeterSerialNumber>
```

Some TDSPs add a manufacturer code to the manufacturer's serial number to guarantee uniqueness. For example, CenterPoint places the manufacturer code before the serial number.

```
<MeterSerialNumber>I6039657245</MeterSerialNumber>
```

Oncor appends the manufacturer code to the end of the serial number.

```
<MeterSerialNumber>6039657245LG</MeterSerialNumber>
```

SMT can accept either format and validates the value in combination with the ESIID value provided.

3.5 Request Priorities

Requesters have the ability to specify a request priority as part of a SMT Header and within display of messages, load control events and price signals. These elements have been defined with the goal of minimizing future changes. However, prioritization is not implemented in the current release.

3.6 Handling Optional Integers

Optional integers are specified for the following interface elements:

- Provisioning request – DeviceClusterSupport
- Load Control event – CoolingTemperationOffset
- Load Control event – HeatingTemperatureOffset
- Load Control event – CoolingTemperationSetPoint
- Load Control event – HeatingTemperatureSetPoint
- Load Control event – AverageLoadAjustPercent
- Load Control event – DutyCycle
- Price Signal – PriceRatio
- Price Signal – GenerationRatio
- Price Signal – AlternateCostUnit
- Price Signal – Generation Price

Requesters must remove the optional tags or must specify an integer value when submitting a request.

4. Common SMT-TDSP Interface Definitions

4.1 Schema Definitions

All provisioning and de-provisioning requests use the following schema definition:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://schemas.esb.ams.com/smtxpprovisiondevice">
```

All messaging requests use the following schema definition:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://schemas.esb.ams.com/smtxpmessaging">
```

4.2 SMT Request Header Information

SMT messages use a common request header element in the SOAP Body element. An example request header appears below.

```
<Header></Header>
<Body>
 <processProvisionDevice xmlns="http://schemas.esb.ams.com/smtxpprovisiondevice">
 <ProvisioningRequest xmlns="">
 <RequestID>123</RequestID>
 <RequesterType>0</RequesterType>
 <RequesterAuthenticationID>111111111</RequesterAuthenticationID>
 <RequesterID>Requester_ID</RequesterID>
 <RequestPriority>M</RequestPriority>
 <CallbackUri></CallbackUri>
 … <!-- Additional elements within the body appear below-->
</Body>
```

All requests sent to SMT will contain header information list in Table 5.

Element	Mandatory	Type	Description
RequestID	N	string(32)	Unique request identifier. Every request sent to SMT will be assigned a RequestID. Requesters may assign their own unique value for a Request ID as part of the request sent to SMT, however it is discouraged because the value is not guaranteed to be unique across all requesters. SMT always generates its own unique value and use it for communications with TDSP systems. The SMT-generated value will be returned to the requester in a request acknowledgement.

Element	Mandatory	Type	Description

RequesterType	Y	Int	Requester Type indicator. Accepted values are:
			0 REP;
			1 TDSP;
			2 Customer;
			3 Third-party;
			4 Host;
			5 Supplemental

RequesterAuthenticationID	Y	string(9,16)	Requesters and TDSPs should input their primary DUNS number
RequesterID	Y	string	System Account ID of the Requester
RequestPriority	Y	string(1)	Priority of request. Accepted values are: H – High M – Medium L – Low
CallbackUri	N	string(256)	(For future use as a callback mechanism with the requester.)

Table 5: SMT-TDSP interface header information

5. In-Home Device Provisioning and De-provisioning

This section describes the device slot management and discusses the interfaces used to Provision and De-provision In-Home Devices.

5.1 Slot Management

A meter may have up to 5 devices provisioned to it. Each device takes a slot. Meter slot management rules are as follows:

- A slot is allocated when a valid provisioning request is accepted by SMT.
- A slot is de-allocated if the TDSP returns a failure status associated with a provisioning request.
- A slot is de-allocated when the TDSP returns a completion status for a de-provisioning request.
- If a de-provisioning request fails, slot allocation is unchanged.

5.2 Provisioning In-Home Device

TDSP submits the device provisioning request through the front-end web services API (Requests can also be entered via the SMT user interface to re-add a device associated with an active In-Home Device Agreement only). SMT performs two levels of validations:

- XML validation is performed by an appliance on the SMT perimeter. If the request fails validation, a SOAP fault is returned. Refer to soap fault codes description in *Section 2.3.*
- If XML is formatted correctly, business validations are performed by SMT. SMT validates that the requester is authorized to provision devices to that ESIID, that the correct meter serial number was supplied, that a slot is available, and that device is not already provisioned or in a provisioning request pending status to the ESIID. If a provisioning request passes business validation checks, SMT assigns a unique identifier, saves the request, and sends an acknowledgement. The request status at this point will be Add Acknowledged[1]. If the request fails any of these checks, a failure is returned in a Request Acknowledgement in the form of request rejected.

When the request is sent, the TDSP will initially send an acknowledgment back to SMT - either a successful acknowledgement with the status of add acknowledged or a failure with a status of request failed. There can be two add acknowledged Status for a provisioned request but the acknowledgement will have different description one indicating that SMT received the request and the other indicating that the TDSP received the request. The TDSP may perform initial validation.

The TDSP may perform additional validations on the provisioning request as stated above. If failure occurs, the TDSP will return the failure status to SMT. Otherwise, the provisioning request is executed and the execution status is returned as Meter Ready. In some cases, the customer will have to take some type of action (such as pushing a button) in order to provision that In-Home Device. The manufacturer’s instructions should provide further details. Once the device has completed its joining process, the TDSP will return a successful status of device added or a failure with add failed.

[1] All provisioning and de-provisioning status are visible through the Portal UI.

Figure 4 shows the interface exchanges for provisioning an In-Home Device.

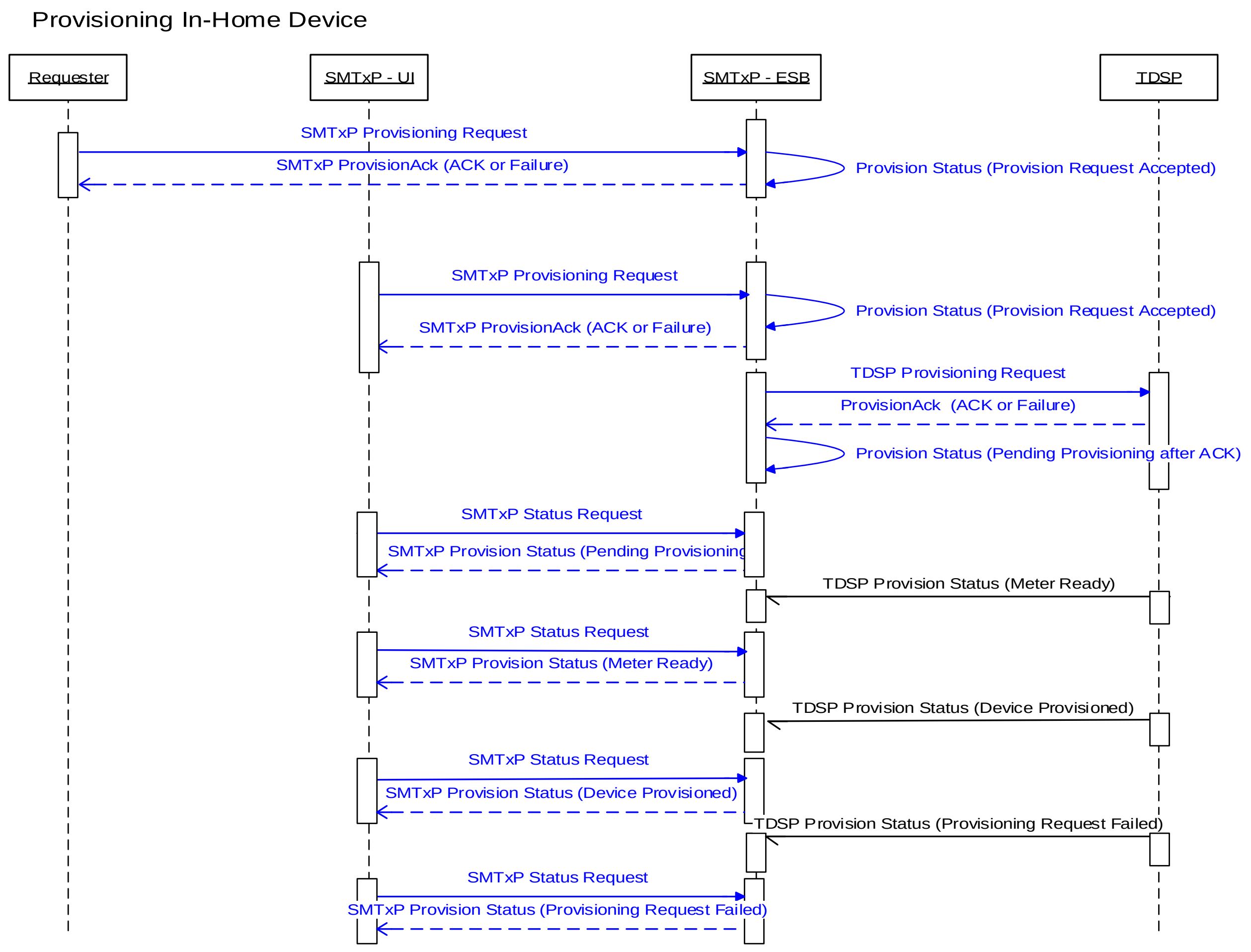

Provisioning In-Home Device
Requester
SMTxP - UI
SMTxP - ESB
TDSP
SMTxP Provisioning Request
SMTxP ProvisionAck (ACK or Failure)
Provision Status (Provision Request Accepted)
SMTxP Provisioning Request
Provision Status (Provision Request Accepted)
SMTxP ProvisionAck (ACK or Failure)
TDSP Provisioning Request
ProvisionAck (ACK or Failure)
Provision Status (Pending Provisioning after ACK)
SMTxP Status Request
SMTxP Provision Status (Pending Provisioning
TDSP Provision Status (Meter Ready)
SMTxP Status Request
SMTxP Provision Status (Meter Ready)
TDSP Provision Status (Device Provisioned)
SMTxP Status Request
SMTxP Provision Status (Device Provisioned)
TDSP Provision Status (Provisioning Request Failed)
SMTxP Status Request
SMTxP Provision Status (Provisioning Request Failed)

Figure 4: Provisioning an In-Home Device

SMT will only allow one In-Home Device to be provisioned per request. A sample provisioning request appears below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://schemas.esb.ams.com/smtxpprovisiondevice">
 <soapenv:Header>
  <wsse:Security xmlns:wsse=
   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <wsse:UsernameToken>
     <wsse:Username>REP_Sytem_Account</wsse:Username>
    </wsse:UsernameToken>
   </wsse:Security>
  </soapenv:Header>
     <soapenv:Body>
     <smt:processProvisionDevice>
    <SMTxPProvisioningRequest>
    <!--Optional  RequestID removed -->
    <RequesterType>3</RequesterType>
    <RequesterAuthenticationID>111111111</RequesterAuthenticationID>
    <RequesterID>ADMIN2</RequesterID>
    <RequestPriority>M</RequestPriority>
    <!--Optional: CallbackUri removed -->
    <DeviceProvisionRequestList>
        <DeviceProvisionRequest>
        <ESIID>00000000000000001</ESIID>
        <MeterSerialNumber>00000000000000000</MeterSerialNumber>
        <DeviceMACAddr>101BC50070000502</DeviceMACAddr>
        <DeviceInstallCode>83FED3407A939723A5C639B26916D505C3B5</DeviceInstallCode>
        <!--Optional: DeviceClusterSupport removed -->
        <!—Optional: DeviceClass removed -->
        <DeviceText>Living Room PCT</DeviceText>
        </DeviceProvisionRequest>
    </DeviceProvisionRequestList>
  </SMTxPProvisioningRequest>
 </smt:processProvisionDevice>
 </soapenv:Body>
</soapenv:Envelope>
```

