new Security Suite 1

1. Introduction

The Honeywell meters that will be used in the TIN/IKB project use some security mechanisms that have not been used in the AT Master framework yet; Security Suite 1. As such some development work will be required in order that sufficient capability exists within the Solution to support these meters.

2. Symmetric vs Asymmetric Cryptography

Security Suite 1 as defined in DLMS COSEM Greenbook can be implemented in many different ways. Therefore, for the avoidance of doubt, our expectations are:

Asymmetric Keys (Certificates) are used for:

- HLS7 Association Authentication (establishing a DLMS COSEM Association)
- Secure Renewal of Meter Symmetric Keys

Symmetric Keys are used for:

- Authentication and Encryption of Requests and Responses (in DLMS COSEM terms: APDUs)

3. Asymmetric General Discussion

The Honeywell Companion Specification explicitly mentions:

- Only a single ECC Keypair is held / owned by the meter.
- That Asymmetric Key Agreement Keys are not required. The only Keys that need to be shared between Device and HES are for the purposes of Digital Signing.
- That although DLMS/COSEM gives provision for delivery of a Certificate to be used to verify the Signature within the AARQ packet, in this implementation that will not be used.

The technical specification of Asymmetric protocol in phase 1:

- Digital Signature - ECDSA with NIST Curve P256
- Public Key Length - 64 bytes
- Private Key Length - 128 bytes

As far as can be seen, the HLS7 Authentication Mechanism involves digitally signing information in the intial AARQ packets exchanged during COSEM Association establishment. All HLS authentication mechanisms are based on mutual authentication, so both Device and HES must exchange Signed information in that exchange and the recipient be able to verify that signature. ECC signature mechanisms for payload signings also rely on both parties being able to verify the signature of the other.

Therefore, implicitly, the Public Keys used for signature verification must be pre-shared:

- The HES / KMS must be in possession of each Devices Signing Key Public Certificate
- The Device must be in possession of the HES / KMS Signing Key Public Certificate

4. Symmetric General Discussion

Several Symmetric Key operations are similar to those implemented to support other Meter types using the AT Master Solution:

- Authenticated Encryption - AES-GCM-128
- Key Transport - AES Key Wrap 128-bit
- Symmetric Key Length - 16 bytes

However there are some slight differences:

- DLMS / COSEM Associations between HHU (Installation, Maintenance and Certification Clients) and Device are established using HLS6

HLS6 is a simple Hash (SHA256) based authentication, it relies on both Client and Server being in possession of the same Symmetric Key:

- HLS6 Key Length - 16 bytes or 32 bytes

The Honeywell Companion Specification and KSM West Schnittstellen mention these Symmetric keys (some are not very well referenced):

- Master Key - KEK - Per Device
- Global Unicast Encryption Key - GUEK - Per Role, Per Device
- Global Authentication Key - GAK - Per Role, Per Device
- HLS6 Authentication Key - HLS_SECRET - Per Device
- Extensible Authentication Protocol PreShared Key - PSK - Per Device
- *(Dedicated Unicast Encryption Key - Volatile, created per association and not stored)*

---

new Security Suite 1 - HES (Client) Certificate Support

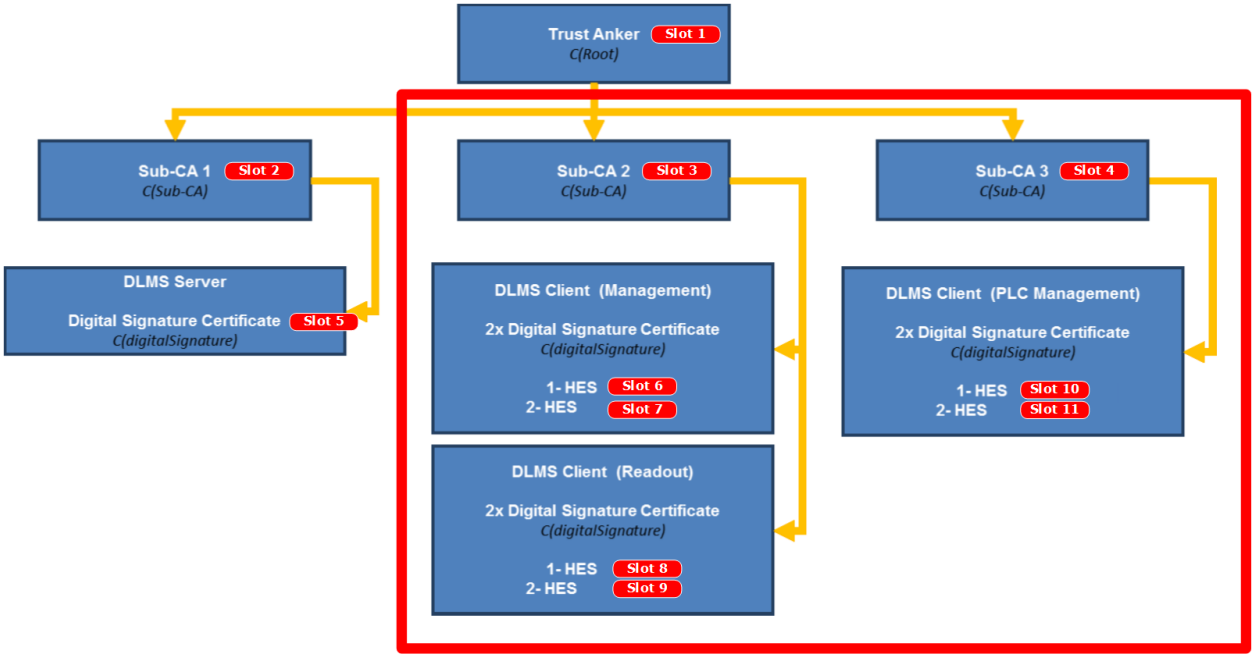
To Support Security Suite 1 Functions the Meter must be able to store several Certificates. These are broadly categorised into two:

- **Client Certificates** - In this context the Client is the HES, hence the Certificate contain Public Keys related to Private Keys 'owned' by the Central System (HES)
- **Server Certificates** - In this context the Server is the Meter, hence the Certificates contains the Public Keys related to Private Keys 'owned' by the Meter

These Certificates and their related Private Keys are used for Authentication between Meter and HES, hence Client Certificates are DLMS COSEM Client Specific. All Certificates are issued in a hierarchy containing a Root Certificate Authority and several Issuing Authorities (Sub-CA).

Certificates are usually issued with an intended finite lifetime, therefore must be replaced in the Meter with new Certificates over time. Due to the way the authentication mechanisms work, if a Certificate Update failed mid-installation, but after the existing certificate had already been over-written, the connecting System (usually the HES) would lose the ability to Authenticate with the Meter and hence would become locked-out. In order to mitigate this risk, the Meters are designed in a such a way that for each COSEM Client, it is possible to have two installed HES Certificates at a given time. This permits the parallel usage of both Old and New HES Certificates, giving a rollback capability.

The diagram below is taken from the Honeywell Companion Standard, modified to better represent the integration with the the Central System (HES, KMS, MDM). It shows the Certificates stored in the Meter and their hierarchical structure. The Certificates in the red box are those that we are concerned with when discussing HES Certificates and their Management:



Terminology

The Meter is capable of storing 11 Certificates. For simplicity, this design documentation refers to these Certificates as being stored in numbered 'Slots'. The mapping between 'Slot' and Certificate Usage is shown in the diagram above.

Slots are further designated to Primary and Backup categories:

- **Primary** - These are the Slots where it is expected the Newest HES Certificate will finally be installed.

- **Backup** - These are the Slots where the Old HES Certificate will be installed to provide rollback capability until the Certificate in a Primary Slot is confirmed working.

Slot Type	Slot ID
Primary	• Slot 6 • Slot 8 • Slot 10
Secondary	• Slot 7 • Slot 9 • Slot 11

new HES (Client) Certificate Import

To facilitate Security Suite 1 functionality and enable the manufacturing process to commence, the Siemens Solution must provide a COSEM Certificate for installation in the Meter for each of the Roles below:

- Management Client
- Readout Client
- PLC Management Client

To clarify subsequently used terminology, these certificates are called 'HES Client Certificates'.

The high level process surrounding the generation of the HES Client Certificates is something like:

1. Order an appropriate Key from Worldline (or generate one using the Worldline Mangement UI), one for each Role, per customer (ie 6 total), each key will be given an alias.
2. Import Keys to HSM
3. Use Command Line JSS Tool to generate a Certificate Signing Request
4. Process the CSR manually through the appropriate Sub-CA of the Siemens Solution PKI
5. Provide the HES Client Certificates to Honeywell / Kaifa for Manufacturing

In order for KMS to be able to create relationships between HES Certificates and Devices (required), and also to enable Lifecycle Management / Reporting (desired), the HES Certificates must be imported to the KMS.

The HES Certificates must be imported in a way allowing the following data to be assigned to / maintained against the Certificate to support further processes / activities:

- Lifecycle Parameters
- That it is a HES Certificate

- The Private Key Label related to the Certifcate
- Fingerprint of the Certificate
- The Role of Certificate
- The Type of Certificate
- The Usage of the Certificate
- The Status of the Certificate

new Solution Proposal (KMS): HES Client Certificate Import

Because there are relatively few HES Client Certificates, the import process will be manual, using the KMS UI Certificate Management.

The existing Certificate Storage Data model will be extended, and the existing UI extended correspondingly.

All extensions will be made ensuring backward compatibility with existing Certificate Records (ie Gateway certs).

User Interface Usage / Modifications

The KMS UI Certificate User Interface will be modified as such to make it less Device-centric, allowing HES Certificates to be stored and viewed appropriately, without confusion of terminology.

A full review of the refactoring of the Certificate UI in order to support import of HES Certificates (and other Certificates and new information required to support new processes can be found here: new Certificate Management - UI and Data Model Refactoring.

New elements specifically relevant to HES Certificate import / storage will be added / possible in the refactored UI:

Field	Type	Usage
Role	Field	To allow the formal assignment of a HES Certificate to a specific DLMS COSEM Role
KeyLabel	Parameter	To allow the definition of a Certificate → HSM KeyLabel relationship for HES Certificates
Fingerprint	Parameter	To allow the definition of a Certificate Fingerprint against a HES Certificate, to support HES Certificate / Meter Relationship Establishment

The required information about the HES certificate will then be stored as such:

Information	Method of Entry	Field	Value
Lifecycle Parameters	Manual (via Cert Template)	Issuer	Cert Issuer ID
That it is a HES Certificate	Manual (via dropdown)	Type	HES
Private Key Label	Manual (via freetext)	Parameter: KeyLabel	TIN_DLMS_SIG_MAN_WK
Certificate Fingerprint	Manual (via freetext)	Parameter: Fingerprint	The fingerprint extracted from the certificate
The Role of Certificate	Manual (via dropdown)	Role	Management || Readout || PLC_Management

		Entity ID	
The Type of Certificate	Manual (via dropdown)	Certificate Type	CRT_ECC
The Usage of the Certificate	Manual (via dropdown)	Sub-Type	Sign || KeyAgreement
The Status of the Certificate	Manual (via dropdown)	Status	Standard enumerated values

Limitations

Certificate / Issuer / Template relationships go:

- Certificate → Key Issuer (on Aussteller) → Certificate Template (on Zertifikatstyp)

Therefore to have a different 'Lifetime' defined for Certificates of a different Role, either Zertifikatstype needs to be Role specific (not great), or each Role is setup as a individual Key Issuer (not so bad).

new HES (Client) Certificate to Meter Relationship Creation

It is important for KMS to maintain a record of which of the HES Certificates are installed in each Meter as the Private Keys behind the certificates are used in HLS7 and Security Suite 1 operations.

These Certificates cannot be statically defined as they will be included in a renewal process, so at some point in the Solution's lifecycle there will be more than one Certificate for each role 'in use'.

The relationship builder process begins when devices are fist notified to the Central System, it is critical prior to initial communication with the Meters that the KMS is informed of which HES Certificates are installed in the Device.

Honeywell / Kaifa will provide that information in the Meter Shipment File. The Shipment File will contain several new elements in the Shipment File Header:

```
<Fingerprint_Root_CA>ad67ee1734fc5a024110cd51f172bf574c06aefc</Fingerprint_Root_CA>
<Fingerprint_Meter_Sub_CA>3f643d5d2597d1b07a41e6f81383765829ded87f</Fingerprint_Meter_Sub_CA>
<Fingerprint_HES_Sub_CA>ad67ee1734fc5a024110cd51f172bf574c06aefc</Fingerprint_HES_Sub_CA>
<Fingerprint_PLC_Sub_CA>ad67ee1734fc5a024110cd51f172bf574c06aefd</Fingerprint_PLC_Sub_CA>
<Fingerprint_HES_Certificate1_MC>ad67ee1734fc5a024110cd51f172bf574c06aefe</Fingerprint_HES_Certi
<Fingerprint_HES_Certificate2_MC>ad67ee1734fc5a024110cd51f172bf574c06aeff</Fingerprint_HES_Certi
<Fingerprint_HES_Certificate1_RC>ad67ee1734fc5a024110cd51f172bf574c06ae01</Fingerprint_HES_Certi
<Fingerprint_HES_Certificate2_RC>ad67ee1734fc5a024110cd51f172bf574c06ae02</Fingerprint_HES_Certi
<Fingerprint_HES_Certificate1_PLC>ad67ee1734fc5a024110cd51f172bf574c06ae03</Fingerprint_HES_Cert
<Fingerprint_HES_Certificate2_PLC>ad67ee1734fc5a024110cd51f172bf574c06ae04</Fingerprint_HES_Cert
```

These elements describe which HES certificates have been installed for every meter in the File. Certificates are defined in terms of 'fingerprints' ⚠️ detail of the HASH functions used etc to follow ⚠️.

new Solution Proposal (KMS): HES Cert to Meter Relationship Maintenance

Data Model

In order to store the HES Certificate → Meter relationships a new table is required in the KMS DB. This table is designed to:

- Allow for tracability of relationships as they change over time
- Minimise data duplication
- Allow future extensibility

New Table: kmsEntityToCertRelationships

Containing, in addition to standard EIP columns:

Column Name	Description	Example value
entityId	The Serial number of the Meter	KFM1000100000001
certId	The Ref ID of the HES Certificate record	3
type	The type of relationship between EntityA and EntityB	deviceToHesCert
effectiveStartDate	The start date / time of the relationship	26-APR-18 03.37.08.000000000 PM
effectiveEndDate	The end date / time of the relationship	NULL || 26-AUG-18 06.54.13.000000000 PM
status	The status of the relationship	Active || Pending Renewal || Renewal Requested || Replaced || Unknown
batchId	The batch ID of the Relationship / Key load, ie per Shipment File Load	1

REST Service Relationship Management

The KMS must provide a REST interface to allow automatic processes to create the relationships between HES Certificate and Meters. The Service definition is described here: new KMS REST Service Definition: crtHesToMeterRel

User Interface Relationship Management

Via the KMS UI, it should be possible to:

- View existing relationships between a Meter and HES Certificates
- Create New relationships between Meter and HES Certificates
- Update the Status of existing Relationships

This will require a new section to the KMS UI and new actions within that section:

View Existing Relationships

A User should be able to see a tabular representation, showing:

- Meter Serial Number
- HES Certificate's Ref ID
- HES Certificate's Role
- Relationship Status
- Relationship Start Date
- Relationship End Date

The User should be able to search / filter on all fields.

Create New Relationships

A User should be able to create a new Meter → HES Certificate relationship via a form made available through an 'Action' button.

This form should allow the User to specify, or obtain via inference:

- The Serial Number of the Meter - Freetext
- The Ref ID of the Certificate to relate to

The remainder of the fields should be set automatically.

The search window for finding a Certificate Ref ID should allow for filtering to find only certificates matching:

- GerateSeriannummer = As required
- GerateType = HES
- Status = Active
- ZertifikatTyp = CRT_ECC
- Verbrauch = Sign || Key Agreement
- Role = Management || Readout || PLC_Management

Update Existing Relationships

A User should be able to update the Status of an existing Relationship in some cases. Permitted Status transitions, via UI should be limited to:

- Unknown → Active

new KMS REST Service Definition: crtHesToMeterRel

The REST service request allows an array of, or individual HES Certificate to Meter relationship to be submitted to the KMS. The Service allows for two modes of Certificate Identification:

- **Fingerprint** - For use by the Shipment File Loader - Uses the 'Fingerprint' parameter to identify the relevant Certificate to include in the relationship record
- **Cert Ref ID** - For use by the KMS UI - Use the RefID of the Certificate directly from the KMS_CERT table to identify the relevant Certificate to include in the relationship record

Request Formats

Request

```
{
  "batchId" : "1",
  "relations" : [{
    "deviceId" : "ISK1050768076703",
    "certIdType" : "FINGERPRINT"
    "crtId" : "1234567890abcdef123456"
  }, {
    "deviceId" : "ISK1050768076704",
    "certIdType" : "REF_ID"
    "crtId" : "3"
  }, {
    "deviceId" : "ISK1050768076705",
    "certIdType" : "REF_ID"
    "crtId" : "4"
  }, {
    "deviceId" : "ISK1050768076706",
    "certIdType" : "REF_ID"
    "crtId" : "5"
  }]
}
```

Element Descriptions

Request element	Required	Description	Type	Value / Example
batchId	Mandatory	The batch ID of the Shipment File Loader process (for enabling rollback)	Integer	1
relations	Mandatory	An array of Meter to HES Cert relationships to create	Array	
deviceId	Mandatory	Meter COSEM Logical Device Name	String	ISK1050768076703
certIdType	Mandatory	The mechanism to be used to identify the Certificate	Enumerated	FINGERPRINT || REF_ID
crtId	Mandatory	The identifier to use in conjunction with the certIdType	String	1234567890abcdef123456 || 3

Error Handling

Prior to creation of any relationship entry, the following conditions should be validated for the Certificate included in the request:

- The certificate must be able to be located using the criteria in the request (fingerprint or refId)
- GerateType / Type = HES
- Status = Active

If any of these conditions are not true, then the relationship should not be created.

Existing Relationship Handling

When a request to create a Relationship is received, before creating the new record a check should be performed to verify if the Relationship is actually new.

If there is an existing Relationship where the conditions below are true, then no new record should be created and the existing Relationship should have it's relationshipStatus reset to 'Active':

- relationshipType = deviceToHesCert
- entityA = The same Meter Serial Number as that in the request
- entityB = The same Cert Reference as that derived from the request
- relationship End Date = NULL

When a new relationship record is inserted any prior relationships should be ended.

A relationship should be identified for uniqueness via a concatenation of:

- Device ID
- Certificate ID
- Certificate Role
- Relationship Type

Where an existing relationship is found, the new entry should be created and the old marked:

- Status = Replaced
- Relationship End Date = The date and time of the replacement

Response Formats

Success Response

```
{
    "globalResultCode": "GLOBAL_SUCCESS",
    "relations": [{
      "resultCode": "OK",
      "deviceId" : "ISK1050768076703",
      "crtId" : "1234567890abcdef123456"
    }, {
      "resultCode": "OK",
      "deviceId" : "ISK1050768076704",
      "crtId" : "3"
    }]
}
```

Partial Success Response

```
{
    "globalResultCode": "PARTIAL_SUCCESS",
    "relations": [{
      "resultCode": "OK",
      "deviceId" : "ISK1050768076703",
      "crtId" : "1234567890abcdef123456"
    }, {
      "resultCode": "CERT_NOT_FOUND",
      "deviceId" : "ISK1050768076704",
      "crtId" : "3"
```

```
    }, {
      "resultCode": "INCORRECT_CERT_TYPE",
      "deviceId" : "ISK1050768076705",
      "crtId" : "4"
    }, {
      "resultCode": "CERT_NOT_ACTIVE",
      "deviceId" : "ISK1050768076706",
      "crtId" : "5"
    }]
}
```

Element Descriptions

Request element	Required	Description	Type	Value / Example
globalResultCode	Mandatory	A global success code detailing the overall success of the request	Enumerated	GLOBAL_SUCCESS || PARTIAL_SUCCESS || PARTIAL_SUCCESS_ROLLED_BACK
relations	Mandatory	An array of Meter to HES Cert relationships to create	Array	
resultCode	Mandatory	A per relationship success status code	Enumerated	OK || CERT_NOT_FOUND || INCORRECT_CERT_TYPE || CERT_NOT_ACTIVE
deviceId	Mandatory	Meter COSEM Logical Device Name for the Meter in the relationship	String	ISK1050768076703
crtId	Mandatory	The identifier to use in conjunction with the certIdType	String	1234567890abcdef123456 || 3

Failure Response

```
{
  "errorCode": "XXX-99999999",
  "errorDescription": "Something went wrong"
}
```

Element Descriptions

Request element	Required	Description	Type	Value / Example
errorCode	Mandatory	An enumerated error code	Enumerated	`XXX-99999999`
errorDescription	Mandatory	A human readable description of the error	String	`Something went wrong`

new KMS REST SERVICE Definition: deleteRelationships

The REST service request allows the Shipment File Loader to delete any Relationships that have been created as part of the processing of an individual Shipment File. This is to allow rollback to pre-import state of Relationship records should there be a problem with the processing of the Shipment File requiring that the Shipment File is re-worked and re-submitted.

This Service is intended to support the Shipment File Loader process, therefore the expectation is that it is dealing with Meters that are wholly new to KMS, hence there are no prior Relationships to re-activate.

Request Formats

Request

```
{
  "batchId" : "1"
}
```

Element descriptions

Request element	Required	Description	Type	Value / Example
batchId	Mandatory	The batch ID of the Shipment File Loader process	Integer	1

Response Formats

Success Response

```
{
   "resultCode": "OK"
}
```

Element Descriptions

Request element	Required	Description	Type	Value / Example
resultCode	Mandatory	Result code indicating success	String	OK

Handled Failure Response

```
{
   "resultCode": "BATCH_NOT_FOUND",
   "resultDescription": "No Relationships relating to the Batch were found"
}
```

Element Descriptions

Request element	Required	Description	Type	Value / Example
resultCode	Mandatory	Result code indicating a problem	String	BATCH_NOT_FOUND
resultDescription	Mandatory	Human readable description	String	No Relationships relating to the Batch were found

Expected Failures

resultCode	resultDescription
BATCH_NOT_FOUND	No Relationships relating to the Batch were found

Unhandled Failure Response

```
{
  "errorCode": "XXX-99999999",
  "errorDescription": "Something went wrong"
}
```

Element Descriptions

Request element	Required	Description	Type	Value / Example
errorCode	Mandatory	An enumerated error code	Enumerated	`XXX-99999999`
errorDescription	Mandatory	A human readable description of the error	String	`Something went wrong`

new Solution Proposal (SFL): Extension

Honeywell / Kaifa will provide the information of which HES Certificates are installed in which Meters via the Meter Shipment File (for Symmetric Keys). The Shipment File will contain several new elements in the Shipment File Header:

```
<Fingerprint_Root_CA>ad67ee1734fc5a024110cd51f172bf574c06aefc</Fingerprint_Root_CA>
<Fingerprint_Meter_Sub_CA>3f643d5d2597d1b07a41e6f81383765829ded87f</Fingerprint_Meter_Sub_CA>
<Fingerprint_HES_Sub_CA>ad67ee1734fc5a024110cd51f172bf574c06aefc</Fingerprint_HES_Sub_CA>
<Fingerprint_PLC_Sub_CA>ad67ee1734fc5a024110cd51f172bf574c06aefd</Fingerprint_PLC_Sub_CA>
<Fingerprint_HES_Certificate1_MC>ad67ee1734fc5a024110cd51f172bf574c06aefe</Fingerprint_HES_Certi
<Fingerprint_HES_Certificate2_MC>ad67ee1734fc5a024110cd51f172bf574c06aeff</Fingerprint_HES_Certi
<Fingerprint_HES_Certificate1_RC>ad67ee1734fc5a024110cd51f172bf574c06ae01</Fingerprint_HES_Certi
<Fingerprint_HES_Certificate2_RC>ad67ee1734fc5a024110cd51f172bf574c06ae02</Fingerprint_HES_Certi
<Fingerprint_HES_Certificate1_PLC>ad67ee1734fc5a024110cd51f172bf574c06ae03</Fingerprint_HES_Cert
<Fingerprint_HES_Certificate2_PLC>ad67ee1734fc5a024110cd51f172bf574c06ae04</Fingerprint_HES_Cert
```

These records define, in terms of 'Certificate Fingerprint' which HES Certificates have been installed for all Meters included in the Shipment File.

The Shipment File Loader should be enhanced to that the information contained about the HES Certificate → Meter realtionships can be provided to the KMS to allow it to build relationships using the REST Service described **here**.

The relevant elements are:

Element	Description
Fingerprint_HES_Certificate1_MC	Certificate fingerprint of the certificate installed for the Management Client
Fingerprint_HES_Certificate1_RC	Certificate fingerprint of the certificate installed for the Readout Client
Fingerprint_HES_Certificate1_PLC	Certificate fingerprint of the certificate installed for the PLC Management Client

Sequencing

1. The process of creating the HES Certificate → Meter relationships should happen before the Symmetric Keys contained within the file are imported.
2. If the HES Certificate → Meter relationship process should fail in any way, then the import of the Symmetric Keys should not be attempted.

Failure Handling

It should be possible to configure, via a System Console parameter, whether a processing failure in the request should initiate a Rollback of the newly created records in the Relationship table.

