elhub

Elhub

Elhub Messaging Interface

Versjon 1.7 | 13.01.2017

Statnett

Table of Contents

1 Introduction

This is a description of the Elhub Messaging Interface (EMIF). The audience of this document are developers and architects of systems that are to integrate with Elhub. The purpose is to give an understanding of the integration patterns of Elhub. Along with this document you will also find the WSDLs for the Elhub integration, a SoapUI package (see Appendix) that demonstrates the integration patterns and the BIM (Business Information Model) documentation that describes the different business documents that are sent and received. In addition, the BRSs (Business Requirement Specifications) will describe the processes. These are not included here.

This document and SoapUI package is for version 1.7 of the BIM.

2 Change Log

Date	Version	Change
30.06.2015	0.5	• Removed message encryption. Will use transport encryption • Removed explicit gzip element and use transport compression • Added optional confirm message • Added physical sender to the message header • Added better filter options to the polling services
28.08.2015	1.0	• Stricter rules for the timing of sending of metering values has been added • Removed a duplicate entry of PortfolioOverview from the PollMarketProcesses WSDL • Added support for WS-Security to the WSDLs • Changed description of requesting positive Acknowledgement to also allow it for service requests • Changed description of what to do in case of transport failures with two more options • Added comments regarding UUID • Removed the MoreValuesAvailable flag from the polling response. • General rephrasings
22.10.2015	1.1	• BIM has been updated (see separate change log) • Changed Soap Faults to being signed • Corrected errors relating to signing in the SoapUI projects • Added more details regarding rules for sending frequency of metering values • Added some information regarding how to connect to the SoapUI package using .Net • CollectedData can now be returned from the metering value polling in accordance with BRS-NO-311 • Added information related to xml contents (namespaces, whitespaces, content type)
05.02.2016	1.5	• Using a common version number for several documents • BIM has been updated (see separate change log) • Added description of rules related to updates of estimated annual consumption • Documentation regarding Security also describes that Soap Faults are signed. The WSDL was updated in the previous release • Added comment that certificates used for signing must be of type non-repudiation • Added information regarding certificates during test • Cardinality in the poll result is changed from 9999 to unbounded. This is done to prevent memory issues when doing XSD validation in Java. There will be no change to the amount of data returned

Date	Version	Change
		• Hourly values for exchange points and production points are not limited by the daily sending rule
31.05.2016	1.6	• Added warning related to parallel sending of messages
13.01.2017	1.7	• Detailed polling frequency rules • Added support for negative Acknowledgement from grid owner for meter indexes and estimated annual consumption received from a balance supplier • New namespaces. All namespaces change from *:v1 to *:v2 • Added description of principles for future version changes • Added new Soap Fault code to indicate the message was rejected due to date restrictions • Changed phrasing to get rid of the word "should" as it is ambiguous • Cleanups in the rules regarding sending interval data • Cleanups regarding polling rules • Added warning regarding repeated namespaces and pretty printing of xml

3 Technology

The Elhub Messaging Interface will be based on SOAP 1.1 web services and uses the following WS-* technologies:

- WS-Security
- WS-Policy

Elhub will provide a set of WSDLs for the integration.

3.1 Security

Messages will be transport encrypted by HTTPS (TLS 1.2). In addition, certificates will be used to sign the messages using WS-Security. The signing requires the use of an enterprise certificate with non-repudiation. The enterprise certificate must be from an issuer trusted by Elhub and messages signed using the enterprise certificate must contain the organization number of the sending market party (the physical sender). Elhub expects to support Buypass and Commfides. We have provided two sets with WSDLs where one set contains WS-Security and one does not (this is shown by the file names of the WSDLs). There are support for both WSDL sets in the SoapUI package . The HTTPS encryption is handled by Elhub with no need for any certificates for the market parties for this.

There will be no message encryption using WS-Security. Due to this, it is the responsibility of the sender of the message to make sure it can only be sent to Elhub (or the Ediel portal for testing/certification) by for example only adding the Elhub certificate to the truststore (Java) or by having some other mechanisms in place to make sure connections only connect to Elhub. Accepting self-signed or outdated certificates and connections without a certificate must be avoided.

Elhub's use of WS-Security is described using WS-Policy. If you cannot use WS-Policy, this is a textual description of Elhub's use of WS-Security: AsymmetricBindingAssertion indicates to use asymmetric encryption, where the requestor's certificate (X509v3) must be used for signature. The InitiatorToken field indicates that the request token must be an X509v3 token and that it must be included with all request messages, while the RecipientToken field indicates that response token has to be X509v3 but will not be included in any message. To identify the token, a keyIdentifier will be used – specified by MustSupportKeyRefIdentitier field. Timestamp is also needed for inclusion to circumvent replay attacks (see https://www.owasp.org/index.php/Web_Services), and as such - by default - this is also signed. The OnlySignEntireHeadersAndBody field dictates that only the entire header or body is allowed to signed – to mitigate XML Signature wrapping. And lastly, we only dictate that the Body-element of the SOAP Envelope needs to be signed – both for the request and for the response. This also includes Soap Faults.

3.1.1 Certificates During Testing

When running tests towards Elhub, like for instance in system vendor trials, it is not necessary to use production certificates from Buypass or Commfides. Both of these will issue test certificates and we recommend using these instead.

3.2 Compression

Compression will be used to minimize the amount of data to send on potentially large data sets. Compression will be used on the transport channel (HTTP). This will be standard HTTP compression.

Elhub will require compression on metering data messages and the client system must also support compression for messages received from Elhub (result from the polling interfaces). Tests have shown that compressing a metering value message can reduce the size of the message by 95%.

When sending compressed data, compress using GZIP. This means the following HTTP header must be set: "Content-Encoding: gzip"

When sending polling messages, the following HTTP header must be set to allow Elhub to send the return data compressed: "Accept-Encoding: gzip". It is allowed to add other compression schemes as well on Accept-Encoding, but GZIP must be one of the supported ones.

4 Data to Send

The data transported using the web services will be data defined in the BIM. There will be some additional SOAP packaging around the data.

5 Versioning

The WSDLs will be versioned in the namespace and the version will change when changes that break the interface are added. The namespace used for this version of the specification end in v2.

In future changes to the specifications we will follow these principles:

- All changes in the interface will require a new namespace
- The different versions will have different endpoints
- Changes will be communicated well in advance of the changes
- Reason for changes will be serious errors in the interface or updates to functionality requiring changes to the information and structure in the messages
- There will be a set date for the changeover that all market parties must respect
- Messages received in the old version will be processed with the rules of the old version and response messages will be in the old version
- The systems must be prepared to poll on both the old and the new interface
- The transition window where there can be data for both versions is typically in the order of magnitude of the longest period of time a business process can be in a waiting state
- Messages sent in before the time of the version change are to be sent in the old version
- Messages sent in after the time of the version change are to be sent in the new version with the exception of polling messages which can still be sent in the old version
- The rules (and in particular the last two ones) might be adjusted for some version changes. This will be done in cooperation with the market.

6 Integration Pattern

Elhub will use synchronous web services. The processing of the request will in most cases be done asynchronously, but Elhub will send an http 200 to signal that the message was received in the SOAP response.

The only messages that will be processed synchronously are some query requests where Elhub is to give a response immediately.

For messages that are processed asynchronously, a separate polling service will return the result of the processing. The same service will also return messages the market party is to receive due to actions by other market parties and actions by Elhub. This means Elhub will not actively send a message to any recipient.

6.1 Service Description

There is one WSDL for each of the services described below (as mentioned above, there are two sets with WSDLs where one supports WS-Security and one does not. The WSDLs are otherwise identical).

- MarketProcesses: Asynchronous processing of all market processes doing updates in Elhub. Examples are registering a change of balance supplier and updating of customer data.
- Query: Handles queries for data. There are three types of queries: Synchronous queries where the result is to be returned immediately, asynchronous queries where the result will be available within a short time on the polling interface and asynchronous queries that will need manual processing and the result of these will be made available on the polling interface at a later time. Examples are queries to be forwarded to the grid owners and upfront validation of metering points.
- MeteringValues: Asynchronous processing of metering values. Examples are hourly values and non-continuous values.
- PollMarketProcesses: Results from asynchronous processing and other data that the market party is to receive. This interface returns everything not related to metering values. Examples are rejections of supplier changes and notification of updated customer data.
- PollMeteringValues: Returns metering values the market party is to receive. This includes both metering point values and aggregated values. In addition, information about rejected metering values, reminders for metering values and rejections related to queries for metering values are returned on this service.

With exception of the polling services, there will be one message in the WSDLs for each BIM type to send. The naming convention used is <BIM type>Request. This means to send a start of supply, the message to send is an RequestStartOfSupplyRequest message with a RequestStartOfSupply BIM type inside it. To send an end of supply you send a RequestEndOfSupplyRequest. The only element inside these types will be the BIM type. For the polling services, the result can be a message containing several BIM types at the same time.

The production version of Elhub will use WSDLs with WS-Security. The WSDLs without WS-Security is to make initial development easier. Possibly the first version of the Ediel Portal test solution for Elhub will use WSDLs without WS-Security.

6.2 Examples of Use

6.2.1 Request Start of Supply

This request is used when a balance supplier is to start supplying a metering point.

When Elhub receives the request, a check of the message signing is done. If this fails, a SOAP Fault will be returned. Next an XSD validation is done and if it fails, a SOAP Fault will be returned. If the message passes both of these, a confirmation that the message was received is returned by an http 200. This does not mean that the change of supplier will be effectuated as the processing of Request Start of Supply request is done asynchronously and may be rejected for functional reasons. If this happens, a Reject Start of Supply message will be returned on the market processes polling service.

If there is some transport failure where the SOAP Response is not received by the sender, the sender does not know if the message was received by Elhub or not as it in general will not know if the error occurred before or after Elhub received the message. We see three ways of handling this situation (Elhub does not utilize WS-ReliableMessaging):

- Send the same message once more with the same message id as the original message. If Elhub received the original message and the problem occurred in the transport of the response, Elhub will have received the same message twice and will reject the second message due to duplicates by a SOAP Fault. It is up to the market party to detect these situations. The duplicate error message can normally be ignored in this situation as the sender knows the same message has been sent twice.
- Always ask for confirmation messages and by doing so you will know after some time if Elhub received the original message or not. If no confirmation or rejection message is received within a reasonable time, the original message is to be resent with a new message id. If no confirmation message is requested and the message is resent in case of a transport failure, the second message may be rejected with an error indicating a crossing process. The sender will then not know if the conflict occurred due to it sending in the original message (the first message was received) or because there is another process that gives the error. In this case we risk Elhub and the sender to be out of sync so resending with a new message id without having requested confirmation messages is not recommended. If a confirmation message was requested, but not received within some time and the sender sent the message once more with a new message id, but Elhub did in fact receive the original message, response messages will eventually be sent for both of these messages. The sender will then either get two rejections, two confirmations, or one confirmation and one rejection, and must based on this figure out the net status. If there are more transport failures, more than two messages may have been sent.
- The third option is to have no automatic handling of these error situations and instead log on to the Elhub Web Portal to check statuses. This may include checking the status of the metering point contract in case of a move out or by checking for initiated processes in case of a future change of supplier. If the status indicates that the message was not received, it is to be resent. In this case you can choose to use the original message id or a new one. There is no guarantee that you will be able to figure out if the message was received or not with 100% accuracy.

Note: Before data are sent to Elhub, they must have been persisted in a state signaling they are to be sent. When Elhub acknowledges it has received the data in the SOAP Response, the data must change state to having been sent. This means that if the sender crashes, the sender will be able to pick up the unsent message and resend it when it starts. If the sender did not persist the data before sending the message, Elhub might have processed the request, but the sending system might not have any trace of sending the message and might also not have the correct state in the business data.

After the message has been processed, any errors (rejections) will be made available on the polling service. When polling, a number of messages of different types can be returned. Elhub will return whatever messages are available (but not necessarily all) at the time of the polling and return no messages if there are none. The data will be compressed using HTTP compression and the calling system must support this. After the market party has persisted the received messages, an Acknowledgement message referring to the polled data set must be sent back to Elhub on the polling service to confirm that the messages has been received. If no acknowledgement is received within a predefined timeout, the same data set will be returned again. This timeout has yet to be defined, but we are thinking in the order of 30 minutes. It is not a requirement to have updated all business objects based on the received polling result before acknowledging to Elhub. It is enough that the polling result has been persisted and so the market party can guarantee that the data will be processed.

It is optional to have a confirmation message returned when Elhub has finished processing the request. It is specified in the message header if such a confirmation is wanted. It is recommended to only request a confirmation if the designed business logic needs it to complete the processing of the update. Also consider if it is necessary to build logic needing the positive confirmation as it is best for Elhub to avoid having to send these messages. The positive confirmation is similar to the positive APERAK in the current market. Messages in the MarketProcesses WSDL will support positive confirmation (with the exception of BRS-NO-317). Metering values will never get any form of positive confirmation. For queries, the query result is the positive confirmation, but for long running queries (like service requests), it is possible to ask for the positive confirmation. The confirmation message will be returned in the polling service.

The rejections in the polling result will contain the id of the message they have rejected. The sending system must keep track of the id of the messages that have been sent in order to be able to correlate the rejection with the message that sent the data and from that figure out which action has to be taken on which data to remedy the error. The optional confirmation messages follow the same pattern with a reference to the original message.

The sequence diagram below shows three different scenarios when sending synchronous messages with asynchronous processing. The first is the normal scenario where the message is received, the second shows a case with XSD validation error, the last shows the case when the sender does not receive the SOAP Response and sends the same message for a second time with the same message id in order to get the proper rejection (duplicate error) if the original message was received by Elhub (as mentioned above, this is one of the three options to handle the transport failure. Another option is to have requested a confirmation message and sending the same message with a new message id if it seems like Elhub did not receive the original message. The third option is to send the message again after having done a manual check in Elhub first and figuring out it was not received by Elhub).

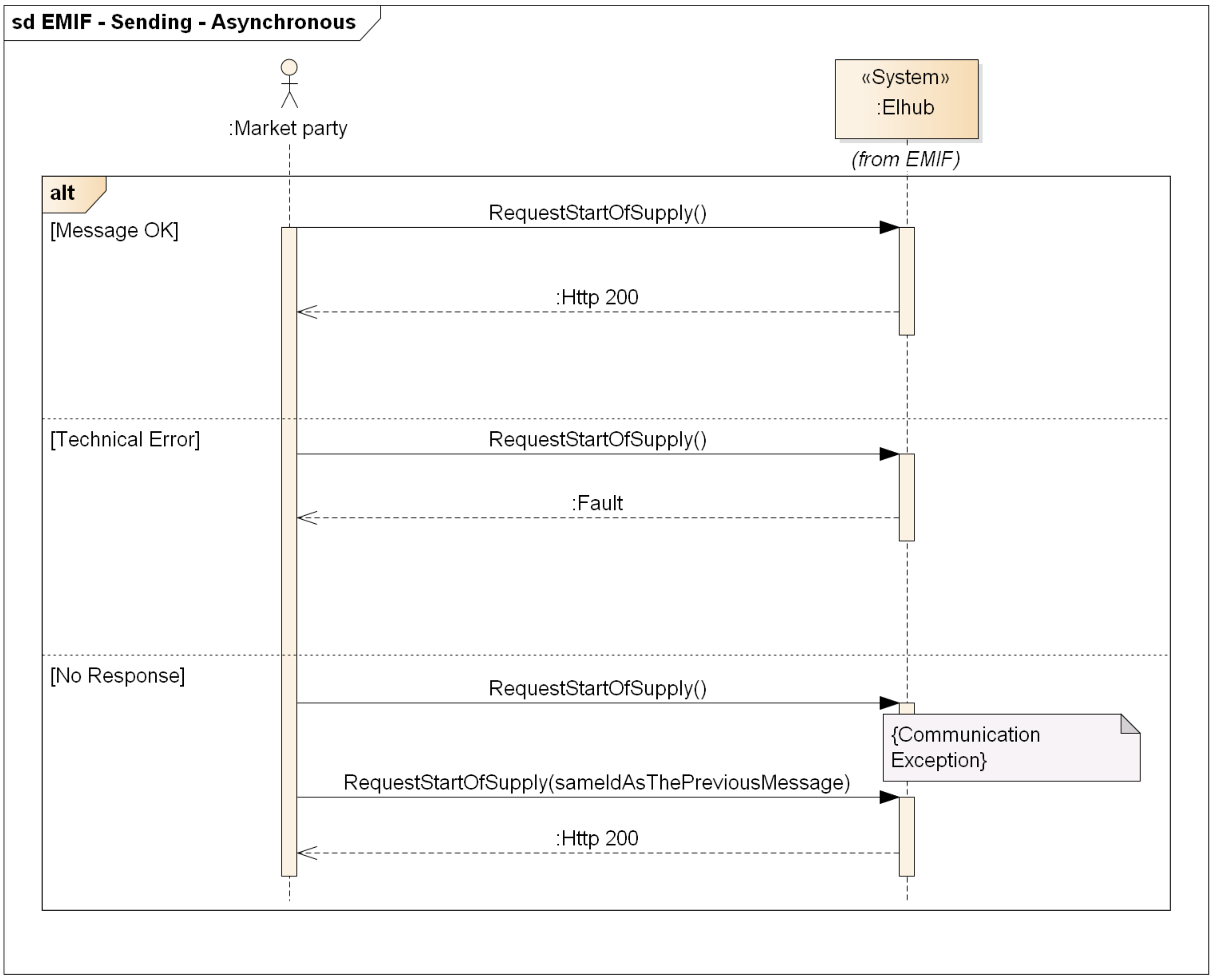


Figure 1 Asynchronous incoming message pattern

6.2.2 Polling

As Elhub process messages asynchronously, it is recommended to run polling separated from the transfer of data to Elhub. Below we can see that you first poll for data to get some data and then send an Acknowledgement to confirm that the data has been received.

Four scenarios are shown for the polling case. In addition to these flows, there can also be SOAP Faults returned for both the Polling and the Acknowledgement. First the normal case with a result set is shown, here you have to acknowledge that the data has been received. Next is the case when there are no data to return in the polling (the returned response object will be empty (NULL)), then there is no acknowledgement. The third and fourth cases shows the cases when the sender does not receive the response based on the sent input. In both cases, the sender is expected to send the request again, but the message id must be different from the original in both cases (this is different from sending in updates as described above). The id that is being acknowledged is to be the same as in the original message in scenario four.