When sending a device provisioning request, the SMT request will include:

- A request header as described in Section *4.2 SMT Request Header Information*.
- The Device provisioning request information as described in Table 6**.**

Element	Mandatory	Type	Description
ESIID	Y	string(17,64)	Energy Service Interface identifier
MeterSerialNumber	Y	string(30)	TDSP Meter manufacturer serial number
DeviceMACAddr	Y	string(16)	Device MAC Address
DeviceInstallCode	Y	string(36)	Device Installation Code
DeviceClusterSupport	N	Int	Sum of 1 - Messaging, 2 - Load Control, 4 Pricing, 0 - unspecified (max integer value = 7), default=0
DeviceClass	N	string(5)	Bitmap values. Bit position(right to left) as 1=InHomeDisplay, 2=LoadControlDevice, 3=ProgrammableThermostat, 4=IntelligentGateway. The SMT API will accept a string(5) value. Characters must be a "0" or a "1". ACK=FLR is returned if this value deviates from these rules.
DeviceText	N	string(256)	User-friendly label for device identification

Table 6: Device Provisioning Request Information

5.3 De-provisioning In-Home Devices

A requester submits the device de-provisioning request through the front-end web services API. (Requests can also be entered via the SMT user interface by removing an In-Home Device associated with an active In-Home Device Agreement.) SMT performs two levels of validations:

- XML validation is performed by an appliance on the SMT perimeter. If the request fails validation, a SOAP fault is returned. Refer to soap fault codes descriptions in *Section 2.3.*
- If XML is formatted correctly, business validations are performed by SMT. SMT validates that the requester is authorized to de-provision the devices to that ESIID, that the correct meter serial number was supplied and that the device is not already de-provisioned or in a de- provisioning request pending status to the ESIID. If a de-provisioning request passes business validation checks, SMT assigns a unique identifier, saves the request, and sends an acknowledgement. The request status at this point will be Remove Acknowledged[2]. If the request fails any of these checks, a failure is returned in a Request Acknowledgement in the form of request rejected.

When the request is sent, the TDSP will initially send an acknowledgment back to SMT - either a successful acknowledgement with the status of remove acknowledged or a failure with a status of request failed. There can be two remove acknowledged Status for a de-provisioned request but the acknowledgment will have different description one indicating that SMT received the request and the other indicating that the TDSP received the request. The TDSP may perform initial validation.

The TDSP would perform additional validations on the de-provisioning request if applicable as stated above. If failure occurs, the TDSP will return the failure status to SMT. Otherwise, the de-provisioning request is executed and the execution status is returned. Once the device has completed its de- provisioning process, the TDSP would return a successful status of device removed or a failure with remove failed.

Figure 5 shows the interface exchanges for de-provisioning an In-Home Device.

[2] All provisioning and de-provisioning status are visible through the Portal UI.

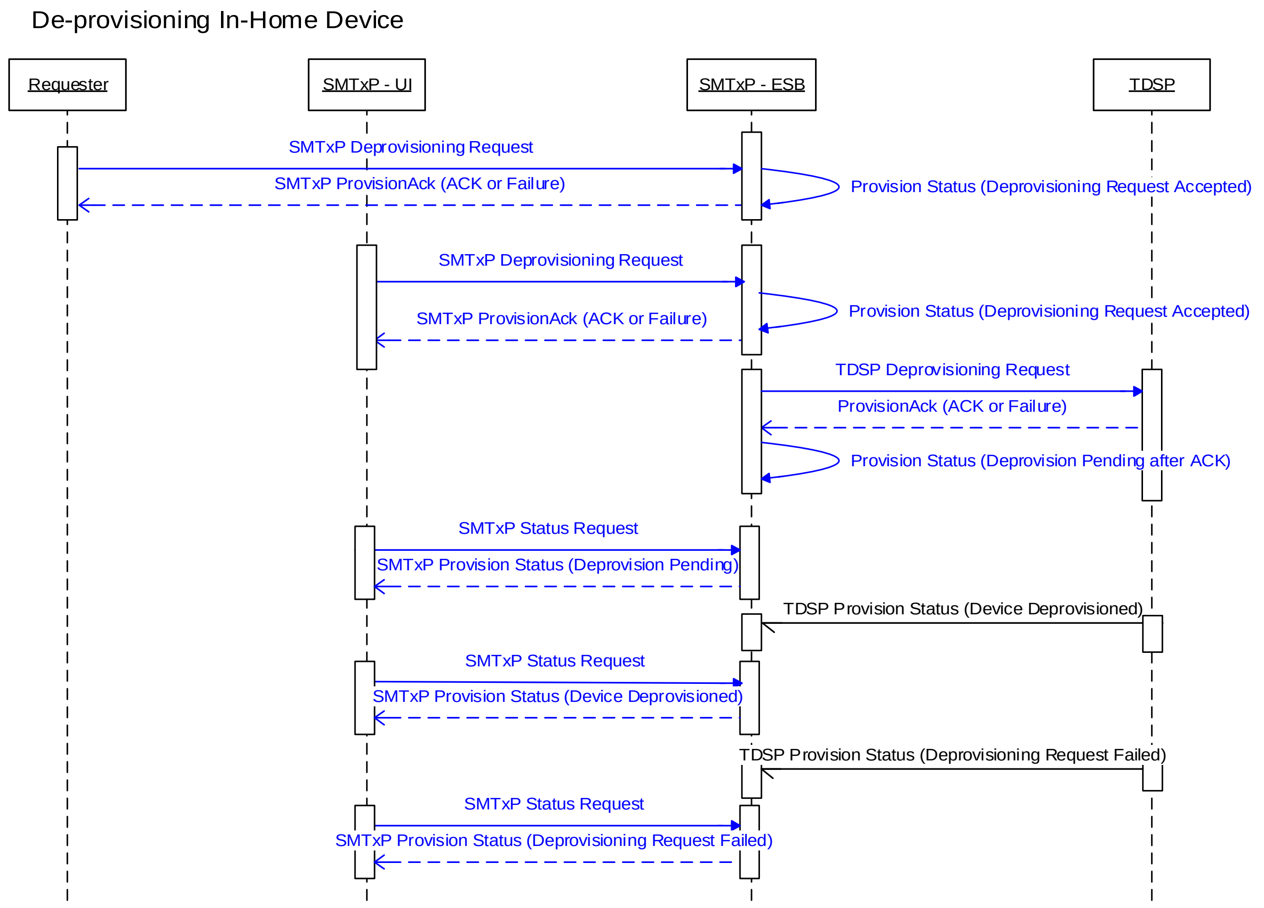

De-provisioning In-Home Device
Requester
SMTxP - UI
SMTxP - ESB
TDSP
SMTxP Deprovisioning Request
SMTxP ProvisionAck (ACK or Failure)
Provision Status (Deprovisioning Request Accepted)
SMTxP Deprovisioning Request
SMTxP ProvisionAck (ACK or Failure)
Provision Status (Deprovisioning Request Accepted)
TDSP Deprovisioning Request
ProvisionAck (ACK or Failure)
Provision Status (Deprovision Pending after ACK)
SMTxP Status Request
SMTxP Provision Status (Deprovision Pending)
TDSP Provision Status (Device Deprovisioned)
SMTxP Status Request
SMTxP Provision Status (Device Deprovisioned)
TDSP Provision Status (Deprovisioning Request Failed)
SMTxP Status Request
SMTxP Provision Status (Deprovisioning Request Failed)

Figure 5: De-provisioning an In-Home Device

SMT will only allow one In-Home Device to be de-provisioned per request. A sample de-provisioning request appears below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://schemas.esb.ams.com/smtxpprovisiondevice">
 <soapenv:Header>
  <wsse:Security xmlns:wsse=
   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <wsse:UsernameToken>
   <wsse:Username>REP_Sytem_Account</wsse:Username>
   </wsse:UsernameToken>
  </wsse:Security>
 </soapenv:Header>
  <soapenv:Body>
   <smt:processDeProvisionDevice>
          <SMTxPDeProvisionRequest>
      <!--Optional  RequestID removed -->
      <RequesterType>3</RequesterType>
```

```
        <RequesterAuthenticationID>111111111</RequesterAuthenticationID>
        <RequesterID> HAN_REP2_ADMIN </RequesterID>
        <RequestPriority>L</RequestPriority>
        <!--Optional: CallbackUri removed -->
        <DeviceDeprovisionRequestList>
           <DeviceDeprovisionRequest>
          <ESIID>00000000000000001</ESIID>
          <MeterSerialNumber>00000000000000000</MeterSerialNumber>
          <DeviceMACAddr>101BC50070000502</DeviceMACAddr>
          <ReasonCode>3</ReasonCode>
          <ReasonComment>Customer requested de-provisioning</ReasonComment>
           </DeviceDeprovisionRequest>
         </DeviceDeprovisionRequestList>
       </SMTxPDeProvisionRequest>
      </smt:processDeProvisionDevice>
    </soapenv:Body>
  </soapenv:Envelope>
```

When sending a device de-provisioning request, the SMT request will include:

- Request header as described in Section *4.2 SMT Request Header Information*
- Display request information as described in Table 7.

Element	Mandatory	Type	Description
ESIID	Y	string(17,64)	Energy Service Interface identifier
MeterSerialNumber	Y	string(30)	TDSP Meter manufacturer serial number
DeviceMACAddr	Y	string(16)	Device MAC Address
ReasonCode	Y	string(1)	De-provisioning reason code are : 1 = RepRequested 2 = CustomerRequested, 3 = MalfunctioningDevice, 4 = UnsupportedDevice, 5 = RequestProvByMistake,
ReasonComment	N	string(64)	Reason description for deprovisioning

Table 7: Device De-provisioning Request information

5.4 Device Provisioning and De-provisioning Request Acknowledgement

When a provisioning or de-provisioning request is sent to SMT, an appliance on the SMT perimeter performs XML validation. If the request fails validation, a SOAP fault is returned. If the request passes XML validation, it is passed to the SMT Enterprise Service Bus to process. The SMT ESB performs business validations and returns a Provisioning Request Acknowledgement.

A sample provisioning acknowledgement appears below.