If rollback is enabled, then should the Shipment File Import Loading process fail for any reason, either during Key Import or Relationship Creation, a request should be made to the REST Service to remove the new Relationships as defined here: new KMS REST SERVICE Definition: deleteRelationships

Supporting Files

Example Honeywell / Kaifa Shipment File:

DSMR_vertraulich.xml

And associated Honeywell / Kaifa Shipment File Schema:

DSMR_vertraulich.xsd

xenc-schema.xsd

new HES (Client) Certificate Lifecycle Management

HES Certificates should be included within Lifecycle Management for automatic semi-automatic renewal in all devices.

The overall Lifecycle Management of the HES Certificates should comprise of:

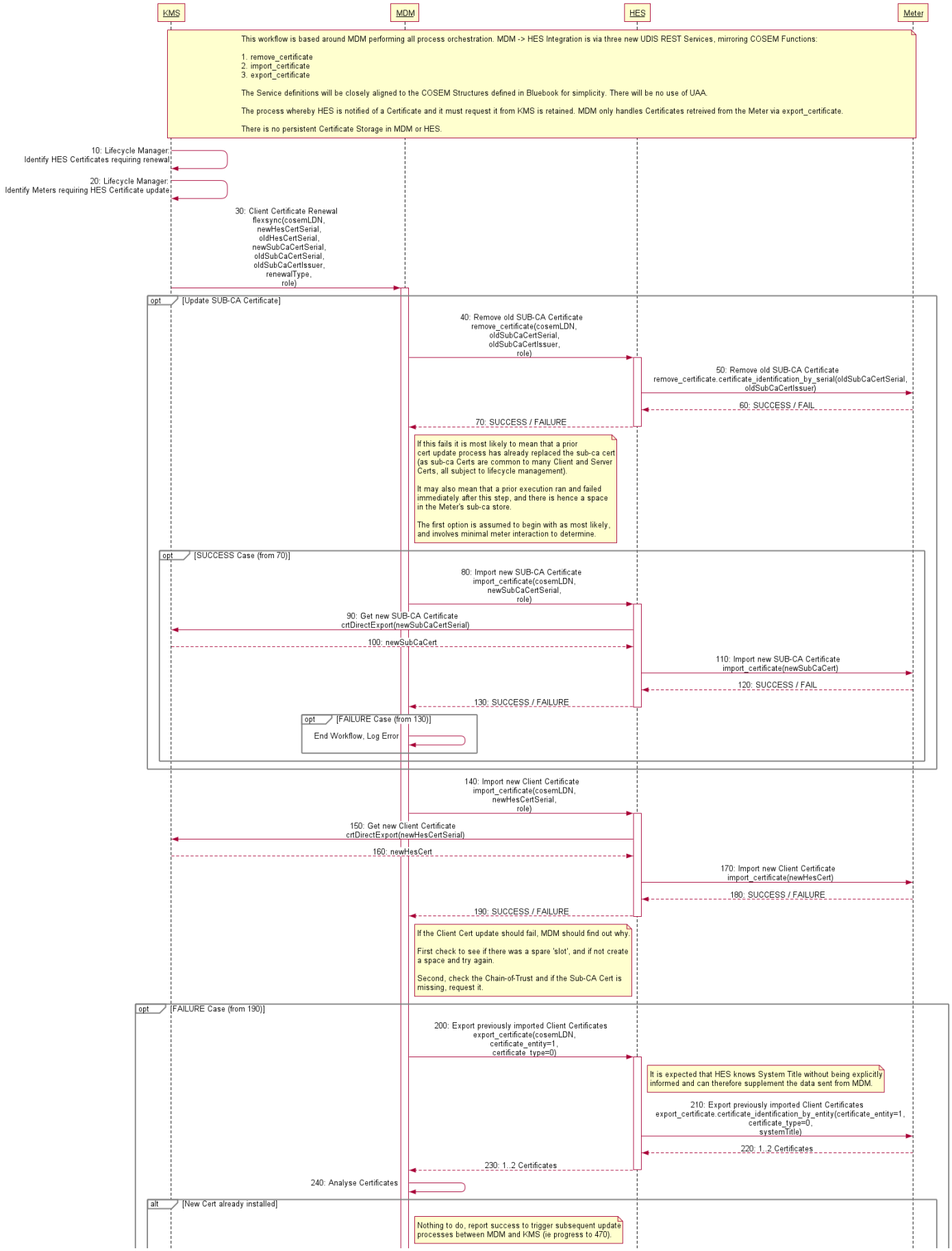
- Notification being raised when a HES Certificate is nearing replacement age
- Manual creation and import to KMS of New HES Certificates
- Manual creation and import to KMS of New Issuing Authority Certificates (if required)
- Automatic identification of which Meters must have new HES Certificates applied to them
- Automatic initiation of the process to renew the Issuing Authorty Certificates in the Meter
- Automatic initiation of the process to renew the HES Certificates in the Meter

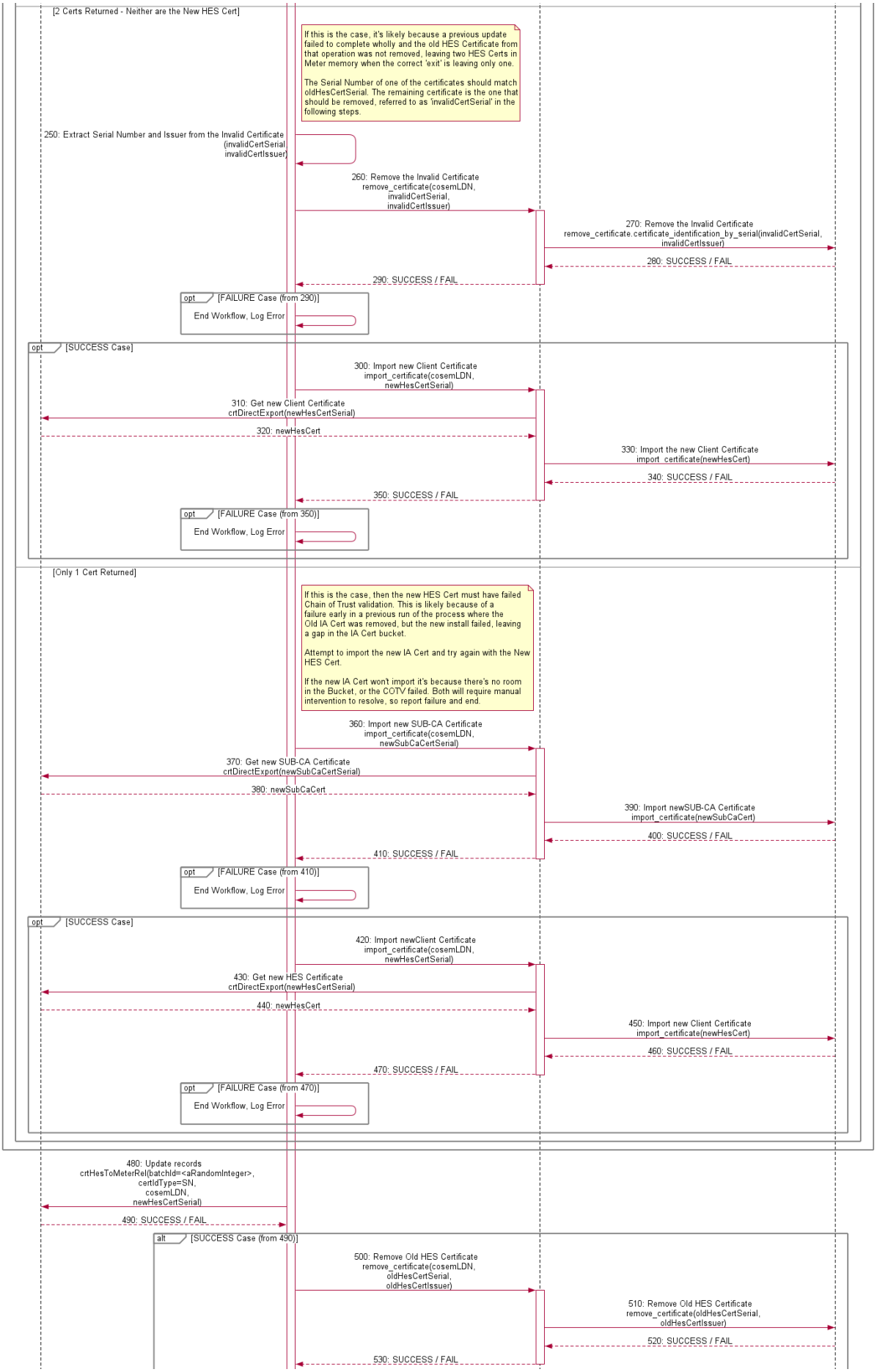
Only HES Certificates will be actively considered for LifeCycle Management, Issuing Authority Certificates will be replaced by implication.

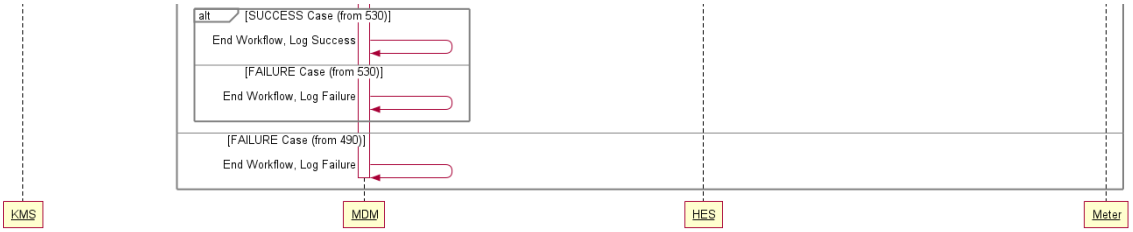
LifeCycle Management Process Overview

A summary is given beneath the diagram, for detailed information see the Solution Proposal sections for individual components:

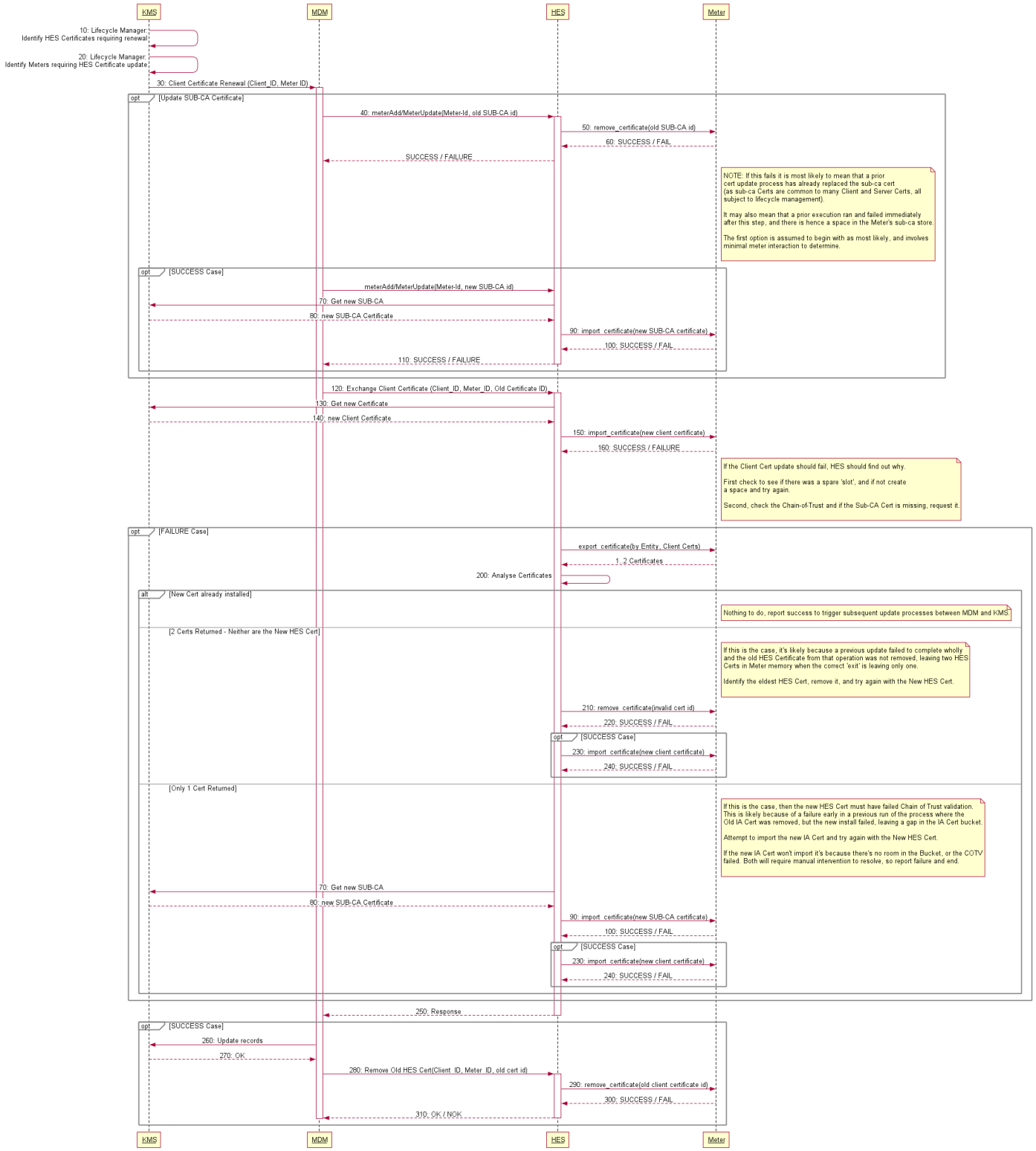
Client Certificate Exchange

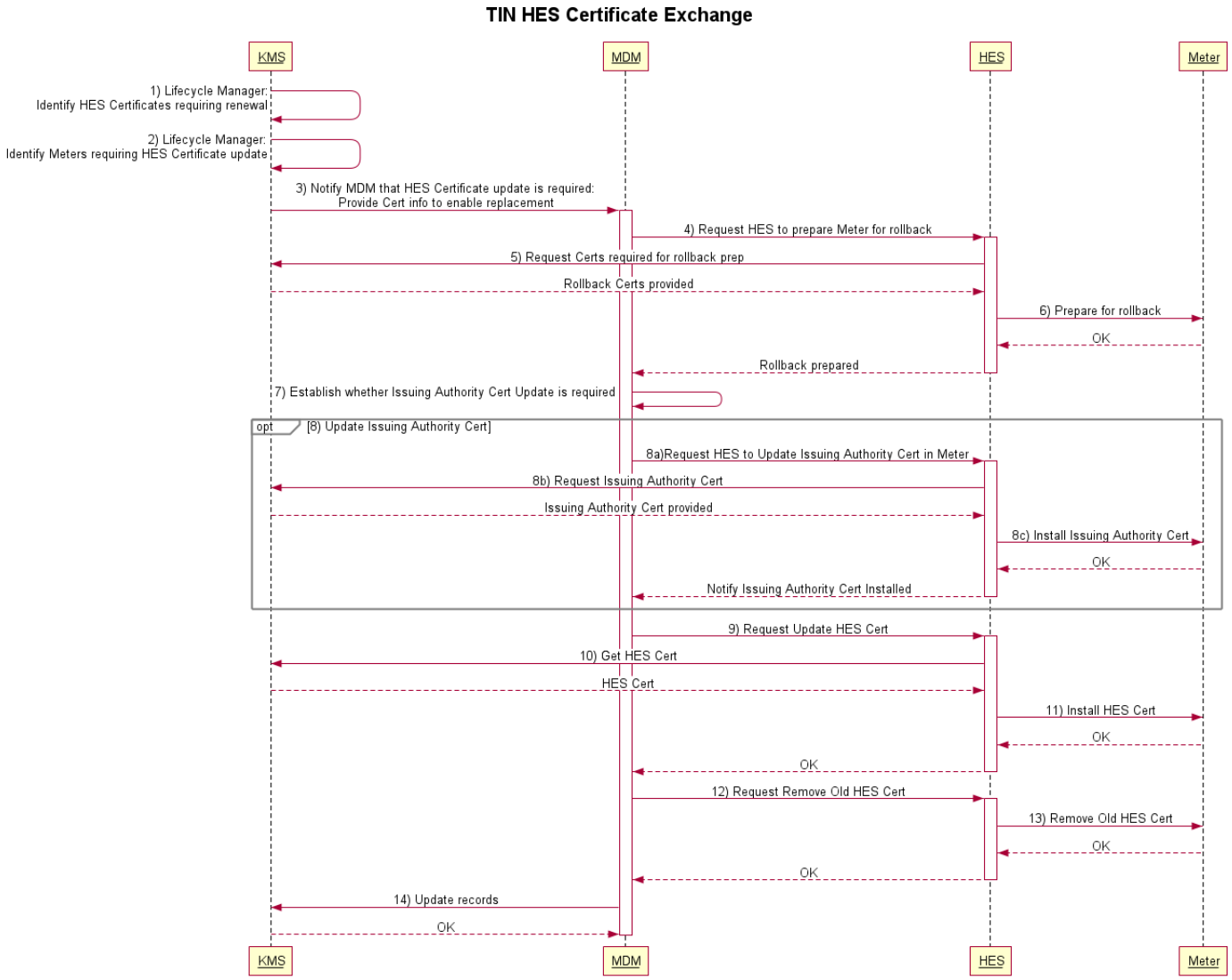



[2 Certs Returned - Neither are the New HES Cert]
If this is the case, it's likely because a previous update failed to complete wholly and the old HES Certificate from that operation was not removed, leaving two HES Certs in Meter memory when the correct 'exit' is leaving only one.
The Serial Number of one of the certificates should match oldHesCertSerial. The remaining certificate is the one that should be removed, referred to as 'invalidCertSerial' in the following steps.
250: Extract Serial Number and Issuer from the Invalid Certificate (invalidCertSerial invalidCertIssuer)
260: Remove the Invalid Certificate remove_certificate(cosemLDN, invalidCertSerial, invalidCertIssuer)
270: Remove the Invalid Certificate remove_certificate.certificate_identification_by_serial(invalidCertSerial, invalidCertIssuer)
280: SUCCESS / FAIL
290: SUCCESS / FAIL
opt [FAILURE Case (from 290)]
End Workflow, Log Error
opt [SUCCESS Case]
300: Import new Client Certificate import_certificate(cosemLDN, newHesCertSerial)
310: Get new Client Certificate crtDirectExport(newHesCertSerial)
320: newHesCert
330: Import the new Client Certificate import_certificate(newHesCert)
340: SUCCESS / FAIL
350: SUCCESS / FAIL
opt [FAILURE Case (from 350)]
End Workflow, Log Error
[Only 1 Cert Returned]
If this is the case, then the new HES Cert must have failed Chain of Trust validation. This is likely because of a failure early in a previous run of the process where the Old IA Cert was removed, but the new install failed, leaving a gap in the IA Cert bucket.
Attempt to import the new IA Cert and try again with the New HES Cert.
If the new IA Cert won't import it's because there's no room in the Bucket, or the COTV failed. Both will require manual intervention to resolve, so report failure and end.
360: Import new SUB-CA Certificate import_certificate(cosemLDN, newSubCaCertSerial)
370: Get new SUB-CA Certificate crtDirectExport(newSubCaCertSerial)
380: newSubCaCert
390: Import newSUB-CA Certificate import_certificate(newSubCaCert)
400: SUCCESS / FAIL
410: SUCCESS / FAIL
opt [FAILURE Case (from 410)]
End Workflow, Log Error
opt [SUCCESS Case]
420: Import newClient Certificate import_certificate(cosemLDN, newHesCertSerial)
430: Get new HES Certificate crtDirectExport(newHesCertSerial)
440: newHesCert
450: Import new Client Certificate import_certificate(newHesCert)
460: SUCCESS / FAIL
470: SUCCESS / FAIL
opt [FAILURE Case (from 470)]
End Workflow, Log Error
480: Update records crtHesToMeterRel(batchId=<aRandomInteger>, certIdType=SN, cosemLDN, newHesCertSerial)
490: SUCCESS / FAIL
alt [SUCCESS Case (from 490)]
500: Remove Old HES Certificate remove_certificate(cosemLDN, oldHesCertSerial, oldHesCertIssuer)
510: Remove Old HES Certificate remove_certificate(oldHesCertSerial, oldHesCertIssuer)
520: SUCCESS / FAIL
530: SUCCESS / FAIL


alt
[SUCCESS Case (from 530)]
End Workflow, Log Success
[FAILURE Case (from 530)]
End Workflow, Log Failure
[FAILURE Case (from 490)]
End Workflow, Log Failure
KMS
MDM
HES
Meter

Client Certificate Exchange v2





Step descriptions:

Step	Description	Reference
1	A process identifies that lifetime threshold parameters have been exceeded for a HES Certificate and marks the record as such.	
2	A process identifies which Meters require HES certificate updates because the installed HES certificate requires replacement and marks records as such.	
3	KMS notifies MDM via FlexSync that a HES Cert update is required. KMS provides the replacement and existing Cert information (not the certificate strings) to MDM to allow subsequent derivations.	
4	MDM requests HES to install the Old HES Certificate in the Backup Slot of the Meter. Theoretically this ensures that the Old Certificate is installed in both Slots, therefore it is not possible to become locked out should a subsequent update fail.	
5	HES must make a REST Service request to KMS to get the Old HES Certificate.	
6	HES installs the Old HES Certificate in the Backup Slot.	

7	MDM checks the data provided to it by KMS and decides whether the Issuing Authority Certificate and the HES Certificate needs to be updated, or just the HES Certificate.	
8	This section is optional, based on the outcome of the check performed in the previous step.	
8a	MDM makes a request to HES to install the Issuing Authority Certificate in the Meter.	
8b	HES must make a REST Service request to KMS to get the Issuing Authority Certificate.	
8c	HES installs the relevant Certificate in the Primary Slot of the Meter.	
9	MDM makes a request to HES to install the New HES Certificate in the Meter.	
10	HES must make a REST Service request to KMS to get the New HES Certificate.	
11	HES installs the relevant Certificate in the Primary Slot of the Meter	
12	MDM makes a request to HES to remove the Old HES Certificate	
13	HES makes a request to the Meter to remove the Certificate in the Backup Slot	
14	MDM makes a REST Request to KMS to update HES Cert to Meter Relationships; either creating a New Relationship (happy path) or re-activating the Old Relationship.	

The LifeCycle Management Process takes into account several important areas:

Area	**Description**
Permitted Integration routes	KMS should not interface directly to HES, only to MDM. HES is permitted to interface directly to MDM or KMS. MDM is permitted to interface directly to HES or KMS.
What can be achieved within MDM via the UAA Adapter (Product)	The existing Meter-centric interface between MDM and HES is via the productised UAA Adapter. Customisations / Extensions to the UAA Adapter would introduce potential support issues and should be avoided.
Workable ways of passing data between Systems	Existing replacement processes for passing Key Material to HES involve the use of multiple MDM parameters with data split across parameters due to length limitations. However to pass the raw Certificate data this way would involve use of approximately 100 additional parameters, this is not workable.
Protecting against 'lock out'	The Meter permits the installation of two HES Certificates per Role, this allows the Old and New Certificates to be active in the Meter in parallel. This ensures that if there is an issue with the update to the New Certificate, the Old Certificate can still be used. The Renewal process should ensure this capability is adopted.
Failure Diagnosis	As far as possible it should be possible to report on information held within MDM to get an overview of where in the replacement process a failure occurs. This may be a compromise against other factors as described in this table.

new Solution Proposal (KMS): HES Certificate Lifecycle Management

Identifying Required Updates

KMS Lifecycle Management shall include a process which identifies which Meters require a HES Certificate to be updated in. This is is a two step process:

Step 1 - Monitor HES Certificates for nearing expiry

The process will evaluate the LifeCycle Parameters of HES Certificates from the Certificate record and linked Certificate Template. Where the lifetime thresholds specified are exceeded, the Certificate will be marked with Status 'Replacement Required'.

The Certificates to include in this monitoring will be filtered by:

- Type = HES
- Status != Retired || Replacement Required

Alternatively the Status of a HES Certificate may be set to 'Replacement Required' manually to trigger a pre-lifetime-threshold replacement.

> For an overview of expectated HES Certificate 'Status' transitions, see the section towards the end of this page.

Step 2 - Mark individual HES Certificate to Meter Relationships involving the HES Certificate requiring replacement

The process will evaluate HES Certificate to Meter Relationships that involve a HES Certificate with a status of 'Replacement Required'. Any Relationships matching the criteria below will have their Relationship Status updated to 'Pending Replacement':

- Cert ID = A HES Certificate with status of 'Replacement Required'
- Relationship Status = 'Active'
- Relationship Type = 'deviceToHesCert'

Replacement Initiation

> ⚠ **Important Control:**
> Only a single HES Certificate Replacement activity should be underway for a specific individual meter at any given time.
>
> HES Certificate Replacement activities should not be undertaken at the same time as any other Key renewal operation in the Meter.