Every message must have a BRS identificator in it. Polling is not defined as a separate BRS, but as the same type of header is used for polling messages as for other messages, a specific POLL BRS is defined in the legal value definition to use when polling. The Document Type to specify in the polling messages, both the initial poll and the later Acknowledgement, is 21 (list agency identifier 6).

The polling message will specify which role to retrieve data for. It is possible to choose to have messages for all roles the market party has or only for a single role. It will not be possible to specify a specific BRS in the polling to only poll data for that BRS.

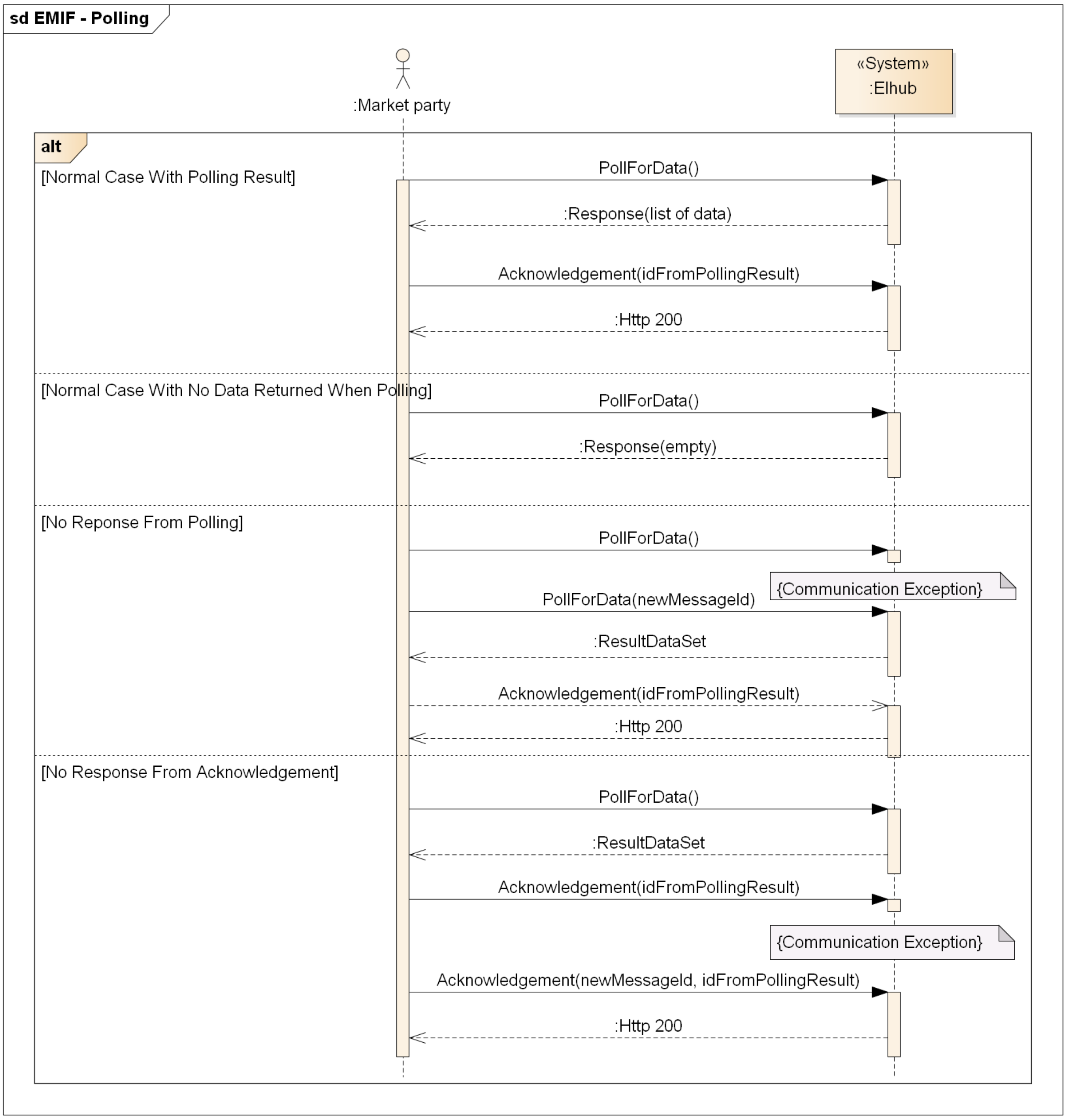

sd EMIF - Polling
:Market party
«System» :Elhub
alt
[Normal Case With Polling Result]
PollForData()
:Response(list of data)
Acknowledgement(idFromPollingResult)
:Http 200
[Normal Case With No Data Returned When Polling]
PollForData()
:Response(empty)
[No Reponse From Polling]
PollForData()
{Communication Exception}
PollForData(newMessageId)
:ResultDataSet
Acknowledgement(idFromPollingResult)
:Http 200
[No Response From Acknowledgement]
PollForData()
:ResultDataSet
Acknowledgement(idFromPollingResult)
{Communication Exception}
Acknowledgement(newMessageId, idFromPollingResult)
:Http 200

Figure 2 Polling timing

6.2.3 CollectedData

Sending in metering values (CollectedData) follow the same pattern as "Request Start of Supply", but uses different services for sending in the data and polling the data. Errors in metering data are sent in Acknowledgement messages. Confirmations that the metering data has been processed will never be sent even though the message format supports requesting it.

In a CollectedData message there is usually data for several metering points. The CollectedData message as a whole has an id and each repeating metering point (payload) has a separate id. Some of the metering values can be accepted and some rejected (one payload element is either rejected or accepted, but one payload might be accepted and another rejected). A separate Acknowledgement message is sent for each payload that is rejected. This acknowledgement message will contain references to both the header of the original CollectedData message and also to the particular payload that was rejected. This means one CollectedData message may result in more than one Acknowledgement message from Elhub. If all the values were accepted, there will be no response from Elhub.

Metering values sent in to Elhub must be compressed and the data received on the polling service will be compressed (both by HTTP compression).

As metering values does not contain a state in the same way as the market processes, you can resend the original message with a new message id in cases of transport failures, but this may lead to duplicate updates in Elhub.

6.2.4 Query

Query messages (except for Request Upfront Metering Point Characteristics) will also follow the asynchronous pattern described for "Request Start of Supply".

Of the query messages, only the service request messages (RequestToGridAccessProvider and RequestToElhub) will support requesting confirmation message. The reason is that for the other queries, the result will either be a rejection or the actual query result which will be made available at the same time as a confirmation message would have been made available. For the service requests it may take a long time to have a query result sent back. Due to this Elhub will support confirmation messages for the service request messages.

The data will be made available on the polling service. If the query is related to metering values (applies to the RequestDataFromElhubRequest message and depends on the specified query type), the responses (including Acknowledgement) will be made available on the PollMeteringValues service. All other query related results will be returned in the PollMarketProcesses service.

For queries that require manual processing from either Elhub operators or the grid owner (service requests), there is no guarantee that there will ever be sent a response to the query. Also the response to a service request may not come for several days so there is no use in resending the query shortly after the initial query if no response is received. Please wait at least some days in order to allow the recipient time to answer the request.

If there is some transport failure where the SOAP Response is not received by the sender, the same integration pattern as for "Request Start of Supply" apply, but instead of having the option to check for confirmation messages, the existence of a query result is to be checked instead in some situations (as only service requests will support confirmation messages). As there is no state related to queries, sending the original query with a new request id is also an option, but this may result in having duplicate search results.

6.2.4.1 Request Upfront Metering Point Characteristics

This query request is used when a balance supplier wants to check the existence of a metering point or find the metering points of a potential new customer.

The processing of Request Upfront Metering Point Characteristics will be done synchronously. When Elhub receives the request, a check of the signing is done. If this fails, a SOAP Fault will be returned. Next an XSD validation is done and if it fails, a SOAP Fault will be returned. If the message passes both of these, Elhub will process the request and return the result of the request in a SOAP

Response. The result can be a negative response (negative Acknowledgement) or the requested data. If there is some internal problem in Elhub that prevents the request from being processed within a predefined timeout, an error will be returned (SOAP Fault). The sender must then retry the request with a new message id at a later time. If there is some transport failure where the SOAP Response is not received by the sender, the message must be sent again with a new message id (note that this differs from the asynchronous case).

The sequence diagram below shows the four different scenarios when sending synchronous messages. The first is the normal scenario where the message is received and the request is ok, the second shows a technically correct message with functional errors (maybe the metering point does not exist), the third shows a case with XSD validation error, the last shows the case when the sender does not receive the SOAP Response and sends the same message for a second time, but with a new message id in order to avoid getting a SOAP Fault if the original message was received by Elhub.

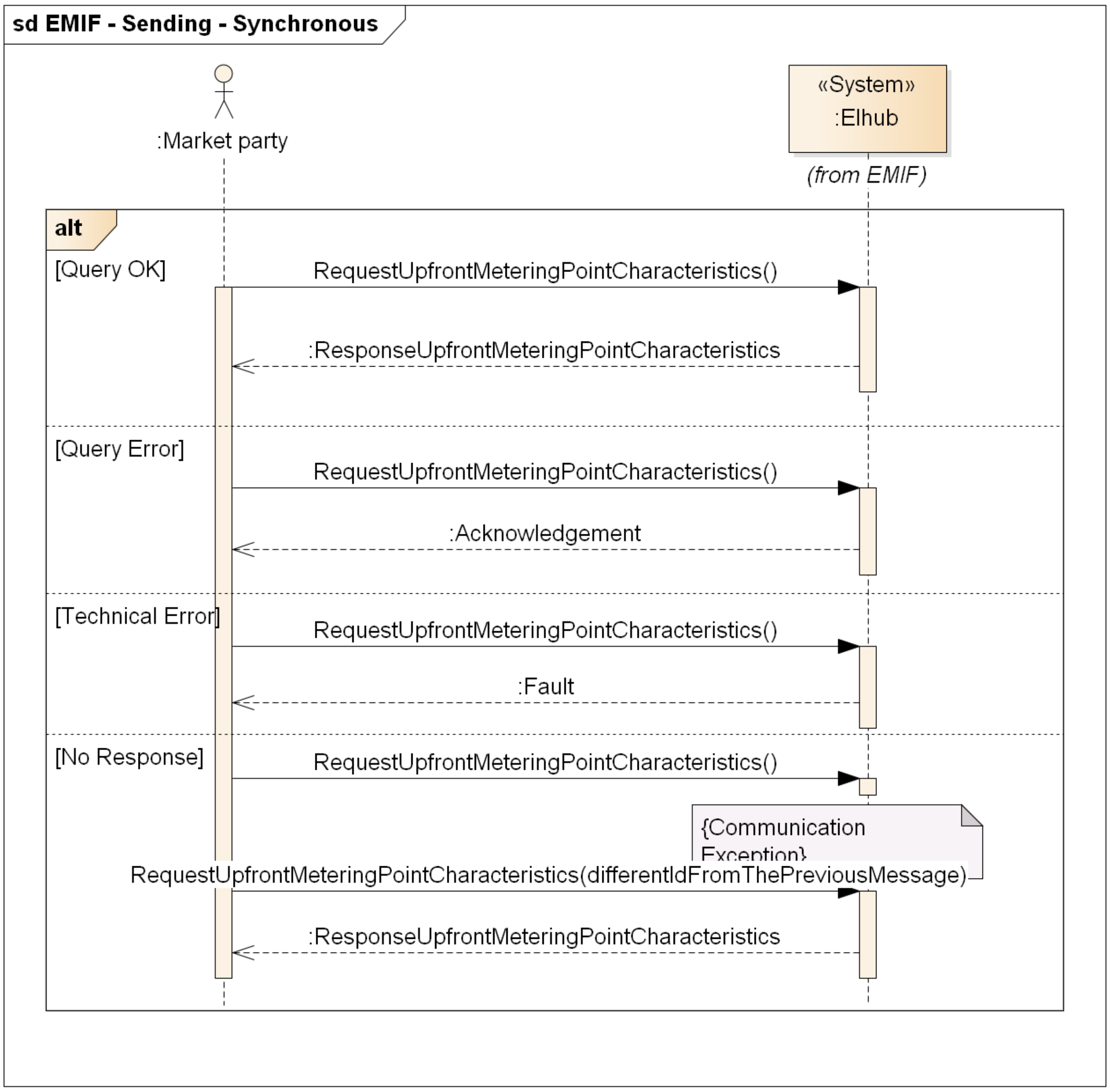

sd EMIF - Sending - Synchronous
:Market party
«System»
:Elhub
(from EMIF)
alt
[Query OK]
RequestUpfrontMeteringPointCharacteristics()
:ResponseUpfrontMeteringPointCharacteristics
[Query Error]
RequestUpfrontMeteringPointCharacteristics()
:Acknowledgement
[Technical Error]
RequestUpfrontMeteringPointCharacteristics()
:Fault
[No Response]
RequestUpfrontMeteringPointCharacteristics()
{Communication Exception}
RequestUpfrontMeteringPointCharacteristics(differentIdFromThePreviousMessage)
:ResponseUpfrontMeteringPointCharacteristics

Figure 3 Synchronous incoming message pattern

6.3 Validation

Elhub expects senders of messages to do an XSD validation of the data sent to Elhub in order to prevent Elhub from failing in the XSD validation. Elhub will still do an XSD validation.

7 Service providers and Roles

If a service provider acts on behalf of some other market party (grid owner A is owning the data, but service provider B is sending the messages), the service provider will specify the id of the real market party as the juridical sender in the header of the messages it sends on the messaging interface. The id of the service provider will be specified as the physical sender (market parties that does not have a service provider will specify its own id in both elements). The service provider will sign the message using its own certificate (messages are always signed by the physical sender). Elhub will check that the service provider is allowed to act on behalf of the specified market party in the specified role (it is possible to have different service providers for different roles, but not a finer granularity like BRS). A SOAP Fault will be returned if the service provider is not allowed to operate on behalf of the market party.

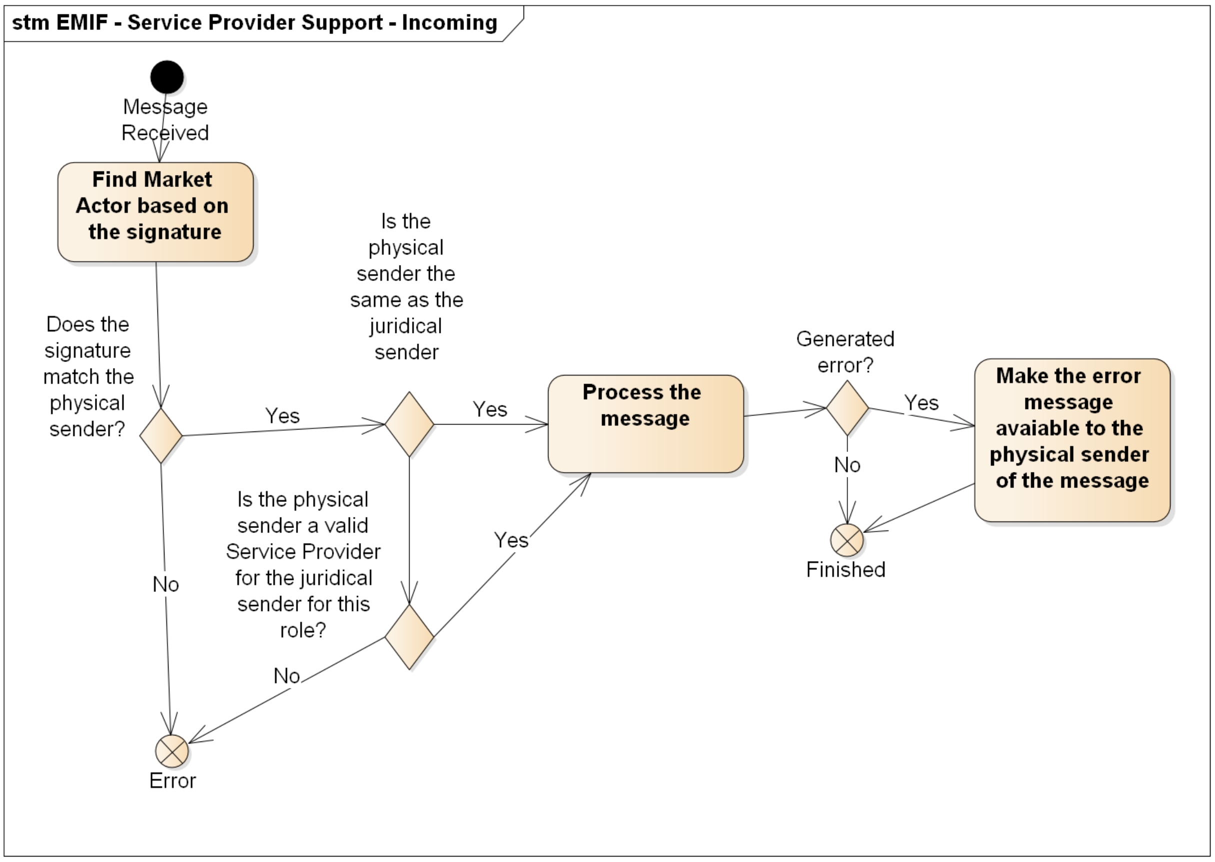

stm EMIF - Service Provider Support - Incoming
Message Received
Find Market Actor based on the signature
Does the signature match the physical sender?
Yes
No
Is the physical sender the same as the juridical sender
Yes
Is the physical sender a valid Service Provider for the juridical sender for this role?
Yes
No
Error
Process the message
Generated error?
Yes
No
Make the error message avaiable to the physical sender of the message
Finished

Figure 4 Service provider, incoming

If Elhub rejected the message (after accepting it technically), the response message will be made available to the physical sender of the original message.

When Elhub generates a message to a market party that is not a direct response to an incoming message from the same market party (for example a NotifyEndOfSupply), it checks if the market party has a service provider for the specific role and forwards the message to the service provider (makes it available for polling by the service provider) if that is the case. The message header in the returned message will specify the id of the real market party as the juridical recipient (there is no physical recipient element).