```
<soapenv:Envelope xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body>
  <sm:processProvisionDeviceResponse xmlns:sm=
    "http://schemas.esb.ams.com/smtxpprovisiondevice">
     <SMTxPProvisionAck>
```

Element	Mandatory	Type	Description
RequestID	Y	string(32)	Unique Request ID which is generated by SMT
RequestStatus	Y	string(3)	Values returned are: ACK - Acknowledgement indicating the request was accepted for further processing. FLR - Failure, indicating the request was not accepted for processing
RequestStatusDesc	N	string(64)	Request Status Description

Table 8: Device Provisioning / De-provisioning Acknowledgement Header Record

If a request is sent with invalid addressing information, the invalid information will returned as one or more invalid request elements. The elements of an invalid request record appear in Table 9.

Element	Mandatory	Type	Description
ESIID	Y	string(17,64)	Energy Service Interface identifier
MeterSerialNumber	Y	string(30)	TDSP Meter manufacturer serial number
DeviceMACAddr	N	string(16)	Device MAC Address
Reason	N	string(128)	Reason of Failure at individual ESIID

Table 9: Device Provisioning and De-provisioning Status Record

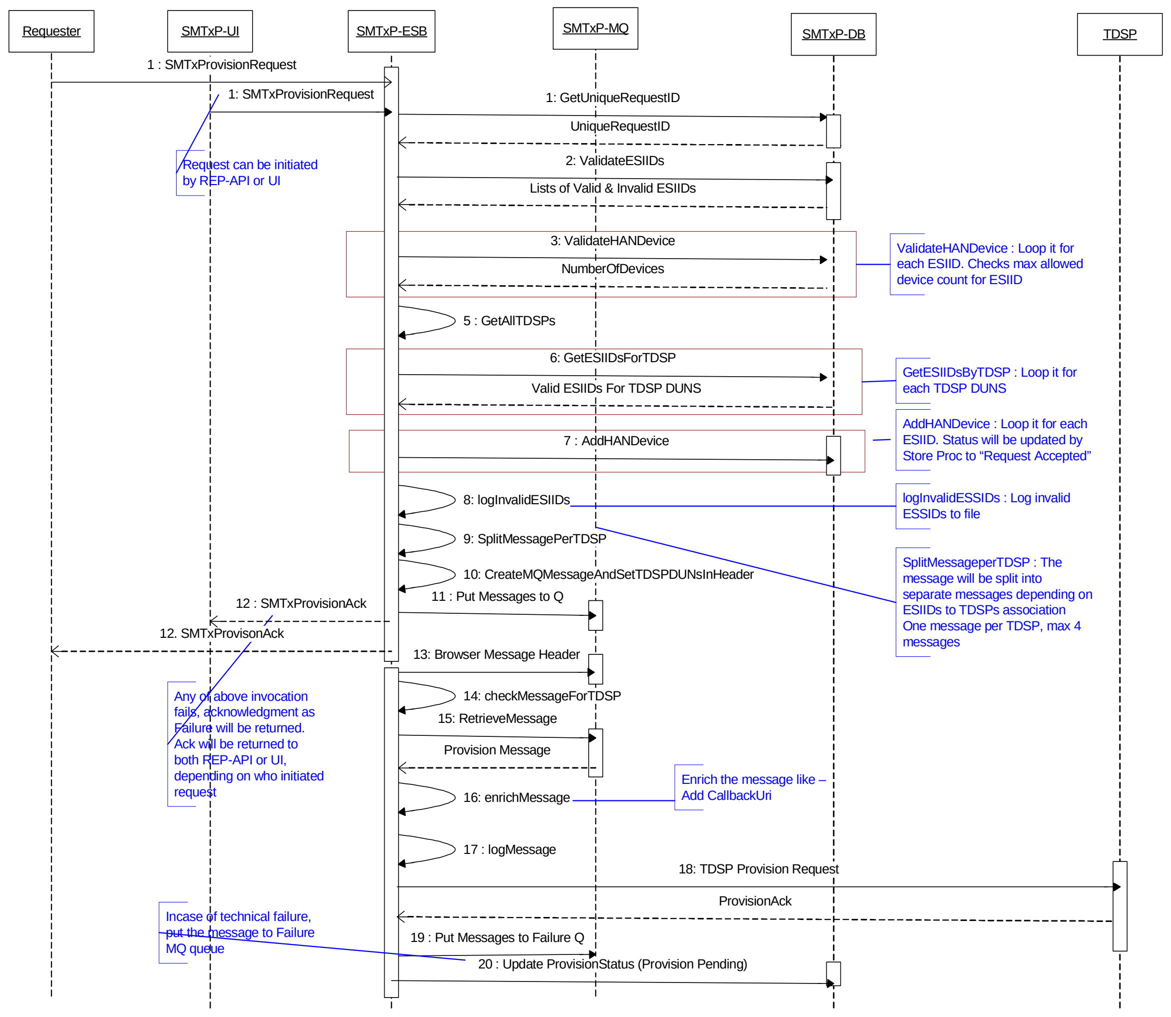

Requester
SMTxP-UI
SMTxP-ESB
SMTxP-MQ
SMTxP-DB
TDSP
1 : SMTxProvisionRequest
1: SMTxProvisionRequest
Request can be initiated by REP-API or UI
1: GetUniqueRequestID
UniqueRequestID
2: ValidateESIIDs
Lists of Valid & Invalid ESIIDs
3: ValidateHANDevice
NumberOfDevices
ValidateHANDevice : Loop it for each ESIID. Checks max allowed device count for ESIID
5 : GetAllTDSPs
6: GetESIIDsForTDSP
Valid ESIIDs For TDSP DUNS
GetESIIDsByTDSP : Loop it for each TDSP DUNS
7 : AddHANDevice
AddHANDevice : Loop it for each ESIID. Status will be updated by Store Proc to "Request Accepted"
8: logInvalidESIIDs
logInvalidESSIDs : Log invalid ESSIDs to file
9: SplitMessagePerTDSP
10: CreateMQMessageAndSetTDSPDUNsInHeader
SplitMessageperTDSP : The message will be split into separate messages depending on ESIIDs to TDSPs association One message per TDSP, max 4 messages
11 : Put Messages to Q
12 : SMTxProvisionAck
12. SMTxProvisonAck
13: Browser Message Header
14: checkMessageForTDSP
15: RetrieveMessage
Provision Message
Any of above invocation fails, acknowledgment as Failure will be returned. Ack will be returned to both REP-API or UI, depending on who initiated request
16: enrichMessage
Enrich the message like – Add CallbackUri
17 : logMessage
18: TDSP Provision Request
ProvisionAck
Incase of technical failure, put the message to Failure MQ queue
19 : Put Messages to Failure Q
20 : Update ProvisionStatus (Provision Pending)

Figure 6: De-provisioning an In-Home Device

5.5 Update Utility Enrollment Group Request

Note: Verify that the TDSP and the In-Home Device Manufacturer support the Utility Enrollment Group functionality before initiating the request.

A requester submits the update UEG request through the front-end web services API. SMT performs two levels of validations:

- XML validation is performed by an appliance on the SMT perimeter. If the request fails validation, a SOAP fault is returned. Refer to soap fault codes in descriptions in *Section 2.3.*
- If XML is formatted correctly, business validations are performed by SMT. SMT validates that the requester is authorized to de-provision the devices to that ESIID, that the correct meter serial number was supplied and that the device is not already de-provisioned or in a de- provisioning request pending status to the ESIID. If a de-provisioning request passes business validation checks, SMT assigns a unique identifier, saves the request, and sends an acknowledgement. The request status at this point will be Remove Acknowledged. If the request fails any of these checks, a failure is returned in a Request Acknowledgement in the form of request rejected.

When the request is sent, the TDSP will initially send an acknowledgment back to SMT - either a successful acknowledgement with the status of remove acknowledged or a failure with a status of request failed. The TDSP may perform initial validation.

The TDSP would perform additional validations on the update UEG request if applicable as stated above. If failure occurs, the TDSP will return the failure status to SMT. Otherwise, the de-provisioning request is executed and the execution status is returned. Once the device enrollment group has been set, the TDSP would return a status.

A sample Update UEG Request message appears below:

```
<soapenv:Body wsu:Id="id-7" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401- wss-
wssecurity-utility-1.0.xsd">
        <smt:processUpdateUEGDevice>
          <SMTxPUpdateUEGRequest>
        <!--Optional:-->
        <RequestID/>
        <RequesterType>3</RequesterType>
        <RequesterAuthenticationID>799530915</RequesterAuthenticationID>
        <RequesterID>reliantuser01</RequesterID>
        <RequestPriority>H</RequestPriority>
        <!--Optional:-->
        <CallbackUri>cnn.com</CallbackUri>
        <DeviceUpdateUEGRequestList>
          <!--1 to 100 repetitions:-->
              <DeviceUpdateUEGRequest>
            <ESIID>1008901020147398480100</ESIID>
            <MeterSerialNumber>61330847</MeterSerialNumber>
            <DeviceMACAddr>1234567890234534</DeviceMACAddr>
            <UtilityEnrollmentGroup>200</UtilityEnrollmentGroup>
              </DeviceUpdateUEGRequest>
        </DeviceUpdateUEGRequestList>
          </SMTxPUpdateUEGRequest>
        </smt:processUpdateUEGDevice>
  </soapenv:Body>
</soapenv:Envelope>
```

A Request header as described in Section *4.2 SMT Request Header Information.*

Display request information as described in Table 10.

Element	Mandatory	Type	Description
ESIID	Y	string(17,64)	Energy Service Interface identifier
MeterSerialNumber	Y	string(30)	TDSP Meter manufacturer serial number
DeviceMACAddr	Y	string(16)	Device MAC Address
UtilityEnrollmentGroup	Y	integer(1)	0 to 255

Table 10: Update UEG Request Information

Figure 7 shows the interface exchanges for updating the Utility Enrollment Group value on an In-Home Device.

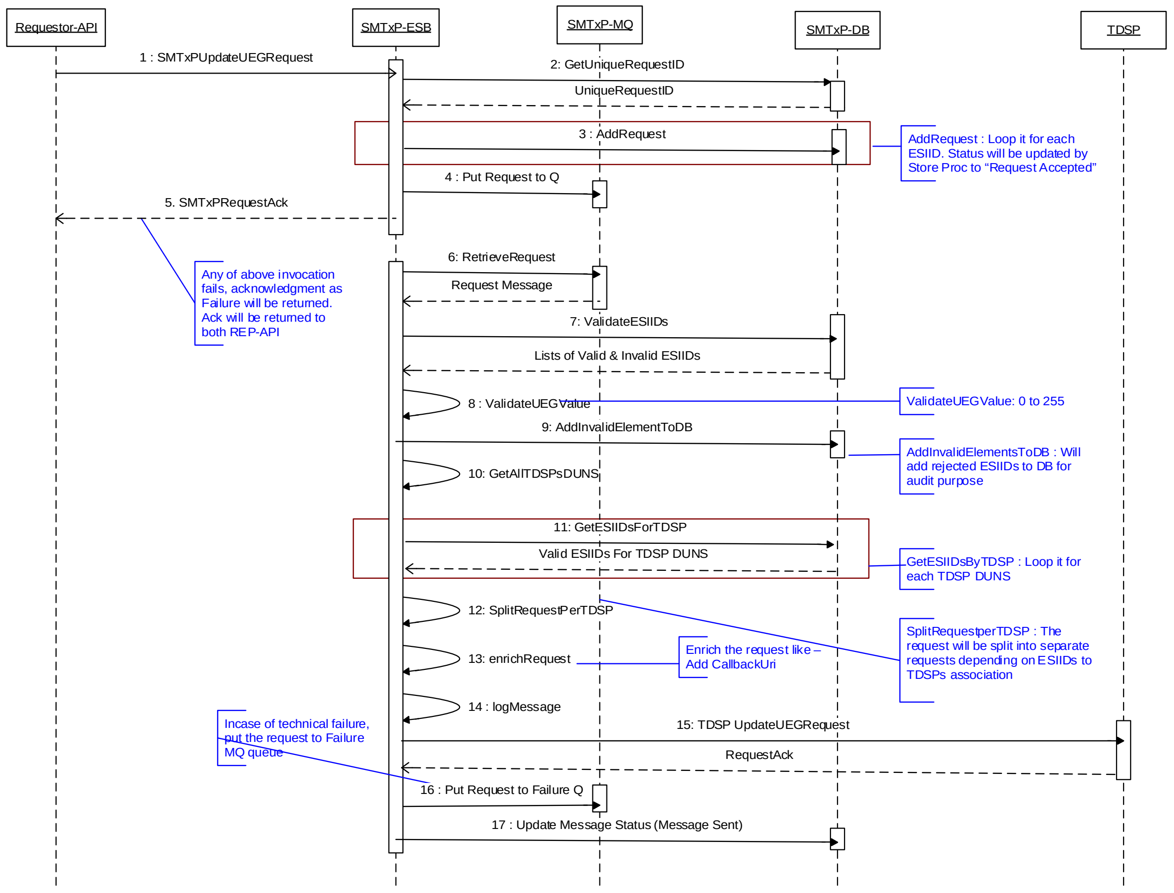

Requestor-API
SMTxP-ESB
SMTxP-MQ
SMTxP-DB
TDSP
1 : SMTxPUpdateUEGRequest
2: GetUniqueRequestID
UniqueRequestID
3 : AddRequest
AddRequest : Loop it for each ESIID. Status will be updated by Store Proc to "Request Accepted"
4 : Put Request to Q
5. SMTxPRequestAck
6: RetrieveRequest
Request Message
Any of above invocation fails, acknowledgment as Failure will be returned. Ack will be returned to both REP-API
7: ValidateESIIDs
Lists of Valid & Invalid ESIIDs
8 : ValidateUEGValue
ValidateUEGValue: 0 to 255
9: AddInvalidElementToDB
AddInvalidElementsToDB : Will add rejected ESIIDs to DB for audit purpose
10: GetAllTDSPsDUNS
11: GetESIIDsForTDSP
Valid ESIIDs For TDSP DUNS
GetESIIDsByTDSP : Loop it for each TDSP DUNS
12: SplitRequestPerTDSP
SplitRequestperTDSP : The request will be split into separate requests depending on ESIIDs to TDSPs association
13: enrichRequest
Enrich the request like – Add CallbackUri
14 : logMessage
Incase of technical failure, put the request to Failure MQ queue
15: TDSP UpdateUEGRequest
RequestAck
16 : Put Request to Failure Q
17 : Update Message Status (Message Sent)

Figure 7: Update UEG Request

5.6 Update Utility Enrollment Group Request Acknowledgement

When a update UEG request is sent to SMT, an appliance on the SMT perimeter performs XML validation. If the request fails validation, a SOAP fault is returned. If the request passes XML validation, it

is passed to the SMT Enterprise Service Bus to process. The SMT ESB performs business validations and returns a Update UEG Request Acknowledgement

A sample update UEG acknowledgement appears below.

```
<soapenv:Body wsu:Id="id-10" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
    <tds:processDeviceUpdateUEGStatus>
      <UpdateUEGStatusResponse>
```

```
              <!--Optional:-->
                <StatusID>313745</StatusID>
              <TDSPDUNSNumber>957877905</TDSPDUNSNumber>
                <DeviceUpdateUEGStatusList>
                <!--Zero or more repetitions:-->
                  <DeviceUpdateUEGStatus>
                  <ESIID>1008901020147398480100</ESIID>
                  <MeterSerialNumber>61330847</MeterSerialNumber>
                  <DeviceMACAddr>1234567890234534</DeviceMACAddr>
                  <EventCategoryID>UEG</EventCategoryID>
                  <StatusCode>COM</StatusCode>
                  <!--Optional:-->
                  <StatusDesc>Testing By Eswar</StatusDesc>
                  <StatusTimestamp>2011-08-10T16:01:12Z</StatusTimestamp>
                  <!--Optional:-->
                  <ReasonCode>Eswar</ReasonCode>
                  <!--Optional:-->
                  <ReasonComment>Eswar</ReasonComment>
                  </DeviceUpdateUEGStatus>
                </DeviceUpdateUEGStatusList>
              </UpdateUEGStatusResponse>
            </tds:processDeviceUpdateUEGStatus>
          </soapenv:Body>
        </soapenv:Envelope>
```

Element	Mandatory	Type	Description
RequestID	Y	string(32)	Unique Request ID which is generated by SMT
RequestStatus	Y	string(3)	Values returned are: ACK - Acknowledgement indicating the request was accepted for further processing. FLR - Failure, indicating the request was not accepted for processing
RequestStatusDesc	N	string(64)	Request Status Description

Table 11: Update UEG Acknowledgement Header Record

If a request is sent with invalid addressing information, the invalid information will returned as one or more invalid request elements. The elements of an invalid request record appear in Table 11.

Element	Mandatory	Type	Description
ESIID	Y	string(17,64)	Energy Service Interface identifier
MeterSerialNumber	Y	string(30)	TDSP Meter manufacturer serial number
DeviceMACAddr	N	string(16)	Device MAC Address
Reason	N	string(128)	Reason of Failure at individual ESIID

Table 12: Update UEG Acknowledgement Record

6. Zigbee Smart Energy Messaging

This section presents information concerning the submission of Smart Energy Profile message requests in the Smart Meter Texas portal.

6.1 Addressing of Zigbee Smart Energy Messages

SMT utilizes the Address Block group in the In-Home Device Messaging API's to validate the destination of the Zigbee Smart Energy Messages.

Table 13 describes the contents of each Address-Block element.

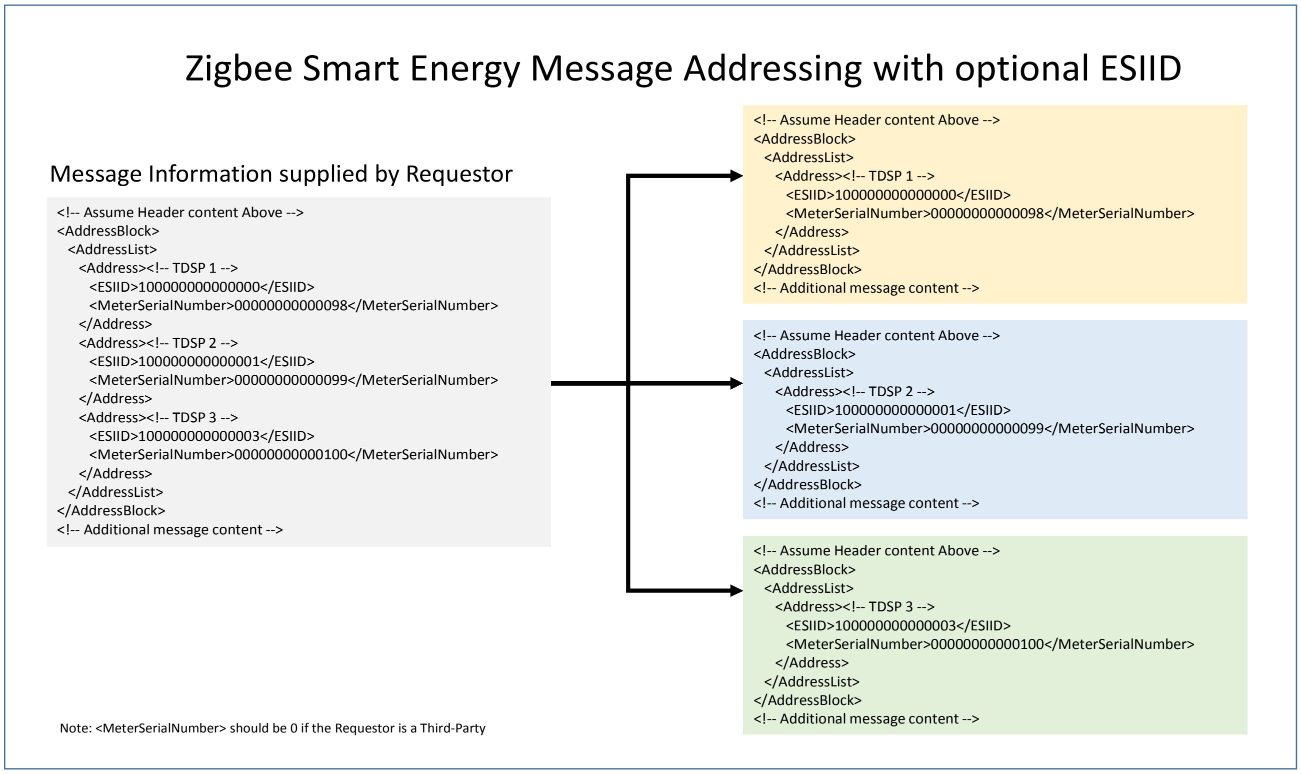
Element			Mandatory	Type	Description
Address List			N		Collection of address elements
	Address	ESIID	Y	string(64)	Energy Service Interface identifier CNP requires that the ESIID field be populated with the target ESIID(s) for Point-to-Point Messaging Oncor and AEP require that the ESIID field be populated with the target ESIID(s)
		Meter Serial Number	Y	string(30)	TDSP Meter manufacturer serial number CNP requires that the Meter Serial Number field be populated with the target Meter Serial Number for Point-to-Point Messaging Oncor/AEP require that the Meter Serial Number field be populated with the target Meter Serial Number(s)
		Device MAC Addr	N	string(16)	All TDSPs require that the Device MAC Addr field be populated with BLANK value.

Table 13: Address Blocks Element Information

The Address block includes a collection of address elements. SMT's Address block validation logic requires that the collection of ESIID/Meter address elements be populated in the In-Home Device Messaging API's to accept the In-Home Device Message. If it is not populated, SMT will generate an exception back to the requester.

6.2 Addressing of Messages with ESIIDs

Figure 8 presents an example of how message addressing will be handled when a requester includes ESIIDs with a message. On the left, the requester has provided three ESIIDs – 1 ESIID and meter serial number each for TDSP 1, TDSP 2 and TDSP 3. SMT creates 3 copies of the message and adds information as required for the message. If appropriate, SMT will evaluate the ESIID list and route and create messages to go to appropriate TDSPs.


Zigbee Smart Energy Message Addressing with optional ESIID
Message Information supplied by Requestor
Note: <MeterSerialNumber> should be 0 if the Requestor is a Third-Party