KMS Lifecyle Management shall include a process that then processes each Relationship with a status of 'Pending Replacement' and submits a hesCertReplacement request to MDM using the interface described here: new MDM FlexSync Definition: MDM_I_METER_HES_CERT_REPLACE

The MDM Interface requires KMS to provide the inputs below:

Input	Description	Derivation
crtExchange.exchangeType	This value indicates the action that should be undertaken, install a Cert, remove a Cert etc.	For KMS initiate Lifecycle Management this should always be: • instOldHes
crtExchange.crtRole	This is the Role of the HES Certificate requiring replacement.	From the Relationship being processed, the current HES Certificate can be located. The crtRole is taken directly from the Role associated with that Cert.
crtExchange. oldCrtFingerprint	This is the Fingerprint value of the HES Certificate currently installed in the Meter.	From the Relationship being processed, the current HES Certificate can be located. The oldCrtFingerprint is taken directly from the Fingerprint Parameter associated with that Cert.
crtExchange. newCrtFingerprint	This is the Fingerprint value of the HES Certificate to be installed in the Meter.	From the Relationship being processed, the current HES Certificate can be located, this will have a Status of 'Active'. From this record the parameters below can be extracted: • Type • Sub Type • Certificate Type • Issuer • Role And used to find the Certificate that matches those parameters, but that has Status = New. The newCrtFingerprint is taken directly from the Fingerprint Parameter associated with the identified Cert.
crt.Exchange. oldCaCrtSubKeyId	This is the Subject Key Identifier of the Issuing Authority Certificate related to the currently installed HES Certificate in the Meter.	Logic to identify a Certificate record is the same as for oldCrtFingerprint. Once the record is located the value for oldCaCrtSubKeyId is taken directly from the Authority Key Identifier field of that record.
crt.Exchange. newCaCrtSubKeyId	This is the Subject Key Identifier of the Issuing	Logic to identify a Certificate record is the same as for new CrtFingerprint. Once the record is located the value for newCaCrtSubKeyId is taken directly from the Authority Key Identifier field of that record.

	Authority Certificate related to the to be installed HES Certificate in the Meter.	

To indicate that a HES Certificate Renewal activity is underway for a specific Relationship, once the hesCertReplacement request is submitted to MDM, the Relationship Status will be set to 'Replacement Requested'. The MDM Workflow will notify KMS of a successful Replacement via the Request to create a New Relationship. Existing Relationship handling will ensure the current Relationship's status is updated appropriately.

Timeout Handling

To enable manual investigation of HES Replacement Operations where the status of the Replacement is unknown, KMS LifeCycle Management shall include a process that identifies Relationships where the criteria below are true, and updates their Relationship Status to 'Active':

- Relationship Status = 'Replacement Requested'
- Last Update Date = Further in the past than a Threshold to be defined via System Console

Timeout logic, in conjunction with the 'Existing Relationship' handling in the KMS REST Service and MDM Workflow error handling, ensures that were the LifeCycle Management process fails leaving the Solution in a non-perfect state, re-tries will be made to 'tidy-up'.

> For an overview of expected HES Certificate to Meter Relationship 'Status' transitions, see the section towards the end of this page.

KMS REST Service Support

ATM-1 Key Renewal processes involve providing the renewal Keys to MDM via FlexSync Parameters. However this approach is not feasible for Certifcates as the number of MDM Parameters that are required to store a Certificate are too high. Therefore, rather than having the Certificates passed to MDM only a reference will be passed. This reference will be forwarded to HES and HES must actively request Certificates from KMS.

Because the HES Certificates are non-device specific, the device specific KMS Certificate Export Services are not appropriate for supporting this process. An additional KMS REST Service will be added to provide a mechanism to allow the HES to more simply request HES Certificates based on the information handed to it by KMS LifeCycle Management. This new Service is defined here: new KMS REST Service Definition: crtDirectExport

LifeCycle Management UI Support

It must be possible to exercise a degree of manual control over HES Certificate Lifecycle Management, this should be achieved via being able to perform the HES Certificate Status transitions defined below through the KMS UI.

HES Certificate & HES Certificate to Meter Status Transition Logic

HES Certificate Status Process Logic

The Status of HES Certificates will be managed by a combination of manual and automatic steps. The status of the HES Certificates is used by Lifecycle Management Processes to monitor and manage HES Certificate Renewal operations in the Meter.

Status Transition	Description	Required Restrictions (via Business Process)
... → New	When a new HES Certificate is imported, its status should be set to 'New'. The 'New' status of the Certificate indicates that it is the Certificate that should be used in any future HES Certificate Renewal operations.	For each 'Role' there should be only a single HES Cert with 'New' status. If at any time (during new HES certificate import) there is either no HES Certificate for the Role with status 'New' or there are multiple HES Certificates for the Role with status 'New', then operations of HES Certificate Renewal in the Meter will fail. Once a singular HES Certificate for the Role is re-established, renewal operations will continue successfully.
New → Valid	After the import of a new HES Certificate, the HES Certificates that previously had their status set as 'New' should have their status set to 'Active'. This ensures they are still usable, but will no longer be used in future HES Certificate Renewal operations.	
New → Replacement Required	When a HES Certificate is nearing the time when it should be replaced and updated in Meters, the Status should be set to 'Replacement Required'. Setting the status of a HES Certificate to 'Replacement Required' causes all Meter to HES Certificate Relationships involving the Certificate to be marked as requiring replacement via Lifecycle Management.	
Valid → Replacement Required	When a HES Certificate is nearing the time when it should be replaced and updated in Meters, the Status should be set to 'Replacement Required'. Setting the status of a HES Certificate to 'Replacement Required' causes all Meter to HES Certificate Relationships involving the Certificate to be marked as requiring replacement via Lifecycle Management.	
Replacement Required → Retired	When all HES Certificate has reached the end of its life and all neccessary replacements have been completed in the relevant Meters, the HES Certificate can be marked as 'Retired'. Setting this status removes the Certifcate from inclusion in Lifecycle evaluations.	

HES Certificate to Meter Relationship Status Process Logic

The Status of HES Certificate to Meter Relationship records will be managed automatically and is used to track the need to replace a HES Certificate in the Meter and to track the process of doing so.

Status Transition	Description	Restrictions (via Automatic Process)
... → Active	When a New HES Certificate to Meter relationship is created, its status will be set to 'Active'.	Automatic process logic ensures that for each Role, per Meter, only a single HES Certificate to Meter relationship has a status of 'Active'.
Active → Replacement Required	When Lifecycle Management identifies a HES Certificate has exceeded its lifetime thresholds, existing relationships involving that HES Certificate will be marked with status 'Replacement Required'.	
Replacement Required → Replacement Requested	When Lifecycle Management initiates a HES Certificate Replacement operation in the Meter, the status of the existing relationship will be set to 'Replacement Requested'.	
Replacement Requested → Active	If a HES Certificate Replacement operation fails or experiences a time-out, the existing Relationship will be reset to 'Active' so that the Lifecycle management process will begin again (ie retry on the next lifecycle management cycle).	
Replacement Requested → Replaced	When a HES Certificate Replacement operation succeeds, the creation of the New Relationship record will automatically end the existing relationship, setting its status to 'Replaced'	

new KMS REST Service Definition: crtDirectExport

The KMS REST Service described below provides a Certificate-Centric export mechanism as opposed to the existing Device-Centric Certificate Export mechanisms which are not suitable for purpose for HES Certificate exports.

The Service Response should yield a single Certificate. If multiple Certificates are located, an error will be thrown.

Request Formats

Request

```
{
  "crtIdType" : "Fingerprint",
  "crtId" : "45454ab5454512e542187",
  "crtType" : "HES",
  "crtSubType" : "Sign"
}
```

Element descriptions:

Request element	Required	Description	Type	Value / Example
crtIdType	Mandatory	The type of Identifier to be used to locate the correct certificate	String	Fingerprint || SubjectKeyId
crtId	Mandatory	The ID of the certificate	String	45454ab5454512e542187
crtType	Optional	The Type of the Certificate being requested for export. Used for additional piece-of-mind, if populated the value should be used as a double check that the Certificate identified by crtIdType and crtId is definitely correct.	Enumerated string	HES
crtSubType	Optional	The Sub-Type of the Certificate being requested for export. Used for additional piece-of-mind, if populated the value should be used as a double check that the Certificate identified by crtIdType and crtId is definitely correct.	Enumerated string	Sign

Response Formats

Success Response


```
{
  "resultCode": "OK",
  "crtIdType" : "1",
  "crtId" : "45454ab5454512e542187",
  "crtType" : "HES",
  "crtSubType" : "Sign"
  "crtString" : "-----BEGIN CERTIFICATE-----MIIFpzCCA4+gAwIBAgIIRcRVSEnS6CEwDQYJKoZIhvcNAQELBQAwI
}
```


Failure Response


```
{
  "resultCode": "CERT_NOT_FOUND",
  "crtIdType" : "1",
  "crtId" : "45454ab5454512e542187",
  "crtType" : "HES",
  "crtSubType" : "Sign"
}
```


Element descriptions:

Request element	Required	Description	Type	Value / Example
resultCode	Mandatory	Result of the Export Request	Enumerated	OK || CERT_NOT_FOUND
crtIdType	Mandatory	Passthrough from the request	String	ISK1050768076703

crtId	Mandatory	Passthrough from the request	String	1234567890abcdef123456 || 3
crtType	Mandatory	If Success: The Type of the Certificate taken from the Certificate record in KMS DB If Failure: Passthrough from the request	String	HES
crtSubType	Mandatory	If Success: The Sub-Type of the Certificate taken from the Certificate record in KMS DB If Failure: Passthrough from the request	String	Sign
crtString	Optional	The Certificate String as stored in the KMS DB	String	-----BEGIN CERTIFICATE----- MIIFpzCCA4+gAwIBAgIIRcR...<removed some of the string for brevity>... gO85nqjx+jkRdXHmzc=-----END CERTIFICATE-----

Failure Result Codes:

Code	Description
CERT_NOT_FOUND	The criteria specified in the Request did not yield a result
MULTIPLE_CERTS_FOUND	The criteria specified did not result in a unique certificate being located

new Solution Proposal (MDM): HES Certificate Replacement Support

The MDM will receive a notification from the KMS that a HES Certificate Update is required using the Interface defined here: new MDM FlexSync Definition: MDM_I_METER_HES_CERT_REPLACE

The receipt of a notification to that interface will trigger a Workflow responsible for the Orchestration of the successful Replacement of a HES Certificate in the Meter. The MDM Workflow will:

- Make best endeavours to ensure a rollback is possible should a failure occur at any stage
- Replace Issuing Authority Certificates in the Meter if required for Chain-of-Trust verification of the New HES Certificate
- Ensure that KMS HES Certificate to Meter Relationships are updated appropriately, either driving retry or indicating successful replacement.
- Provide reporting capabilities to identify where the end state is workable non-perfect.

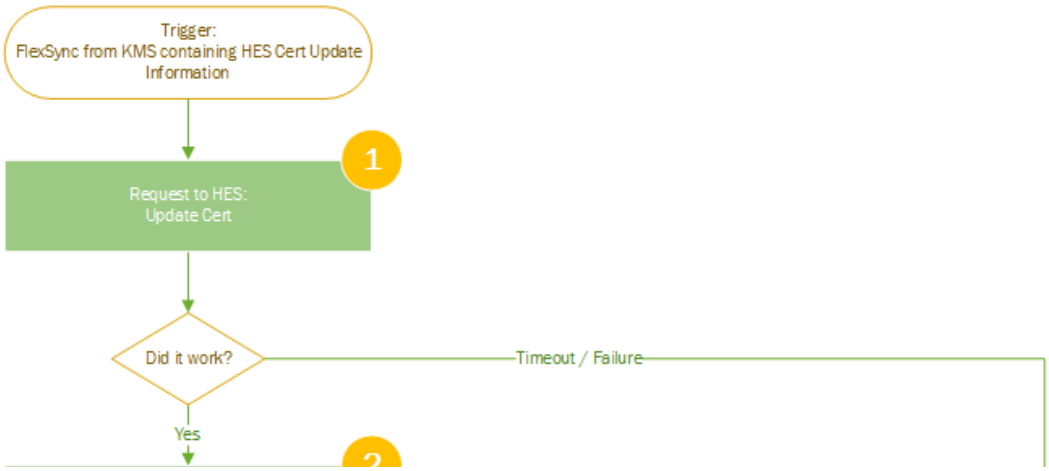
SDP Data Model Extension

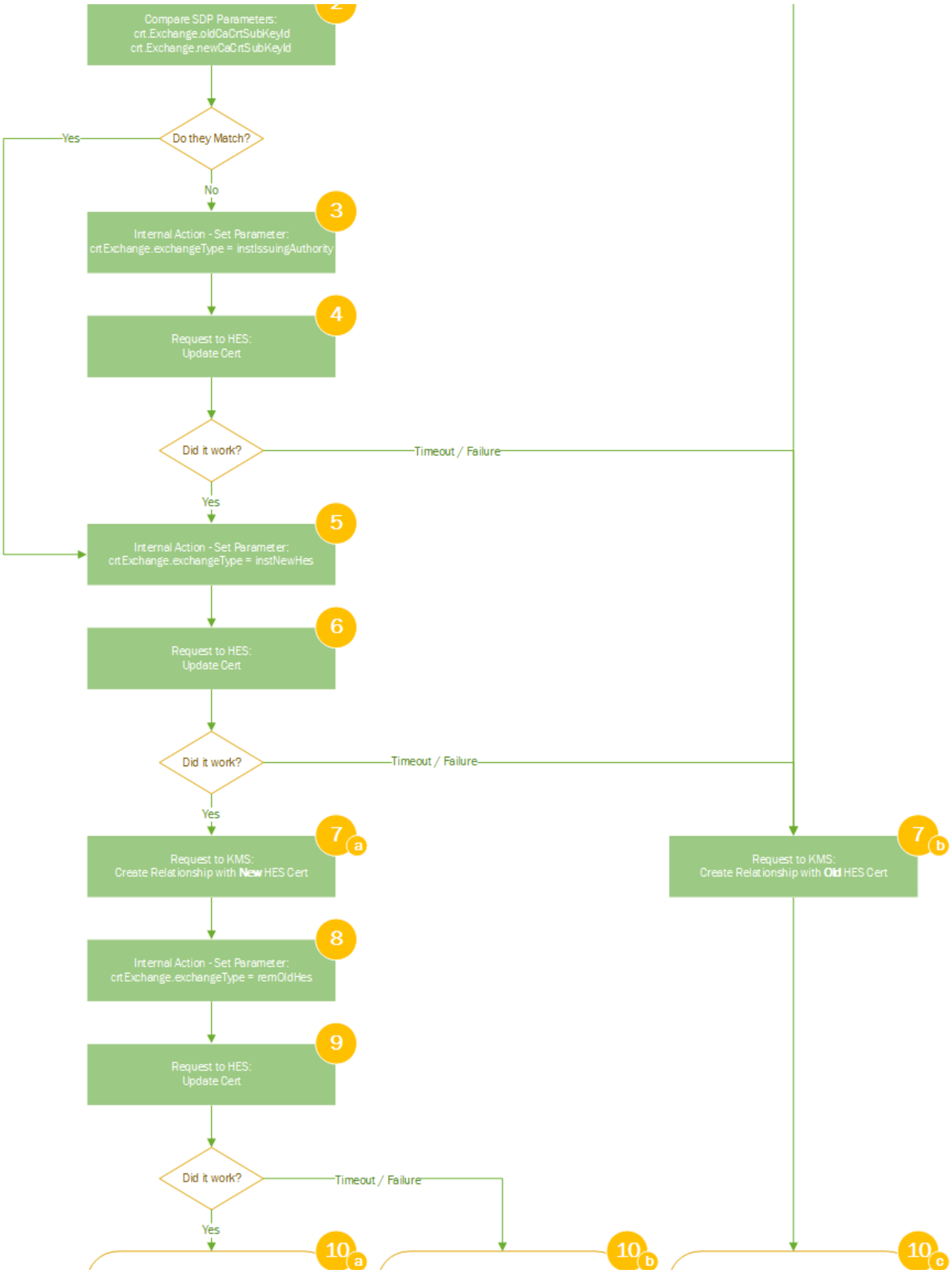
The MDM_I_METER_HES_CERT_REPLACE interface will populate several SDP Parameters with information that is used throughout the Certificate replacement process. This will require an extension to the EnergyIP SDP Data Model to add additional Parameters (these will be re-used by the Meter Certificate Lifecycle Management Process too):

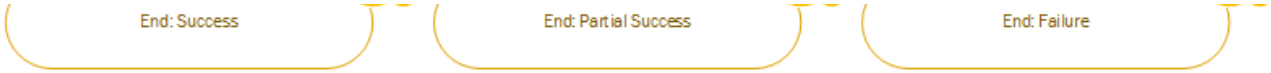
Parameter Label	Values / Enumerations Required for HES Lifecycle Management
crtExchange.exchangeType	Enumerated String: • instOldHes • instNewHes • instIssuingAuthority • remOldHes
crtExchange.crtRole	Enumerated String: • Management • Readout • PLC_Management
crtExchange.oldCrtFingerprint	Hex String
crtExchange.newCrtFingerprint	Hex String
crt.Exchange.oldCaCrtSubKeyId	Hex String
crt.Exchange.newCaCrtSubKeyId	Hex String

Process Orchestration

Having recieved the FlexSync notification and created a Service Request / Workflow, the actions that the Workflow must perform are described below:




2
Compare SDP Parameters:
crt.Exchange.oldCaCrtSubKeyId
crt.Exchange.newCaCrtSubKeyId
Do they Match?
Yes
No
3
Internal Action - Set Parameter:
crtExchange.exchangeType = instIssuingAuthority
4
Request to HES:
Update Cert
Did it work?
Timeout / Failure
Yes
5
Internal Action - Set Parameter:
crtExchange.exchangeType = instNewHes
6
Request to HES:
Update Cert
Did it work?
Timeout / Failure
Yes
7 a
Request to KMS:
Create Relationship with New HES Cert
7 b
Request to KMS:
Create Relationship with Old HES Cert
8
Internal Action - Set Parameter:
crtExchange.exchangeType = remOldHes
9
Request to HES:
Update Cert
Did it work?
Timeout / Failure
Yes
10 a
10 b
10 c



#	Description
1	Send a UAA Meter Update Request to HES. If successful this results in the Old HES Cert being stored in both Primary and Backup Slots, ensuring rollback possibility.
2	Compare New and Old Issuing Authority Cert IDs to establish whether the Issuing Authority Certificate requires updating before the HES Certificate.
3	If required, set the ExchangeType Parameter to instIssuingAuthority so the subsequent request sent to HES can be identified correctly as a request to install a different Issuing Authority Certificate.
4	Send a UAA Meter Update Request to HES. If successful this results in the Issuing Authority Certificate being updated in the relevant Slot. This paves the way for the subsequent HES Certificate update.
5	Set the ExchangeType Parameter to instNewHes so the subsequent request sent to HES can be identified correctly as a request to install the New HES Certificate.
6	Send a UAA Meter Update Request to HES. If successful this results in the New HES Certificate being installed in the relevant Primary Slot.
7a	If previous steps are successful, send a request to KMS to create a new HES Certificate to Meter relationship. This will automatically terminate the Old KMS Relationship, therefore subsequent Meter Ineteractions will use the New HES Certificate for Authentication.
7b	If previous steps have Failed or Timed-out then send a request to KMS to reset the existing HES Certificate to Meter Relationship. This means the Old HES Certificate remains in use and the Relationship will be re-avaulated for Lifecycle Management on the next Cycle.
8	Set the ExchangeType Parameter to remOldHes so the subsequent request sent to HES can be identified correctly as a request to remove the Old HES Certificate.
9	Send a UAA Meter Update Request to HES. If successful this results in the Old HES Certificate being removed from the Backup Slot.
10a	If the whole process worked successfully, the SR should be marked as Complete with a Status indicating an overall Success.
10b	If the process appeared to work but removal of the Old HES Certificate failed or timed-out the SR should be marked as Complete with a Status indicating that some 'cleanup' may still be required.
10c	If the process reached the installation of the New HES Certificate, but that action failed or timed-out the SR should be marked as Complete with a Status indicating that the overall process failed (indicating it will be retried in the next cycle of Lifecycle Management).

new MDM FlexSync Definition: MDM_I_METER_HES_CERT_REPLACE

An SDP SDPSyncMessage is sent to FlexSyncWS application by KMS Lifecycle Management in order to initiate the MDM Workflow that orchestrates the HES Certificate replacement. The SDP Sync request creates meter parameters which contain details about HES Certificates requiring update.

> ⚠ It's assumed that a required service request is derived on MDM side that's why it's not explicitly included in a request.