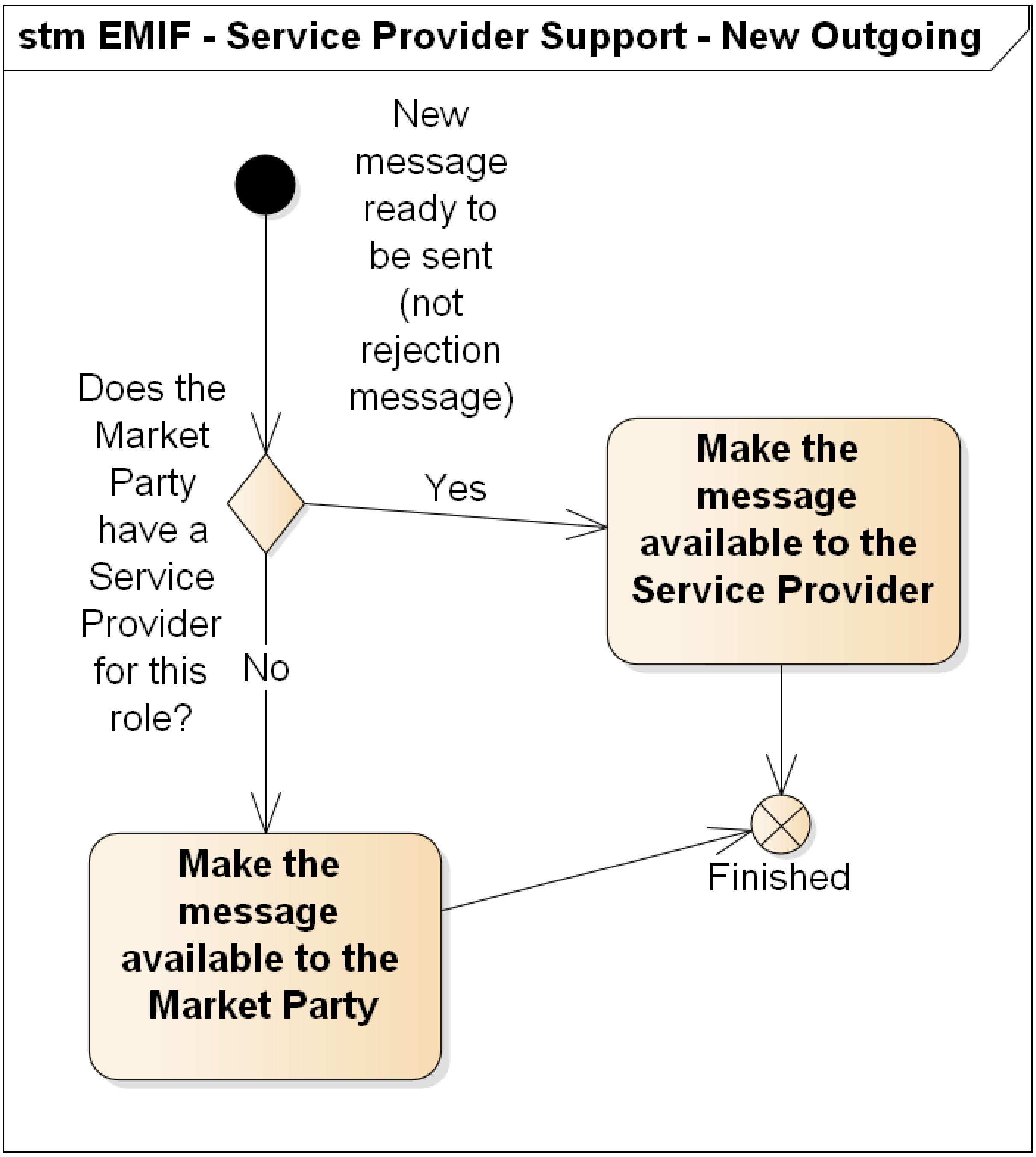

stm EMIF - Service Provider Support - New Outgoing
New message ready to be sent (not rejection message)
Does the Market Party have a Service Provider for this role?
Yes
Make the message available to the Service Provider
No
Make the message available to the Market Party
Finished

Figure 5 Service provider, outgoing

On the polling interface, the service provider is to specify its own id in the message header (both physical and juridical). In the message payload there is an option to specify the market party to get data for (along with the role). This allows the service provider to choose to get data belonging to itself (if it operates both as a service provider and a market party), one of the market parties it serves or all data that is available for it, regardless of market party (both itself and the market parties it serves). The messages returned from Elhub in the polling interface will always identify the market party owning the data in the message header. Similar options is also available to filter on the roles in order to get data for a single role or all roles.

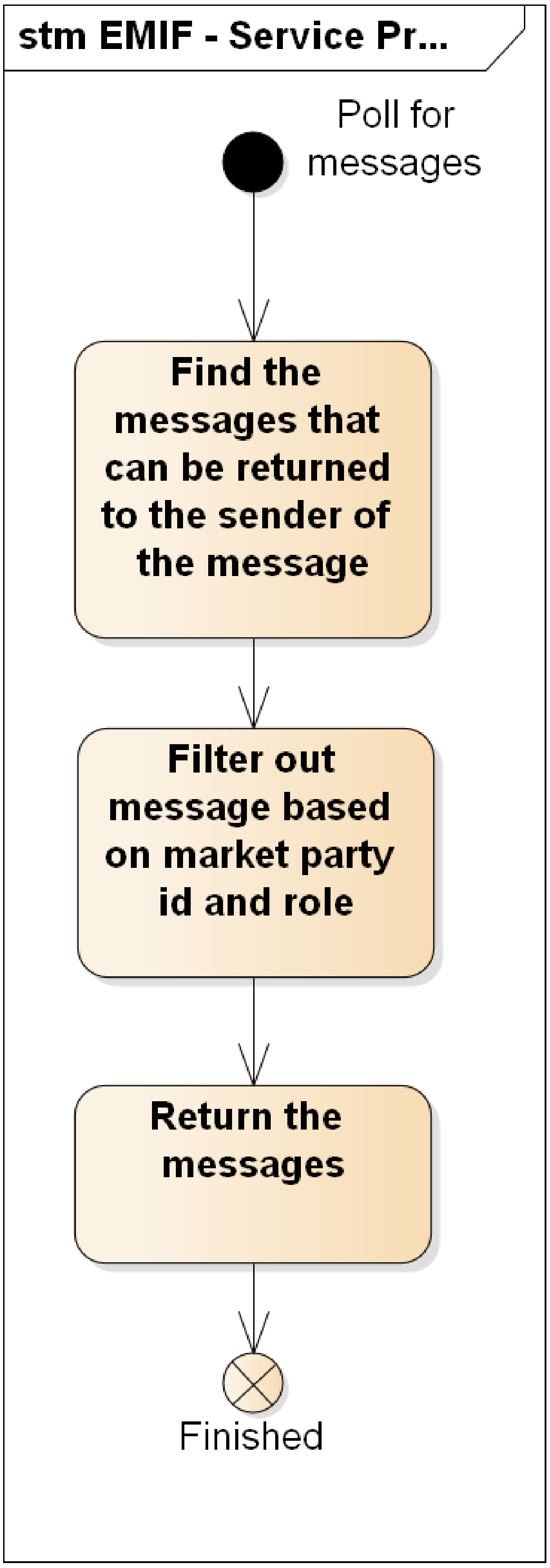

stm EMIF - Service Pr...
Poll for messages
Find the messages that can be returned to the sender of the message
Filter out message based on market party id and role
Return the messages
Finished

Figure 6 Service provider, polling

8 Polling Format

The polling message format is described in the BIM. The polling message will use the standard Header and Process Context. The juridical sender and role in the header of the polling message are not relevant, but are mandatory and must be specified. The payload part will specify the market party and role to get data for. The market party id is only relevant for service providers (as described above). The role is most relevant for service providers and grid owners. This allows the sender of the message to specify that messages for all its roles are to be returned or only a single role. The valid codes for roles and scheme/list identifiers are the same as the similar definitions in the header structure as shown in the BIM.

The returned message from the polling will be a PollForDataResponse message. This message will contain an identificator of the returned data set that is to be used when Acknowledging that the data has been received and the returned messages. The returned message will have a size limited to a number of MB and a number of messages. At the moment we are thinking of never returning more than 100MB of uncompressed data. There will never be more than 9999 messages in a returned data set, but some of these messages can be big (for example metering value messages). Also Elhub does not guarantee to return all available messages in one polling.

The Acknowledgement message with status 39 (Accepted) is to be used when confirming the receipt of the poll result by referring to the message returned from the polling.

Description of the poll response message (PollForDataResponse):

Element name	Description	Data Type
Identification	Identifier of the data set. Used as input in the Acknowledgement of the polling	UUID
ResultDataSet	The BIM messages returned in the polling	List of BIM messages of different types

All elements are mandatory, but the PollForDataResponse itself is optional (nillable="true" in the XSD) as there does not have to be anything to return.

9 File Descriptions

The files (WSDLs, XSDs and other documentation) will be stored in a folder representing the version of the interface. The first production release will have version v1 and all versions up to the first production release will also have version v1. Inside the version folder there are several folders:

- bim: Contains the XSDs for the BIM.
- example xmls: Contains examples xmls based on the BIM XSDs.
- wsdl: Contains the web service definitions. At the root of this folder you find the WSDLs and the XSD sub folder contains data structures used by the WSDL to envelope the BIM documents and to define non-BIM types like the polling messages. There are two sets of WSDLs, one with support for WS-Security and one without (separated with name suffix).
- bindings: Contains optional binding files to use when generating code (currently only JAXB binding files are included).
- soapui: Contains the different SoapUI projects.
- documentation: This document and the BIM description.

9.1 Binding Files

The main purpose of the binding files are to help JAXB to generate enumerations for XSD enumerations with numeric values. The C# code generator does this by default. There is currently one binding file available:

- jaxb_numeric.xml maps numeric enumerations to a textual representation. It does this by prefixing the numeric value with "VALUE_". This means an enumeration with value 6 is mapped to VALUE_6 and 89 is mapped to VALUE_89.

The mappings only apply to the generated code, the xml generated when serializing objects will still be the same. You can change the binding file if you do not like the names. It is optional to use it.

To use the mapping file with wsimport use the "-b" switch and refer to the binding file.

10 Request/Response Descriptions

Below is a short description of the different WSDL files.

10.1 MarketProcesses.wsdl

As mentioned previously, the incoming message has the name <BIM type >Request. The response is empty (http 200). The request will contain the corresponding BIM type. If the update is rejected, the rejection message will be returned in PollMarketProcesses. If a positive acknowledgement is requested, that will also be returned in PollMarketProcesses. The values can be sent compressed.

10.2 MeteringValues.wsdl

There are two messages in this file:

- The message is called CollectedDataRequest is used for sending metering values and contains a CollectedData element from the BIM. The response will be empty (http 200). If the update is rejected, the rejection message (negative Acknowledgement) will be returned in PollMeteringValues. There will be one Acknowledgement for each payload (metering point) that is rejected. The values must be sent compressed.
- AcknowledgeRequest uses the standard BIM Acknowledgement type. It is used to reject meter indexes or estimated annual consumption received from the balance suppliers. Only negative Acknowledgment messages are to be sent (PayloadResponseEvent must be 41). The OriginalBusinessDocumentReference in the PayloadResponseEvent must refer to the Identification in the header of the received CollectedData. The OriginalPayloadReference must refer to the Identification of the payload to reject. If several payloads are to be rejected, there must be one AcknowledgeRequest for each payload that is rejected.

10.3 Query.wsdl

The queries follow the same naming convention with <BIM type>Request. There is one synchronous message with a response message called <BIM type>Response (RequestUpfrontMeteringPointCharacteristicsResponse). The other queries are asynchronous and will respond with an empty response (http 200).

The result of the asynchronous queries will be returned in PollMarketProcesses except for queries for metering values (applies to RequestDataFromElhubRequest and depends on the specified query type) which will be returned in PollMeteringValues. Any acknowledgement message for an asynchronous query will be returned in the same polling service as the expected data would have been returned in. This also apply to positive Acknowledgement where applicable.

The values can be sent compressed.

10.4 PollMarketProcesses.wsdl

There are two messages in this file:

- PollForDataRequest uses the BIM type PollForData. The payload specifies which market party and role to get data for. As mentioned previously, the BRS identificator is to be POLL. It will return a set of messages that are available for retrieval based on the specified market party and role. The result data will be a set of BIM types where there can be several different types in a single response. In addition an id is returned for the data set. If there are no values to return, the response object will be empty (NULL). Confirmation messages will be returned as positive Acknowledgement message (status 39, Accepted) except for confirmation of Start and End of Supply which have specific confirmation messages. The result values will be sent compressed in a PollForDataResponse message. The input request can be sent compressed. Compression must be supported for the response.

- AcknowledgePollRequest uses the standard BIM Acknowledgement type. The OriginalBusinessDocumentReference in the PayloadResponseEvent must refer to the id returned in the top of the PollForDataResponse. The status type in PayloadResponseEvent must be 39 (Accepted) as Elhub does not accept rejections of pollings. There will also be no option for partial accept of the polling result. The reponse message is empty (http 200). The request can be sent compressed.

10.5 PollMeteringValues.wsdl

The messages in this file are similar to those in PollMarketProcesses.wsdl, but different data will be returned.

11 Message Transfer Intervals

11.1 Market Processes

One message in the Market Processes interface relates to only one metering point for one specific process. We encourage that these messages are sent as soon as practically possible instead of bulking them up and sending a relatively large set of messages in close succession. This still does not mean we discourage having a schedule that looks for messages to send as long as the interval between each sending is not too long (at most a couple of hours). In theory messages can be sent as often as you like on the MarketProcesses service. This does in particular apply to messages related to start and end of supply. Only send messages for updates of master data and customer information when there are changes to the data sent to Elhub, not when some other related data that is not sent to Elhub is changed in the source system.

11.2 Query

The Query service can be used as often as needed, but as the result from the queries only benefit the sender of the query, it must not be misused by querying for data untimely. This in particular applies to query for metering values which is not allowed be used as part of a normal flow as there are options to subscribe for metering values and have Elhub make new values available on the polling service automatically.

11.3 Metering Values

Sending of metering values must be bulked. The CollectedData message supports sending values for several metering points (9999) in the same message and also for more than one hour at a time (9999).

11.3.1 Hourly Values

Elhub will require that all metering values for a consumption metering point channel for a day is sent in the same message as of now, with the possibility of more frequent options in later versions of Elhub. You are not allowed to send the hourly values as they come in to your systems, but must buffer these and send them once the day is complete. Also the hourly values for a day for a metering point channel must be grouped together in a single payload element in the CollectedData message, not split on several payload elements. Note that this allows sending different channels for a metering point in different messages, but you are also allowed to send all or several channels in a single message.

A single CollectedData message must contain data for as many metering points as practically possible. As metering values are collected after midnight for the preceding day (either all values for the day or the last hour of the day), you can start sending the metering values for the metering points that have been completed in bulks. You do not have to wait until all metering points have been collected, but at the same time you are not allowed to send the metering points one by one. The general rule is you must delay sending until you have metering values for at least 1000 metering points and then send these in a single message (you are allowed to wait until you have metering values for 9999 metering points). If the 07:00 limit for metering values is closing, you can send the metering values in smaller batches in order to not risk sending them after the 07:00 limit. The reason for doing so is when you fear the data will not be processed by Elhub by 07:00. There will be no hard enforcing of these rules by rejecting small messages, but this will be monitored. In addition to this, in the 00:00-07:00 interval you are allowed to send messages with fewer than 1000 metering points if it is more than 1 hour since the last time a message was sent. That means you do not have to buffer more than 1000 messages if it is more than 1 hour since the last sending of metering values and that you can send fewer than 1000 metering points if it is more than 1 hour since the last sending of metering values. Note that sending messages every hour on the hour are to be avoided as that would give an unnatural peak.

The rules above also applies to corrections done outside the 00:00-07:00 period (07:00-00:00). These messages are also be buffered and sent in bulks of 1000 metering points or more, but you are allowed to send fewer than 1000 metering points in a message if it is more than 1 hour since the last message. For corrections, you are allowed to send only the corrected hours, you do not have to send the entire day. If there are several corrections for a single day, it is recommended that these are sent in the same message. This also apply if there are corrections for two non-consecutive hours, these are not be sent as two single hourly corrections, but as a single correction with at least all values in the interval, but sending the entire day is also allowed.

For exchange points and larger production points, you are allowed, after agreement with Elhub, to send metering values hourly, but the others rules regarding grouping of messages still apply. This means you can handle exchange points and production points similar as described in the correction scenario.

Note that these rules applies per combination of physical and juridical sender. This means a service provider does not have to consider these rules across all market parties it is serving. It also means the physical senders for a market party with more than one physical sender of metering values do not have to consider these rules across the different physical senders. However a market party with several head-end instances will have to look at the rules combined for all nodes as long as there is a single physical sender of the metering values. This means the different head-end instances cannot build messages independently of each other and must go through an MDM or similar.

As mentioned, the CollectedData message supports 9999 metering points with 9999 hourly values, but Elhub will not support large amount of both metering points and hourly values in the same message. If the number of metering points is high, the number of hourly values must be low. Similarly, if you need to transfer a lot of hourly values for some metering points, the number of metering points must be low. The general rule is to limit the total number of hourly values (summed for all metering points) in one message to no more than 250 000. This means if you send hourly values for a single day, you can send 9999 metering points, but if you want to increase the number hourly values, the number of metering points must be lowered. Elhub will reject metering value messages with more than 250 000 hourly values. The rules here may seem to conflict with the rule of at least 1000 metering points per message, but that is not the case as the main rule can be translated to sending at least 25 000 hourly values per message.

Metering values are to be sent in as early as possible with regard to the 07:00 (Central European Time with daylight saving) limit and not delayed until close to 07:00. By starting the data collection just after midnight and sticking to the rules described above, it is expected data to be sent to Elhub before 07:00 with a good margin. Metering values not processed (received and stored) by Elhub within 07:00 will not be included in the calculations. In order to make sure all metering values are processed by Elhub, it is recommended that the large bulk of metering values have been sent by 05:00. Note that for the day of transition to daylight saving time (last Sunday in March), the period for sending metering values will be 1 hour shorter than normal. This means your systems must be able to finish all collection and sending of metering values in 6 hours. At this transition day, Elhub will still need to receive the messages early enough to be able to process the message before 07:00 in order to have the values included in the calculations.