```
<!-- Assume Header content Above -->
<AddressBlock>
  <AddressList>
    <Address><!-- TDSP 1 -->
      <ESIID>100000000000000</ESIID>
      <MeterSerialNumber>00000000000098</MeterSerialNumber>
    </Address>
    <Address><!-- TDSP 2 -->
      <ESIID>100000000000001</ESIID>
      <MeterSerialNumber>00000000000099</MeterSerialNumber>
    </Address>
    <Address><!-- TDSP 3 -->
      <ESIID>100000000000003</ESIID>
      <MeterSerialNumber>00000000000100</MeterSerialNumber>
    </Address>
  </AddressList>
</AddressBlock>
<!-- Additional message content -->
```

```
<!-- Assume Header content Above -->
<AddressBlock>
  <AddressList>
    <Address><!-- TDSP 1 -->
      <ESIID>100000000000000</ESIID>
      <MeterSerialNumber>00000000000098</MeterSerialNumber>
    </Address>
  </AddressList>
</AddressBlock>
<!-- Additional message content -->
```

```
<!-- Assume Header content Above -->
<AddressBlock>
  <AddressList>
    <Address><!-- TDSP 2 -->
      <ESIID>100000000000001</ESIID>
      <MeterSerialNumber>00000000000099</MeterSerialNumber>
    </Address>
  </AddressList>
</AddressBlock>
<!-- Additional message content -->
```

```
<!-- Assume Header content Above -->
<AddressBlock>
  <AddressList>
    <Address><!-- TDSP 3 -->
      <ESIID>100000000000003</ESIID>
      <MeterSerialNumber>00000000000100</MeterSerialNumber>
    </Address>
  </AddressList>
</AddressBlock>
<!-- Additional message content -->
```

Figure 8: Message Addressing with ESIIDs

SMT will limit the number of address elements to 10,000. Requests with more than 10,000 address elements will be rejected.

6.3 Simple Text/Display Messaging

A Requester submits simple text message request via the SMT API. SMT performs two levels of validations:

- XML validation is performed by SMT. If the request fails validation, a SOAP fault is returned. Please refer to table 2.1.3 for further details.
- If XML is formatted correctly, business validations are performed by SMT. SMT validates that the requester is authorized to send a text message. If the simple text messaging request passes business validation checks, SMT assigns a unique identifier, saves the request, and returns an acknowledgement to the requester. If the request fails any of these checks, a failure is returned in a Request Acknowledgement in the form of request rejected.

SMT will create a copy and send to the respective TDSPs based on addressing schemes explained in section 6.1. When the request is sent, the TDSP may perform initial validation on the simple text messaging request and then acknowledge the request or return a failure. The request status becomes *Request Acknowledged by TDSP* when the TDSP returns an acknowledgement or *Request Failed* if a failure condition is returned.

The TDSP will perform additional validations on the simple text messaging request after returning the acknowledgement. If a failure occurs, the TDSP will return the failure status to SMT. Otherwise, the simple text messaging request is executed and the execution status is returned. In either case, SMT will store the simple text messaging request status that is returned.

The failure status will be different for each TDSP and depends on the failure condition.

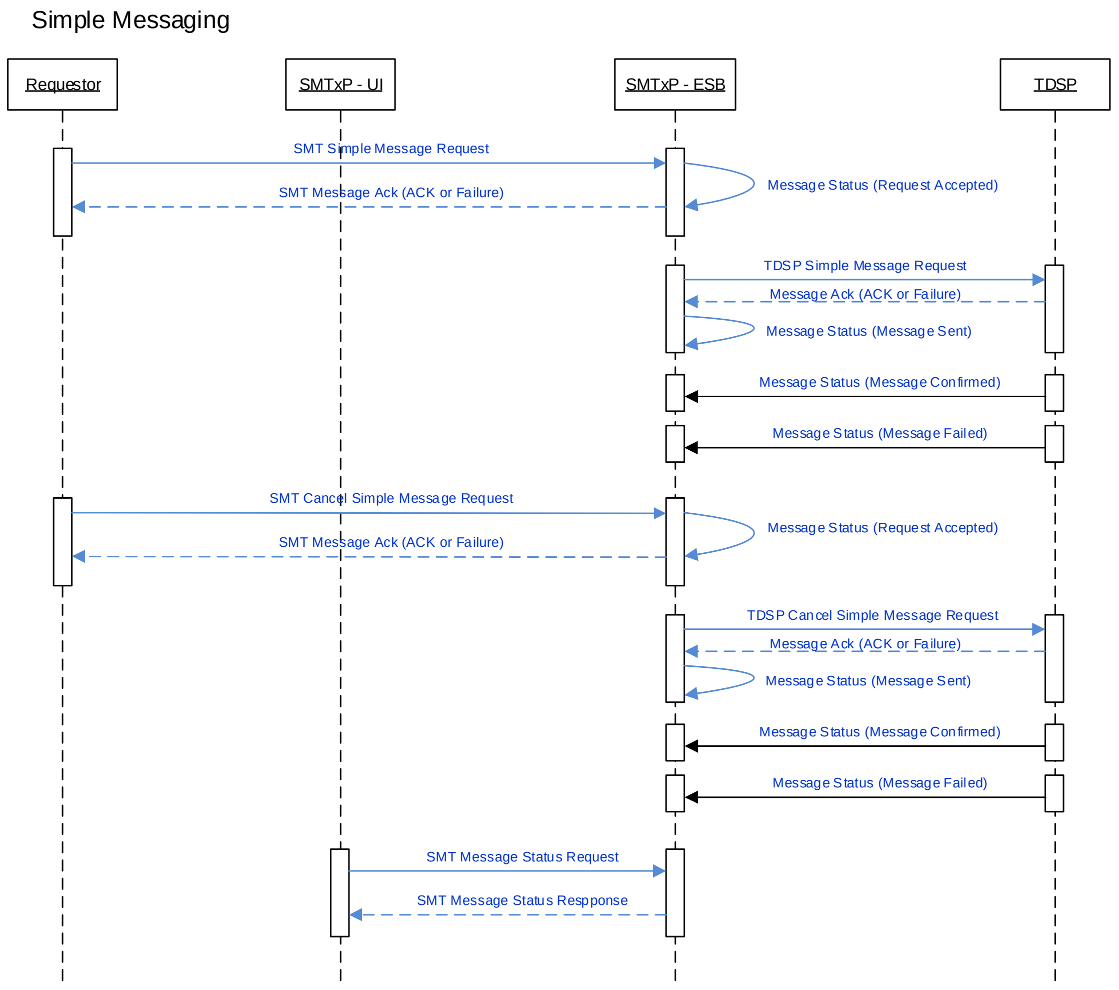

Simple Messaging
Requestor
SMTxP - UI
SMTxP - ESB
TDSP
SMT Simple Message Request
SMT Message Ack (ACK or Failure)
Message Status (Request Accepted)
TDSP Simple Message Request
Message Ack (ACK or Failure)
Message Status (Message Sent)
Message Status (Message Confirmed)
Message Status (Message Failed)
SMT Cancel Simple Message Request
SMT Message Ack (ACK or Failure)
Message Status (Request Accepted)
TDSP Cancel Simple Message Request
Message Ack (ACK or Failure)
Message Status (Message Sent)
Message Status (Message Confirmed)
Message Status (Message Failed)
SMT Message Status Request
SMT Message Status Respponse

Figure 9: Sending and Canceling Simple Text / Display Message

Figure 9 shows the interface exchanges for sending and canceling simple text/display messages.

6.3.1 Sending Simple Text / Display Messages

A sample simple text message request appears below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:smt="http://schemas.esb.ams.com/smtxpmessaging">
 <soapenv:Header>
  <wsse:Security xmlns:wsse=
    "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <wsse:UsernameToken>
    <wsse:Username>Entity_Sytem_Account</wsse:Username>
```

```
        </wsse:UsernameToken>
       </wsse:Security>
     </soapenv:Header>
       <soapenv:Body>
         <smt:processSimpleMessaging>
           <SMTxPSimpleMessageRequest>
             <!--Optional  RequestID removed -->
             <RequesterType>3</RequesterType>
             <RequesterAuthenticationID>111111111</RequesterAuthenticationID>
               <RequesterID> HAN_REP2_ADMIN </RequesterID>
               <RequestPriority>L</RequestPriority>
               <!--Optional: CallbackUri removed -->
               <AddressBlock>
                 <!--Optional: GroupID removed -->
                 <AddressList>
                      <!--0 to 10000 repetitions:-->
                      <Address>
                     <ESIID>00000000000000001</ESIID>
                     <MeterSerialNumber>00000000000000098</MeterSerialNumber>
                     <DeviceMACAddr>101BC50070000502</DeviceMACAddr>
                      </Address>
                 </AddressList>
               </AddressBlock>
               <SimpleMessageBlock>
                 <MessageID>123</MessageID>
                 <StartTime>2000-01-01T00:00:00Z</</StartTime>
                 <DurationTime>20</DurationTime>
                 <Message>Welcome to the Smart Meter Texas Portal!</Message>
                 <MCTransmission>0</MCTransmission>
                 <MCPriority>0</MCPriority>
                 <MCConfirmation>0</MCConfirmation>
               </SimpleMessageBlock>
               </SMTxPSimpleMessageRequest>
             </smt:processSimpleMessaging>
        </soapenv:Body>
      </soapenv:Envelope>
```