SDP Sync Request


```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v8="http://www.
   <soapenv:Header/>
   <soapenv:Body>
      <v8:SDPSyncMessage>
         <v8:header>
            <v8:noun>SDPSync</v8:noun>
            <v8:revision>1</v8:revision>
            <v8:dateTime>2016-05-05T00:00:00</v8:dateTime>
            <v8:source>CIS</v8:source>
            <v8:messageID>0de28253-c9af-4676-b63b-7d30745e36f3</v8:messageID>
            <v8:asyncReplyTo>none</v8:asyncReplyTo>
            <v8:syncMode>sync</v8:syncMode>
            <v8:optimizationLevel>Full</v8:optimizationLevel>
         </v8:header>
         <v8:payload>
            <v8:device>
               <v8:mRID>ISK1050768076703</v8:mRID>
               <v8:type>Meter</v8:type>
               <v8:parameter>
                  <v8:name>crtExchange.exchangeType</v8:name>
                  <v8:value>instOldHes</v8:value>
                  <v8:startDate>2016-05-05T00:00:00</v8:startDate>
               </v8:parameter>
               <v8:parameter>
                  <v8:name>crtExchange.crtRole</v8:name>
                  <v8:value>Management</v8:value>
                  <v8:startDate>2016-05-05T00:00:00</v8:startDate>
               </v8:parameter>
               <v8:parameter>
                  <v8:name>crtExchange.oldCrtFingerprint</v8:name>
                  <v8:value>1F43389BCE874F51844D191FF7B546721E4309F61E8B3CA15E</v8:value>
                  <v8:startDate>2016-05-05T00:00:00</v8:startDate>
               </v8:parameter>
               <v8:parameter>
                  <v8:name>crtExchange.newCrtFingerprint</v8:name>
                  <v8:value>07F530563BCCB54D4274FAAF561F3466BF0F75E603ADC6FBFF</v8:value>
                  <v8:startDate>2016-05-05T00:00:00</v8:startDate>
               </v8:parameter>
               <v8:parameter>
                  <v8:name>crt.Exchange.oldCaCrtSubKeyId</v8:name>
                  <v8:value>65841AB54565F65CD654654EF5654AA654654A654654E65465</v8:value>
                  <v8:startDate>2016-05-05T00:00:00</v8:startDate>
               </v8:parameter>
               <v8:parameter>
                  <v8:name>crt.Exchange.oldCaCrtSubKeyId</v8:name>
                  <v8:value>07F530563BCCB54D4274FAAF561F3466BF0F75E603ADC6FBFF</v8:value>
                  <v8:startDate>2016-05-05T00:00:00</v8:startDate>
               </v8:parameter>
            </v8:device>
         </v8:payload>
```

```
      </v8:SDPSyncMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Attribute	Data Type	Required	Configurable	Source System	Values	Descript
//SDPSyncMessage/header/						
noun	string	Y	N	KMS	SDPSync	
revision	string	Y	N	KMS	1	
dateTime	date	Y	N	KMS	<current time>	
source	string	Y	N	KMS	CIS	
messageID	string	Y	N	KMS	<UUID>	
asyncReplyTo	string	Y	N	KMS	none	
syncMode	string	Y	N	KMS	sync	
optimizationLevel	string	Y	N	KMS	Optimistic	
//SDPSyncMessage/payload/device/						
mRID	string	Y	N	KMS	<Entity ID>	
type	string	Y	N	KMS	Meter	
//SDPSyncMessage/payload/device/parameter						
name	string	Y	N	KMS	crtExchange.exchangeType crtExchange.crtRole crtExchange.oldCrtFingerprint crtExchange.newCrtFingerprint crt.Exchange.oldCaCrtSubKeyId crt.Exchange.newCaCrtSubKeyId	
value	string	Y	N	KMS	<parameter value>	
startDate	date	Y	N	KMS	<current time>	

SDP Sync Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v8="http://www.
   <soapenv:Header/>
   <soapenv:Body>
```

```
      <v8:SDPSyncReplyMessage>
         <v8:header>
            <v8:verb>?</v8:verb>
            <v8:noun>?</v8:noun>
            <v8:revision>1</v8:revision>
            <v8:dateTime>2016-05-05T00:00:00</v8:dateTime>
            <v8:source>?</v8:source>
            <v8:messageID>e40153ba-ea99-4cb4-9733-270a84a08893</v8:messageID>
         </v8:header>
         <v8:reply>
            <v8:replyCode>0</v8:replyCode>
            <v8:replyText>?</v8:replyText>
            <v8:correlationId>0de28253-c9af-4676-b63b-7d30745e36f3</v8:correlationId>
         </v8:payload>
      </v8:SDPSyncMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Attribute	Data Type	Required	Source System	Values	Description
//SDPSyncReplyMessage/reply/					
replyCode	string	Y	MDM	0 | 1 | etc.	
replyText	string	Y	MDM		
correlationId	string	Y	MDM	<request: //SDPSyncMessage/header /messageID>	

new Solution Proposal (UDIS): HES Certificate Replacement Support

> **Note:**
> Several sections of this page refer to 'Slots', for a explanation of this terminology see here: [new Security Suite 1 - HES (Client) Certificate Support]

UDIS will receive requests to perform actions on HES Certificates and related Issuing Authority Certificates from MDM. These requests will be delivered to UDIS and Responses expected via the Interface described here: new UAA to UDIS Adapter Interface Extension for HES Certs: MeterAssetUpdateRequestMessage

From this request, one of four actions relating to HES Certificates will be requested based on the value passed in the crt.Exchange, exchangeType:

Exchange Type	Description
instOldHes	This is a request to install the Old HES Certificate identified in the request into the relevant COSEM Client's Backup Slot.
instNewHes	This is a request to install the New HES Certificate identified in the request into the relevant COSEM Client's Primary Slot.
instIssuingAuthority	This is a request to install the Issuing Authority Certificate identified in the request into the relevant COSEM Client's Sub-CA Slot.
remOldHes	This is a request to remove / delete the OLD HES Certificate from the relevant COSEM Client's Backup Slot

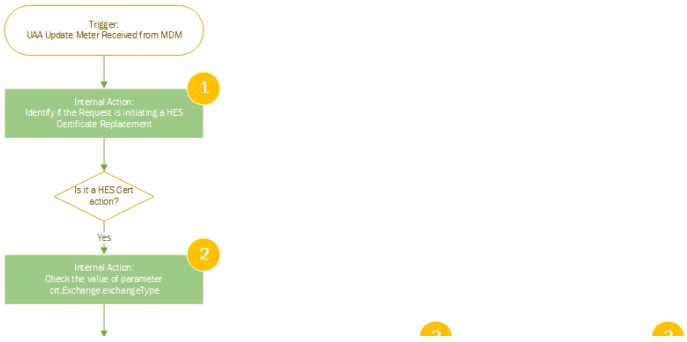
The replacement of a HES Certificate in the Meter is a multi-step process involving several Meter interactions. Responsibility for the orchestration of these actions is not an UDIS function. MDM will orchestrate the process and requires UDIS only to perform simple actions upon request. See new Solution Proposal (MDM): HES Certificate Replacement Support for more information about the orchestration.

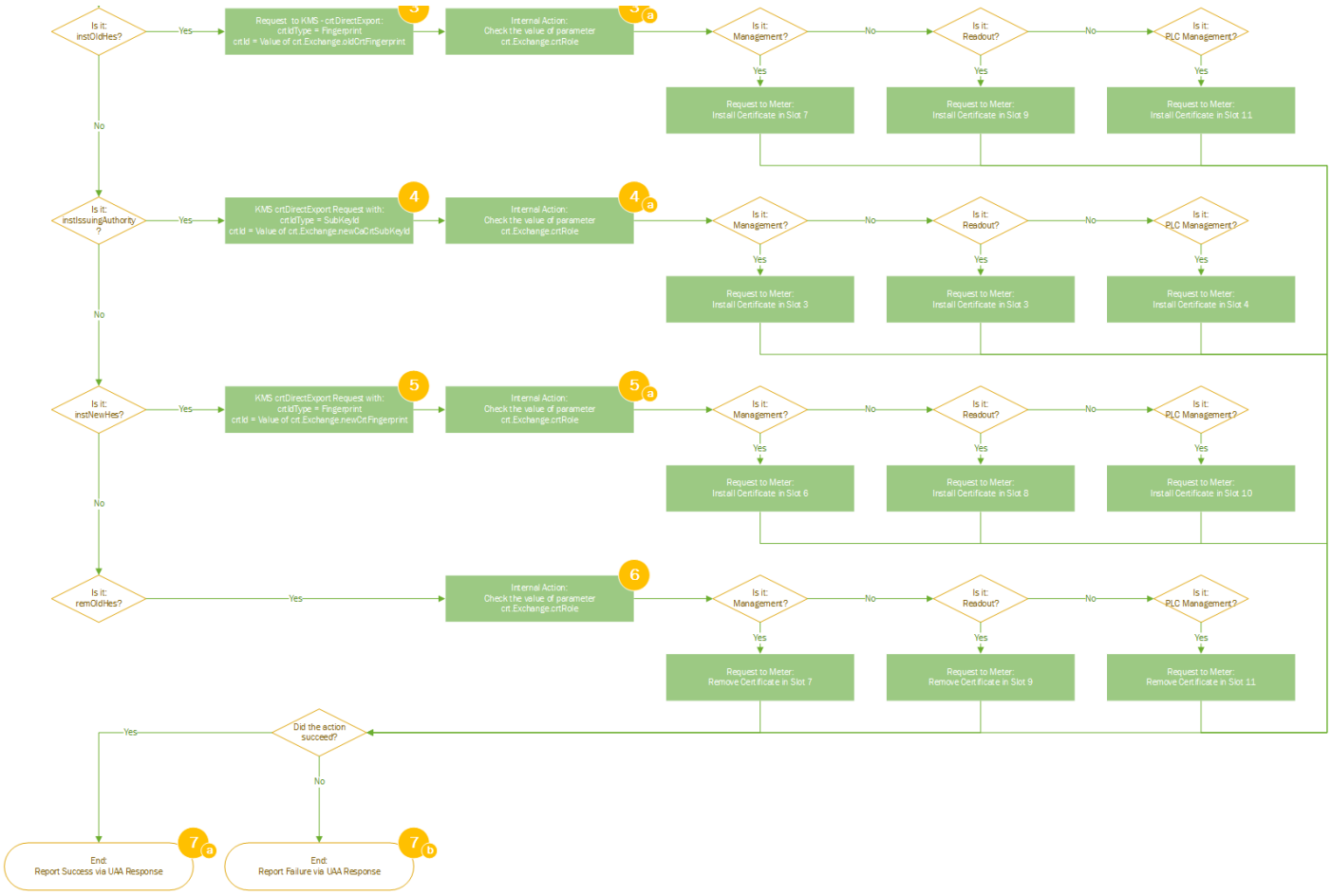
Process

UDIS is responsible for taking the information provided in the request from MDM and initiating the action, UDIS is therefore responsible for:

- Identification of which Slot a given Certificate should be installed in, based on Exchange Type and Role
- Identification of which COSEM Client should be used to perform the action
- Retrieval of the Certificate to Install from KMS using the interface defined here: new KMS REST Service Definition: crtDirectExport
- Making the Installation / Removal request to the Meter, including negotiating Authentication and Encryption mechanisms (with KMS REST services supporting)
- Providing a response to MDM indicating the success / failure status of the requested action

The required overall UDIS process logic can be summarised as below, however this can be implemented in whatever way most efficiently fulfills all the potential paths through this flow:





Step #	Description	Supporting Information
1	Upon receipt of the MeterAssetUpdateRequestMessage request, identify if the request is intended to trigger a HES Certificiate Update activity.	new UAA to UDIS Adapter Interface Extension for HES Certs: MeterAssetUpdateRequestMessage
2	Check which of the 4 types of HES Certificate related activity is being requested.	
3	If the activity is to install the Old HES Certificate in the Backup Slot of the relevant COSEM Client, retrieve the Certificate from KMS via the REST Service, using the Fingerprint parameters provided in the request.	new KMS REST Service Definition: crtDirectExport
3a	Using the Role from the request, establish which Slot the Certificate should be installed in, and make the request to the Meter.	new HES (Client) Certificate Lifecycle Management
4	If the activity is to install a New Issuing Authority Certificate, retrieve the Certificate from KMS via the REST Service, using the Subject Key Identifier parameters provided in the request.	new KMS REST Service Definition: crtDirectExport
4a	Using the Role from the request, establish which Slot the Certificate should be installed in, and make the request to the Meter.	new HES (Client) Certificate Lifecycle Management
5	If the activity is to install the New HES Certificate in the Primary Slot of the relevant COSEM Client, retrieve the Certificate from KMS via the REST Service, using the Fingerprint parameters provided in the request.	new KMS REST Service Definition: crtDirectExport

5a	Using the Role from the request, establish which Slot the Certificate should be installed in, and make the request to the Meter.	new HES (Client) Certificate Lifecycle Management
6	If the activity is to remove / delete the Old HES Certificate from the Backup Slot, use the Role from the request to establish the Slot requiring deletion / removal and make the request to the Meter.	
7a	If the requested action was a success, report the success status back to MDM via the UAA Adapter.	new UAA to UDIS Adapter Interface Extension for HES Certs: MeterAssetUpdateRequestMessage
7b	If the requested action was a failure, report the failure status back to MDM, including information describing what went wrong, via the UAA Adapter.	new UAA to UDIS Adapter Interface Extension for HES Certs: MeterAssetUpdateRequestMessage

new UAA to UDIS Adapter Interface Extension for HES Certs: MeterAssetUpdateRequestMessage

The MeterAssetUpdateRequestMessage interface is an existing Interface between MDM and UDIS, it is documented here: UDIS Schnittstellenbeschreibung

The MeterAssetUpdateRequestMessage interface is currently used to trigger actions in UDIS that result in something being updated in the Meter. The definition will be extended to give provision for sufficient data to be passed to UDIS to enable the management of HES Certificates and related Issuing Authority Certificates in the Meter.

The proposed extensions use the following as a Baseline:

- UDIS V6.0 Adapters V1.8
  - Section 3.5 UAA Provisioning
    - Updating a Meter: Request
    - Updating a Meter: Response

Request Extension: Additional Parameters

Six new parameters will be included to support request of HES Certificate actions, and provide HES with all neccesary information to initiate the requested process:

> **Note:**
> All parameters are optional within the context of a MeterAssetUpdateRequestMessage, however for a Replace HES Certificate initiating request; if any one is present then all should be present.

Parameter Name	Parameter Value	Required	Description
crtExchange.exchangeType	String	N	This parameter denotes the action being requested of HES: • instOldHes - Install a Certificate in the Client's Backup Slot

			• instNewHes - Install a Certificate in the Client's Primary Slot • instIssuingAuthority - Install a Certificate in the SubCA Slot related to the Client • remOldHes - Delete the Certificate in the Client's Backup Slot
crtExchange.crtRole	String	N	The Role / Client that the Certificate action is being requested for: • Management • Readout • PLC_Management
crtExchange. oldCrtFingerprint	String	N	A hex value allowing the unique identification of a specific HES Certificate.
crtExchange. newCrtFingerprint	String	N	A hex value allowing the unique identification of a specific HES Certificate.
crt.Exchange. oldCaCrtSubKeyId	String	N	A hex value allowing the unique identification of a specific Issuing Authority Certificate.
crt.Exchange. newCaCrtSubKeyId	String	N	A hex value allowing the unique identification of a specific Issuing Authority Certificate.

Reply Codes

The following responses (or very close equivalents) are to be provided as a minimum. More can be added to the list if they are available and giving more detailed / granular responses is also acceptable.

> **Note:**
> Non-zero replyCodes can be assigned as required

replyCode	replyText
0	Success
	Meter offline
	Meter association could not be established
	No response from Meter
	Request failed at Meter

new Security Suite 1 - Meter (Server) Certificate Support

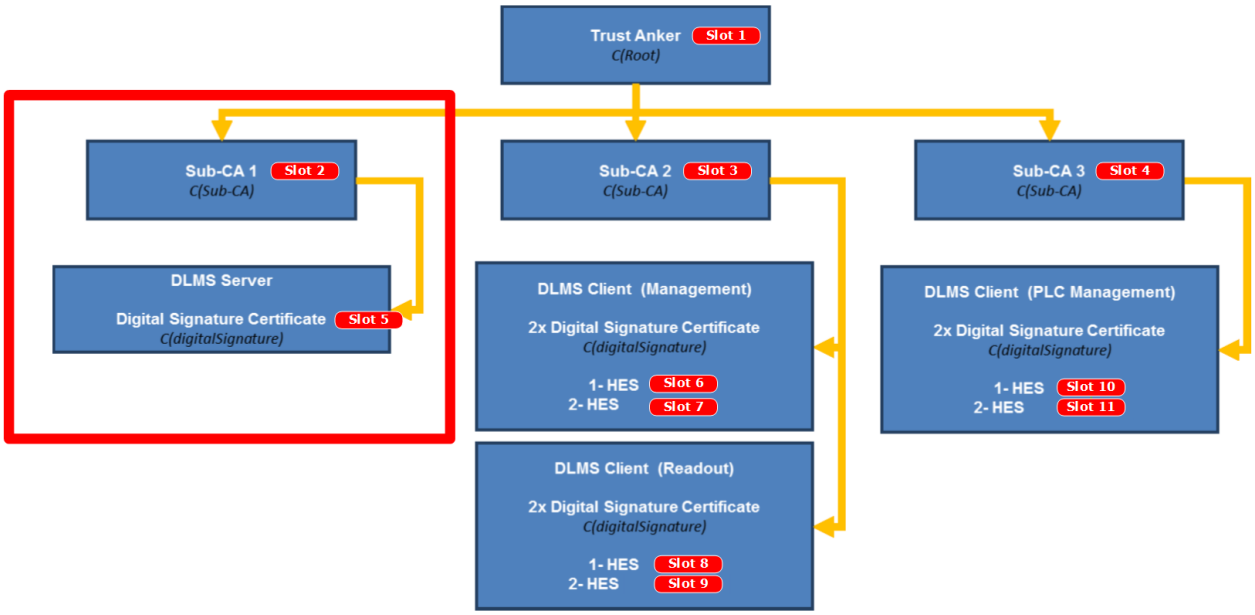
To Support Security Suite 1 Functions the Meter must be able to store several Certificates. These are broadly categorised into two:

- **Client Certificates** - In this context the Client is the HES, hence the Certificate contain Public Keys related to Private Keys 'owned' by the Central System (HES)
- **Server Certificates** - In this context the Server is the Meter, hence the Certificates contains the Public Keys related to Private Keys 'owned' by the Meter

These Certificates and their related Private Keys are used for Authentication between Meter and HES. All Certificates are issued in a hierarchy containing a Root Certificate Authority and several Issuing Authorities (Sub-CA).

Certificates are usually issued with an intended finite lifetime, therefore must be replaced, along with their related Private Keys, in the Meter with new Certificates over time.

The diagram below is taken from the Honeywell Companion Standard, modified to better represent the integration with the the Central System (HES, KMS, MDM). It shows the Certificates stored in the Meter and their hierarchical structure. The Certificates in the red box are those that we are concerned with when discussing Meter Certificates and their Management:



new Meter (Server) Certificate Import

This topic remains under discussion with the Customer. However there are two options 'on the table':

- **Option #1** - Import of Meter Certificates via Shipment File
- **Option #2** - Automatic synchronisation of Meter Certificates from PKI

The second option appears to be the preference at the moment and would probably be technically possible.

Option #2 Evaluation

According to Honeywell and Worldline documentation, when a CSR is submitted to the PKI, the Common Name from the Subject of the CSR is used to create or update a 'User' entity in EJBCA.

- TINetz_SMPKI_DTD_v0.2.docx - Section 6.1.3.3
- CSR Prozessbeschreibung-v12-20181004_vertraulich.pdf - Section 3, point 9

According to Greenbook, Meter Certificates (and hence Meter Certificate Signing Requests) contain the System Title of the Meter as the Subject Parameter. Therefore, for each Meter CSR processed, a 'User' within the context of PKI will be created with the Meter's System Title as the ID.

The EJBCA Web Service provides a function:

Name: **findCerts**

Taking Parameters:

- username - a unique username
- onlyValid - only return valid certs not revoked or expired ones.

Returns: A collection of Certificates or an empty list if no certificates, or no user, could be found.

Following on from, or during Key Import; KMS knows the COSEM Logical Device Name of the Meters for which it hold some information. The System Title can be logically derived from the Logical Device Name.

Therefore during or after Key Import, the KMS could:

1. Make a request to the findCerts function to retrieve an array of Meter Certificates for the Device (there *should* only be one)
    a. If there is only a single Certificate returned, store the Certificate
    b. If there are multiple Certificates returned, stored the most recently created. This would require programatic inspection of the Certificate content.

This process could be added:

- As an extension to the Shipment File Loader Process
- As an extension to the actions a call to the Key Import REST Service initiates
- As a scheduled Periodic Job that identifies Meters that are in KMS with Symmetric Keys, but no Meter Certificate

The last option is my preference as it does not change any existing processes.

new Meter (Server) Certificate Lifecycle Management

Meter Certificates (aka Asymmetric Keys) should be included within Lifecycle Management for automatic semi-automatic renewal in all devices.

The overall Lifecycle Management of the Meter Certificates should comprise of:

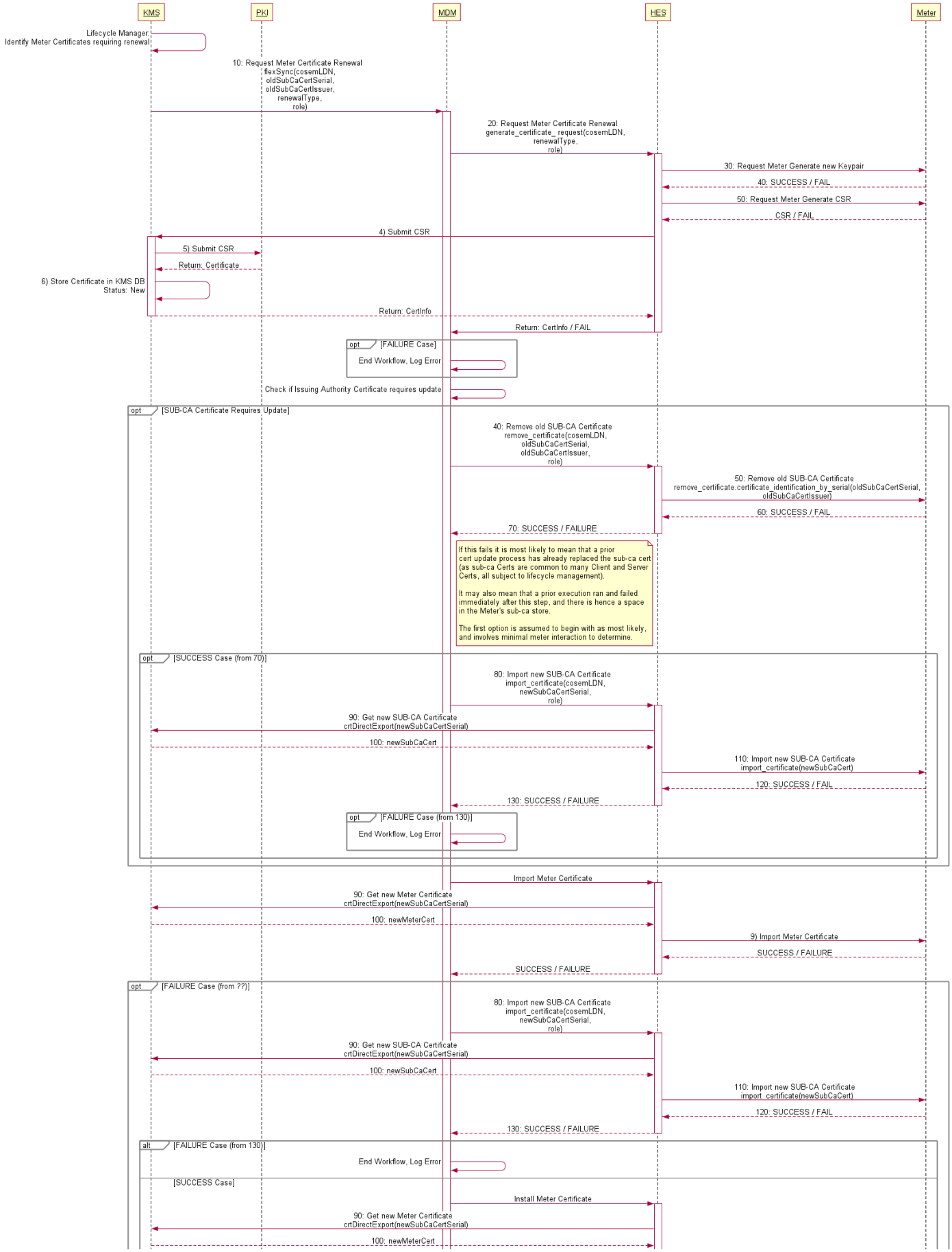
- Automatic identification of which Meters are required to go through the Meter Certificate renewal process
- Automatic initiation of the process to renew the Meter Certificates

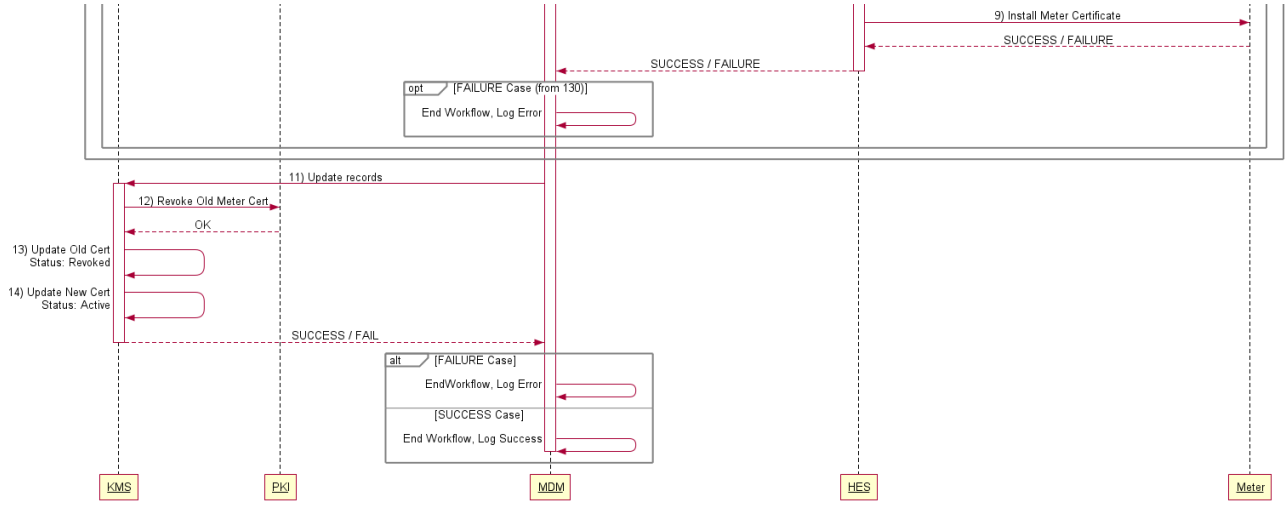
Only the LifeCycle of Meter Certificates will be actively considered in LifeCycle Management, however the renewal of the Meter Certificate may necessitate the update of the Issuing Authority Certificate in the Meter. This will be replaced where necessary by implication.

LifeCycle Management Process Overview

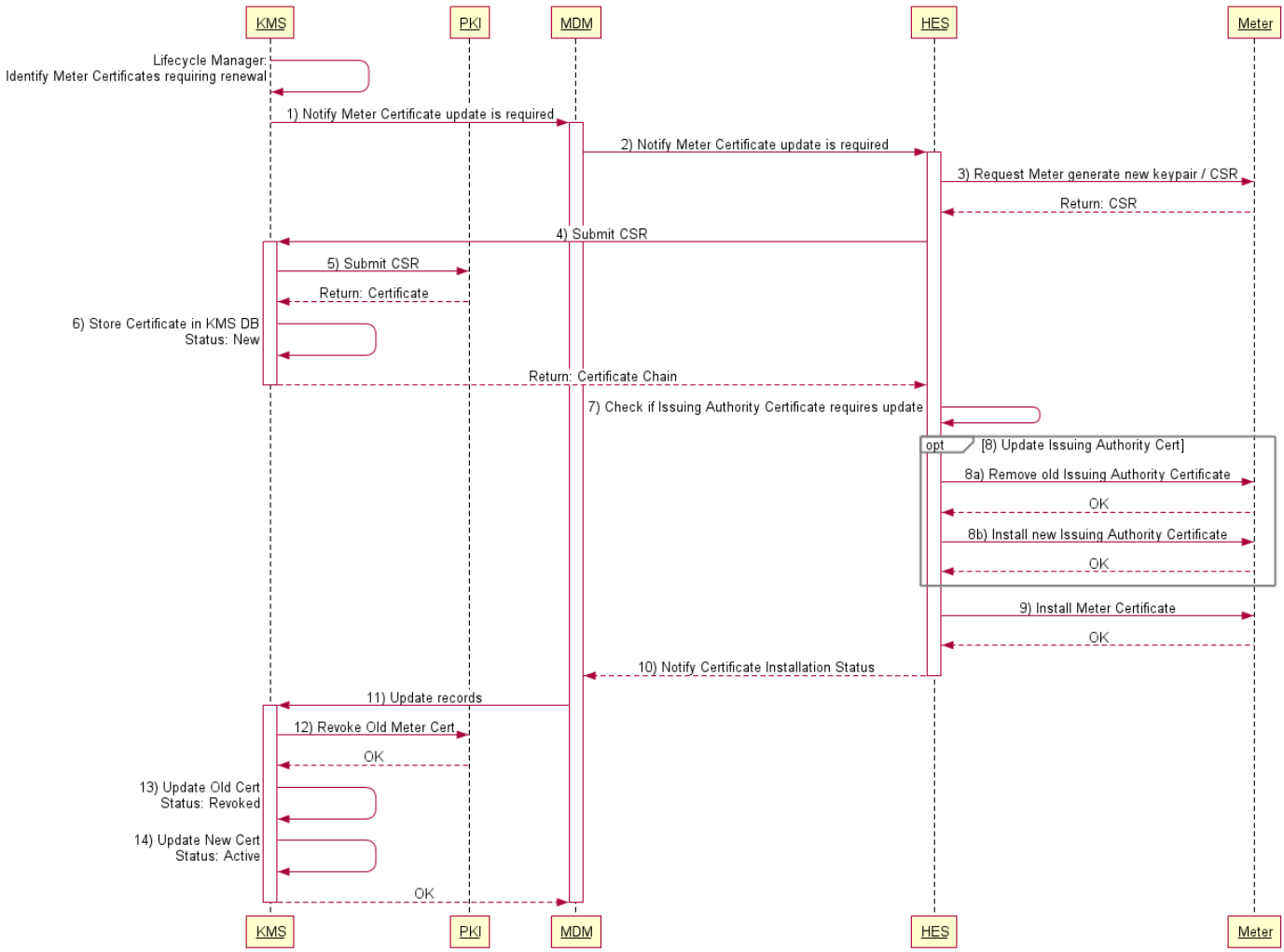
A summary is given beneath the diagram, for detailed information see the Solution Proposal sections for individual components:

TIN Meter Certificate Replacement v2





TIN Meter Certificate Replacement



Step Descriptions:

Step	Description	Reference
1	Once a Meter Certificate has been identified as exceeding LifeCycle parameters, the renewal process is triggered. This results in a FlexSync notification being sent to MDM requesting a Meter Certificate renewal process be initiated.	
2	On receipt of the request in step #1, MDM will forward the Meter Certificate Renewal request to UDIS via the UAA Adapter.	
3	UDIS will perform the necessary actions to initiate the Meter Certificate Renewal in the Meter.	
4	The outcome of step #3 is that the Meter provides a Certificate Signing Request back to UDIS. UDIS must then submit the CSR, via REST Service, to the KMS for processing.	
5	Having received the CSR from UDIS, KMS will submit the CSR for processing to the relevant PKI via Web Service.	
6	The response from the PKI to the submission of the CSR is a Meter Certificate Chain. KMS stores a copy of the Meter Certificate for future reference and then returns the Certificate Chain to UDIS.	
7	UDIS must extract information from the Certificate Chain to compare against information provided in the initial request; to identify whether steps #8 and #9 (the update of the Issuing Authority Certificate) are necessary.	
8a	To enable to installation of a new Issuing Authority Certificate, first the old one must be removed from the Meter.	
8b	Following successful remove of the old Issuing Authority Certificate, the new Issuing Authority Certificate can be installed in the Meter.	
9	If either no update of the Issuing Authority Certificate was required, or if it was completed successfully, the new Meter Certificate should be installed in the Meter.	
10	The overall status of the Renewal process should then be reported back to MDM via UAA response.	
11	If the overall Renewal process has been successful between HES and Meter, then the KMS should be notified of the sucessful Renewal so it can get it's Certificate records synchronised.	
12	The KMS should submit revocation requests to PKI (via Web Service) for any old Meter Certificates matching the 'Type' of the Certificate just renewed.	
13	Following successful revocation requests, the old Certificates should have their status marked as 'Revoked'	
14	The first action of the Certificate Syncronisation process is to set the status of the New Certificate to 'Active' and the status of the old Meter Certificate to 'Retired'	

The LifeCycle Management Process takes into account several important areas:

Area	Description

Permitted Integration routes	KMS should not interface directly to HES, only to MDM. HES is permitted to interface directly to MDM or KMS. MDM is permitted to interface directly to HES or KMS.
What can be achieved within MDM via the UAA Adapter (Product)	The existing Meter-centric interface between MDM and HES is via the productised UAA Adapter. Customisations / Extensions to the UAA Adapter would introduce potential support issues and should be avoided.
Failure Diagnosis	As far as possible it should be possible to report on information held within MDM to get an overview of where in the replacement process a failure occurs. This may be a compromise against other factors as described in this table.

new Solution Proposal (KMS): Meter Certificate Lifecycle Management

KMS must be able to:

- Monitor Meter Certificates for impending expiry
- Notify MDM to initiate a Meter Certificate Renewal process
- Receive a Meter CSR from UDIS and respond with a Certificate
- Store the Certificate that results from the above
- Receive notification that the new Cert should be made 'active'

Replacement Initiation

> ⚠ **Important control:**
> Only a single HES Certificate Replacement activity should be underway for a specific individual meter at any given time.
>
> HES Certificate Replacement activities should not be undertaken at the same time as any other Key renewal operation in the Meter.

KMS Lifecyle Management shall include a process that then processes each Meter Certificate requiring renewal and submits a meterCertRenew request to MDM using the interface described here: new MDM FlexSync Definition: MDM_I_METER_METER_CERT_RENEW

The MDM Interface requires KMS to provide the inputs below:

Input	Description	Derivation
Input	**Description**	**Derivation**
crtExchange.exchangeType	This value indicates the action that should be	For KMS initiate Lifecycle Management this should always be:

	undertaken, install a Cert, remove a Cert etc.	• renewMeterDS
crt.Exchange. oldCaCrtSubKeyId	This is the Subject Key Identifier of the Issuing Authority Certificate related to the currently installed Meter DS Certificate in the Meter.	The current 'active' Meter DS Certificate can be located in the KMS DB, from this record the Authority Key Identifier can be taken directly and used as the value for oldCaCrtSubKeyId of this request.

KMS REST Service Support

Certificate Update Request?

new KMS REST Service Definition: ../certificate/meter /status_update

This REST Service allows KMS to be sent notification of when a certificate was successfully installed for Meter, indicating that the Meter Key Renewal process has completed and KMS can update its records accordingly (see the Request Handling section for further detail).

Request

Request:

```
{
    "deviceSerialNumber" : "KFE01000000001",
    "exchangeType" : "renewMeterDS"
}
```

Attribute	Data Type	Required	Source System	Values	Description
/					
deviceSerialNumber	string	Y	UDIS/GW	See example Request	Determines the device which the certificate belongs to (corresponds to the MFG_SERIAL_NUM in KMS_CERT table, this will be the COSEM Logical Device Name in most deployments).
exchangeType	string	Y		renewMeterDS	This is the type of successful renewal that the calling System is notifying KMS of, this can be taken from the crtExchange. exchangeType parameter value (set to initiate the Workflow).

Request Handling

Upon receipt of this request the following actions will be completed:

> ⚠ **Important note:**
> The order of these steps is important

- Any other existing Certificates for the Meter matching the criteria below will have their status marked as 'Retired'
  - Criteria:
    - Entity ID = deviceSerialNumber from the Request
    - Type = Meter
    - Sub Type = The Sub Type of the Certificate identified by certSerialNumber from the Request
    - Status = 'Active'

- The Certificate record of the New Meter Certificate will be identified via the criteria below, and its status set to 'Active'
  - Criteria:
    - Entity ID = deviceSerialNumber from the Request
    - Type = Meter
    - Sub Type = Based on the value of exchangeType from the Request; 'renewMeterDS' mapped to 'Sign'
    - Status = 'New'

- Any other existing Certificates for the Meter matching the criteria below will be revoked via Web Service request to the PKI and their status set to 'Revoked'
  - Criteria:
    - Entity ID = deviceSerialNumber from the Request
    - Type = Meter
    - Sub Type = The Sub Type of the Certificate identified by certSerialNumber from the Request
    - Status = 'Retired'

Response

Success Response:

```
{
    "result": "Success"
}
```

Attribute	Data Type	Required	Source System	Values	Description
/					

result	string	Y	UDIS/GW	Success	The response can mean success only. In case of failure, error response is retrieved.

Error response

Failure Response:

```
{
    "errorCode": "KMS-xxxxxxx",
    "errorDescription": "Gateway not present in database" // this is only sample, various message
}
```

Attribute	Data Type	Required	Source System	Values	Description
/					
errorCode	string	Y	UDIS/GW	TBD	The short error response code
errorDescription	string	Y	UDIS/GW	TBD	A human readable summary of the error

new KMS REST Service Definition: Extension of ../certificate /meter/retrieve

The existing /certificate/meter/retrieve KMS REST Service must be extended as per the definition below.

Request

Request:


```
{
  "csrPayload" : "-----BEGIN CERTIFICATE REQUEST-----
MIIDFjCCAf4CAQIwcjEPMA0GA1UEBwwGVmllbm5hMQ8wDQYDVQQIDAZWaWVubmEx
EzARBgNVBAoMClNpZW1lbnMgQUcxFTATBgNVBAsMDERpZ2l0YWwgR3JpZDEVMBMG
A1UEAwwMRFZMMTYxMjAwMDAwMQswCQYDVQQGEwJBVDCCASIwDQYJKoZIhvcNAQEB
BQADggEPADCCAQoCggEBANb/cUe7049nT1rcmyPEAnXX6sL3mI34UCSMNwOT3WMP
4T+dtMjaWefaBmFcbPT7ajj4DTCBJjZ8S6hepkqeXdZefrXgAYgG5WGqgHFX/AQS
TXYWu8XakiQxP2rc9Xgex4V1wB/7yCnHZWD/N5kZSChIFQpTgcQ6HmG9EPImnTKb
pvwRWgJuCFr/IVw8wWYwwc8pFqGP+Rpryvqer+f9CG3a0/zMS3j1jUmw29cvwwch
ulKu+CuZaNPBzbyi6nfp+mGidg1Zv27p6enr+EVFatQp3rjMEL8MJ7c8EhRHix24
rKqyKcaSWSaHmYLMMTWOqys3XSDCLVcE51oy5tPapz0CAwEAAaBfMF0GCSqGSIb3
DQEJDjFQME4wCQYDVR0TBAIwADALBgNVHQ8EBAMCA7gwHQYDVR0lBBYwFAYIKwYB
BQUHAwEGCCsGAQUFBwMCMBUGA1UdEQQOMAyHBKwQAAOHBAoJnYMwDQYJKoZIhvcN
AQELBQADggEBANLQZbTtDaQVO3vYJUhZo80tzR7ObNBuS3TNfafdCAj+L+7GtcBd
OMidG7jzeu6AGoHS3N5jgFmdA+SjmJoeFb3sDeHMftpg9K6+nvknSIt9FqgspDYE
4czYBesdnGNhy7tRz6hRvLFxpVuS3+34zqJpsrW3rUiYXrpTK+AiG0H4DumNwSr5
6jiQK4oUPJDuZk/Rkib0u4jl/7klpP1jJHlDQtA7CLxewKy1nwUVrId0K8gG8meK
ymivf0twX3E/n8ltg5yprQtwSDtUb6syE1NuVQ5LNNiqcZEU0s5q3q+rj9RcP/ay
```

```
5PwmPfBI6rDOIMKgyuA2GBZYNnidmrOlwAw=
-----END CERTIFICATE REQUEST-----",
  "csrSign" : "mIK5+E8wL+u/h5VkFfnPBxfqydyLJHfULn3LkxaendBEVuE4aVK6a2lTdZPYCXJsJW29VSgouXBaM62vRd
  "deviceSerialNumber" : "KFE10000000010"
}
```


Attribute	Data Type	Required	Values	Description
/				
csrPayload	base64	Y	See example Request	Certificate signing request (originated by Meter, forwarded by UDIS to KMS).
csrSign	base64	N	See example Request	Signature of the certificate signing request.
deviceSerialNumber	String	Y	KFE10000000010	The COSEM Logical Device Name (Entity ID in KMS UI, manufacturer serial number in KMS DB).This must be present because the 'Subject' of the CSR is the System Title of the Meter. The System Title does not provide enough information for KMS to derive the COSEM Logical Device Name (KMS's primary identifier).

Request Handling

- If Signature is present, it should be verified
- If Signature is not present, then the status of the existing Meter Cert in KMS should be verified. Only where a CSR is expected (due to lifecycle management) should the CSR be forwarded to PKI (as configured).
- Response from PKI will be a Certificate Chain in PEM format
  - Whole Chain should be returned to UDIS
  - Only Meter Cert should be stored in KMS DB (with Authority Key Identifier extracted)

Response

> ⚠ **Warning:**
> The Certificates in this Response example are not real Certificates and will not decode. This example is provided only to show how the Chain will be represented in the Response.
>
> See the Companion Standard and/or Greenbook for the full Certificate examples.

Response:


```
{
  "certChain" : "-----BEGIN CERTIFICATE-----
MIIFvTCCA6WgAwIBAgIIQK8NOgVWS9EwDQYJKoZIhvcNAQELBQAwTDEfMB0GA1UE
AwwWU00tMjAxNy1FWFQtVExTLVNVQi1DQTEOMAwGA1UECwwFU01QS0kxDDAKBgNV
BAoMA0tORzELMAkGA1UEBhMCQVQwHhcNMTcxMTA3MTAwOTU1WhcNMjAxMTA3MTAw
FjzIAKUUbCdn4Y/qWMOPs8hUfAXWO/4ocX9G7/u/3mxGqgkq12cl4VJ0BVCnHV0f
Ykkfw+rFXF0lxWk00yDzPzsuhoqd9mbKSO4KoJk6fCI3Wp7/OoSCypvSI7BHDCxR
kpIsxQvdSIc51LmbqxTrqwu6y8aMI4JtQtpqFRTnBE7E
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFfTCCA2WgAwIBAgIINxM7Ek1OxxwwDQYJKoZIhvcNAQELBQAwSTEcMBoGA1UE
```

```
AwwTU00tMjAxNy1SU0EtUk9PVC1DQTEOMAwGA1UECwwFU01QS0kxDDAKBgNVBAoM
A0tORzELMAkGA1UEBhMCQVQwHhcNMTcwMzE0MTEwNDA2WhcNMzkwMzE0MTEwNDA2
iLx32+NEUWx8nvI9NqOpERpnGrXSkiAJKurJWx6HIrPzFOskGOwndFoGvSyoqw0t
zcaSOmPtE7sEMibuoNmC0KRbZ14T/gzdLatE612u+uDmbGZAiFPy0Iq601wzK0E5
AhJcbGXiWI4MQgEov0fsAvo/vrymLqUF30L0VbGGsm+hmZScrlB1X5MNHWnnmlDS
0Qa+I2GCVlrVAVVirTR9dwk=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFeTCCA2GgAwIBAgIIJdpbwVBG/HAwDQYJKoZIhvcNAQELBQAwSTEcMBoGA1UE
AwwTU00tMjAxNy1SU0EtUk9PVC1DQTEOMAwGA1UECwwFU01QS0kxDDAKBgNVBAoM
A0tORzELMAkGA1UEBhMCQVQwIBcNMTcwMzA5MTcyNzIzWhgPMjA2MjAzMDkxNzI3
2wVpiw5MsE5e3jeGFWWPOprhoagd+b2i4qR4PkwoQKc8xbdq3R0XhMzU2in5mtmJ
+G1UotUi0hxUQHHzxXLkq5DMfda0jlHWG/BYHof8f1rwzpC2y3PHu7BzV86BBYqz
bWk6/6CDmhHqVv/PHW9qZwiY1/49kXQmXfq2sdZhtxRUxw9eVbIpnawrTXvpbTcO
+AtzPlPy8ChKozShzQ==
-----END CERTIFICATE-----",
"deviceSerialNumber" : "KFE10000000010"
}
```


Attribute	Data Type	Required	Values	Description
/				
certChain	PEM Certificate Chain	Y	See example Response	Certificate Chain in PEM format, individual Certificate Base64 encoded. The order of the Certificates in the Certificate Chain will always be: 1. Meter Certificate 2. Issuing Authority Certificate 3. Root Certificate
deviceSerialNumber	String	Y	KFE10000000010	The COSEM Logical Device Name (Entity ID in KMS UI, manufacturer serial number in KMS DB).This is a simple pass-through of data from the Request.

new Solution Proposal (MDM): Meter Certificate Renewal Support

The MDM will receive a notification from the KMS that a Meter Certificate Renewal is required using the Interface defined here: MDM_I_METER_METER_CERT_RENEW

The receipt of a notification to that interface will trigger a Workflow responsible for the submission of the update request to HES and the feedback to KMS of success.

SDP Data Model Extension

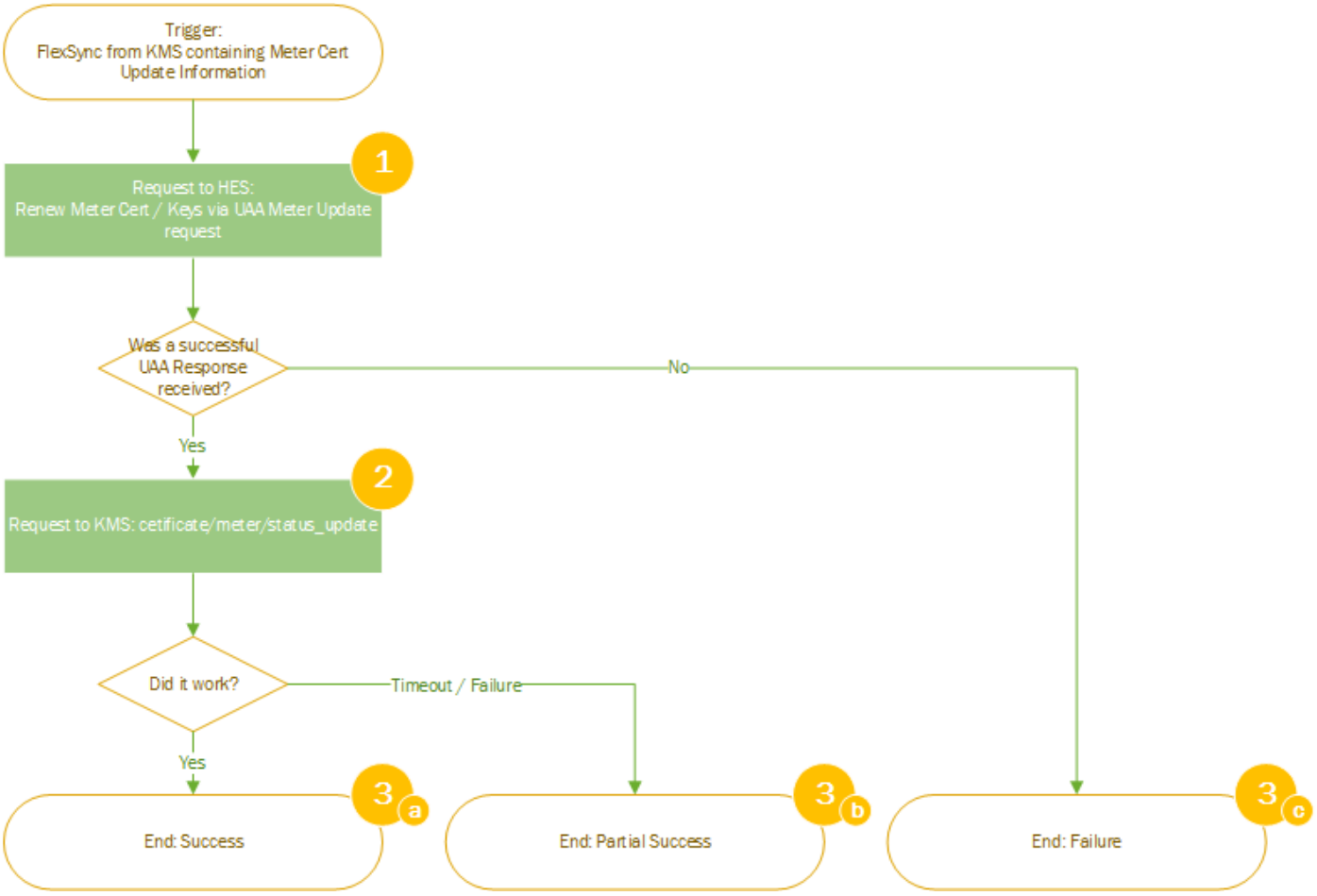
The MDM_I_METER_METER_CERT_RENEW interface will populate several SDP Parameters with information that is used throughout the Certificate renewal process. This will require an extension to the EnergyIP SDP Data Model to add additional Parameters (these will be re-used by the HES Certificate Lifecycle Management Process too):

Parameter Label	Values / Enumerations Required for HES Lifecycle Management
crtExchange.exchangeType	

	Enumerated String: • renewMeterDS
crt.Exchange.oldCaCrtSubKeyId	Hex String

Process Orchestration

Having received the FlexSync notification and created a Service Request / Workflow, the actions that the Workflow must perform are described below:



#	Description
1	Following on from the initiation of the Workflow via FlexSync, a UAA Meter Update request should be sent to UDIS using the definition: MDM_I_METER_METER_CERT_RENEW
2	If the Meter Certificate / Key renewal process is a success in HES / Meter, a success UAA Response is received and should cause the MDM Workflow to make a 'status_update' request to KMS in order that KMS promotes the new Certificate to Active and

	Retires / Revokes the old.
3a	If the previous step have all succeeded then the Workflow should end, marking the Workflow Complete, with a status indicating a total Success.
3b	If the status_update to KMS failed, but the Certificate / Key update was successful, then the Workflow should end marked as Complete, with a status indicating the KMS record was not updated (allowing resolutory actions to be made manually).
3c	If the Meter Certificate / Key Update in the Meter Failed, then the Workflow should end marked as Complete, with a status indicating the Failure response received in the UAA Response.

new MDM FlexSync Definition: MDM_I_METER_METER_CERT_RENEW

An SDP SDPSyncMessage is sent to FlexSyncWS application by KMS Lifecycle Management in order to initiate the MDM Workflow that orchestrates the Meter Certificate replacement. The SDP Sync request creates meter parameters which contain details about the Meter Certificate requiring Replacement.

> ⚠ It's assumed that a required service request is derived on MDM side that's why it's not explicitly included in a request.

SDP Sync Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v8="http://www.
   <soapenv:Header/>
   <soapenv:Body>
      <v8:SDPSyncMessage>
         <v8:header>
            <v8:noun>SDPSync</v8:noun>
            <v8:revision>1</v8:revision>
            <v8:dateTime>2016-05-05T00:00:00</v8:dateTime>
            <v8:source>CIS</v8:source>
            <v8:messageID>0de28253-c9af-4676-b63b-7d30745e36f3</v8:messageID>
            <v8:asyncReplyTo>none</v8:asyncReplyTo>
            <v8:syncMode>sync</v8:syncMode>
            <v8:optimizationLevel>Full</v8:optimizationLevel>
         </v8:header>
         <v8:payload>
            <v8:device>
               <v8:mRID>ISK1050768076703</v8:mRID>
               <v8:type>Meter</v8:type>
               <v8:parameter>
                  <v8:name>crtExchange.exchangeType</v8:name>
                  <v8:value>renewMeterDS</v8:value>
                  <v8:startDate>2016-05-05T00:00:00</v8:startDate>
               </v8:parameter>
               <v8:parameter>
                  <v8:name>crt.Exchange.oldCaCrtSubKeyId</v8:name>
                  <v8:value>65841AB54565F65CD654654EF5654AA654654A654654E65465</v8:value>
                  <v8:startDate>2016-05-05T00:00:00</v8:startDate>
               </v8:parameter>
            </v8:device>
         </v8:payload>
      </v8:SDPSyncMessage>
```

```
   </soapenv:Body>
</soapenv:Envelope>
```

Attribute	Data Type	Required	Configurable	Source System	Values	Descript
//SDPSyncMessage/header/						
noun	string	Y	N	KMS	SDPSync	
revision	string	Y	N	KMS	1	
dateTime	date	Y	N	KMS	<current time>	
source	string	Y	N	KMS	CIS	
messageID	string	Y	N	KMS	<UUID>	
asyncReplyTo	string	Y	N	KMS	none	
syncMode	string	Y	N	KMS	sync	
optimizationLevel	string	Y	N	KMS	Optimistic	
//SDPSyncMessage/payload/device/						
mRID	string	Y	N	KMS	<Entity ID>	
type	string	Y	N	KMS	Meter	
//SDPSyncMessage/payload/device/parameter						
name	string	Y	N	KMS	crtExchange.exchangeType crt.Exchange.oldCaCrtSubKeyId	
value	string	Y	N	KMS	<parameter value>	
startDate	date	Y	N	KMS	<current time>	

SDP Sync Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v8="http://www.
   <soapenv:Header/>
   <soapenv:Body>
      <v8:SDPSyncReplyMessage>
         <v8:header>
            <v8:verb>?</v8:verb>
            <v8:noun>?</v8:noun>
```

```
            <v8:revision>1</v8:revision>
            <v8:dateTime>2016-05-05T00:00:00</v8:dateTime>
            <v8:source>?</v8:source>
            <v8:messageID>e40153ba-ea99-4cb4-9733-270a84a08893</v8:messageID>
         </v8:header>
         <v8:reply>
            <v8:replyCode>0</v8:replyCode>
            <v8:replyText>?</v8:replyText>
            <v8:correlationId>0de28253-c9af-4676-b63b-7d30745e36f3</v8:correlationId>
         </v8:payload>
      </v8:SDPSyncMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Attribute	Data Type	Required	Source System	Values	Description
//SDPSyncReplyMessage/reply/					
replyCode	string	Y	MDM	0 | 1 | etc.	
replyText	string	Y	MDM		
correlationId	string	Y	MDM	<request: //SDPSyncMessage/header /messageID>	

new Solution Proposal (UDIS): Meter Certificate Renewal Support

> **Note:**
> Several sections of this page refer to 'Slots', for a explanation of this terminology see here: new Security Suite 1 - Meter (Server) Certificate Support

UDIS will receive a request to initiate the required actions to perform a renewal and subsequent activation of a new Asymmetric Digital Signing Meter Key. These requests will be delivered to UDIS and Responses expected via the Interface described here: new UAA to UDIS Adapter Interface Extension for Meter Certs: MeterAssetUpdateRequestMessage

From this request, actions relating renewal of the Meter's Keys will be requested based on the value passed in the crt.Exchange, exchangeType:

Exchange Type	Description
renewMeterDS	Renew the Meter's Digital Signing Asymmetric Keys, including replacing the relevant Issuing Authority Certificate if required.