11.3.2 Non-hourly Metering Values

Non-hourly metering values must have the same sending pattern as the hourly values with regard to bulking metering points. These values are in no way limited to the 00:00-07:00 period. As hourly metering values and non-hourly metering values belong to different BRSes, it is not possible to mix these in a single message. The non-hourly metering points are also to be bulked into messages with at least 1000 metering points in a single message and you will be allowed to send a message with fewer metering points if it has been 1 hour or more since the last sending of a message with non-hourly metering points (hourly and non-hourly messages are seen separately here so the 1 hour rule runs independently, which means you can send two small messages per hour as long as they are for different types of metering points).

11.4 Estimated Annual Consumption

Estimated annual consumption in BRS-NO-317 is to be grouped with the same rules as for non-hourly metering values with at least 1000 metering points per message and the 1 hour rule.

11.5 Polling

You will not know when there will be data available on the polling services. The expectation is that the outgoing messages from Elhub will not be needed immediately so continuous polling by sending a new polling request immediately after the previous one ended will not be needed. This means a more or less fixed polling schedule is recommended. The polling scheme will be like this (does not show processing of the received data and acknowledgement as the important thing to show here is the waiting scheme):

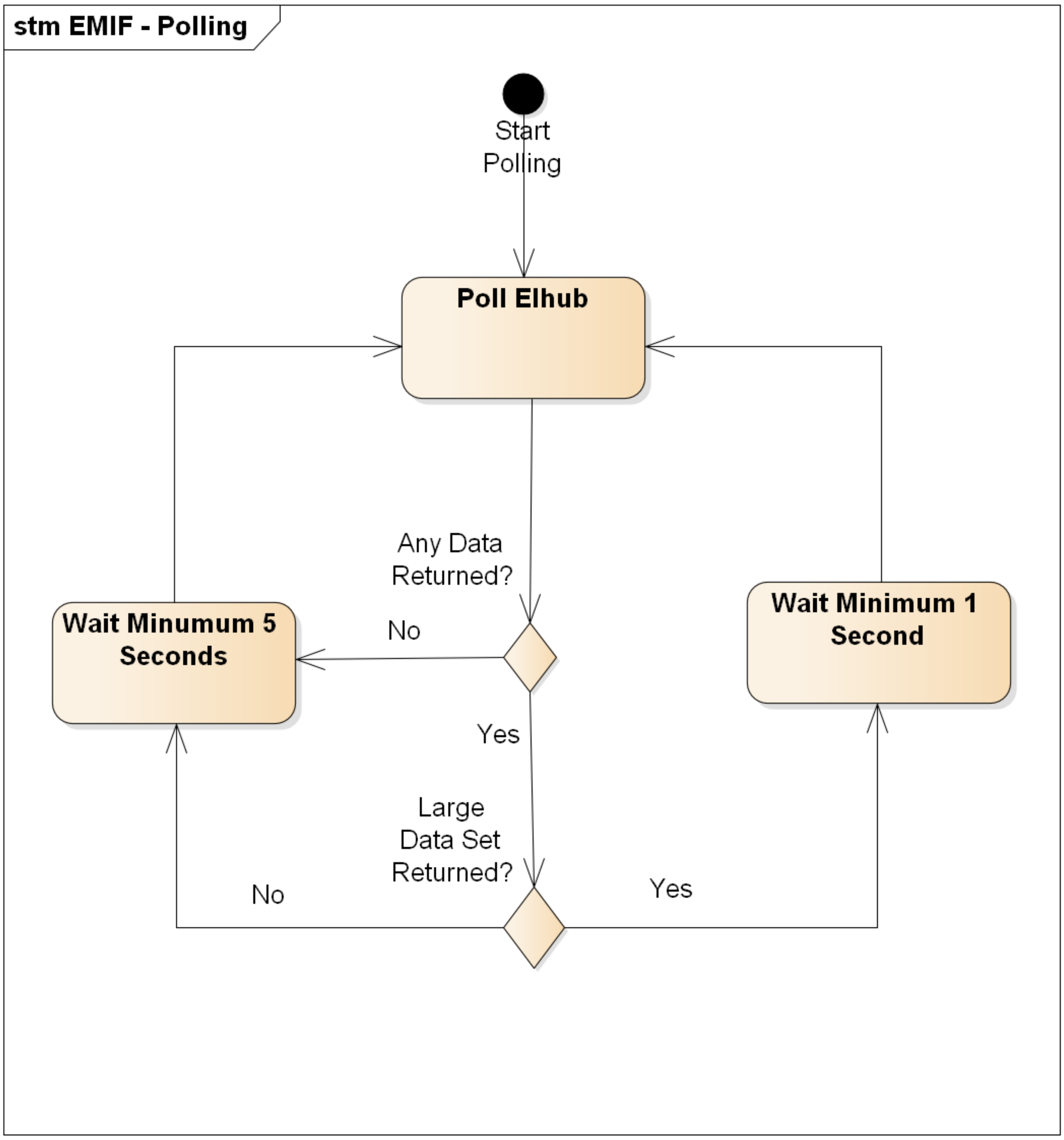


Figure 7 Polling pattern

- If no data or a small data set was returned, you must wait at least 5 seconds before polling again (counting from the time the response was received).
- If the response included a large dataset you are allowed to do a new polling after minimum a 1 second delay (counting from the time the response was received) as Elhub may have more values. You are not required to do this. Elhub has configurable options for how many messages to include in a polling result and how many MB will at most be returned. At the moment these configurations are set to 1000 messages and 100MB. This means if you received a dataset less than this, you are to wait for at lest 5 seconds before polling again. If you received this amount of data (either 1000 messages or close to 100MB in xml) you are allowed to poll after minimum a 1 second delay. These message size limits can change.
- Polling must be done at least once per hour in order to not have huge outgoing queues building up. If the polling interval is too long you risk losing data. We have not defined how

long a period we guarantee to keep the messages, but it is expected to at least be a number of days, but that is only to support downtime issues.

- Even though it is allowed to poll every 5 seconds, Elhub would appreciate if polling intervals were longer. Polling every 5 seconds could be done when you are waiting for a query result, but in other circumstances, longer wait periods would mean less stress on Elhub.

The waiting above is until you can query for the same market party and role. If the query is split into querying per role, you are allowed to query for the next role immediately, but when all roles have been polled, the waiting must be done.

Elhub has two polling interfaces and the polling rules are the same for both interfaces. You are not required to synchronize polling on the two interfaces. This also means there is no requirement that after polling on the market processes interface, you must wait 5 seconds before polling on the metering values interface.

11.6 Monitoring

The message transfer rates will be monitored in order to detect any misuse of the interface. This will in particular apply to the "Request Upfront Metering Point Characteristics" message (the replacement of NUBIX).

11.7 Parallel Sending of Messages

Elhub supports receiving messages in parallel. This means is it possible to send in multiple messages simultaneously, but for the same metering point we recommend not not to send messages in parallel. The reason for this is in this case Elhub cannot guarantee processing the messages in the expected order as Elhub may receive the second message before the first one. The recommendation is to at least wait until the first message has been received by Elhub (http response received) before sending the next message for the same metering point.

12 SOAP Fault Definition

When there is a technical error, a SOAP Fault will be returned. The SOAP Fault will have these properties:

XML element	Description	Card	Max Length	Content
faultcode	Who made the mistake that resulted in the fault. Standard part of a SOAP Fault soapenv:Client - The client made a mistake soapenv:Server - The server has en error	1..1		http://schemas.xmlsoap.org/soap/envelope/
faultstring	Standard part of a SOAP Fault. Description of the fault. This element cannot be processed automatically.	1..1		http://schemas.xmlsoap.org/soap/envelope/
CodeGroup	The type of fault: • XSD: The message failed the XSD validation • Compression: Message data is not compressed or the client does not accept to receive compressed data • Security: The message failed in the signature check, authorization etc.	1..1	Enumeration	

XML element	Description	Card	Max Length	Content
	• System: An internal error caused Elhub to be unable to process the request • UUID: The message ID is not unique (has been received previously) • Size: The XML is too big (in MB). Currently only relevant for CollectedData • Date: The message is not allowed to be sent in at this date (the BRS is not active at the moment) • Other: Some other error situation. Used to be able to update with new validations without having to update the interface immediately. "Other" is not intended for long time use and a			

XML element	Description	Card	Max Length	Content
	proper code will, normally, be introduced in the first upcoming update of the interface.			
Description	Short description of the error. This element cannot be processed automatically.	1..1	A100	
ExceptionDateTime	When the fault was created	1..1		YYYY-MM-DDTHH:MM:SSZ or YYYY-MM-DDTHH:MM:SS[+-][HH:MM]
FaultText	Details of the fault like information about the element that failed XSD validation. This element cannot be processed automatically.	0..1	A1000	

13 Message Identification

As specified in the BIM, all message identifications are to be UUIDs. Note that these are to be real UUIDs generated by standard UUID generators and not just a string that follows the UUID pattern. We recommend using the version 4 (random) of UUID generation, but this is not a requirement.

In the CollectedData messages, there is an Identification element inside the payload part (PayloadEnergyTimeSeries). Elhub will not validate that this identification id unique over time, but it must be unique within a single CollectedData message. Elhub will always send unique Identification values in the payload part when it sends metering values (NotifyValidatedDataForBillingEnergy and PriceVolumeCombinationForReconciliation). It is recommended to use a payload id that is unique over time. The id of the CollectedData message itself (Identification inside the He ader element) must be unique over time. This will be validated by Elhub.

14 XML contents

14.1 Namespace

Elhub highly recommends to define namespaces in the header of the message and not on the individual elements in order to reduce message size. If the namespace is put on the individual xml elements, there is a risk that the message will be rejected due to size constraints (applies to CollectedData). If it turns out that messages with repeated namespaces results in problems, we will define a hard requirement regarding the use of namespace prefixes.

Namespace prefixes are to be small like ns1, ns2 etc and urn1, urn2 etc.

14.2 Whitespaces

Elhub highly recommend to send messages without pretty printing (newline and whitespaces). If pretty printing is used, there is a risk that the message will be rejected due to size constraints (applies to CollectedData). If it turns out that messages with pretty printing results in problems, we will define a hard requirement regarding pretty printing.

14.3 Encoding

Messages sent to Elhub must be encoded as utf-8 (Content-Type: text/xml; charset=UTF-8). Data in responses from Elhub will be similarly encoded. Messages with other encodings will be rejected. If the wrong encoding is used, the "Other" Fault Code will be returned.

15 Appendix A - SoapUI package

To show the main services and interaction patterns we have included a SoapUI package with support for the Elhub WSDLs. This appendix includes instructions for this SoapUI package.

The intention of this package is for developers and architects to understand the web services and the integration patterns required to integrate with Elhub.

You may connect your own applications to the mock projects. The returned values are quite fixed and for the most part do not relate directly to the request sent in. Some timestamps and message ids will be automatically generated. There is also no memory/state in the SoapUI package so no checks for duplicate message ids or similar will be done.

15.1 Security

The projects shown below does not have support for WS-Security. In addition to those projects, there are also duplicate projects with a suffix "WS-Security" that use WSDLs where WS-Security is enabled.

The projects with WS-Security is running on the same ports as the projects without WS-Security. This means you cannot run both in parallel without doing the proper port changes.

The projects with WS-Security will generate signed messages (the signed request can be seen in the Raw tab in the request). The signed response is shown directly.

The messages are signed using dummy certificates. The projects supporting security does not do any security validations and will accept unsigned requests as well.

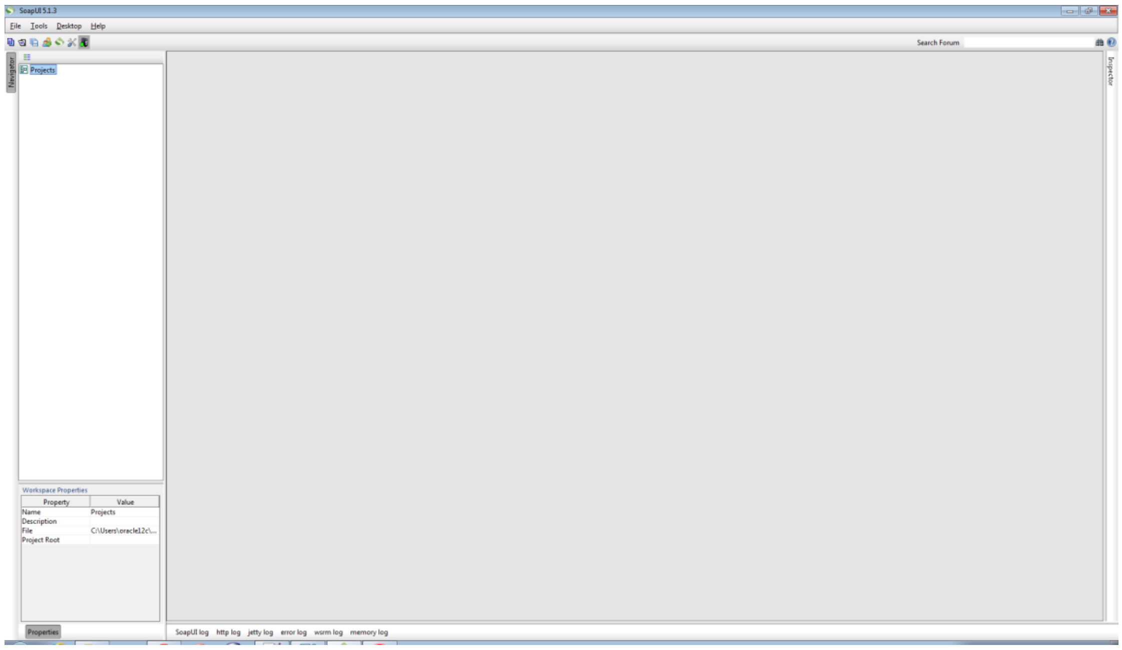
15.2 Preparations

15.2.1 Prerequisite

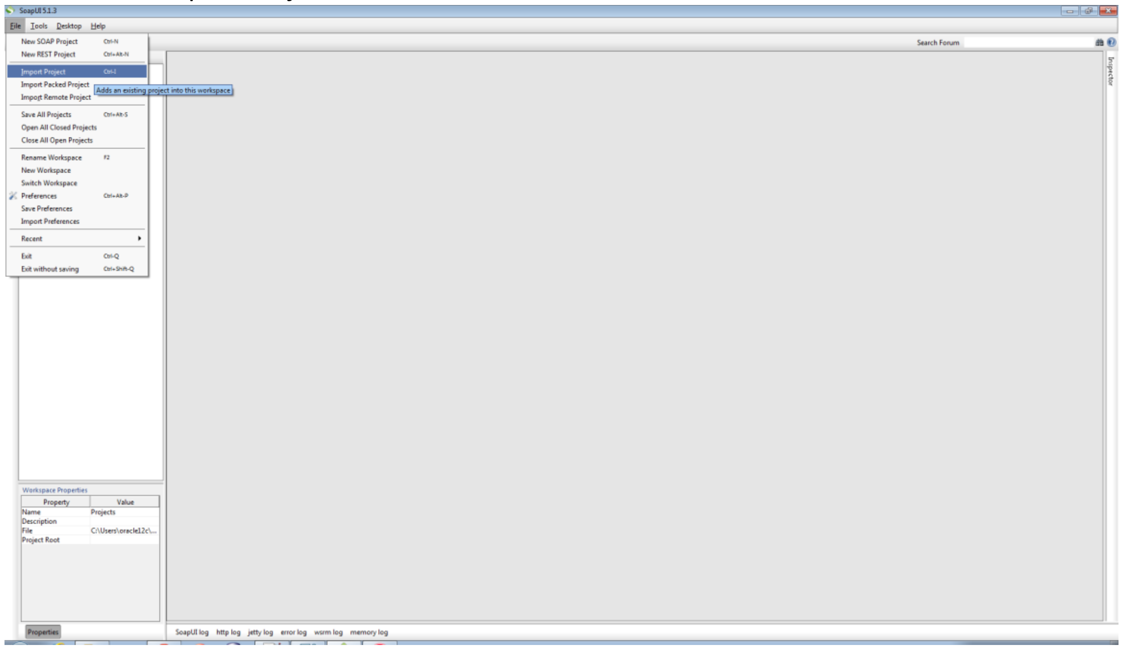
- If you haven't installed SoapUI (http://www.soapui.org/) already, begin with downloading it and installing it.