When sending a request to display a simple text message, the SMT request will include:

- A message header as described in Section 4.2
- An address block as described in Section 6.1
- Display request information as described in Table 14.

Element	Mandatory	Type	Description
MessageID	Y	Int	Message Identifier, Requester issued

Element	Mandatory	Type	Description
StartTime	Y	dateTime	The time at which the message becomes valid. Requester provided value. For "now", use 2000-01-01T00:00:00Z
DurationTime	Y	Int	SEP defines a range from 0 to 0xFFFF. In Release 1, the maximum number of minutes allowed is 255.
Message	Y	string(80)	As per SEP
MCTransmission	Y	Int	Values are per SEP: 0 Secured/normal 1 InterPAN 2 Secured & InterPAN In Release 1, SMT will only allow a value of 0.
MCPriority	Y	Int	Values are per SEP: 0 Low 1 Medium 2 High 3 Critical
MCConfirmation	Y	Int	Values are per SEP: 0 confirmation not required 1 confirmation required

Table 14: Smart Energy Text Display Request Information

6.3.2 Canceling Simple Text / Display Messages

SMT will perform validations, save the request, and send a cancellation acknowledgement. The message status for the cancellation at this point will be *request Accepted*. If there is a failure, the status will be request rejected.

SMT will formulate cancellation requests and send it to the TDSPs. The TDSP is will acknowledge the message or return a failure .The request status becomes *Request Acknowledged by TDSP* when the TDSP returns an acknowledgement or *Failure* if a failure condition is returned.

The TDSP may return additional Status regarding the request later. SMT will store the simple text messaging request cancellation Status that are returned. The format of the cancel text message is described in table 11.

A sample simple text message cancellation request appears below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:smt="http://schemas.esb.ams.com/smtxpmessaging">
 <soapenv:Header>
  <wsse:Security xmlns:wsse=
   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <wsse:UsernameToken>
   <wsse:Username>Entity_Sytem_Account</wsse:Username>
     </wsse:UsernameToken>
  </wsse:Security>
 </soapenv:Header>
```

```
<soapenv:Body>
  <smt:processCancelSimpleMessage>
        <SMTxPCancelSimpleMessageRequest>
      <!--Optional  RequestID removed -->
      <RequesterType>3</RequesterType>
      <RequesterAuthenticationID>111111111</RequesterAuthenticationID>
      <RequesterID> HAN_REP2_ADMIN </RequesterID>
      <RequestPriority>L</RequestPriority>
      <!--Optional: CallbackUri removed -->
      <AddressBlock>
            <!--Optional: GroupID removed -->
            <AddressList>
              <!--0 to 10000 repetitions:-->
              <Address>
            <ESIID>00000000000000001</ESIID>
            <MeterSerialNumber>00000000000000098</MeterSerialNumber>
            <DeviceMACAddr>101BC50070000502</DeviceMACAddr>
              </Address>
        </AddressList>
      </AddressBlock>
      <CancelSimpleMessageBlock>
        <MessageRequestID>123</MessageRequestID>
        <MCTransmission>0</MCTransmission>
        <MCPriority>0</MCPriority>
        <MCConfirmation>0</MCConfirmation>
      </CancelSimpleMessageBlock>
        </SMTxPCancelSimpleMessageRequest>
  </smt:processCancelSimpleMessage>
</soapenv:Body>
  </soapenv:Envelope>
```

When sending a request to cancel display of a simple text message, the request will include:

- A message header as described in Section 4.2
- An address block as described in Section 6.1
- Display cancellation request information as described in Table 15.

Element	**Mandatory**	**Type**	**Description**
MessageRequestID	Y	string(32)	This is the SMT-generated value sent in the header of the original text display request.
MessageID	Y	Int	This message id assigned by the requester in the original display text request.
MCTransmission	Y	Int	Values are per SEP: 0 Secured/normal 1 InterPAN 2 Secured & InterPAN SMT will only allow a value of 0.

Element	Mandatory	Type	Description
MCPriority	Y	Int	Values are per SEP: 0 Low 1 Medium 2 High 3 Critical
MCConfirmation	Y	Int	Values are per SEP: 0 confirmation not required 1 confirmation required

Table 15: Smart Energy Text Display Cancellation Information

6.4 Load Control Messaging

A Requester submits a load control event request to be displayed through the SMT API. SMT performs two levels of validations:

- XML validation is performed by an appliance on the SMT perimeter. If the request fails validation, a SOAP fault is returned. Please refer to table 2.1.3 for further details.
- If XML is formatted correctly, business validations are performed by SMT. SMT validates that the requester is authorized to send a load control event to addressed devices. If the load control event request passes business validation checks, SMT assigns a unique identifier, saves the request, and sends an acknowledgement. The request status at this point will be *Request Accepted*. If the request fails any of these checks, a failure is returned in a Request Acknowledgement in the form of request rejected.

SMT will create a copy and send to the respective TDSPs based on addressing schemes explained in section 6.1. When the request is sent, the TDSP may perform initial validation on the load control event request and then to acknowledge the request or return a failure indicator.

The TDSP will perform additional validations on the load control event request after returning the acknowledgement. If a failure occurs, the TDSP will return the failure status to SMT. Otherwise, the load control event request is executed and the execution status is returned. In either case, SMT will store the load control event request status that is returned.

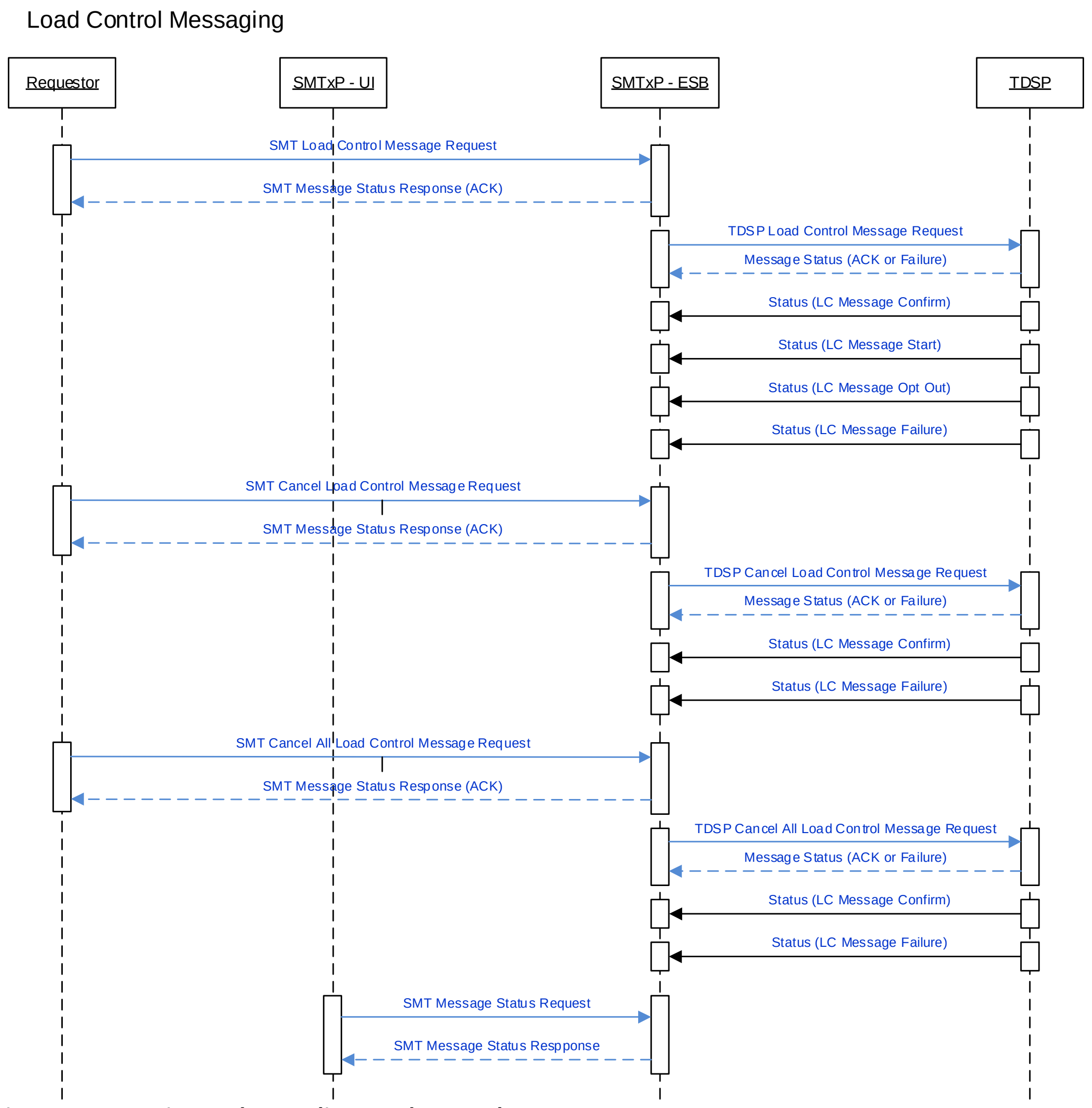

Load Control Messaging
Requestor
SMTxP - UI
SMTxP - ESB
TDSP
SMT Load Control Message Request
SMT Message Status Response (ACK)
TDSP Load Control Message Request
Message Status (ACK or Failure)
Status (LC Message Confirm)
Status (LC Message Start)
Status (LC Message Opt Out)
Status (LC Message Failure)
SMT Cancel Load Control Message Request
SMT Message Status Response (ACK)
TDSP Cancel Load Control Message Request
Message Status (ACK or Failure)
Status (LC Message Confirm)
Status (LC Message Failure)
SMT Cancel All Load Control Message Request
SMT Message Status Response (ACK)
TDSP Cancel All Load Control Message Request
Message Status (ACK or Failure)
Status (LC Message Confirm)
Status (LC Message Failure)
SMT Message Status Request
SMT Message Status Respponse

Figure 10: Creating and Canceling Load Control Events

Figure 10 shows the interface exchanges for creating and cancelling load control events. The requester can cancel a single event or cancel all load control events through the front-end web services API. SMT will perform validations, save the request, and send a cancellation acknowledgement. The message status for the cancellation at this point will be *Message Accepted*. (Change this)

SMT will formulate the appropriate cancellation request and send it to the TDSPs. The TDSPs will acknowledge the message or return a failure. SMT will log acknowledgements from all TDSPs. The TDSP may return additional status regarding the request later.

6.4.1 Creating a Load Control Event

A sample load control event request appears below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:smt="http://schemas.esb.ams.com/smtxpmessaging">
 <soapenv:Header>
  <wsse:Security xmlns:wsse=
   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <wsse:UsernameToken>
   <wsse:Username>Entity_Sytem_Account</wsse:Username>
      </wsse:UsernameToken>
  </wsse:Security>
 </soapenv:Header>
  <soapenv:Body>
    <smt:processLoadControlEvent>
         <SMTxPLoadControlEventRequest>
       <!--Optional  RequestID removed -->
       <RequesterType>3</RequesterType>
       <RequesterAuthenticationID>111111111</RequesterAuthenticationID>
       <RequesterID> HAN_REP2_ADMIN </RequesterID>
       <RequestPriority>L</RequestPriority>
       <!--Optional: CallbackUri removed -->
       <AddressBlock>
         <!--Optional: GroupID removed -->
         <AddressList>
              <!--0 to 10000 repetitions:-->
              <Address>
            <ESIID>00000000000000001</ESIID>
            <MeterSerialNumber>00000000000000098</MeterSerialNumber>
              </Address>
         </AddressList>
       </AddressBlock>
       <LCMessageBlock>
         <EventID>135</EventID>
         <StartTime>2000-01-01T00:00:00Z</StartTime>
         <DurationTime>180</DurationTime>
         <DeviceClass>0000000000000000</DeviceClass>
         <UtilityEnrollmentGroup>0</UtilityEnrollmentGroup>
         <CriticalityLevel>4</CriticalityLevel>
         <EventControl>0</EventControl>
         </LCMessageBlock>
       </SMTxPLoadControlEventRequest>
     </smt:processLoadControlEvent>
     </soapenv:Body>
    </soapenv:Envelope>
```