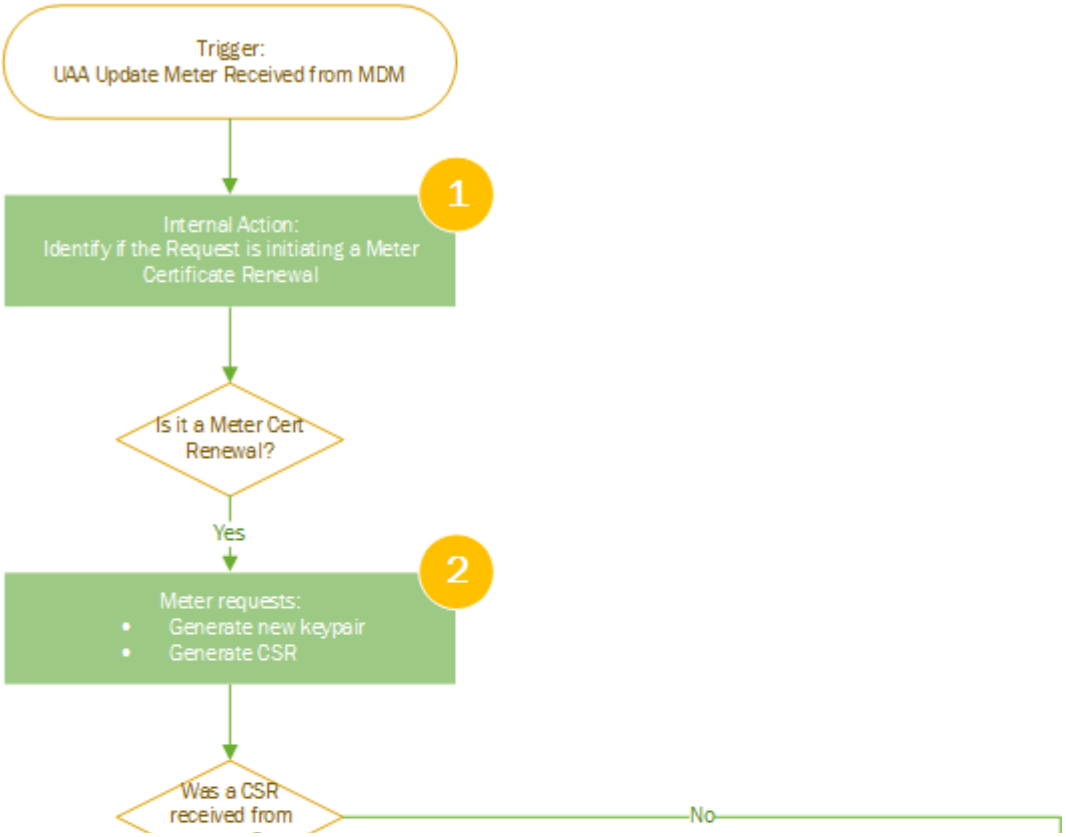
The renewal of a Meter Certificate in the Meter is a multi-step process involving several Meter interactions. Responsibility for the orchestration of these actions is an UDIS function.

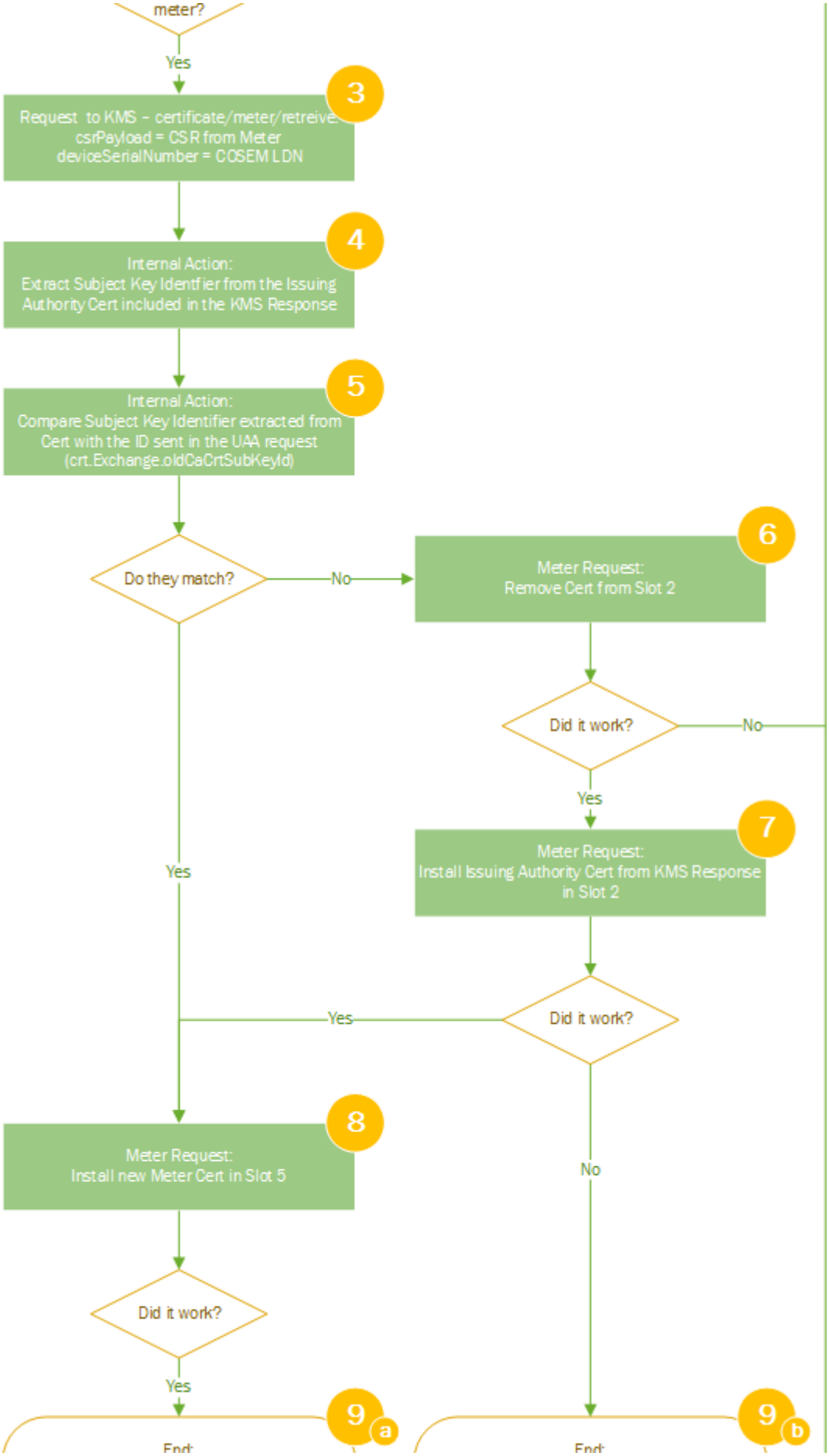
Process

UDIS is responsible for taking the information provided in the request from MDM and initiating the action, UDIS is therefore responsible for:

- Identification of which COSEM Client should be used to perform the action
- Determination of whether the Issuing Authority Certificate should be updated
- Making the Generation / Installation / Removal requests to the Meter, including negotiating Authentication and Encryption mechanisms (with KMS REST services supporting)
- Submitting the Meter CSR to the Central System for processing
- Providing a response to MDM indicating the success / failure status of the requested action

The required overall UDIS process logic can be summarised as below, however this can be implemented in whatever way most efficiently fulfills all the potential paths through this flow:




meter?
Yes
3
Request to KMS - certificate/meter/retreive:
csrPayload = CSR from Meter
deviceSerialNumber = COSEM LDN
4
Internal Action:
Extract Subject Key Identfier from the Issuing Authority Cert included in the KMS Response
5
Internal Action:
Compare Subject Key Identifier extracted from Cert with the ID sent in the UAA request (crt.Exchange.oldCaCrtSubKeyId)
Do they match?
No
6
Meter Request:
Remove Cert from Slot 2
Did it work?
No
Yes
7
Meter Request:
Install Issuing Authority Cert from KMS Response in Slot 2
Yes
Did it work?
Yes
8
Meter Request:
Install new Meter Cert in Slot 5
No
Did it work?
Yes
9
a
9
b

Step #	Description	Supporting Information
1	Receive via UAA, a request to initiate the Meter Key / Certificate renewal process.	new UAA to UDIS Adapter Interface Extension for Meter Certs: MeterAssetUpdateRequestMessage
2	Orchestrate the required process to request that the meter goes through all neccessary steps to generate a new Digital Signing Keypair and produce a Certificate Signing Request based on those Keys.	
3	Submit the CSR created by the Meter to the Central System PKI, via KMS. The deviceSerialNumber should be the COSEM Logical Device Name of the Meter whose Key / Certificate is being renewed.	new KMS REST Service Definition: Extension of ../certificate/meter/retrieve
4 / 5	KMS will respond to the previous step with a Certificate issued by the PKI. This certificate will be presented as a Certificate Chain. The Meter Certificate cannot be installed unless the related Issuing Authority Certificate is installed in the Meter. To establish whether this is required UDIS needs to extract the Authority Key Identifier from the Meter Certificate and compare it with the value sent in the UAA Request field: crt.Exchange.renewMeterDS. If these values do not match then it indicates that before the Meter will accept the installation of the Meter Certificate, the Issuing Authority Certificate in Slot 2 must first be updated. An alternative comparison is to extract the Subject Key Identifier from the Issuing Authority Certificate contained in the Chain.	new KMS REST Service Definition: Extension of ../certificate/meter/retrieve
6	If the previous step identified that an update of the Issuing Authority Certificate is required, then the first step of that process is to first make suitable requests to the Meter to remove the Certificate stored in Slot 2. The new Issuing Authority Certificate cannot be installed without first removing the old from the Meter memory.	
7	Assuming the previous step succeeded; the Issuing Authority Certificate should be extracted from the Chain received in response to step #3 and the neccessary requests made to the Meter to install the new Issuing Authority Certificate in Slot 2.	
8	If no update of the Issuing Authority Certificate in Slot 2 was required, or of it was successfully updated in previous steps, then UDIS should make the neccessary requests to the Meter to install the New Meter Key. Unlike the Issuing Authority Certificate, the old Certificate does not need to be removed beforehand.	
9a	If all actions have been successful, and the Meter Keys / Certificate have been successfully updated in the Meter, then a UAA 'Success' response should be sent to MDM.	new UAA to UDIS Adapter Interface Extension for Meter Certs: MeterAssetUpdateRequestMessage
9b	If there has been a failure during the process, or if the status of the update in the Meter is unknown, then an appropriate UAA 'Failure' response should be sent to MDM.	new UAA to UDIS Adapter Interface Extension for Meter Certs: MeterAssetUpdateRequestMessage

new UAA to UDIS Adapter Interface Extension for Meter Certs: MeterAssetUpdateRequestMessage

⚠️ WIP: Copied from another section ⚠️

The MeterAssetUpdateRequestMessage interface is an existing Interface between MDM and UDIS, it is documented here: [UDIS Schnittstellenbeschreibung]

The MeterAssetUpdateRequestMessage interface is currently used to trigger actions in UDIS that result in something being updated in the Meter. The definition will be extended to give provision for sufficient data to be passed to UDIS to enable the management of Meter Certificates in the Meter.

The proposed extensions use the following as a Baseline:

- UDIS V6.0 Adapters V1.8
  - Section 3.5 UAA Provisioning
    - Updating a Meter: Request
    - Updating a Meter: Response

Request Extension: Additional Parameters

Six new parameters will be included to support request of HES Certificate actions, and provide HES with all neccesary information to initiate the requested process:

> **Note:**
> All parameters are optional within the context of a MeterAssetUpdateRequestMessage, however for a Renew Meter Certificate initiating request; they are mandatory.

Parameter Name	Parameter Value	Required	Description
crtExchange.exchangeType	String	N	This parameter denotes the action being requested of HES: • renewMeterDS - Triggers the process to renew the Digital Signing Key / Certificate in the Meter • renewMeterKA - Triggers the process to renew the Key Agreement Key / Certificate in the Meter
crt.Exchange. oldCaCrtSubKeyId	String	N	This parameter provides the Subject Key Identifier of the Meter Certificate's Issuing Authority Certificate currently installed in the Meter.

Reply Codes

The following responses (or very close equivalents) are to be provided as a minimum. More can be added to the list if they are available and giving more detailed / granular responses is also acceptable.

> **Note:**
> Non-zero replyCodes can be assigned as required

replyCode	replyText
0	Success
	Meter offline
	Meter association could not be established
	No response from Meter
	Request failed at Meter

new Security Suite 1 - HLS7 Authentication

For all HES -> Meter communcations, the COSEM Association must be established using HLS7 authentication. HLS7 is a mutual authentication mechanism based on generation of a digital signature (ECDSA). Generation of a digital signature requires use of a private key. Verification of the signature requires the use of a previously distributed public key (distributed via certificate).

The process goes roughly:

1. Both Meter and HES generate and send each other a Challenge, KMS provides an existing Service to generate a Challenge of the appropriate length (rest/crypto/getChallenge)
2. Each party then concatenate each other's Challenges, and each other's System Titles to create a string that they generate an ECDSA signature for. These signatures are then exchanged.
3. Each party is able to construct the string that formed the input to the signature generation of the other party. This is used in conjunction with the public key of the sender and the signature they sent to verify the signature, and hence the authenticity of the sender.

HES does not have access to private or public key material, so will rely on KMS to provide functions to support the above.

new Solution Proposal (KMS): Provide Signing and Signature Verification Services

Two new KMS REST Services will be exposed to provide the necessary support functions to UDIS:

- cosemGenerateSignature - new KMS REST Service Definition: cosemGenerateSignature
- cosemVerifySignature - new KMS REST Service Definiton: cosemVerifySignature

new KMS REST Service Definition: cosemGenerateSignature

This KMS REST Service is used to expose the Worldline JSS Function:

- **JSS Service:** EnergyService

- **Function:** cosemGenerateSignature

The Worldline function consumes the following inputs and these must be derived by the KMS Service or provided directly in the REST Service request:

- **securitySuite:** Which DLMS Security Suite should be used
- **signatureKey:** The Label of the Private HES Key to be used in Signature Generation
- **dataToBeSigned:** The payload of data to be signed

Request Format

Request:

```
{
  "deviceSerialNumber" : KFE503222912,
  "manufacturer" : "Kaifa",
  "oeRoleIdentifier" : "Management",
  "securitySuite" : "SUITE_1",
  "payloadHex" : "000102030405060708090A0B0C0D0E0F"
}
```

Element descriptions:

Request element	Required	Description	Type	Value / Example
deviceSerialNumber	Mandatory	The Meter Serial Number (normally COSEM System Title). Used in conjunction with other fields to establish the 'signatureKey'.	String	KFE503222912
manufacturer	Optional	This is the Key Issuer associated with the HES Certificate related to the HES Key that should be generate the Signature.	String	Kaifa
oeRoleIdentifier	Mandatory	The COSEM Role the signature should be generated in the context of. Used in conjunction with other fields to establish the 'signatureKey'.	Enumerated	Management || Readout || PLC Management
securitySuite	Mandatory	The security suite the signature shouldbe generated in the context of.	Enumerated	SUITE_1 || SUITE_2
payloadHex	Mandatory	The payload to generate the Signature for.	String	000102030405060708090A0B0C0D0E0F

signatureKey / Key Label derivation:
The Label of the Key used as the signatureKey value can be established by the KMS, because the following are true:

- HES Certificate records in KMS contain the Key Label of their 'owning Key'
- HES Certificate records in KMS are assigned to Roles
- Relationships between Meters and the HES Certificates installed in them are maintained in KMS

Use of Manufacturer element:
The Manufacturer element can be used as an additional filter where Meter Serial Number is non-unique between manufacturers. Normally this is not a problem because COSEM System Title contains a manufacturer code. By creating a specific Key Issuer per manufacturer, uniqueness of Certificate and Relationship records can be retained.

Error Handling:
If the query to establish signatureKey / Key Label yields multiple results, then an error should be thrown and no signature generated.

Response Formats

Success Response:

```
{
  "resultCode": "OK"
  "signatureHex": "714A286D976BF3E58D9D671E37CBCF7C",
}
```

Element Descriptions:

Request element	Required	Description	Type	Value / Example
signatureHex	Mandatory	The ECDSA signature (r,s) is returned as binary concatenation of r and s.	Hex String	`714A286D976BF3E58D9D671E37CBCF7C` 2912

Handled Failure Response:

```
{
   "resultCode": "MULTIPLE_KEYS",
```

```
   "resultDescription": "Multiple Signing Keys identified"
}
```

Element Descriptions:

Request element	Required	Description	Type	Value / Example
resultCode	Mandatory	Result code indicating a problem	String	`MULTIPLE_KEYS`
resultDescription	Mandatory	Human readable description	String	`Multiple Signing Keys identified`

Expected Failures:

resultCode	resultDescription
MULTIPLE_KEYS	Multiple Signing Keys identified
NO_KEY	No Signing Key identified
INCORRECT_KEY	Signing Key Length incorrect
PARAM_MISSING	Mandatory Parameter Missing

Unhandled Error Response:

```
{
   "errorDescription": "xyz",
   "errorCode": "XXX-99999999"
}
```

Element Descriptions:

Request element	Required	Description	Type	Value / Example
errorDescription	Mandatory	Human readable error description	String	`xyz`
errorCode	Mandatory	Technical error code related to the description	String	`XXX-99999999`

new KMS REST Service Definiton: cosemVerifySignature

A request to this Service results in the provided Signature being verified. It does not require the use of the Worldline JSS as no private key material is use in the verification of a Signature. The following information is required to be able to verify a signature:

- Public Key
- Information about the cryptographic curve and hash algorithm used in Signature Generation
- The Signature
- The Payload the Signature was generated for

Request Format

Request:

```
{
  "deviceSerialNumber" : KFE503222912,
  "manufacturer" : "Kaifa",
  "securitySuite" : "SUITE_1",
  "payloadHex" : "000102030405060708090A0B0C0D0E0F"
  "signatureHex" : "714A286D976BF3E58D9D671E37CBCF7C"
}
```

Element descriptions:

Request element	Required	Description	Type	Value / Example
deviceSerialNumber	Mandatory	The Meter Serial Number (normally COSEM System Title).	String	KFE503222912
manufacturer	Optional	This is the Key Issuer associated with the HES Certificate related to the HES Key that should be generate the Signature.	String	Kaifa
securitySuite	Mandatory	The encryption suite the Signature was generated in the context of.	Enumeration	SUITE_1 || SUITE_2
payloadHex	Mandatory	The payload to generate the Signature was generated for.	Hex String	000102030405060708090A0B0C0D0E0F
signatureHex	Mandatory	The Signature for verification.	Hex String	714A286D976BF3E58D9D671E37CBCF7C

Public Key Derivation:
The Public Key to use in the Signature Verification can be extracted from the Meter's Signing Certificate, stored in KMS_CERT. It can be identified using the deviceSerialNumber from the request:

- Owning entity ID = Device Serial Number
- Owning entity type = Meter
- ZertifikatTyp = CRT_ECC
- Verbrauch = Sign
- Status = Activ

Cryptographic Curve derivation:
The Cryptographic Curve to use in Verification operations can be derived from the encryptionSuite value provided in the request:

- 1 = P-256; AES 128
- 2 = P-384; AES 256

Response Formats

Response:

```
{
  "resultCode": "OK"
  "result": "true|false"
}
```

Element descriptions:

Request element	Required	Description	Type	Value / Example
resultCode	Mandatory	Result code indicating success	String	OK
result	Mandatory	The result of the signature verification.	Boolean	True = Signature verified successfully False = Signature did not verify

Handled Failure Response:

```
{
  "resultCode": "MULTIPLE_CERTS"
  "resultDescription": "Multiple certificates identified"
}
```

Element Descriptions:

Request element	Required	Description	Type	Value / Example
resultCode	Mandatory	Result code indicating a problem	String	`MULTIPLE_CERTS`
resultDescription	Mandatory	Human readable description	String	`Multiple certificates identified`

Expected Failures:

resultCode	resultDescription
MULTIPLE_CERTS	Multiple certificates identified
NO_CERT	No certificate identified
INCORRECT_CERT	Key Length incorrect
PARAM_MISSING	Mandatory Parameter Missing

Unhandled Error Response:

```
{
   "errorDescription": "xyz",
   "errorCode": "XXX-99999999"
}
```

Element Descriptions:

Request element	Required	Description	Type	Value / Example
errorDescription	Mandatory	Human readable error description	String	`xyz`
errorCode	Mandatory	Technical error code related to the description	String	`XXX-99999999`

new Solution Proposal (UDIS): Replace HLS5 Authentication with HLS7

For Honeywell Meters, the existing HLS5 based Client Authentication will no longer be acceptable. Client Authentication for all Associations established with the Meter via HES must use HLS7.

UDIS functionality must be modified appropriately, using the the KMS Support Functions: [new Solution Proposal (KMS): Provide Signing and Signature Verification Services]

new Security Suite 1 - Key Agreement Key Exchange Mechanism

In previous implementations, key exchange / replacement for symmetric COSEM keys has been performed using Key-Wrap mechanisms. This provides a simple end-to-end process of:

1. Generate a new key
2. Wrap the new key with the meter KEK
3. Send the wrapped new key to the meter

The Honeywell meter uses capabilities provided by Security Suite 1 to make the key renewal process more secure, using the Key Agreement mechanism *'Ephemeral Unified Model C(2e, 0s, ECC CDH)'*. This process avoids the need to ever send the actual final key between Client and Server. In this process data is exchanged resulting in the ability of both Client and Server to compute the same Key, this Key is then stored by both Parties and becomes the new symmetric key.

The Worldline JSS provides two functions to support the process, which is broadly a 2-step process, where the output of the first is used to supplement the inputs to the second.

Re-Use of Existing Key Management Functionality

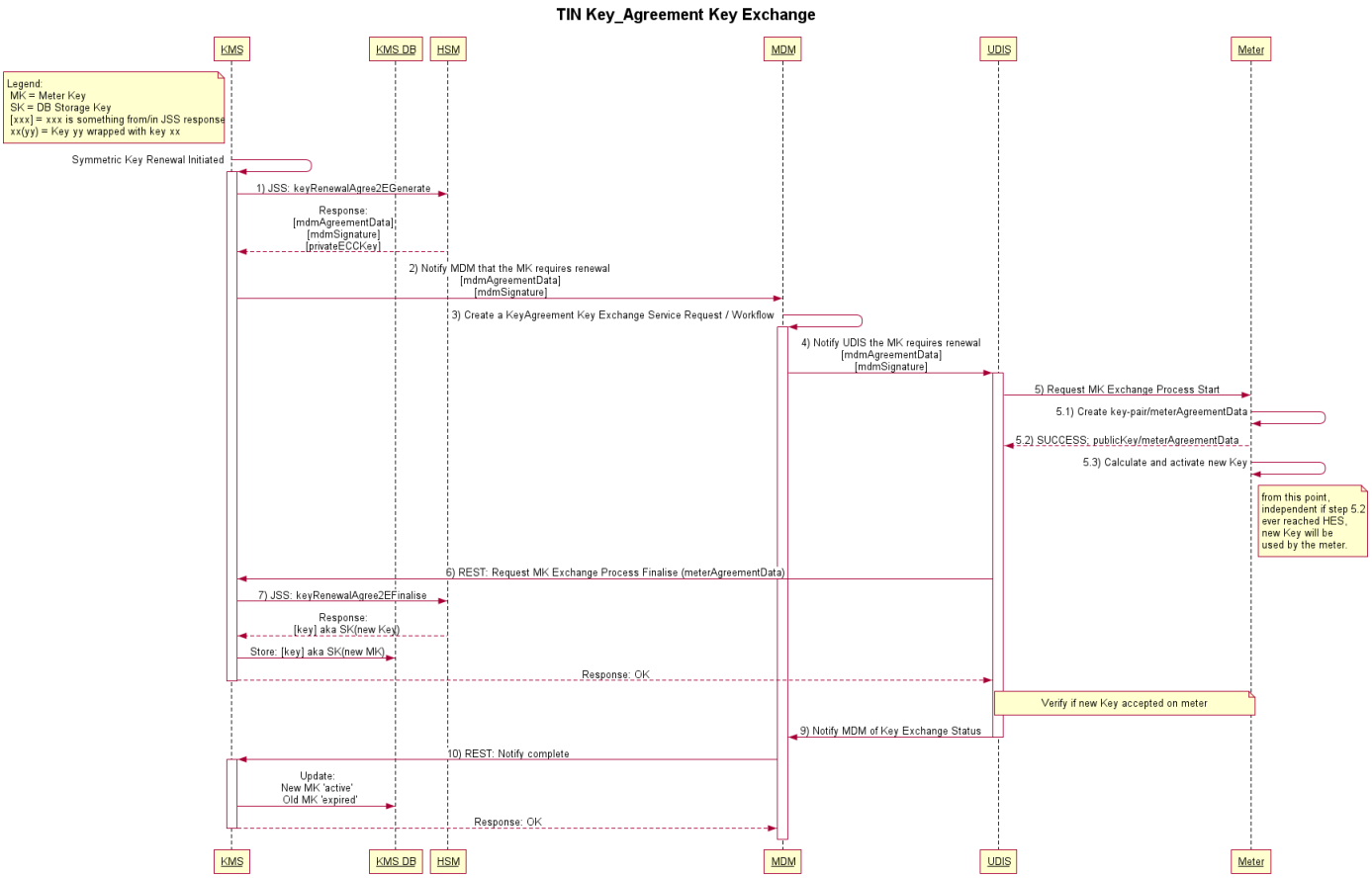
Existing KMS mechanisms to enable a Key replacement to be initiated should be retained, including initiation by:

- Lifecycle
- Direct Request

Also, existing rollback capabilities, for example the KMS functionality where both old and new Keys are retained until an explicit confirmation of the successful adoption of the new Key is received should also be retained. In essence the process between initiation of the renewal request, and the notification to KMS of a successful swap, should be different for Honeywell Meters, the processes outside that section should remain the same as they are for existing Key Exchange Key Replacement processes.