15.2.2 Importing projects to SoapUI

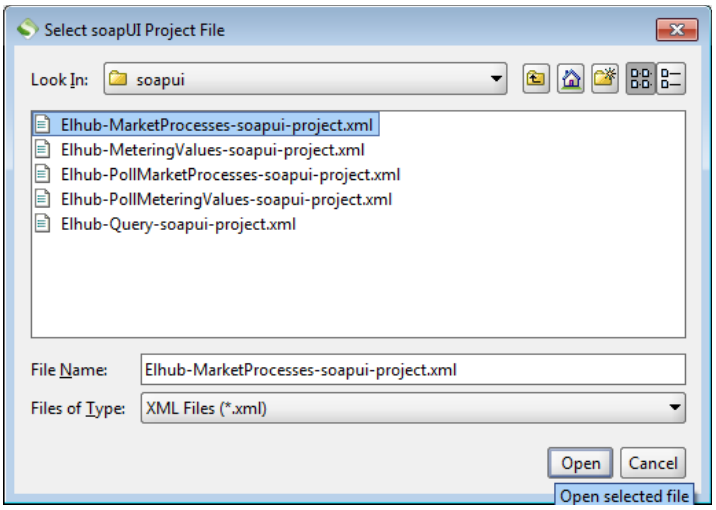
1. Start by opening SoapUI



2. Go to File -> Import Project



3. Navigate to the soapui-folder in the package and select a project and press "Open".



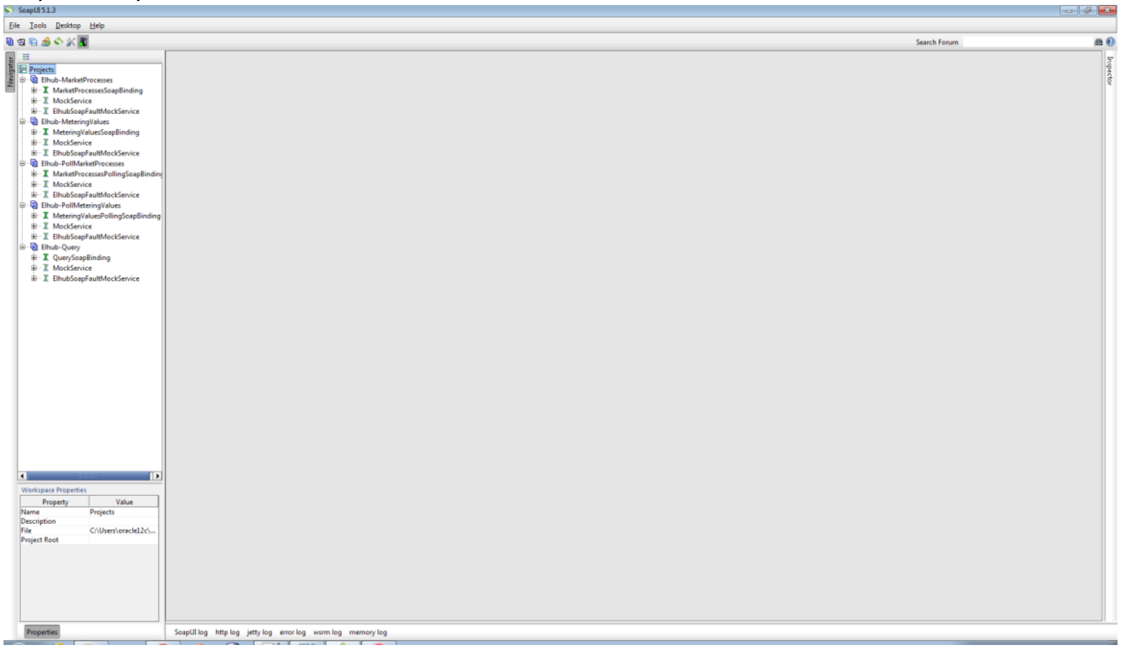
Look In: soapui

- Elhub-MarketProcesses-soapui-project.xml
- Elhub-MeteringValues-soapui-project.xml
- Elhub-PollMarketProcesses-soapui-project.xml
- Elhub-PollMeteringValues-soapui-project.xml
- Elhub-Query-soapui-project.xml

File Name: Elhub-MarketProcesses-soapui-project.xml

Files of Type: XML Files (*.xml)

4. Repeat steps 1-3 for all SoauUI-files.

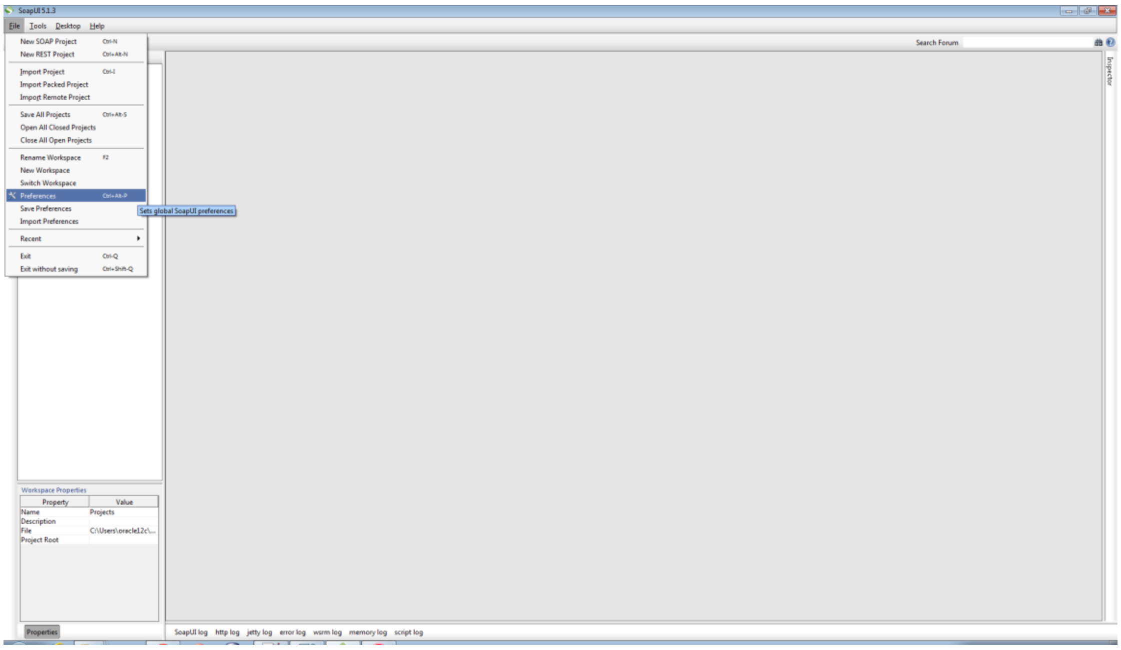


Workspace Properties

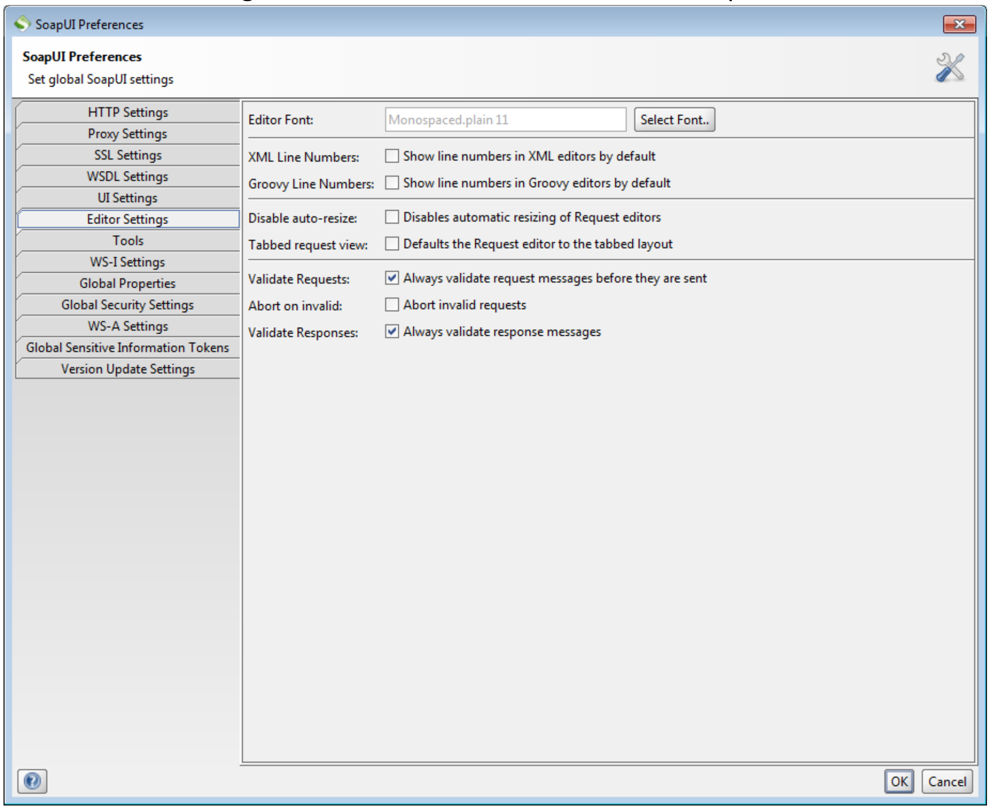
Property	Value
Name	Projects
Description	
File	C:\Users\oracle12c\...
Project Root	

15.2.3 SoapUI Preferences

1. Go to "File -> Preferences"



2. Go to "Editor Settings" and make sure that "Abort invalid requests" is not checked.



SoapUI Preferences
Set global SoapUI settings

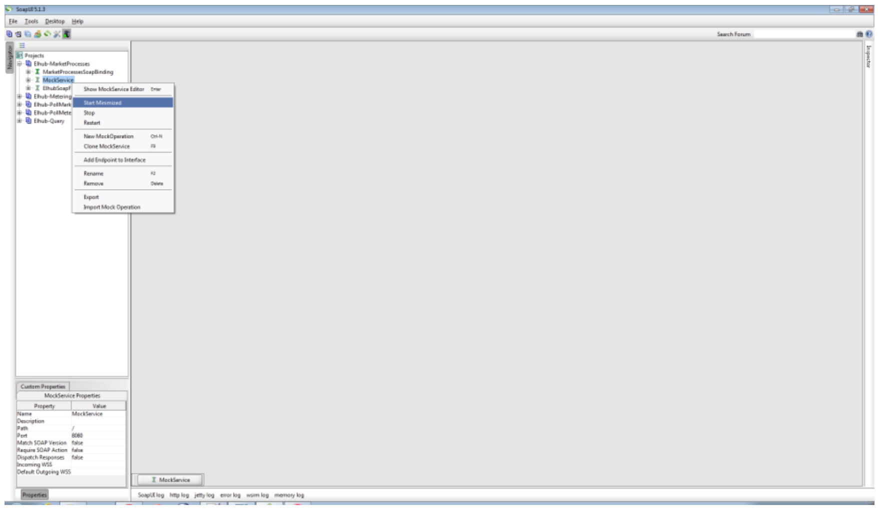
Editor Font:	Monospaced.plain 11
XML Line Numbers:	☐ Show line numbers in XML editors by default
Groovy Line Numbers:	☐ Show line numbers in Groovy editors by default
Disable auto-resize:	☐ Disables automatic resizing of Request editors
Tabbed request view:	☐ Defaults the Request editor to the tabbed layout
Validate Requests:	☑ Always validate request messages before they are sent
Abort on invalid:	☐ Abort invalid requests
Validate Responses:	☑ Always validate response messages

15.3 Sample 1 - Elhub-MarketProcesses

Before you start using the requests, you need to start the two mock-services.

15.3.1 Start mockservices

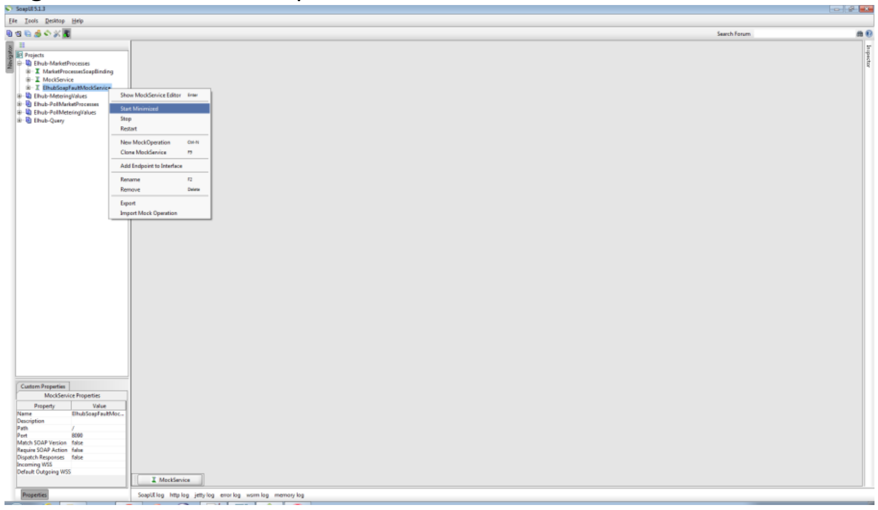
1. Expand "Elhub-MarketProcess"-project and right click on "MockService" and select "Start Minimized".



MockService Properties

Property	Value
Name	MockService
Description	
Path	/
Port	8080
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

2. Right click on "ElhubSoapFaultMockService" and select "Start Minimized".

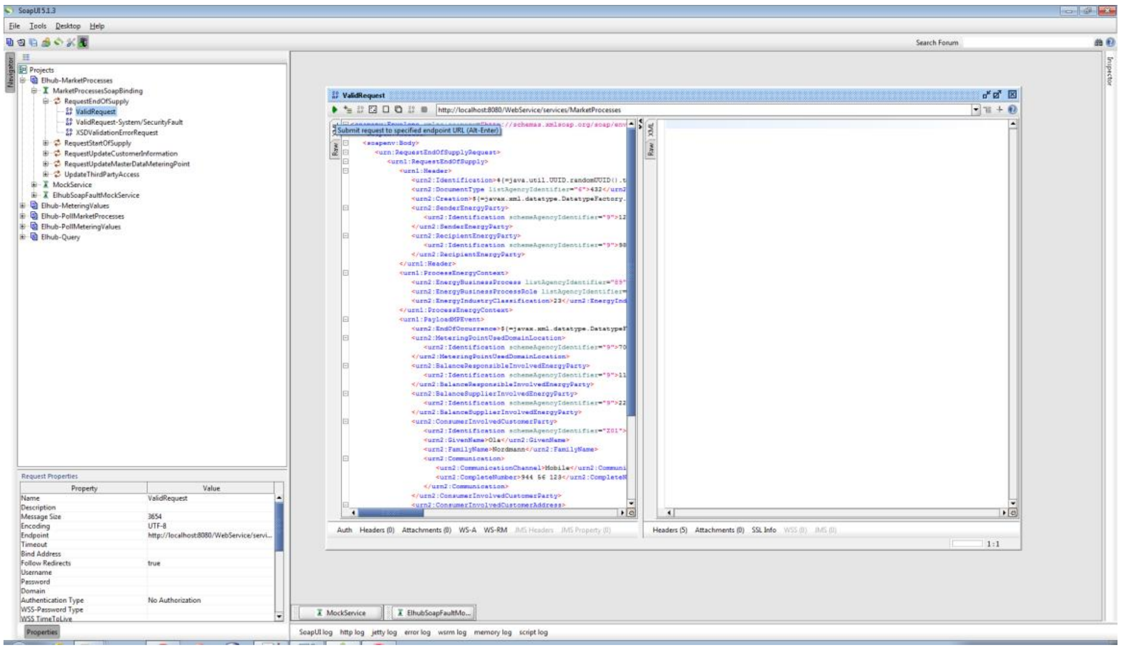


MockService Properties

Property	Value
Name	ElhubSoapFaultMoc...
Description	
Path	/
Port	8080
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

15.3.2 Calling the mock services:

1. Expand "Elhub-MarketProcesses" -> "MarketProcessSoapBinding" and select any of the operations. For each of the operations, there are defined three different types of requests:
   a. ValidRequest - This is a syntactically correct message.
   b. ValidRequest - System/SecurityFault - This is a syntactically correct message, however the mock service returns a ElhubSoapFault response illustrating that there were some sort of error related to either security or Elhub. This is a fault scenario.
   c. XSDValidationErrorRequest - This is not a syntactically valid request and therefore, the response returned by the mock service is a ElhubSoapFault of type XSD.
2. Double click on any of the requests and click on the top-left icon (play) to send a request.



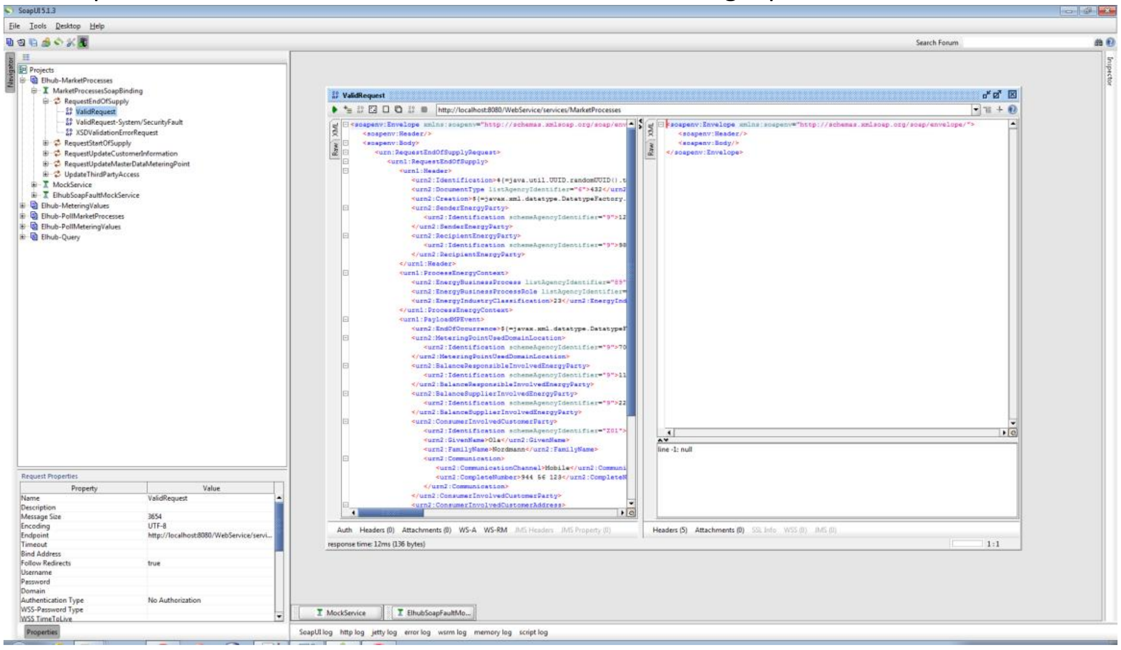
http://localhost:8080/WebService/services/MarketProcesses