When sending a request to create a load control event, the SMT request will include:

- A message header as described in Section 4.2
- An address block as described in Section 6.1
- Display cancellation request information as described in Table 16.

Element	Mandatory	Type	Description
EventID	Y	Int	Event identifier, unique within messages. REPS assigned EventID
StartTime	Y	dateTime	The time at which the message becomes valid. Requester provided value. For "now", use 2000-01-01T00:00:00Z
DurationTime	Y	Int	As per SEP: allowed values are 1 to 1440
DeviceClass	Y	string(16)	This is a bitmap value as related to Table D.2 in Section D.2.2.3.1.1.1 in revision 15 of the Smart Energy Profile Spec (page 143). That table defines a bit map for Load Control the device class. Bit 0 (right- most bit) controls HVAC compressors or furnaces. Bit 1 controls Strip and baseboard heaters...... Bit 11 controls Generation systems. Bits 12-15 are reserved. The SMT API will accept a string(16) value. Counting characters from the right, characters 1-12 must be a "0" or a "1". ACK=FLR is returned if this value deviates from these rules.
UtilityEnrollmentGroup	Y	Int	Per SEP: Range 0 - 255
CriticalityLevel	Y	Int	Per SEP: Range of 1-9
CoolingTemperationOffset	N	Int	Per SEP
HeatingTemperatureOffset	N	Int	Per SEP
CoolingTemperationSetPoint	N	Int	Per SEP, Range -27315 to 32766 & 32768
HeatingTemperatureSetPoint	N	Int	Per SEP, Range -27315 to 32766 & 32768
AverageLoadAdjustPercent	N	Int	As per SEP: Range of -100 to 100 or 128(0x80) which indicates the field is not used.
DutyCycle	N	Int	As per SEP: Range of 0 to 100 & 255 for NULL
EventControl	Y	Int	Values are per Zigbee: 0 – do not randomize start time, do not randomize end time 1 – randomize start time, do not randomize end time 2 – do not randomize start time, randomize end time 3 – randomize start and end times

Table 16: Smart Energy Load Control Request Information

6.4.2 Cancelling a Load Control Event

A sample load control event cancellation request appears below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:smt="http://schemas.esb.ams.com/smtxpmessaging">
 <soapenv:Header>
```

```
     <wsse:Security xmlns:wsse=
       "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
     <wsse:UsernameToken>
       <wsse:Username>Entity_Sytem_Account</wsse:Username>
      </wsse:UsernameToken>
     </wsse:Security>
    </soapenv:Header>
     <soapenv:Body>
       <smt:processCancelLCEvent>
             <SMTxPCancelLoadControlEventRequest>
           <!--Optional  RequestID removed -->
           <RequesterType>3</RequesterType>
           <RequesterAuthenticationID>111111111</RequesterAuthenticationID>
           <RequesterID> HAN_REP2_ADMIN </RequesterID>
           <RequestPriority>L</RequestPriority>
           <!--Optional: CallbackUri removed -->
           <AddressBlock>
                 <!--Optional: GroupID removed -->
             <AddressList>
                   <!--0 to 10000 repetitions:-->
                   <Address>
                 <ESIID>00000000000000001</ESIID>
                 <MeterSerialNumber>00000000000000098</MeterSerialNumber>
                 <DeviceMACAddr>101BC50070000502</DeviceMACAddr>
                   </Address>
             </AddressList>
           </AddressBlock>
           <CancelLCMessageBlock>
             <LCMessageID>?</LCMessageID>
             <EventID>135</EventID>
             <StartTime>2000-01-01T00:00:00Z</</StartTime>
             <DeviceClass>0000000000000000</DeviceClass>
             <UtilityEnrollmentGroup>0</UtilityEnrollmentGroup>
             <CancelControl>0</CancelControl>
           </CancelLCMessageBlock>
             </SMTxPCancelLoadControlEventRequest>
       </smt:processCancelLCEvent>
     </soapenv:Body>
      </soapenv:Envelope>
```

When sending a request to cancel a load control event, the SMT request will include:

- A message header as described in Section 4.2
- An address block as described in Section 6.1
- Display cancellation request information as described in Table 16.

If a load control event is cancelled for a specific load control event then the event id should be included in the load control cancellation request information.

If a load control event is cancelled for a set of ESIIDs, those ESIIDs and meter serial numbers will be included in individual AddressBlock elements with an AddressBlockList element. See section 6 for more information on these elements.

In the current release, CNP will cancel all load cancel events if no specific load control event id is included in the request.

Element	Mandatory	Type	Description
LCRequestID	Y	string	Original RequestID of the LC Message
EventID	Y	string(32)	EventID of issued event that needs to be cancelled
EffectiveTime	Y	dateTime	The time at which the message becomes valid. Requester provided value. For "now", use 2000-01-01T00:00:00Z
DeviceClass	Y	string(16)	This is a bitmap value as related to Table D.3 in Section D.2.2.3.2.1. in revision 15 of the Smart Energy Profile Spec (page 148). That table defines a bit map for Load Control the device class. Bit 0 (right-most bit) controls HVAC compressors or furnaces. Bit 1 controls Strip and baseboard heaters...... Bit 11 controls Generation systems. Bits 12-15 are reserved. The SMT API will accept a string(16) value. Counting characters from the right, characters 1-12 must be a "0" or a "1". ACK=FLR is returned if this value deviates from these rules.
UtilityEnrollmentGroup	Y	Int	Per SEP: Range 0 - 255
CancelControl	Y	Int	As per SEP: 0 do not randomize 1 randomizes

Table 17: Smart Energy Load Control Cancellation Request Information

6.4.3 Cancelling ALL Load Control Events

A sample request to cancel all load control events appears below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://schemas.esb.ams.com/smtxpmessaging">
 <soapenv:Header>
  <wsse:Security xmlns:wsse=
    "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <wsse:UsernameToken>
    <wsse:Username>Entity_Sytem_Account</wsse:Username>
   </wsse:UsernameToken>
  </wsse:Security>
 </soapenv:Header>
 <soapenv:Body>
    <smt:processCancelAllLCEvents>
```

```
          <SMTxPCancelAllLoadControlEventRequest>
            <!--Optional  RequestID removed -->
            <RequesterType>3</RequesterType>
            <RequesterAuthenticationID>111111111</RequesterAuthenticationID>
            <RequesterID> HAN_REP2_ADMIN </RequesterID>
            <RequestPriority>L</RequestPriority>
            <!--Optional: CallbackUri removed -->
            <AddressBlock>
                 <!--Optional: GroupID removed -->
                 <AddressList>
                   <!--0 to 10000 repetitions :-->
                   <Address>
                 <ESIID>00000000000000001</ESIID>
                 <MeterSerialNumber>00000000000000098</MeterSerialNumber>
                   </Address>
                 </AddressList>
            </AddressBlock>
            <CancelAllLCEventMessageBlock>
                 <CancelControl>0</CancelControl>
            </CancelAllLCEventMessageBlock>
          </SMTxPCancelAllLoadControlEventRequest>
        </smt:processCancelAllLCEvents>
      </soapenv:Body>
    </soapenv:Envelope>
```

When cancelling all load control events, the SMT request will include:

- A message header as described in Section 4.2
- An address block as described in Section 6.1
- Display cancellation request information as described inTable 16.