Overall Process

The overall process is shown below as a sequence diagram. The detail of individual steps is discussed further in specific sub-sections on a per-component basis.



Process Considerations

This overall process has been designed with a view to:

- Pro-actively providing as much information to HES as possible, reducing the number of HES initiated interactions with MDM / KMS
- Enabling MDM to hold enough information about the Replacement operation that MDM records can be used for success / failure reporting

new Solution Proposal (KMS): Implementation of KeyAgreement Key Exchange

The KMS already has mechanisms in place that are used to enable centrally controlled replacement of DLMS COSEM Symmetric Keys, however these may need extending or modifying to varying degrees to cater for the different Key Replacement mechanism used by the Honeywell Meters.

Initiation of Key Replacement

The KMS already provides two mechanisms for the initiation of Key Replacement in Meters:

- Directly requesting renewal
- Monitoring Lifetime parameters (defined in a Key Template) and initiating renewal based on them

These mechanisms should continue to be used for Replacement initiation, and should continue operating in the same way.

Existing Replacement Processes need to be extended so that:

- Upon renewal initiation a check in made to find which 'type' of renewal is required.
- Based on the information determined in the step above, a new KMS Workflow is initiated where the 'type' is Key Agreement.

Identification of Required Renewal Type

Once a Symmetric Key has been identified as requiring renewal, the KMS needs to be able to determine whether the Key should be renewed using the existing Key Exchange protocol, or the Key Agreement protocol. It is a necessary requirement that the KMS be able to support both protocols simultaneously.

The existing KMS Data Architecture allows an individual Key Record to always be linked back to a Key Template, therefore this requirement can be fulfilled via the addition of a new field added to the Key Templates:

Field	Enumerations
Replacement Protocol	• Key Exchange • Honeywell Key Agreement

Parallel Replacement Restrictions

Controls should be put in place, or existing controls extended to ensure that:

- Only one Symmetric Key is replaced at a time
- Meter Certificate renewal processes should not overlap Symmetric Key Replacement operations
- Operations for the updating of HES Certificates in the Meter should not overlap Symmetric Key Replacement operations

KeyAgreement KMS Workflow

Unike the Key Exchange protocol, the Key Agreement Protocol is not 'fire-and-forget' from a KMS perspective. The Key Agreement process involves a two-step cryptographic process, with the response to step 1 feeding into the request for step 2 (supplemented with additional information).

Because of this, the KMS will need a more advanced workflow to be created upon initiation of the Key Agreement Replacement request, rather than a simple Service Request.

The overall positioning of this workflow can be seen on the overview page, with more detailed descriptions of the KMS and HES interactions given below: new Security Suite 1 - Key Agreement Key Exchange Mechanism

Overall process step	Description
1	Upon initiation of a Key Agreement replacement, a KSM workflow is created.

	The first action of this workflow is to internally call the JSS function keyRenewalAgree2EGenerate, either internally or by using the KMS REST Service defined here: new KMS REST Service Definition: keyRenewalAgree2EGenerate The keyRenewalAgree2EGenerate request provides three pieces of information in its response: • mdmAgreementData • mdmSignature • privateECCKey The privateECCKey part of the response is the part of this request that is used directly to feed into the secondary stage (step #7) and is hence held in the Workflow.
2	The Workflow initiated in step #1 makes a request to MDM to broker a Key Replacement request onwards to HES. This request is made via FlexSync according to the definition here: new MDM FlexSync Definition: MDM_I_METER_KEY_AGREEMENT_KEY_REPLACE The mdmAgreementData and mdmSignature values from the JSS response in step #1 are included in the MDM request.
7	MDM, HES and the Meter do what they need to do with the content of the Key Replacement request and as a result HES makes a request to the KMS REST Service defined here: new KMS REST Service Definition: keyRenewalAgree2EFinalise (step #6). As a result of this REST Service Request, the KMS Workflow must make a request to the JSS function : keyRenewalAgree2EFinalise, it is this step that requires the privateECCKey to be extracted from the Workflow for inclusion in the secondary stage of the cryptographic process. The response to this JSS request is the 'Key Agreed' new Symmetric Key, wrapped with the KMS DB Storage Key. This should be stored in the KMS DB, following the usual process where both New and Old Keys are stored pending final confirmation of successful Replacement. The KMS Workflow sends a simple 'OK' Response to HES to tell it that KMS-side processing has completed and the process in the Meter can be continued.
10	HES and the Meter do what they need to do to set the new 'Agreed' Key to 'Active' in the Meter, and then notify (via MDM) the KMS (via the existing keyUpdate REST Service) of the Successful Replacement. This triggers the KMS to mark the New Key as 'Active' and end the Old Key records appropriately.

new KMS REST Service Definition: keyRenewalAgree2EGenerate

Calling this REST Service results in a subsequent request being made to the Worldline JSS:

- **JSS Service:** EnergyService
- **Function:** CosemKeyRenewalAgree2EGenerate

This page is split into two sections; the first defines the REST Service itself, the second describes the JSS Function and the actions that must be triggered upon calling the REST Service.

Request Format

Request:

```
{
  "deviceSerialNumber" : KFE503222912,
  "oeRoleIdentifier" : "Default",
  "securitySuite" : "SUITE_1",
  "keyType" : "KEK"
}
```

Element descriptions:

Request element	Required	Description	Type	Value / Example
deviceSerialNumber	Mandatory	The Meter's COSEM Logical Device Name.	String	KFE503222912
oeRoleIdentifier	Mandatory	The COSEM Role of the Key being renewed.	Enumerated	Default || Management || Readout || PLC Management || Installation || Certification || Maintenance || CIP
securitySuite	Mandatory	The security suite the renewal should be performed in the context of.	Enumerated	SUITE_1 || SUITE_2
keyType	Mandatory	The type of Key being replaced.	Enumerated	KEK || GUEK || GAK

Response

{

"resultCode": "OK"

"mdmAgreementData": "length 65 || 97 bytes"

"mdmSignature": "06F0607702AA0E2435A183E2F6B1ECD19629712E389A213610C03F77B2590860EA840AF5C3FA1F2BCDF055D4744E9A01CE9A0E55026BCAA4I
aka 64 bytes"

}

Handled Failure

```
{
  "resultCode": "Unknown Key | Unknown Device | etc"
  "resultDescription": "tbd"
}
```

Unhandled Failure

```
{
  "errorCode": "XXX-99999999",
  "errorDescription": "Something went wrong"
}
```

JSS Function Overview

The Function requires the input parameters below:

JSS Parameter	Description
securitySuite	The DLMS COSEM Security Suite to use
keyID	The type of Key being renewed
privateEccMdmSigningKey	The Key Label of the Private Key of the HES Client to be used to renew the target Key
mdmStorageKey	The Key Label of the DB Storage Key

Derivation of JSS inputs:
The links between the REST Service input and the JSS Inputs are described below:

JSS Input	Derived From KMS REST Service Parameter(s)	Notes
securitySuite	securitySuite	1 → SecuritySuite.SUITE1 2 → SecuritySuite.SUITE2
keyID	keyType	GUEK → 0x00

		GAK → 0x02 KEK → 0x03
privateEccMdmSigningKey	deviceSerialNumber && oeRoleIdentifier && keyType	All three KMS parameters are used to identify the Key Template of the Key being replaced. In turn this allows the Renewal Role to be established. In turn that allows the 'active' HES Certificate for the Renewal Role to be established. The HES Certificate record contains the Key Label to use, stored as a Parameter.
mdmStorageKey	deviceSerialNumber && oeRoleIdentifier && keyType	All three KMS parameters are used to identify the Key Template of the Key being replaced. The Key Template contained the DB Storage Key Label to use.

new KMS REST Service Definition: keyRenewalAgree2EFinalise

Calling this KMS REST Service results in a subsequent request being made to the Worldline JSS:

- **JSS Service:** EnergyService
- **Function:** CosemKeyRenewalAgree2EFinalise

This page is split into two sections; the first defines the REST Service itself, the second describes the JSS Function and the actions that must be triggered upon calling the REST Service.

REST Service Definition

Request Format

Request:

```
{
  "deviceSerialNumber" : KFE503222912,
  "oeRoleIdentifier" : "Default",
  "securitySuite" : "SUITE_1",
  "keyType" : "KEK"
  "ephemeralEccPubKeyForSmAgreementData" : "2914D60E10AB705F62ED6CC349D7CB99B9AB3F3978E59278C7AF5
  "smSignature" : "A92995225CEE004ED4376057EEE9536E97EE6F5BAE43E59BDBBD515A89FB2CB83F2A270871A31B
```

```
  "otherInfo" : "608574060803004D4D4D0000BC614E4D4D4D0000000001"
}
```

Element descriptions:

Request element	Required	Description	Type	Value / Example
deviceSerialNumber	Mandatory	The Meter's COSEM Logical Device Name. Used in conjunction with other fields to establish the 'signatureKey'.	String	KFE503222912
oeRoleIdentifier	Mandatory	The COSEM Role of the Key being renewed.	Enumerated	Default || Management || Readout || PL
securitySuite	Mandatory	The security suite the renewal should be performed in the context of.	Enumerated	SUITE_1 || SUITE_2
keyType	Mandatory	The type of Key being replaced.	Enumerated	KEK || GUEK || GAK
ephemeralEccPubKeyForSmAgreementData	Mandatory	As described in JSS Function Overview. The Meter's Ephemeral Public Key	HEX String	2914D60E10AB705F62ED6CC349D7CB
smSignature	Mandatory	As described in JSS Function Overview. The Signature generated by the Meter.	HEX String	A92995225CEE004ED4376057EEE9536
otherInfo	Mandatory	As described in JSS Function Overview. A concatenation of information.	HEX String	608574060803004D4D4D0000BC614E4

JSS Function Overview

The JSS Function requires the input parameters below:

JSS Parameter	Description
securitySuite	Specification of the DLMS/COSEM Security Suite. This also defines the accepted ECC curve and the length of the generated EEK according to [COSEM-GREEN-81].
keyID	This member contains the key-ID of the key to agree.
privateEccKey	This member contains the encrypted MDM ephemeral ECC private key which was generated by this function in the first phase generate and returned as result.
ephemeralEccPubKeyForSmAgreementData	This ephemeral ECC public key given as byte array will be prefixed with the given key-ID (member keyID), this will result in the key agreement data (FE2OS(xp) || FE2OS(yp) representation of COSEM-8.1 chapter 9.2.3.4.3.6).
smSignature	This value is the signature over the key agreement data (key-ID and the SM ephemeral ECC public key) generated by Smart Meter. The data representation is like this in COSEM as octet string without any leading zeros.
caCertificate	The Key Label of the Root CA of the PKI that issues Meter and HES Certificates.
certificateChain	The Certificate of the Meter, in chain form.
otherInfo	Concatenation of data as described in COSEM 8.1 chapter 9.2.3.4.6.5 Key Derivation Function - Table 17 OtherInfo. This data is a concatenation of the following data: • AlgorithmID according COSEM 8.1 chapter 9.2.3.4.6.5 Key Derivation Function - Table 18 Security algorithm ID-s as 7 byte encoded value (e.g. '60857406080300'H for AES-GCM-128). • System Title Client (SysTC) normally coded in 8 byte (e.g. '4D4D4D0000BC614E' H) • System Title Server (SysTS) normally coded in 8 byte (e.g. '4D4D4D0000000001' H) The length of this concatenated data is defined in COSEM and fixed at 23 bytes.
mdmStorageKey	The Key Label of the DB Storage Key

Derivation of JSS inputs:

The links between the REST Service input and the JSS Inputs are described below:

JSS Input	REST Service Inputs Derived From KMS Parameter(s)	Notes
securitySuite	securitySuite	1 → SecuritySuite.SUITE1 2 → SecuritySuite.SUITE2
keyID	keyType	GUEK → 0x00 GAK → 0x02 KEK → 0x03
privateEccMdmSigningKey	deviceSerialNumber && oeRoleIdentifier && keyType	All three KMS parameters are used to identify the Key Template of the Key being replaced. In turn this allows the Renewal Role to be established. In turn that allows the 'active' HES Certificate for the Renewal Role to be established. The HES Certificate record contains the Key Label to use.
ephemeralEccPubKeyForSmAgreementData	ephemeralEccPubKeyForSmAgreementData	Direct pass-through from REST Parameter.
smSignature	smSignature	Direct pass-through from REST Parameter.
caCertificate	deviceSerialNumber && oeRoleIdentifier	The two KMS parameters noted allow the identification of the Issuing Authority Certificates used in the construction of certificateChain. The Issuing Authority Certificate has a parameter associated with it: ParentKeyLabel. This can be used for caCertificate.
certificateChain	deviceSerialNumber && oeRoleIdentifier	

		This JSS input is derived from the Meter Certificate and it's related Issuing Authority Certificate. The two KMS parameters noted, in conjunction with Status and Usage, allow the identification of the relevant Meter Cert in KMS_CERT table. Once the Meter Cert is identified, its associated Authority Key Identifier can be used to identify the Issuing Authority Certificate.
otherInfo	otherInfo	Direct pass-through from REST Parameter.
mdmStorageKey	deviceSerialNumber && oeRoleIdentifier && keyType	All three KMS parameters are used to identify the Key Template of the Key being replaced. The Key Template contains the DB Storage Key Label to use.

new Solution Proposal (MDM): Meter Key Replacement via Key Agreement Workflow / SR

Upon receipt of a Key Agreement Key Renewal Replacement request (via the FlexSync interface defined here: new MDM FlexSync Definition: MDM_I_METER_KEY_AGREEMENT_KEY_REPLACE), a simple MDM Workflow should be initiated.

The overall positioning of this workflow can be seen on the overview page, with more detailed descriptions of the MDM and HES interactions given below: Security Suite 1 - Key Agreement Key Exchange Mechanism

Overall process step	Description
4	Upon initiation (via FlexSync) of the Key Agreement Key Renewal Workflow, the Key Replacement request should be brokered onwards to HES using the Update Meter Provisioning Interface described here:new UAA to UDIS Adapter Interface Extension for Key Agreement Key Exchange: MeterAssetUpdateRequestMessage The SDP Parameters populated by KMS in relation to the Key Agreement Key Renewal request are passed through to HES in this request.
10	HES, the Meter and KMS do what they need to do 'Agree' the Replacement Key at both ends and HES sends a positive confirmation of the New Key activation in the Meter in the form of a 'success' UAA Response (step #9). The MDM Workflow can then mark the Service Request indicating the request to HES as complete, and finally notify KMS of the successful 'activation' of the New Key in the Meter via the 'Key Update' interface defined here: KMS_I_KEY_RENEWAL

new MDM FlexSync Definition: MDM_I_METER_KEY_AGREEMENT_KEY_REPLACE

An SDP SDPSyncMessage is sent to FlexSyncWS application by KMS Lifecycle Management in order to initiate the MDM Workflow that orchestrates (in conjunction with HES) the process of a Symmetric Meter Key Renewal using the Key Agreement Mechanism. The SDP Sync request creates meter parameters which contain details required by HES to begin the process in the Meter.

⚠ It's assumed that a required service request is derived on MDM side that's why it's not explicitly included in a request.

SDP Sync Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v8="http://www.
   <soapenv:Header/>
   <soapenv:Body>
      <v8:SDPSyncMessage>
         <v8:header>
            <v8:noun>SDPSync</v8:noun>
            <v8:revision>1</v8:revision>
            <v8:dateTime>2016-05-05T00:00:00</v8:dateTime>
            <v8:source>CIS</v8:source>
            <v8:messageID>0de28253-c9af-4676-b63b-7d30745e36f3</v8:messageID>
            <v8:asyncReplyTo>none</v8:asyncReplyTo>
            <v8:syncMode>sync</v8:syncMode>
            <v8:optimizationLevel>Full</v8:optimizationLevel>
         </v8:header>
         <v8:payload>
            <v8:device>
               <v8:mRID>ISK1050768076703</v8:mRID>
               <v8:type>Meter</v8:type>
               <v8:parameter>
                  <v8:name>KeyExchange.keyType</v8:name>
                  <v8:value></v8:value>
```

```
                <v8:startDate>2016-05-05T00:00:00</v8:startDate>
             </v8:parameter>
             <v8:parameter>
                <v8:name>KeyExchange.role</v8:name>
                <v8:value>Management</v8:value>
                <v8:startDate>2016-05-05T00:00:00</v8:startDate>
             </v8:parameter>
             <v8:parameter>
                <v8:name>KeyExchange.AgreementDataPart1</v8:name>
                <v8:value>06F0607702AA0E2435A183E2F6B1ECD19629712E389A213610C03F77B2590860EA840
                <v8:startDate>2016-05-05T00:00:00</v8:startDate>
             </v8:parameter>
             <v8:parameter>
                <v8:name>KeyExchange.AgreementDataPart2</v8:name>
                <v8:value>A0E55026BCAA4EEBEB764CED64BB3</v8:value>
                <v8:startDate>2016-05-05T00:00:00</v8:startDate>
             </v8:parameter>
             <v8:parameter>
                <v8:name>KeyExchange.SignaturePart1</v8:name>
                <v8:value>2914D60E10AB705F62ED6CC349D7CB99B9AB3F3978E59278C7AF595B3AF987941372D
                <v8:startDate>2016-05-05T00:00:00</v8:startDate>
             </v8:parameter>
             <v8:parameter>
                <v8:name>KeyExchange.SignaturePart2</v8:name>
                <v8:value>F43414611058D1B48F894</v8:value>
                <v8:startDate>2016-05-05T00:00:00</v8:startDate>
             </v8:parameter>
          </v8:device>
       </v8:payload>
    </v8:SDPSyncMessage>
  </soapenv:Body>
</soapenv:Envelope>
```


Attribute	Data Type	Required	Configurable	Source System	Values	Descrip
//SDPSyncMessage/header/						
noun	string	Y	N	KMS	SDPSync	
revision	string	Y	N	KMS	1	
dateTime	date	Y	N	KMS	<current time>	
source	string	Y	N	KMS	CIS	
messageID	string	Y	N	KMS	<UUID>	
asyncReplyTo	string	Y	N	KMS	none	
syncMode	string	Y	N	KMS	sync	
optimizationLevel	string	Y	N	KMS	Optimistic	
//SDPSyncMessage/payload/device/						
mRID	string	Y	N	KMS	<Entity ID>	
type	string	Y	N	KMS	Meter	
//SDPSyncMessage/payload/device/parameter						

name	string	Y	N	KMS	KeyExchange.keyType KeyExchange.role KeyExchange.AgreementDataPart1 KeyExchange.AgreementDataPart2 KeyExchange.SignaturePart1 KeyExchange.SignaturePart2	
value	string	Y	N	KMS	<parameter value>	
startDate	date	Y	N	KMS	<current time>	

SDP Sync Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v8="http://www.
   <soapenv:Header/>
   <soapenv:Body>
      <v8:SDPSyncReplyMessage>
         <v8:header>
            <v8:verb>?</v8:verb>
            <v8:noun>?</v8:noun>
            <v8:revision>1</v8:revision>
            <v8:dateTime>2016-05-05T00:00:00</v8:dateTime>
            <v8:source>?</v8:source>
            <v8:messageID>e40153ba-ea99-4cb4-9733-270a84a08893</v8:messageID>
         </v8:header>
         <v8:reply>
            <v8:replyCode>0</v8:replyCode>
            <v8:replyText>?</v8:replyText>
            <v8:correlationId>0de28253-c9af-4676-b63b-7d30745e36f3</v8:correlationId>
         </v8:payload>
      </v8:SDPSyncMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Attribute	Data Type	Required	Source System	Values	Description
//SDPSyncReplyMessage/reply/					
replyCode	string	Y	MDM	0 | 1 | etc.	
replyText	string	Y	MDM		
correlationId	string	Y	MDM	<request: //SDPSyncMessage/header /messageID>	

new Solution Proposal (UDIS): Key Agreement Key Update UAA / UDIS Adapter Interface Extension

Upon receipt of a Key Agreement Key Renewal Replacement request (via the UAA interface defined here: new Solution Proposal (UDIS): Key Agreement Key Update UAA / UDIS Adapter Interface Extension), a HES Workflow should be initiated.

The overall positioning of this workflow can be seen on the overview page, with more detailed descriptions of the MDM and KMS interactions given below: Security Suite 1 - Key Agreement Key Exchange Mechanism

> **Note:**
> UDIS will a notification only that a particular key, for a particular Role needs Replacement. UDIS is responsible for using the appropriate COSEM Client to actually make the neccessary requests to the Meter. The Clients to be used should be aligned with KMS, as KMS also requires configuration on this topic in order that the correct Private Keys are used in Cryptographic operations.

Overall process step	Description
5	Upon receipt of the Key Agreement Key Replacement UAA request, the information provided in the request should be used to construct and send appropriate messages to the Meter to trigger the derivation of the New Symmetric Key as defined by Ephemeral Unified Model C(2e, 0s, ECC CDH) in Greenbook v8. There is a process overview of the model in Annex C of Greenbook.
6	The Meter's responses to the step #5 requests should leave HES in a position to make a REST Service request to the KMS Service: new KMS REST Service Definition: keyRenewalAgree2EFinalise This action triggers the deriviation on the KMS side of the same Symmetric Key that was derived by the Meter in step #5.
8	The KMS will respond to step #6 with a simple 'OK', which the HES Workflow should interpret as instruction to tell the Meter to 'Activate' the New Key.

9	The success of failure of step #8 should be returned to MDM as part of the UAA Response message: new Solution Proposal (UDIS): Key Agreement Key Update UAA / UDIS Adapter Interface Extension This enables MDM to subsequently notify the KMS of the successful Replacement of the Key so KMS can update its internal records appropriately.

new UAA to UDIS Adapter Interface Extension for Key Agreement Key Exchange: MeterAssetUpdateRequestMessage

The MeterAssetUpdateRequestMessage interface is an existing Interface between MDM and UDIS, it is documented here: UDIS Schnittstellenbeschreibung

The MeterAssetUpdateRequestMessage interface is currently used to trigger actions in UDIS that result in something being updated in the Meter. The definition will be extended to give provision for sufficient data to be passed to UDIS to enable the initiation of the Key Agreement mechanism required to replace Symmetric COSEM Keys for Honeywell Meters.

The proposed extensions use the following as a Baseline:

- UDIS V6.0 Adapters V1.8
  - Section 3.5 UAA Provisioning
    - Updating a Meter: Request
    - Updating a Meter: Response

> **Note:**
> All parameters described below are optional within the context of a MeterAssetUpdateRequestMessage, however within the context of a specific Key Agreement Key Renewal request; if any one is present then all should be present.

Parameter Re-use

Some existing parameters will be re-used in the Key Agreement Key Exchange context:

Parameter Name	Parameter Value	Required	Description
KeyExchange.keyType	String	N	

KeyExchange.role	String	N	

New Parameters

Four new parameters will be included to supplement the two re-used parameters to support request of Key Agreement Key Replacement operations and provide HES with all necessary information to initiate the requested process:

Parameter Name	Parameter Value	Required	Description
KeyExchange. AgreementDataPart1	String	N	For passthrough of data from KMS, output by the JSS Function: keyRenewalAgree2EGenerate This is based on the Greenbook definition of Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme. The data will be presented in HEX format and will always be split between two parameters, giving a combined length of 65bytes for Security Suite 1 and 97bytes for Security Suite 2.
KeyExchange. AgreementDataPart2	String	N	
KeyExchange.SignaturePart1	String	N	For passthrough of data from KMS, output by the JSS Function: keyRenewalAgree2EGenerate This is based on the Greenbook definition of Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme. The data will be presented in HEX format and will always be split between two parameters, giving a total length of 64bytes.
KeyExchange.SignaturePart2	String	N	

Reply Codes

The following responses (or very close equivalents) are to be provided as a minimum. More can be added to the list if they are available and giving more detailed / granular responses is also acceptable.

> **Note:**
> Non-zero replyCodes can be assigned as required

replyCode	replyText
0	Success
	Meter offline
	Meter association could not be established
	No response from Meter
	Request failed at Meter

new Key Management - Additional Key of type HLS_SECRET

Honeywell meters require HLS6 authentication for Local connections. This means that they have an additional Key installed in Memory and provided to the Siemens Solution via Shipment File.

We expect to receive an HLS_SECRET key for three Clients; Installation, Certification and Maintenance.

The HLS_SECRET Keys will be delivered in the same Shipment File as other Keys. It will use the same Transport Key as other Symmetric Keys.

The Worldline JSS is going to be modified to include a new 'ProtectedSessionKeyCapability' for the HLS_SECRET. This is in part because the definition of the HLS_SECRET is such that it may require padding, the purpose of the new KeyCapability is to standardize that padding (TBD with Worldline). The JSS function will strip off any PKCS#5 padding added to the HLS_SECRET and replace with the correct padding. The nature of the padding is currently under discussion between Worldline and Honeywell, it has no impact to KMS.

new HLS_SECRET - Key Import

The implementation of changes to support the Import from Shipment File of the HLS_SECRET for the relevant Clients will involve:

- Extension of the SFL Interface SFL_I_SHIP_FILE_M and necessary changes to backend processes that the calling of this service triggers
- Extension of the KMS Interface KMS_I_UPLOAD_KEYS and necessary changes to backend processes that the calling of this service triggers

HES and MDM are not affected by the new Key Import requirements.

new Solution Proposal (KMS): Extension to include HLS_SECRET import

Changes to interface definition

The interface to KMS KMS_I_UPLOAD_KEYS must be extended to include the element described below:

Data field	Data type	Description	Mandatory	Example value
`device/oeRole/hlsSecret`	XML object	*HLS Secret for HLS6 authentication*	✔	see scheme for keys

Backend Changes

Upon receipt of an HLS_SECRET Key via the Shipment File Loader process, the HLS_SECRET Keys should be stored in the KMS, according to the same rules and restrictions as any other existing key types.

They must be created as an HLS_SECRET specific 'Type' in HSM terms:

- **Protected Session Key Capability:** SM_WK_HLSAUTH_AUTHENTIC

The HLS_SECRET Keys can be any length between 128bytes and 256bytes, however the JSS will deal with all necessary (un)padding from DLMS COSEM padding to an appropriate storage padding, by use of the correct Protected Session Key Capability.

new Solution Proposal (SFL): Extension to include HLS_SECRET import in Mappings

The SFL definition should be extended to ensure that mappings are included to map the Honeywell Shipment file to the KMS interface specification: KMS_I_UPLOAD_KEYS (including exensions).

In the Honeywell Shipment File, keys will be presented like:

Honeywell Shipment Format