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/env
   <soapenv:Body>
      <urn:RequestEndOfSupplyRequest>
         <urn1:RequestEndOfSupply>
            <urn1:Header>
               <urn2:Identification>${=java.util.UUID.randomUUID().t
               <urn2:DocumentType listAgencyIdentifier="6">432</urn2
               <urn2:Creation>${=javax.xml.datatype.DatatypeFactory.
               <urn2:SenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">12
               </urn2:SenderEnergyParty>
               <urn2:RecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">98
               </urn2:RecipientEnergyParty>
            </urn1:Header>
            <urn1:ProcessEnergyContext>
               <urn2:EnergyBusinessProcess listAgencyIdentifier="89"
               <urn2:EnergyBusinessProcessRole listAgencyIdentifier=
               <urn2:EnergyIndustryClassification>23</urn2:EnergyInd
            </urn1:ProcessEnergyContext>
            <urn1:PayloadMPEvent>
               <urn2:EndOfOccurrence>${=javax.xml.datatype.DatatypeF
               <urn2:MeteringPointUsedDomainLocation>
                  <urn2:Identification schemeAgencyIdentifier="9">70
               </urn2:MeteringPointUsedDomainLocation>
               <urn2:BalanceResponsibleInvolvedEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">11
               </urn2:BalanceResponsibleInvolvedEnergyParty>
               <urn2:BalanceSupplierInvolvedEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">22
               </urn2:BalanceSupplierInvolvedEnergyParty>
               <urn2:ConsumerInvolvedCustomerParty>
                  <urn2:Identification schemeAgencyIdentifier="Z01">
                  <urn2:GivenName>Ola</urn2:GivenName>
                  <urn2:FamilyName>Nordmann</urn2:FamilyName>
                  <urn2:Communication>
                     <urn2:CommunicationChannel>Mobile</urn2:Communi
                     <urn2:CompleteNumber>944 56 123</urn2:CompleteN
                  </urn2:Communication>
               </urn2:ConsumerInvolvedCustomerParty>
               <urn2:ConsumerInvolvedCustomerAddress>
```

Request Properties

Property	Value
Name	ValidRequest
Description	
Message Size	3654
Encoding	UTF-8
Endpoint	http://localhost:8080/WebService/servi...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Type	No Authorization
WSS-Password Type	
WSS TimeToLive	

3. The response from the mock service can be viewed on the right pane:



http://localhost:8080/WebService/services/MarketProcesses

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/env
   <soapenv:Header/>
   <soapenv:Body>
      <urn:RequestEndOfSupplyRequest>
         <urn1:RequestEndOfSupply>
            <urn1:Header>
               <urn2:Identification>${=java.util.UUID.randomUUID().t
               <urn2:DocumentType listAgencyIdentifier="6">432</urn2
               <urn2:Creation>${=javax.xml.datatype.DatatypeFactory.
               <urn2:SenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">12
               </urn2:SenderEnergyParty>
               <urn2:RecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">98
               </urn2:RecipientEnergyParty>
            </urn1:Header>
            <urn1:ProcessEnergyContext>
               <urn2:EnergyBusinessProcess listAgencyIdentifier="89"
               <urn2:EnergyBusinessProcessRole listAgencyIdentifier=
               <urn2:EnergyIndustryClassification>23</urn2:EnergyInd
            </urn1:ProcessEnergyContext>
            <urn1:PayloadMPEvent>
               <urn2:EndOfOccurrence>${=javax.xml.datatype.DatatypeF
               <urn2:MeteringPointUsedDomainLocation>
                  <urn2:Identification schemeAgencyIdentifier="9">70
               </urn2:MeteringPointUsedDomainLocation>
               <urn2:BalanceResponsibleInvolvedEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">11
               </urn2:BalanceResponsibleInvolvedEnergyParty>
               <urn2:BalanceSupplierInvolvedEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">22
               </urn2:BalanceSupplierInvolvedEnergyParty>
               <urn2:ConsumerInvolvedCustomerParty>
                  <urn2:Identification schemeAgencyIdentifier="Z01">
                  <urn2:GivenName>Ola</urn2:GivenName>
                  <urn2:FamilyName>Nordmann</urn2:FamilyName>
                  <urn2:Communication>
                     <urn2:CommunicationChannel>Mobile</urn2:Communi
                     <urn2:CompleteNumber>944 56 123</urn2:CompleteN
                  </urn2:Communication>
               </urn2:ConsumerInvolvedCustomerParty>
               <urn2:ConsumerInvolvedCustomerAddress>
```

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

```
line -1: null
```

response time: 12ms (136 bytes)

Request Properties

Property	Value
Name	ValidRequest
Description	
Message Size	3654
Encoding	UTF-8
Endpoint	http://localhost:8080/WebService/servi...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Type	No Authorization
WSS-Password Type	
WSS TimeToLive	

15.3.3 Endpoints:

There are two mock services:

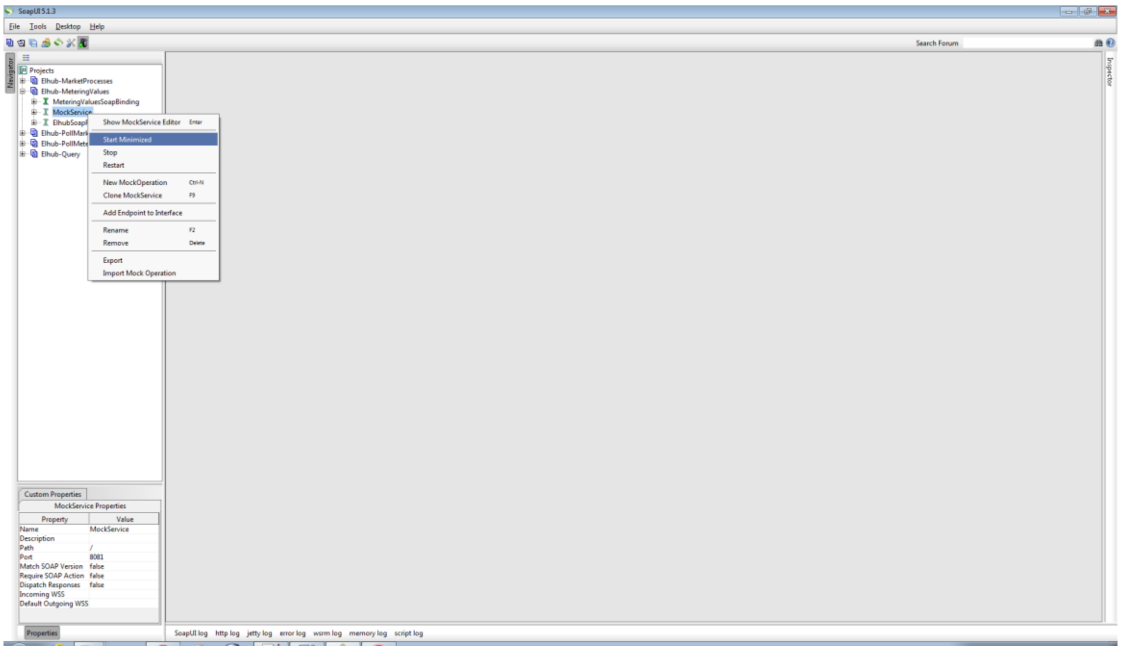
- MockService:
  - Listens on: http://localhost:8080/WebService/services/MarketProcesses
  - Returns a syntactically correct response
  - Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.
- ElhubSoapFaultMockService
  - Listens on: http://localhost:8090/WebService/services/MarketProcesses
  - Returns a ElhubSoapFault of type Security or System
  - Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.

15.4 Sample 2 - Elhub-MeteringValues

Before you start using the requests, you need to start the two mock-services.

15.4.1 Start mockservices

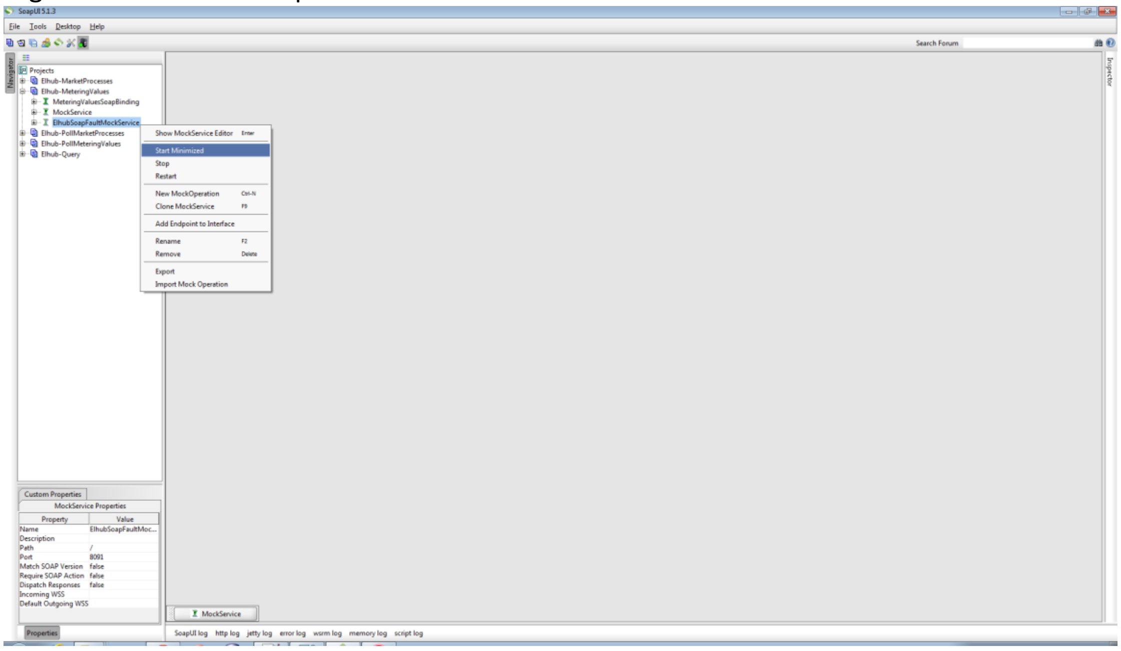
1. Expand "Elhub-MeteringValues"-project and right click on "MockService" and select "Start Minimized".



MockService Properties

Property	Value
Name	MockService
Description	
Path	/
Port	8081
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

2. Right click on "ElhubSoapFaultMockService" and select "Start Minimized".

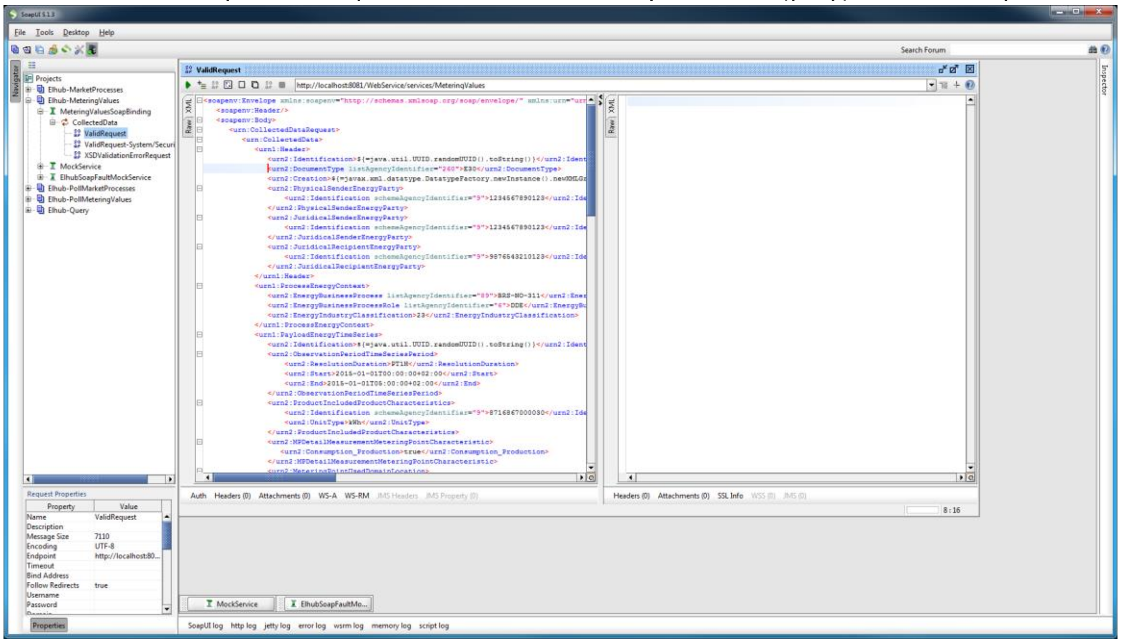


MockService Properties

Property	Value
Name	ElhubSoapFaultMoc...
Description	
Path	/
Port	8091
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

15.4.2 Calling the mock services:

1. Expand "Elhub-MeteringValues" -> "MeteringValuesSoapBinding" and select the CollectedData operation. For the operations, there are defined three different types of requests:
   a. ValidRequest - This is a syntactically correct message.
   b. ValidRequest - System/SecurityFault - This is a syntactically correct message, however the mock service returns a ElhubSoapFault response illustrating that there were some sort of error related to either security or Elhub. This is a fault scenario.
   c. XSDValidationErrorRequest - This is not a syntactically valid request and therefore, the response returned by the mock service is a ElhubSoapFault of type XSD.
2. Double click on any of the requests and click on the top-left icon (play) to send a request.



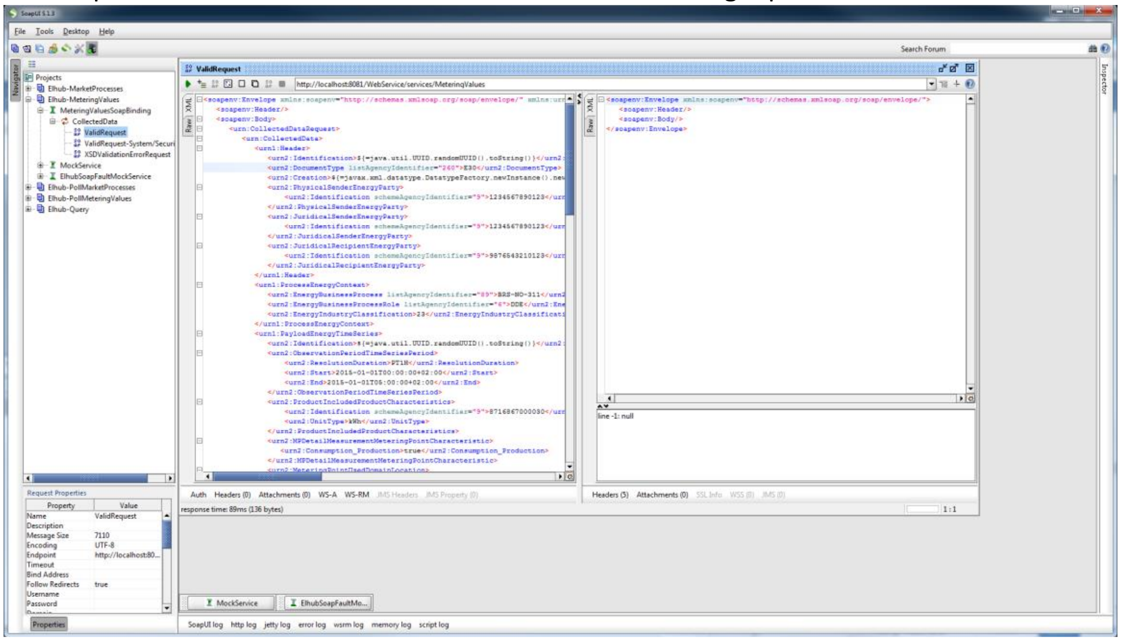
http://localhost:8081/WebService/services/MeteringValues