Element	Mandatory	Type	Description
CancelControl	Y	Int	Values are per Zigbee: 0 do not randomize 1 randomize

Table 18: Smart Energy Request Information for Cancelling ALL Load Control Events

6.5 Sending Price Signals

A Requester submits a price signal request to be passed through the SMT API. SMT performs two levels of validations:

- XML validation is performed by an appliance on the SMT perimeter. If the request fails validation, a SOAP fault is returned. Please refer to section 2.1.3 for further details.
- If XML is formatted correctly, business validations are performed by SMT. SMT validates that the requester is authorized to send pricing signals to the devices. If the price signal request passes business validation checks, SMT assigns a unique identifier, saves the request, and sends an

acknowledgement. The request status at this point will be *Request Accepted*. If the request fails any of these checks, a failure is returned in a Request Acknowledgement in the form of request rejected.

SMT will create a copy and send to the respective TDSPs based on addressing schemes explained in section 6.1 when the request is sent, the TDSP may perform minimal validation on the price signal request and then will acknowledge the request or return a failure.

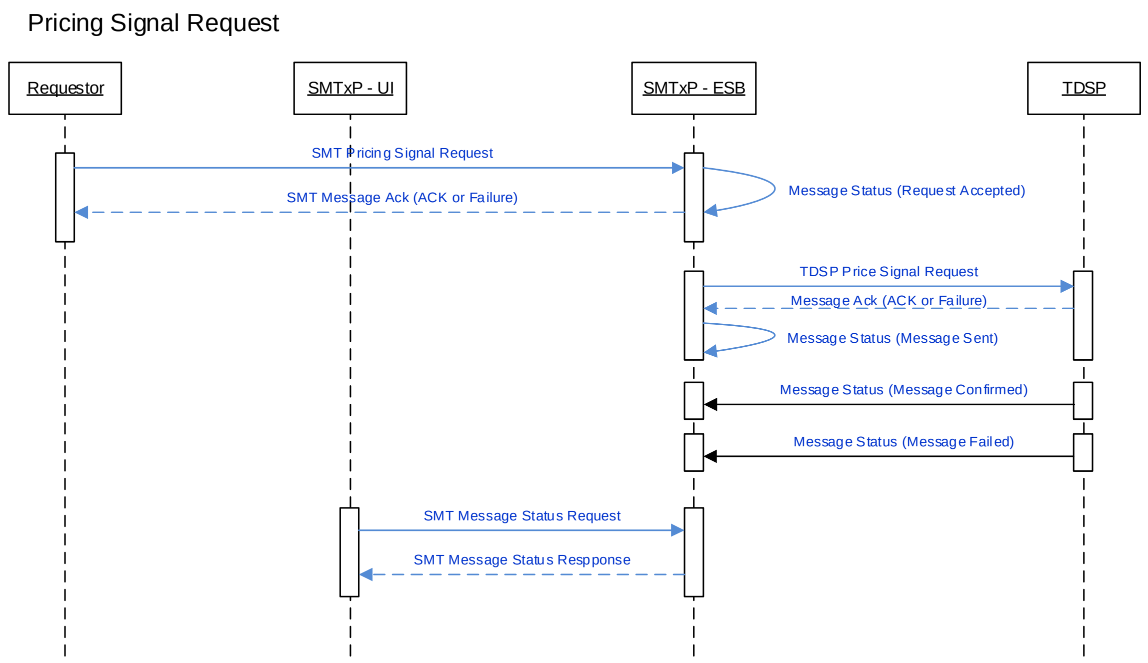

Pricing Signal Request
Requestor
SMTxP - UI
SMTxP - ESB
TDSP
SMT Pricing Signal Request
SMT Message Ack (ACK or Failure)
Message Status (Request Accepted)
TDSP Price Signal Request
Message Ack (ACK or Failure)
Message Status (Message Sent)
Message Status (Message Confirmed)
Message Status (Message Failed)
SMT Message Status Request
SMT Message Status Respponse

Figure 11: Sending a Price Signal Request

Figure 11 shows the interface exchange for sending a price signal message.

6.5.1 Sending Price Signal Request

A sample price signal request appears below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:smt="http://schemas.esb.ams.com/smtxpmessaging">
 <soapenv:Header>
  <wsse:Security xmlns:wsse=
    "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <wsse:UsernameToken>
    <wsse:Username>Entity_Sytem_Account</wsse:Username>
   </wsse:UsernameToken>
  </wsse:Security>
 </soapenv:Header>
 <soapenv:Body>
```

```
<smt:processPricingMessage>
    <SMTxPPriceSignalRequest>
   <!--Optional  RequestID removed -->
   <RequesterType>3</RequesterType>
   <RequesterAuthenticationID>1234567890123</RequesterAuthenticationID>
   <RequesterID> TestREPUser1</RequesterID>
   <RequestPriority>L</RequestPriority>
    <!--Optional: CallbackUri removed -->
    <AddressBlock>
      <!--Optional: GroupID removed -->
       <AddressList>
        <!--0 to 10000 repetitions:-->
         <Address>
             <ESIID>12345678901234567</ESIID>
           <MeterSerialNumber>1234567890123456789012</MeterSerialNumber>
           <DeviceMACAddr> ABCD123456789012</DeviceMACAddr>
         </Address>
      </AddressList>
      </AddressBlock>
    <PriceMessageBlock>
      <ProviderID>123456789</ProviderID>
      <RateLabel>Rate Label-1</RateLabel>
      <IssuerEventID>123456789</IssuerEventID>
      <CurrentTime></CurrentTime>
      <UOM>0</UOM>
      <Currency>USD</Currency>
      <PriceTier>1</PriceTier>
      <PriceTrailingDigit>4</PriceTrailingDigit>
      <RegisterTier>1</RegisterTier>
      <StartTime>2000-01-01T00:00:00Z</</StartTime>
      <Duration>180</Duration>
      <Price>12545</Price>
      <GenerationPrice>12400</GenerationPrice>
    </PriceMessageBlock>
    </SMTxPPriceSignalRequest>
   </smt:processPricingMessage>
  </soapenv:Body>
</soapenv:Envelope>
```

When sending a price signal request, the SMT request will include:

- A message header as described in Section 4.2
- An address block as described in Section 6.1
- Display cancellation request information as described in Table 19.

Element	Mandatory	Type	Description
ProviderID	Y	Int	Provider ID (limited to 9 digits) – Requester will provide a unique number. Discussion of using the first 9 digits of the Requester's DUNS possibly, or a different identifier chosen by the Requester.
RateLabel	Y	String(12)	Rate Label
IssuerEventID	Y	Int	Issuer Event ID – Mandatory in SEP, unique identifier controlled by requester for message.
CurrentTime	Y	dateTime	Current Time – API will include current time field as mandatory for requester.
UOM	Y	Int	Unit Of Measure – Per SEP.
Currency	Y	String(3)	Currency (USD= US Dollars) - Use ISO 4217, US Dollar=USD=840, use character value.
Price Tier	Y	Int	Price Tier – Requester will provide. As per SEP - 0 means no Tiers, values 1 thru 6 indicate a Price Tier. Tier 1 is the least expensive. Tier 6 is the most expensive.
Price Trailing Digit	Y	Int	Price Trailing Digit – Requester will provide. As per SEP - number of digits to right of the decimal point in the price
Register Tier	Y	Int	Register Tier (4 bits Unsigned – Range 0 to 15) -As per SEP - Table D.31 in spec. 0 means no tier related. Register Tier values of 1 thru 6 allowed.
Start Time	Y	dateTime	The time at which the message becomes valid. Requester provided value. For "now", use 2000-01-01T00:00:00Z
Duration	Y	Int	Duration (in minutes) - Requester will provide. As per SEP, 16 bits, a max value of 0xffff (=65535) would mean stay in place until changed, by a subsequent Price message.
Price	Y	Int	Price – Requester will provide. As per SEP.
Price Ratio	N	Int	Price Ratio per SEP.
GenerationPrice	N	Int	Generation Price per SEP.
GenerationRatio	N	Int	Generation Price Ratio per SEP.
AlternateCostDelivered	N	Int	Alternate Cost Delivered per SEP.
AlternateCostUnit	N	Int	Alternate Cost Unit per SEP.
AlternateCostTrailingDigit	N	Int	Alternate Cost Trailing Digit per SEP.

Table 19: Smart Energy Price Signal Request Information

Per SEP - Nested and overlapping Publish Price commands are not allowed. The current active price will be replaced if new price information is received by the ESI.

6.6 Smart Energy Messaging Request Acknowledgement

When a SEP messaging request is sent to SMT, an appliance on the SMT perimeter performs XML validation. If the request fails validation, a SOAP fault is returned. If the request passes XML validation, it is passed to SMT. SMT ESB performs business validations and returns a Messaging Request Acknowledgement

A sample messaging acknowledgement appears below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header/>
    <soapenv:Body>
      <sm:processLoadControlEventResponse
  xmlns:sm="http://schemas.esb.ams.com/smtxpmessaging">
        <SMTxPMessageAck>
          <RequestID>2930</RequestID>
          <RequestStatus>Success</RequestStatus>
          <RequestStatusDesc>Request Accepted</RequestStatusDesc>
        </SMTxPMessageAck>
      </sm:processLoadControlEventResponse>
      </soapenv:Body>
    </soapenv:Envelope>
```

The Smart Energy message acknowledgement contains header information and zero or more invalid request elements. The Smart Energy messaging acknowledgement header record includes the 3 elements described in Table 20.

Element	Mandatory	Type	Description
RequestID	Y	string(32)	Unique Request ID which is generated by SMT
RequestStatus	Y	string(3)	Values returned are: ACK - Acknowledgement indicating the request was accepted for further processing. FLR - Failure, indicating the request was not accepted for processing
RequestStatusDesc	N	string	Request Status Description

Table 20: Smart Energy Messaging Request Acknowledgement Header Record

If a Smart Energy messaging request is sent with invalid addressing information, the invalid information will be returned as one or more invalid request elements. The elements of an invalid request record appear in Table 21.

Element	Mandatory	Type	Description
ESIID	Y	string(17,64)	Energy Service Interface identifier
MeterSerialNumber	Y	string(30)	TDSP Meter manufacturer serial number
DeviceMACAddr	N	string(16)	Device MAC Address
Reason	N	string(128)	Reason of Failure at individual ESIID

Table 21: Invalid Request Record