```
<Device>
<SerialNumber>1KFM0200000001</SerialNumber>
<DeviceType>MA309MH4LAT1</DeviceType>
<ModuleType>CP115</ModuleType>
<MAC_Address>00237EFA6CC3</MAC_Address>
<IMEI></IMEI>
<Attribute name="FirmwareAPP">V11.00.12</Attribute>
<ModuleSerialNumber">1KFM1000000001</ModuleSerialNumber>
<Attribute name="YearOfConstruction">2018</Attribute>
<Attribute name="FirmwareCertificate">V10.00.07</Attribute>
<Attribute name="FirmwareModule">VB04V02_B05V07_SIM7500E-KF</Attribute>
<Attribute name="DeviceLogicalDeviceName">KFM2000200000001</Attribute>
<Systemtitle>4B464D000BEBC201</Systemtitle>
<Key name="HLSK_SM_CC" WrapKeyLabel="Pub_WK_SM_EX_P0">
<xenc:CipherData>
<xenc:CipherValue>50ZXJkdW0gdGVtcG9yLCBvZGlvIG51bGxhIGNG5lcXVlIGN1cnN1cG5lcXVlIGmn</xenc:CipherVa
</xenc:CipherData>
</Key>
```

The three Clients for which HLS_SECRET Keys are provided for Honeywell devices are:

Honeywell Shipment 'keyName'	KMS oeRole
HLSK_SM_CC	Certification
HLSK_SM_IC	Installation
HLSK_SM_MA	Maintenance

new HLS_SECRET - Key Export

new Solution Proposal (KMS): Extension of getKey Interface to include HLS_SECRET

Interface Definition

The interface definition: new HLS_SECRET - Key Export should be extended to add support for export of HLS_SECRET by the additions below:

Request

data field	data type	description	mandatory	example value
keyType	enum	Key Type; possible values • GAK • GUEK • GBEK • **HLS_SECRET**	✅	GUEK

Backend Processing

The result of a getKey request for an HLS_SECRET should behave in the same way as for any other Key, the result should contain the HLS_SECRET, wrapped by the KEK of the target System. The HLS_SECRET can theoretically be any length between 128bytes and 256bytes, but the JSS will deal with all necessary (un)padding to provide an export that is acceptable to all Parties who have been in talks with Worldline around that part of the design. No changes are expected to be required in KMS.

new HLS_SECRET - Lifecycle Management

In overall Solution terms, HLS_SECRET is an additional Symmetric Key that requires the same lifecycle operations as any other Symmetric Key. Therefore KMS and UDIS should be extended to allow for its replacement in the Meter.

The impact of this extension is:

- Relatively minor for KMS - Solution Proposal (KMS): Inclusion of HLS_SECRET in Lifecycle Management Operations
- More complex for UDIS - new Solution Proposal (UDIS): Inclusion of HLS_SECRET in Key Replacement Operations

There is no impact to MDM, therefore there is no specific section for an MDM proposal.

new Solution Proposal (KMS): Inclusion of HLS_SECRET in Lifecycle Management Operations

KMS already has mechanisms and monitoring processes for triggering of Replacement of DLMS COSEM Symmetric Keys. These should be extended to include HLS_SECRET.

The HLS_SECRET is not specifically defined to follow a particular Key Standard, however the definiton (below) allows it to be treated as if it were a DLMS COSEM Symmetric Key:

"For HLS mode 6 (SHA-256) the new HLS secret length must be between 128bit (16byte) and 256bit (32byte)"

The Replacement process for HLS_SECRET can follow the established Symmetric Key Replacement process in KMS terms (this is just a brief recap of the standard process):

1. The Key is identified (via the mechanisms listed later) for renewal
2. The New HLS_SECRET is generated by the KMS
3. The New HLS_SECRET is stored in KMS DB wrapped by the KMS DB Storage Key
4. The New HLS_SECRET is wrapped with the appropriate Meter Key (defined in the HLS_SECRET Key Templates)
5. A Key Replacement MDM Workflow is initiated, passing the wrapped Key in Parameters and setting the KeyExchange.keytype parameter to HLS_SECRET
6. MDM, HES and Meter perform the neccessary actions to install the New HLS_SECRET in the Meter
7. A notification is later sent to the KMS of the successful installation of the New HLS_SECRET

The following KMS mechanisms should be updated to give provision for HLS_SECRET:

- Lifecycle Management Monitoring / Marking Processes
- Lifecycle Management Automatic Key Renewal Initiation
- Key Renewal initiation via REST Service
- Key Update via REST Service

Implied in the list of mechanisms above is the 'extension' of the FlexSync interface between KMS and MDM. In reality this is just an additional expected value in the KeyExchange.keytype parameter, these are simply passed through to HES so do not require any technical or configuration change in MDM. As such it is not documented / described separately in MDM terms.

new Solution Proposal (UDIS): Inclusion of HLS_SECRET in Key Replacement Operations

There is an existing Interface between MDM and UDIS for the notification of Symmetric Key Replacement requests, MeterAssetUpdateRequestMessage. It is documented here: [UDIS Schnittstellenbeschreibung]

The MeterAssetUpdateRequestMessage interface is currently used to trigger actions in UDIS that result in something being updated in the Meter. The definition should be extended to give provision for it to be used for the request of an HLS_SECRET replacement operation. This extension is only to describe the use of existing fields, no addtional fields are required, hence the interface extension is not documented separately as it is in other areas.

The proposed interface for HLS_SECRET Replacement Requests uses the following as a Baseline:

- UDIS V6.0 Adapters V1.8
  - Section 3.5 UAA Provisioning
    - Updating a Meter: Request
    - Updating a Meter: Response

Upon receipt of a MeterAssetUpdateRequestMessage that indicates a HLS_SECRET Replacement is required, UDIS should take the necessary actions to Replace the HLS_SECRET in the Meter (for the required Client) and notify MDM / KMS of the status of the action.

> **Note:**
> The notification from MDM to Replace the HLS_SECRET will contain the Role / Client for which the replacement is required. However, the Client that should be used to undertake the action in the Meter must be determined by UDIS, along with the estblishment and negotiation of the required security controls to make the request of the Meter.

Request Parameter Usage

To identifiy an HLS_SECRET Replacement update request, the following parameters will be present.

> **Note:**
> In the context of a general MeterAssetUpdateRequestMessage request, these parameters are all optional. However when the request is used to initiate an HLS_SECRET replacement, all parameters below should be present.

Parameter Name	Parameter Value	Required	Value / Description
KeyExchange.keyType	String	N	HLS_SECRET
KeyExchange. keyNumberOfParts	String	N	The number of 'keyPart' parameters the New HLS_SECRET Key is distributed across
KeyExchange.keyPart1...x	String	N	The New HLS_SECRET is passed (wrapped by a Meter Key) in these parameters. Each parameter can hold only 50bytes, therefore the Key will be split over multiple parameters.
KeyExchange.role	String	N	One of: • Installation • Certificate

			• Maintenance

Reply Codes

The following responses (or very close equivalents) are to be provided as a minimum. More can be added to the list if they are available and giving more detailed / granular responses is also acceptable.

> **Note:**
> Non-zero replyCodes can be assigned as required

replyCode	replyText
0	Success
	Meter offline
	Meter association could not be established
	No response from Meter
	Request failed at Meter

new Certificate Management - UI and Data Model Refactoring

To support the import to KMS of several new types of Certificate and assignment of information to those records of data to enable new processes, it is necessary to Refactor the KMS UI to include several new fields.

In turn, this means that the underlying DB data model must also be updated to permit the persistent storage of the additional fields.

- new Solution Proposal (KMS): UI Refactoring
- new Solution Proposal (KMS): Data Model Extension

new Solution Proposal (KMS): Data Model Extension

To support the storage of the additional information required to enable new processes and storage of records several changes to the KMS internal DB model are required.

New Columns

The following new Columns will be required in existing tables:

Table	Column	Description

KMS_CERT	OE_Role	This is used to store the value of the DLMS COSEM Role (where appropriate) the Certificate record is associated to.
KMS_CERT	AuthKeyId	This is used to store the value of the Authority Key Identifier extracted from the Certificate (where present).
KMS_CERT	SubKeyId	This is used to store the value of the Authority Key Identifier extracted from the Certificate (where present).

New Tables

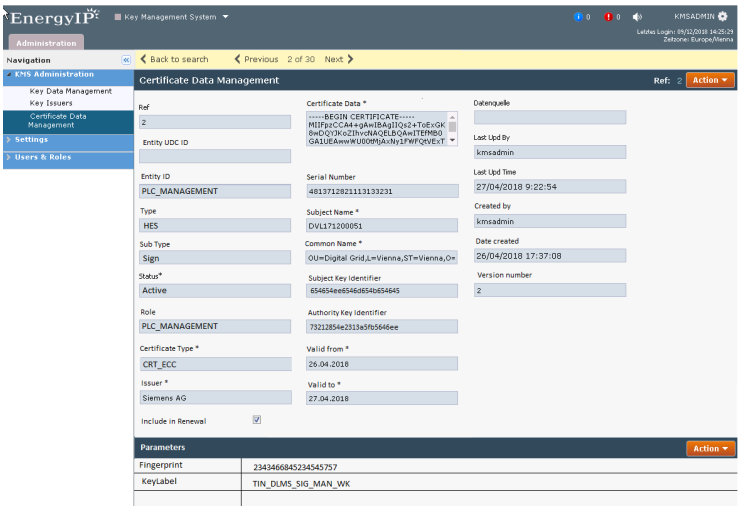
Some data is not generic enough to justify inclusion in the KMS_CERT table directly, and is hence added to the Certificate record as Parameters. In order to store these parameters a new table will be required:

New Table	Structure Notes
KMS_CERT_PARAMS	The structure of this table should mirror the standard EnergyIP structure where this type of parameter assignment / storage is used extensively.

new Solution Proposal (KMS): UI Refactoring

Certificate Management UI

The Certificate Management UI should undergo several changes as can be seen in the UI mockup below. Specific information on the changes can be found in the following sections:



New fields

The Certificate Management UI needs to have three new fields added to cater for the import / storage and maintenance (lifecycle management) of Meter Certificates and two new types of Certificate; HES and Issuing Authority:

New field	Description	Type	Reference

Role	An optional field to allow the assignment of a Certificate record to a specific DLMS COSEM Role. Important for HES Certificates.	Enumerated: • Management • Readout • PLC_Management	new Solution Proposal (KMS): HES Client Certificate Import
Subject Key Identifier	An optional field that will be auto-populated on Certificate Import with the (if present) Subject Key Identifier extracted from the Certificate.	Hex String	new Solution Proposal (KMS): Subject Key Identifier and Authority Key Identifier
Authority Key Identifier	An optional field that will be auto-populated on Certificate Import with the (if present) Authority Key Identifier extracted from the Certificate.	Hex String	new Solution Proposal (KMS): Subject Key Identifier and Authority Key Identifier

New panel

The Certificate Management UI needs to have a new panel added to give provision for the addition and visibility of per Certificate record parameters. This should use the same parameter UI panel as seen throughout the standard EnergyIP UI. The new parameters section will be used for the addition (where necessary) of the parameters described below to a Certificate Record:

Parameter	Description	Type	Reference
Fingerprint	A parameter relevant only to HES Certificates, allowing the addition of a Certificate Fingerprint to the Certificate record. This is required to support HES Certificate to Meter Relationship creation.	Hex String	new Solution Proposal (KMS): HES Cert to Meter Relationship Maintenance
KeyLabel	A parameter relevant only to HES Certificates, allowing the addition of a KeyLabel to the Certificate record. This enables, via deriviation, HLS7 mechanisms to use the correct Private Signing Key based on the HES Certificate known to be installed in the Meter.	String	new Solution Proposal (KMS): HES Certificate Lifecycle Management new Solution Proposal (KMS): Provide Signing and Signature Verification Services

New Field Labels

Several existing field labels should be changed to allow for the more generic presentation of non-device-centric records:

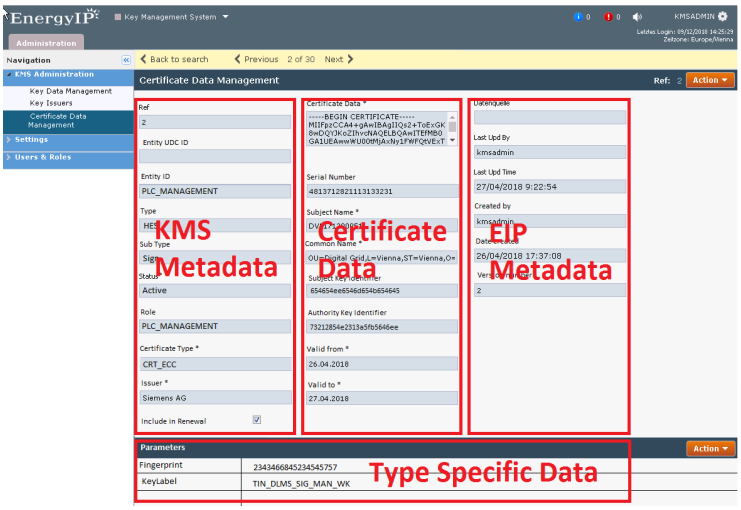
Old Label	New Label	Field Content Type
Device UDC ID	Entity UDC ID	String

Device ID	Entity ID	String
Device Type	Type	Enumerated: • Meter • Gateway • HES • Issuing Authority
Usage	Sub Type	Enumerated: • Sign • KeyAgreement • TLS

New layout

The fields and panels present in the Certificate Management UI should be re-arranged as shown in the mockup below to group information into logical sections:

Section	Description
KMS Metadata	These fields present the information that KMS uses to categorise, or add additional supporting information to, Certificate records internally.
Certificate Data	These fields present data directly representing the Certificate, or data extracted directly from the Certificate.
EIP Metadata	These fields present the information the KMS uses to maintain administrative records of Certificate records.
Type Specific Data	These fields are predominantly used for the addition of data to a Certificate Record that doesn't fit into the KMS_CERT DB structure directly, typically 'type specific' data (eg data that's relevant only to HES certificates.



Certificate Search UI

To support debugging activities it should be possible to use the Search facility of the KMS Certificate UI to search for Certificate Records based on Subject Key Identifier.

If would also be useful (not mandatory) to be able to seatch for Certificate Records based on the Fingerprint Parameter.

new Security Suite 1 - Additional Authentication Data Support

In order to support the Honeywell implementation of Security Suite 1, Wordline have added two new JSS Functions to support a slightly different mechanism of Encryption and Decryption of DLMS COSEM APDU.

This means that the KMS requires an extension to be made in order to expose those new functions.

new Solution Proposal (KMS) - Provision of New REST Services to Support Encryption and Decryption with AAD

Worldline have added two new JSS functions that are required to support the Honeywell implementation of Security Suite 1:

- cosemAuthDataDecryptWithAAD
- cosemAuthDataEncryptWithAAD

The JSS API definitions for these two new Functions are identical to the older functions of cosemAuthDataDecrypt and cosemAuthDataEncrypt, but take an additional parameter:

- additionalAuthenticationData

Defined as:

> "Optional additional authenicated data for SC-E or SC-AE. It contains the concatenation of the following fields as A-XDR encoded OCTET STRINGs. The length and the value of each field is included in the AAD:
>
> - transaction-id
> - originator-system-title
> - recipient-system-title
> - date-time
> - other-information
>
> The data provided in this member will be added to the AAD generated internally from the security control byte and authentication key. If this member is absent the internal AAD are constructed as normal (see [COSEM-GREEN-8.2])."

The KMS should expose the new JSS Functions via the extensions of the REST Services below, by adding an new optional parameter 'aad':

- KMS_I_DECRYPT - decryptUCast
- KMS_I_DECRYPT - decryptUCastBulk
- KMS_I_ENCRYPT - encryptUCast
- KMS_I_ENCRYPT - encryptUCastBulk

new KMS REST Service Definition: Extension of KMS_I_DECRYPT

The extensions of the services below are defined using the following definition as a baseline: KMS_I_DECRYPT

Unless specifically defined here, no further extensions are required.

Extensions to decryptUCast

Request

Data field	Data type	Description	Mandatory	Example value
aad	string	Optional additional authenticated data for SC-E or SC-AE. It contains the concatenation of the following fields as A-XDR encoded OCTET STRINGs. The length and the value of each field is included in the AAD: • transaction-id • originator-system-title • recipient-system-title • date-time • other-information The data provided in this member will be added to the AAD generated internally from the security control byte and authentication key. If this member is absent the internal AAD are constructed as normal (see [COSEM-GREEN-8.2]).		

Request Handling

The new JSS Functions are enabled in the Solution by an update to the JSS and an update to the HSM itself. The KMS should retain backward compatibility in projects where the version of KMS is updated, but because the xxxWithAAD functions are not reuqired, the JSS and/or HSM are not updated.

Therefore depending on the way the request is made to KMS, KMS must decide whether to call the old or new JSS function:

'aad' Parameter...	JSS Function call
Present and populated	cosemAuthDataDecryptWithAAD
Empty	cosemAuthDataDecrypt
Missing	cosemAuthDataDecrypt

This retains backwards compatibility. If the request received to KMS is in the non-AAD format, then the old JSS function will be called. If the use of the xxxWithAAT variant is required, then the inclusion of the appropriate parameter in the request ensures the xxxWithAAD function is called.

Extensions to decryptUCastBulk

Request

Data field	Data type	Description	Mandatory	Example value
aad	string	Optional additional authenticated data for SC-E or SC-AE. It contains the concatenation of the following fields as A-XDR encoded OCTET STRINGs. The length and the value of each field is included in the AAD: • transaction-id • originator-system-title • recipient-system-title • date-time • other-information The data provided in this member will be added to the AAD generated internally from the security control byte and authentication key. If this member is absent the internal AAD are constructed as normal (see [COSEM-GREEN-8.2]).		

Request Handling

The new JSS Functions are enabled in the Solution by an update to the JSS and an update to the HSM itself. The KMS should retain backward compatibility in projects where the version of KMS is updated, but because the xxxWithAAD functions are not reuqired, the JSS and/or HSM are not updated.

Therefore depending on the way the request is made to KMS, KMS must decide whether to call the old or new JSS function:

'aad' Parameter...	JSS Function call
Present and populated	cosemAuthDataDecryptWithAAD
Empty	cosemAuthDataDecrypt

Missing	cosemAuthDataDecrypt

This retains backwards compatibility. If the request received to KMS is in the non-AAD format, then the old JSS function will be called. If the use of the xxxWithAAT variant is required, then the inclusion of the appropriate parameter in the request ensures the xxxWithAAD function is called.

new KMS REST Service Defintion: Extension of KMS_I_ENCRYPT

The extensions of the services below are defined using the following definition as a baseline: KMS_I_ENCRYPT

Unless specifically defined here, no further extensions are required.

Extensions to encryptUCast

Request

Data field	Data type	Description	Mandatory	Example value
aad	string	Optional additional authenticated data for SC-E or SC-AE. It contains the concatenation of the following fields as A-XDR encoded OCTET STRINGs. The length and the value of each field is included in the AAD: • transaction-id • originator-system-title • recipient-system-title • date-time • other-information The data provided in this member will be added to the AAD generated internally from the security control byte and authentication key. If this member is absent the internal AAD are constructed as normal (see [COSEM-GREEN-8.2]).		

Request Handling

The new JSS Functions are enabled in the Solution by an update to the JSS and an update to the HSM itself. The KMS should retain backward compatibility in projects where the version of KMS is updated, but because the xxxWithAAD functions are not reuqired, the JSS and/or HSM are not updated.

Therefore depending on the way the request is made to KMS, KMS must decide whether to call the old or new JSS function:

'aad' Parameter...	JSS Function call
Present and populated	cosemAuthDataEncryptWithAAD
Empty	cosemAuthDataEncrypt
Missing	cosemAuthDataEncrypt

This retains backwards compatibility. If the request received to KMS is in the non-AAD format, then the old JSS function will be called. If the use of the xxxWithAAT variant is required, then the inclusion of the appropriate parameter in the request ensures the xxxWithAAD function is called.

Extensions to encryptUCastBulk

Request

Data field	Data type	Description	Mandatory	Example value
aad	string	Optional additional authenticated data for SC-E or SC-AE. It contains the concatenation of the following fields as A-XDR encoded OCTET STRINGs. The length and the value of each field is included in the AAD: • transaction-id • originator-system-title • recipient-system-title • date-time • other-information The data provided in this member will be added to the AAD generated internally from the security control byte and authentication key. If this member is absent the internal AAD are constructed as normal (see [COSEM-GREEN-8.2]).		

Request Handling

The new JSS Functions are enabled in the Solution by an update to the JSS and an update to the HSM itself. The KMS should retain backward compatibility in projects where the version of KMS is updated, but because the xxxWithAAD functions are not reuqired, the JSS and/or HSM are not updated.

Therefore depending on the way the request is made to KMS, KMS must decide whether to call the old or new JSS function:

'aad' Parameter...	JSS Function call
Present and populated	cosemAuthDataEncryptWithAAD
Empty	cosemAuthDataEncrypt
Missing	cosemAuthDataEncrypt

This retains backwards compatibility. If the request received to KMS is in the non-AAD format, then the old JSS function will be called. If the use of the xxxWithAAT variant is required, then the inclusion of the appropriate parameter in the request ensures the xxxWithAAD function is called.

new Solution Proposal (KMS): Issuing Authority Certificate Import

It should be possible via the KMS Certificate UI to manually import an Issuing Authority Certificate. The following values should be set manually accordingly on import (categories are as described here: new Solution Proposal (KMS): UI Refactoring):

Category	Label	Value
KMS Metadata	Entity ID	*<the SubCA name>*
KMS Metadata	Type	Issuing Authority
KMS Metadata	Sub Type	Sign
KMS Metadata	Status	Active
KMS Metadata	Role	Default
KMS Metadata	Certificate Type	CRT_ECC
KMS Metadata	Issuer	*<as appropriate>*
KMS Metadata	Include in renewal	False
Type Specific Data (parameter)	ParentKeyLabel	*<the Key Label of the related Root Cert imported to the HSM>*

Several other parameters important parameters should be set automatically (see: new Solution Proposal (KMS): Subject Key Identifier and Authority Key Identifier).

new Solution Proposal (KMS): Subject Key Identifier and Authority Key Identifier

For Certificate and Key Lifecycle operations to be undertaken successfully, it is necessary for KMS to have a mechanism to allow the establishment of relationships between Base level Certificates and the Issuing Authority Certificate.

For DLMS COSEM Certificates (and it's a widely adopted RFC standard, see RFC 5280) these relationships can be established via two elements in the Certificates, used to assist in building a Chain-of-Trust Certificate Hierarchy:

Element	RFC Description	OID
Subject Key Identifier	The subject key identifier extension provides a means of identifying certificates that contain a particular public key.	2.5.29.14
Authority Key Identifier	The authority key identifier extension provides a means of identifying the public key corresponding to the private key used to sign a certificate.	2.5.29.35

The Issuing Authority Certificate related to a Base Certificate can be established by direct comparison / search of these values where:

- Base Certificate's Authority Key Identifier = Issuing Authority Certificate's Subject Key Identifier

The KMS Certificate Import functions should be extended so that for all Certificates imported to the KMS DB, these values are extracted from the Certificate and stored in the appropriate DB Columns as defined here: new Solution Proposal (KMS): Data Model Extension

This extraction of data from the Certificate should happen automatically via all mechanisms used to obtain Certificates in KMS, including:

- Certificate Import via UI
- Certificate Import via Shipment File
- Certificate Import via CSR Response 'Interception' / 'Data scraping'
- Certificate Import via Reconcilliation

⚠ This process will only extract and store information that can be obtained directly from imported Certificates. It will not establish if a Chain-of-Trust can be constructed or verified.