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn
   <soapenv:Header/>
   <soapenv:Body>
      <urn:CollectedDataRequest>
         <urn:CollectedData>
            <urn1:Header>
               <urn2:Identification>${=java.util.UUID.randomUUID().toString()}</urn2:Ident
               <urn2:DocumentType listAgencyIdentifier="260">E30</urn2:DocumentType>
               <urn2:Creation>${=javax.xml.datatype.DatatypeFactory.newInstance().newXMLGr
               <urn2:PhysicalSenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">1234567890123</urn2:Ide
               </urn2:PhysicalSenderEnergyParty>
               <urn2:JuridicalSenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">1234567890123</urn2:Ide
               </urn2:JuridicalSenderEnergyParty>
               <urn2:JuridicalRecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">9876543210123</urn2:Ide
               </urn2:JuridicalRecipientEnergyParty>
            </urn1:Header>
            <urn1:ProcessEnergyContext>
               <urn2:EnergyBusinessProcess listAgencyIdentifier="89">BRS-NO-311</urn2:Ener
               <urn2:EnergyBusinessProcessRole listAgencyIdentifier="6">DDE</urn2:EnergyBu
               <urn2:EnergyIndustryClassification>23</urn2:EnergyIndustryClassification>
            </urn1:ProcessEnergyContext>
            <urn1:PayloadEnergyTimeSeries>
               <urn2:Identification>${=java.util.UUID.randomUUID().toString()}</urn2:Ident
               <urn2:ObservationPeriodTimeSeriesPeriod>
                  <urn2:ResolutionDuration>PT1H</urn2:ResolutionDuration>
                  <urn2:Start>2015-01-01T00:00:00+02:00</urn2:Start>
                  <urn2:End>2015-01-01T05:00:00+02:00</urn2:End>
               </urn2:ObservationPeriodTimeSeriesPeriod>
               <urn2:ProductIncludedProductCharacteristics>
                  <urn2:Identification schemeAgencyIdentifier="9">8716867000030</urn2:Ide
                  <urn2:UnitType>kWh</urn2:UnitType>
               </urn2:ProductIncludedProductCharacteristics>
               <urn2:MPDetailMeasurementMeteringPointCharacteristic>
                 <urn2:Consumption_Production>true</urn2:Consumption_Production>
               </urn2:MPDetailMeasurementMeteringPointCharacteristic>
               <urn2:MeteringPointUsedDomainLocation>
```

Request Properties

Property	Value
Name	ValidRequest
Description	
Message Size	7110
Encoding	UTF-8
Endpoint	http://localhost:80...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	

3. The response from the mock service can be viewed on the right pane:



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

line -1: null

response time: 89ms (136 bytes)

15.4.3 Endpoints:

There are two mock services:

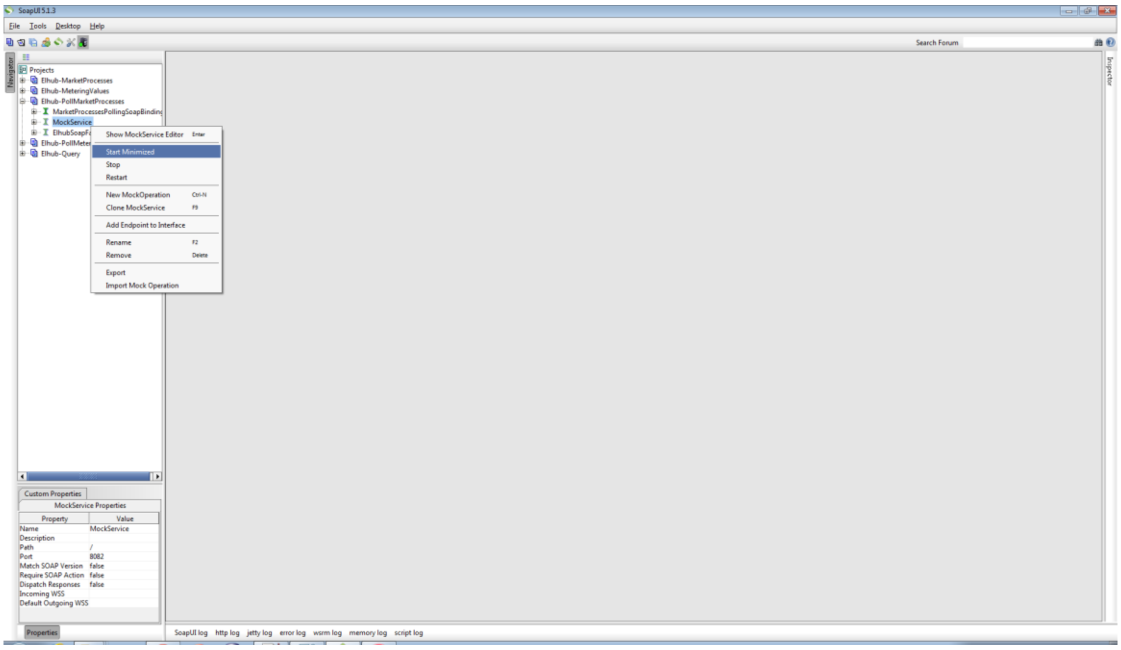
- MockService:
  - Listens on: http://localhost:8081/WebService/services/MeteringValues
  - Returns a syntactically correct response
  - Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.
- ElhubSoapFaultMockService
  - Listens on: http://localhost:8091/WebService/services/MeteringValues
  - Returns a ElhubSoapFault of type Security or System
  - Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.

15.5 Sample 3 - Elhub-PollMarketProcesses

Before you start using the requests, you need to start the two mock-services.

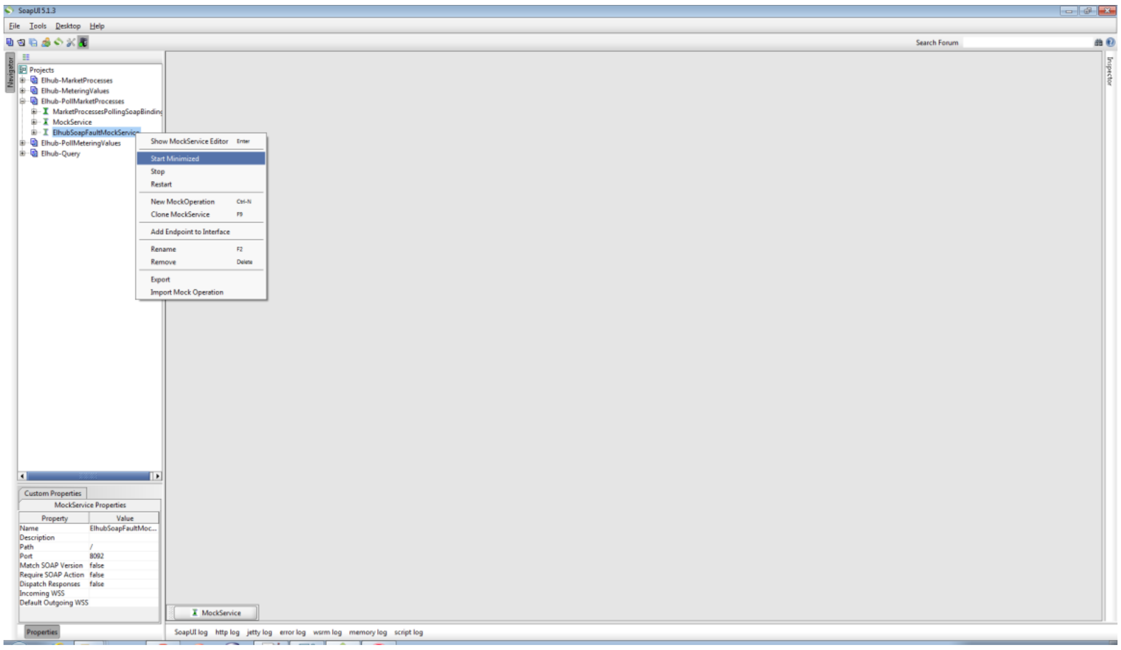
15.5.1 Start mockservices

1. Expand "Elhub-PollMarketProcess"-project and right click on "MockService" and select "Start Minimized".



Property	Value
Name	MockService
Description	
Path	/
Port	8082
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

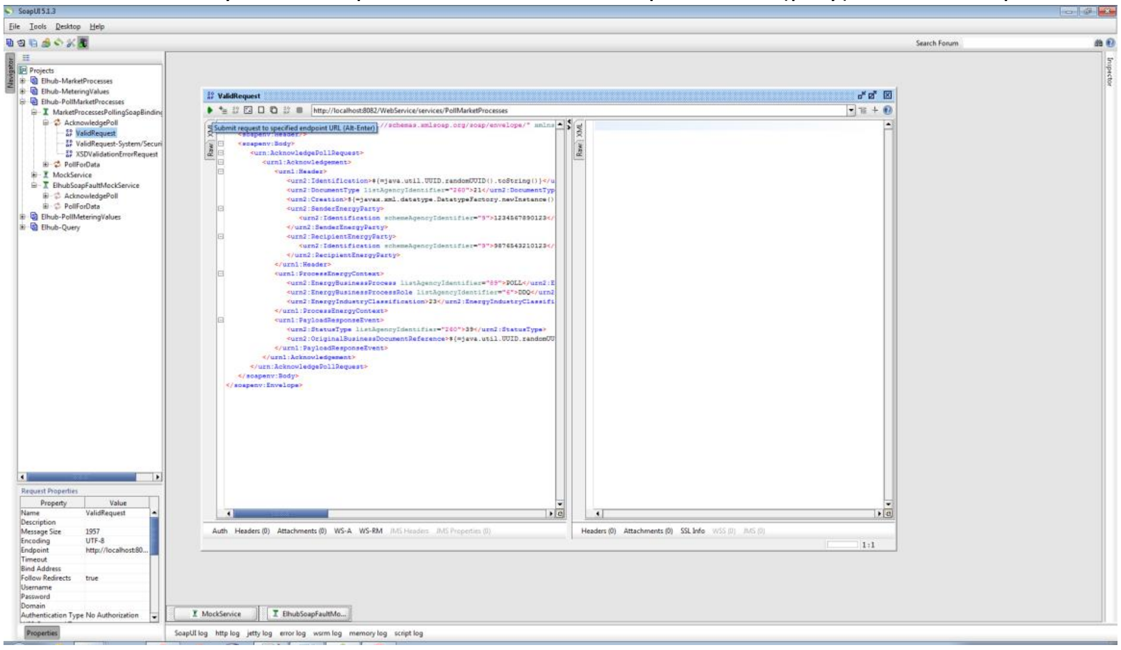
2. Right click on "ElhubSoapFaultMockService" and select "Start Minimized".



Property	Value
Name	ElhubSoapFaultMoc...
Description	
Path	/
Port	8092
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

15.5.2 Calling the mock services:

1. Expand "Elhub-PollMarketProcesses" -> "MarketProcessesPollingSoapBinding" and select any of the two operations. For each of the operations, there are defined three different types of requests:
   a. ValidRequest - This is a syntactically correct message. It will either return a some messages or an empty data set.
   b. ValidRequest - System/SecurityFault - This is a syntactically correct message, however the mock service returns a ElhubSoapFault response illustrating that there were some sort of error related to either security or Elhub. This is a fault scenario.
   c. XSDValidationErrorRequest - This is not a syntactically valid request and therefore, the response returned by the mock service is a ElhubSoapFault of type XSD.
2. Double click on any of the requests and click on the top-left icon (play) to send a request.

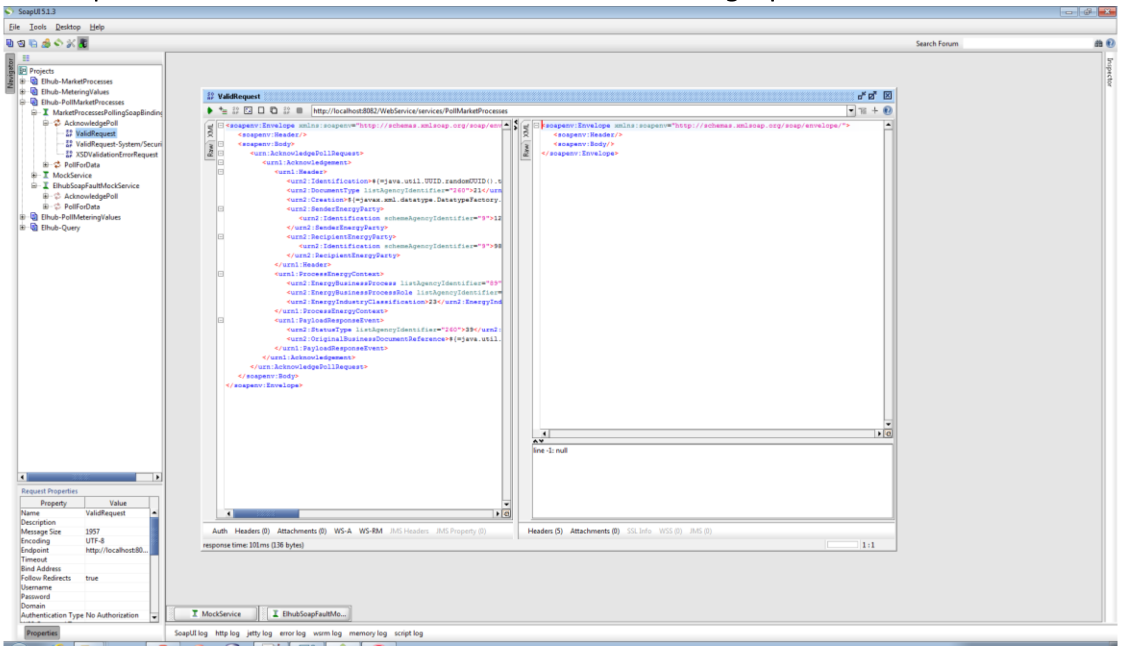

SoapUI 5.1.3
File Tools Desktop Help
Search Forum
Navigator
Projects
Elhub-MarketProcesses
Elhub-MeteringValues
Elhub-PollMarketProcesses
MarketProcessesPollingSoapBinding
AcknowledgePoll
ValidRequest
ValidRequest-System/Securi
XSDValidationErrorRequest
PollForData
MockService
ElhubSoapFaultMockService
AcknowledgePoll
PollForData
Elhub-PollMeteringValues
Elhub-Query
ValidRequest
Submit request to specified endpoint URL (Alt-Enter)
XML
Raw
Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Properties (0)
Headers (0) Attachments (0) SSL Info WSS (0) JMS (0)
1 : 1
MockService
ElhubSoapFaultMo...
Request Properties
Properties
SoapUI log http log jetty log error log wsrm log memory log script log
Inspector

```
http://localhost:8082/WebService/services/PollMarketProcesses
```

```
                    ://schemas.xmlsoap.org/soap/envelope/" xmlns
   <soapenv:Header/>
   <soapenv:Body>
      <urn:AcknowledgePollRequest>
         <urn1:Acknowledgement>
            <urn1:Header>
               <urn2:Identification>${=java.util.UUID.randomUUID().toString()}</u
               <urn2:DocumentType listAgencyIdentifier="260">21</urn2:DocumentTyp
               <urn2:Creation>${=javax.xml.datatype.DatatypeFactory.newInstance()
               <urn2:SenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">1234567890123</
               </urn2:SenderEnergyParty>
               <urn2:RecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">9876543210123</
               </urn2:RecipientEnergyParty>
            </urn1:Header>
            <urn1:ProcessEnergyContext>
               <urn2:EnergyBusinessProcess listAgencyIdentifier="89">POLL</urn2:E
               <urn2:EnergyBusinessProcessRole listAgencyIdentifier="6">DDQ</urn2
               <urn2:EnergyIndustryClassification>23</urn2:EnergyIndustryClassifi
            </urn1:ProcessEnergyContext>
            <urn1:PayloadResponseEvent>
               <urn2:StatusType listAgencyIdentifier="260">39</urn2:StatusType>
               <urn2:OriginalBusinessDocumentReference>${=java.util.UUID.randomUU
            </urn1:PayloadResponseEvent>
         </urn1:Acknowledgement>
      </urn:AcknowledgePollRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Property	Value
Name	ValidRequest
Description	
Message Size	1957
Encoding	UTF-8
Endpoint	http://localhost:80...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Type	No Authorization

3. The response from the mock service can be viewed on the right pane:



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/env
   <soapenv:Header/>
   <soapenv:Body>
      <urn:AcknowledgePollRequest>
         <urn1:Acknowledgement>
            <urn1:Header>
               <urn2:Identification>${=java.util.UUID.randomUUID().t
               <urn2:DocumentType listAgencyIdentifier="260">21</urn
               <urn2:Creation>${=javax.xml.datatype.DatatypeFactory.
               <urn2:SenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">12
               </urn2:SenderEnergyParty>
               <urn2:RecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">98
               </urn2:RecipientEnergyParty>
            </urn1:Header>
            <urn1:ProcessEnergyContext>
               <urn2:EnergyBusinessProcess listAgencyIdentifier="89"
               <urn2:EnergyBusinessProcessRole listAgencyIdentifier=
               <urn2:EnergyIndustryClassification>23</urn2:EnergyInd
            </urn1:ProcessEnergyContext>
            <urn1:PayloadResponseEvent>
               <urn2:StatusType listAgencyIdentifier="260">39</urn2:
               <urn2:OriginalBusinessDocumentReference>${=java.util.
            </urn1:PayloadResponseEvent>
         </urn1:Acknowledgement>
      </urn:AcknowledgePollRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

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

15.5.3 Endpoints:

There are two mock services:

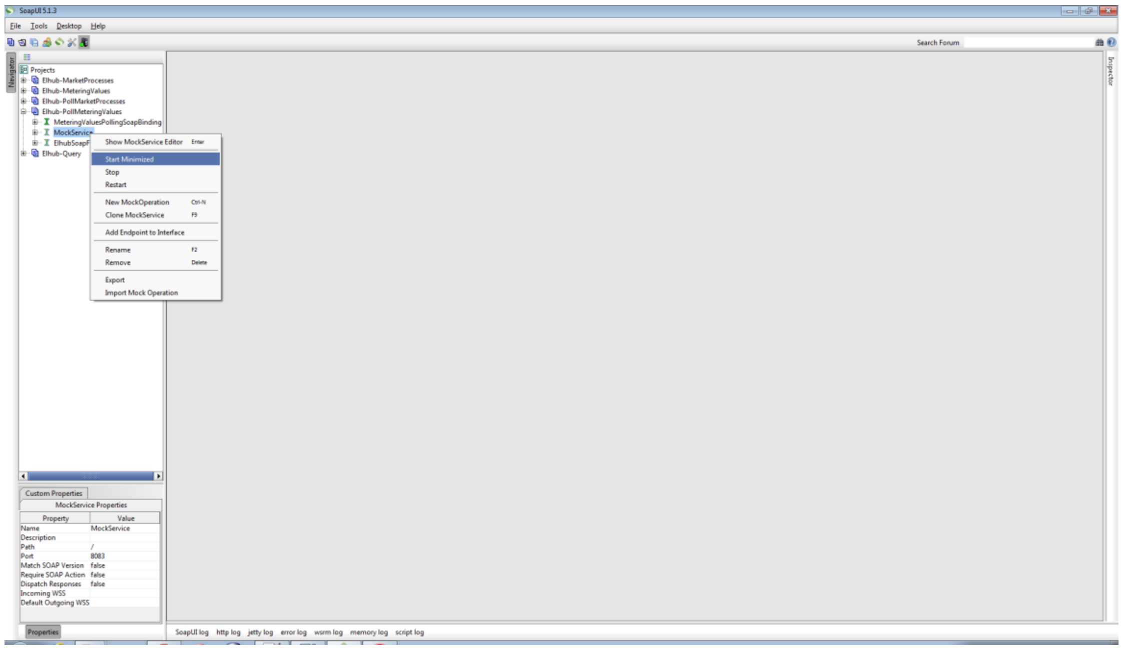
- MockService:
  - Listens on: http://localhost:8082/WebService/services/PollMarketProcesses
  - Returns a syntactically correct response
  - Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.
- ElhubSoapFaultMockService
  - Listens on: http://localhost:8092/WebService/services/PollMarketProcesses
  - Returns a ElhubSoapFault of type Security or System
  - Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.

15.6 Sample 4 - Elhub-PollMeteringValues

Before you start using the requests, you need to start the two mock-services.

15.6.1 Start mockservices

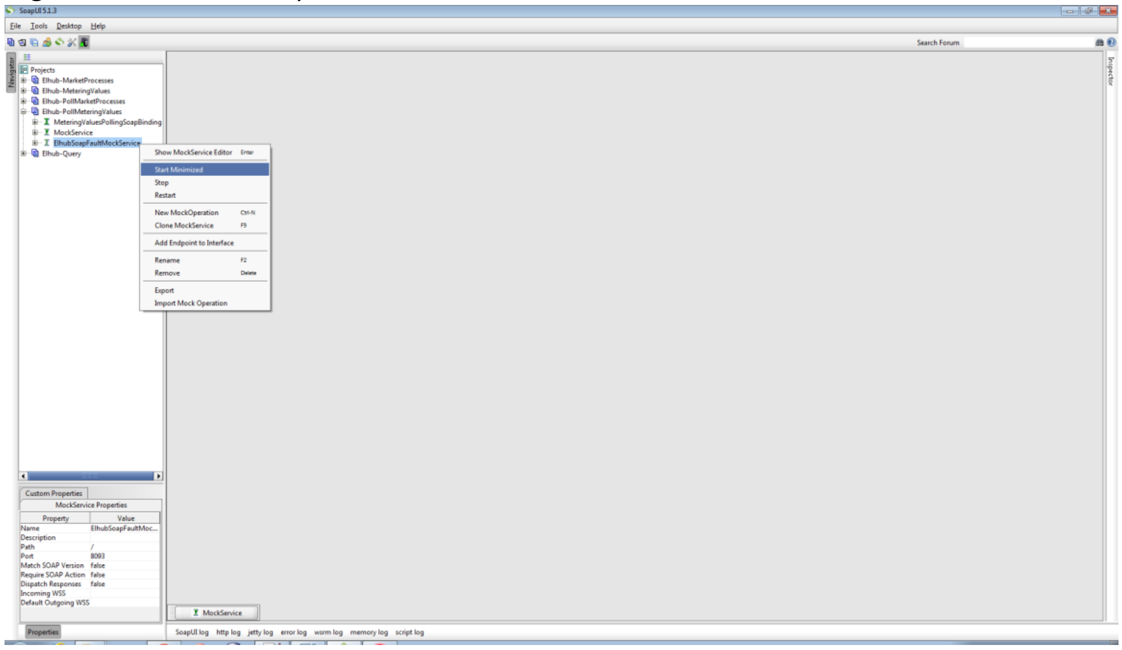
1. Expand "Elhub-PollMeteringValues"-project and right click on "MockService" and select "Start Minimized".



MockService Properties

Property	Value
Name	MockService
Description	
Path	/
Port	8083
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

2. Right click on "ElhubSoapFaultMockService" and select "Start Minimized".



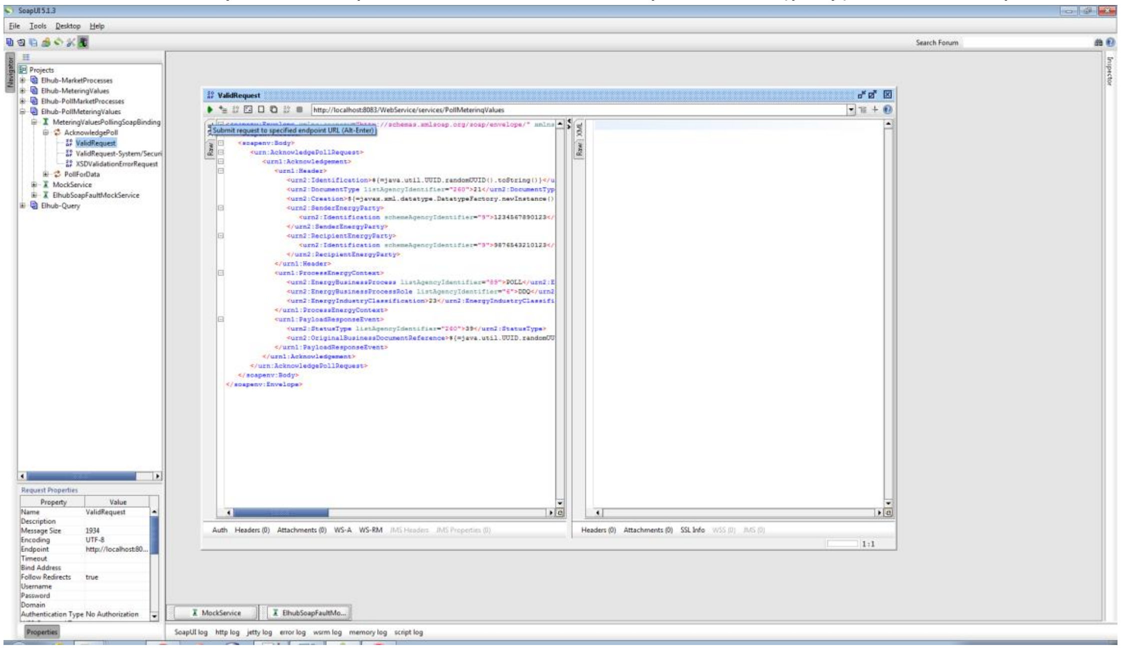
MockService Properties

Property	Value
Name	ElhubSoapFaultMoc...
Description	
Path	/
Port	8093
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

15.6.2 Calling the mock services:

1. Expand "Elhub-PollMeteringValues" -> "MeteringValuesPollingSoapBinding" and select any of the two operations. For each of the operations, there are defined three different types of requests:
    a. ValidRequest - This is a syntactically correct message. It will either return a metering value message or an empty data set.
    b. ValidRequest - System/SecurityFault - This is a syntactically correct message, however the mock service returns a ElhubSoapFault response illustrating that there were some sort of error related to either security or Elhub. This is a fault scenario.
    c. XSDValidationErrorRequest - This is not a syntactically valid request and therefore, the response returned by the mock service is a ElhubSoapFault of type XSD.

2. Double click on any of the requests and click on the top-left icon (play) to send a request.



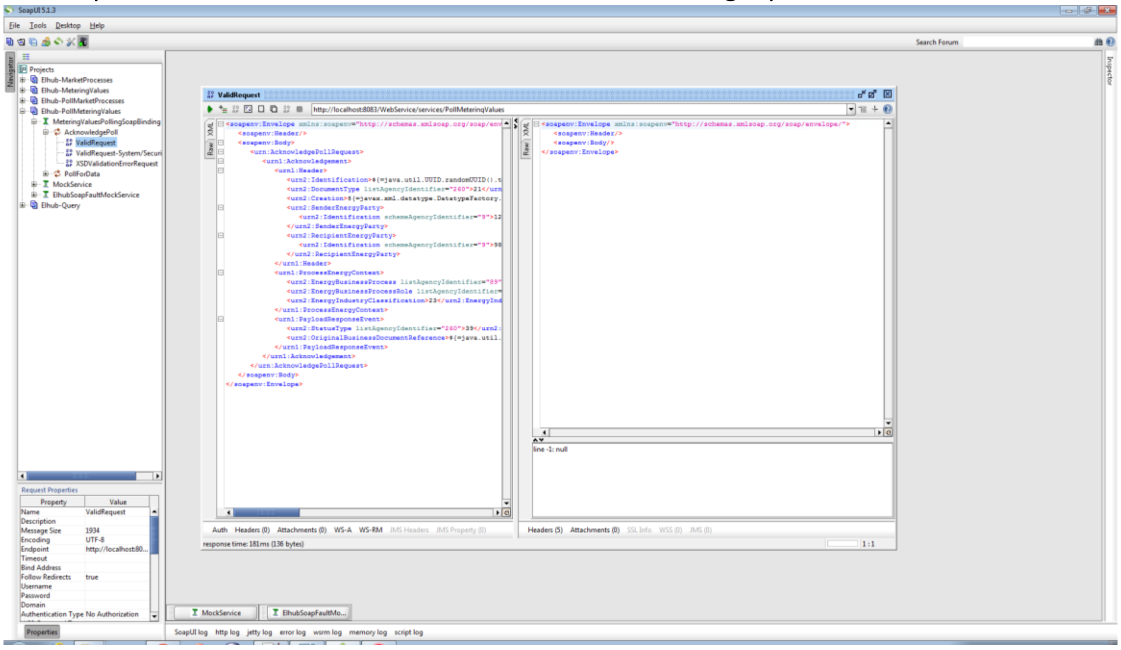
```
http://localhost:8083/WebService/services/PollMeteringValues
Submit request to specified endpoint URL (Alt-Enter)
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns
   <soapenv:Header/>
   <soapenv:Body>
      <urn:AcknowledgePollRequest>
         <urn1:Acknowledgement>
            <urn1:Header>
               <urn2:Identification>${=java.util.UUID.randomUUID().toString()}</u
               <urn2:DocumentType listAgencyIdentifier="260">21</urn2:DocumentTyp
               <urn2:Creation>${=javax.xml.datatype.DatatypeFactory.newInstance()
               <urn2:SenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">1234567890123</
               </urn2:SenderEnergyParty>
               <urn2:RecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">9876543210123</
               </urn2:RecipientEnergyParty>
            </urn1:Header>
            <urn1:ProcessEnergyContext>
               <urn2:EnergyBusinessProcess listAgencyIdentifier="89">POLL</urn2:E
               <urn2:EnergyBusinessProcessRole listAgencyIdentifier="6">DDQ</urn2
               <urn2:EnergyIndustryClassification>23</urn2:EnergyIndustryClassifi
            </urn1:ProcessEnergyContext>
            <urn1:PayloadResponseEvent>
               <urn2:StatusType listAgencyIdentifier="260">39</urn2:StatusType>
               <urn2:OriginalBusinessDocumentReference>${=java.util.UUID.randomUU
            </urn1:PayloadResponseEvent>
         </urn1:Acknowledgement>
      </urn:AcknowledgePollRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

SoapUI 5.1.3
File Tools Desktop Help
Search Forum
Projects
Elhub-MarketProcesses
Elhub-MeteringValues
Elhub-PollMarketProcesses
Elhub-PollMeteringValues
MeteringValuesPollingSoapBinding
AcknowledgePoll
ValidRequest
ValidRequest-System/Securi
XSDValidationErrorRequest
PollForData
MockService
ElhubSoapFaultMockService
Elhub-Query
ValidRequest
Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Properties (0)
Headers (0) Attachments (0) SSL Info WSS (0) JMS (0)
MockService
ElhubSoapFaultMo...
SoapUI log http log jetty log error log wsrm log memory log script log
Properties

Request Properties

Property	Value
Name	ValidRequest
Description	
Message Size	1934
Encoding	UTF-8
Endpoint	http://localhost:80...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Type	No Authorization

3. The response from the mock service can be viewed on the right pane:



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

```
line -1: null
```

```
response time: 181ms (136 bytes)
```

SoapUI 5.1.3
File Tools Desktop Help
ValidRequest
http://localhost:8083/WebService/services/PollMeteringValues
Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)
Headers (5) Attachments (0) SSL Info WSS (0) JMS (0)
MockService
ElhubSoapFaultMo...
SoapUI log http log jetty log error log wsrm log memory log script log
Properties

15.6.3 Endpoints:

There are two mock services:

- MockService:
  - Listens on: http://localhost:8083/WebService/services/PollMeteringValues
  - Returns a syntactically correct response
  - Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.
- ElhubSoapFaultMockService
  - Listens on: http://localhost:8093/WebService/services/PollMeteringValues
  - Returns a ElhubSoapFault of type Security or System

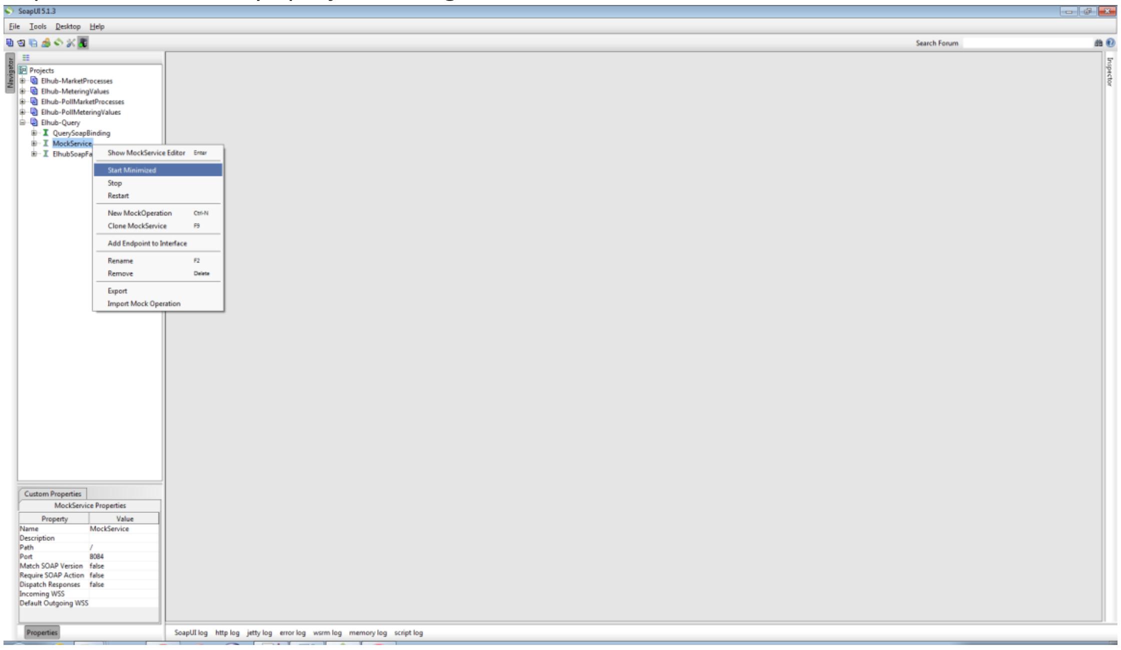
- Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.

15.7 Sample 5 - Elhub-Query

Before you start using the requests, you need to start the two mock-services.

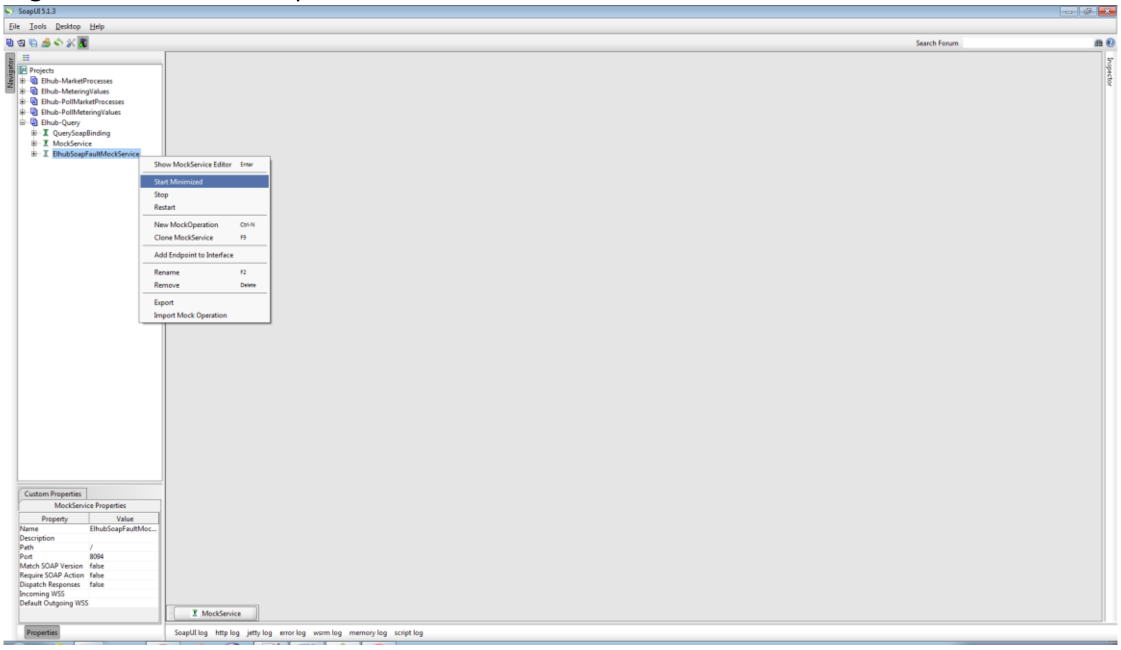
15.7.1 Start mockservices

1. Expand "Elhub-Query"-project and right click on "MockService" and select "Start Minimized".



Property	Value
Name	MockService
Description	
Path	/
Port	8084
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

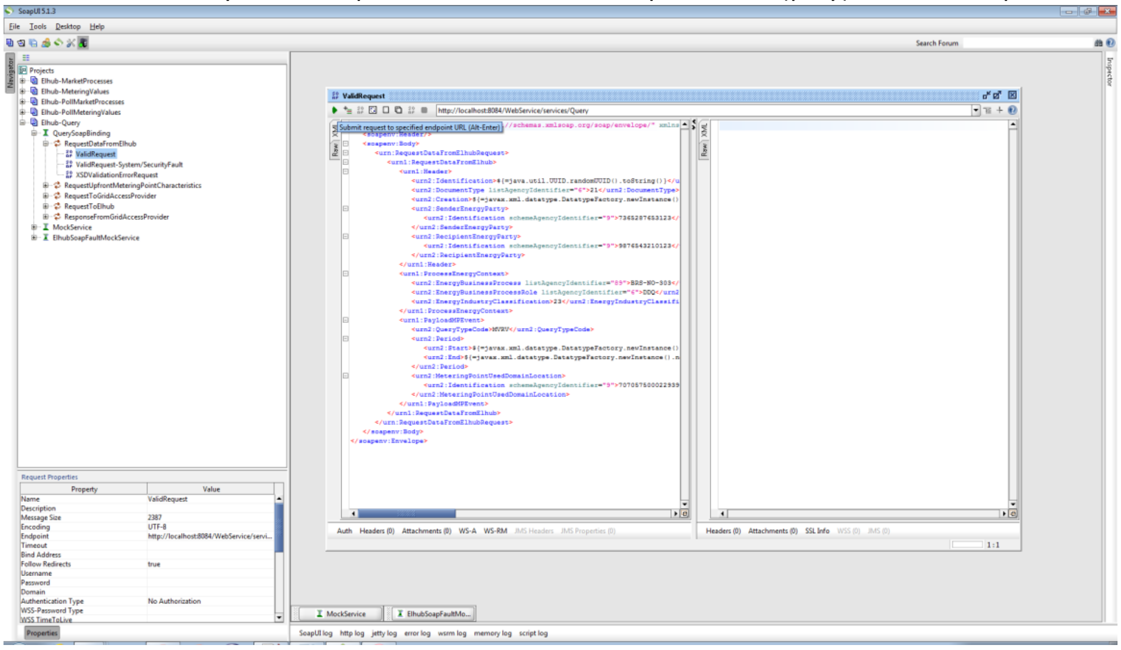
2. Right click on "ElhubSoapFaultMockService" and select "Start Minimized".



Property	Value
Name	ElhubSoapFaultMoc...
Description	
Path	/
Port	8094
Match SOAP Version	false
Require SOAP Action	false
Dispatch Responses	false
Incoming WSS	
Default Outgoing WSS	

15.7.2 Calling the mock services:

1. Expand "Elhub-Query" -> "QuerySoapBinding" and select any of the operations. For each of the operations, there are defined three different types of requests:
   a. ValidRequest - This is a syntactically correct message.
   b. ValidRequest - System/SecurityFault - This is a syntactically correct message, however the mock service returns a ElhubSoapFault response illustrating that there were some sort of error related to either security or Elhub. This is a fault scenario.
   c. XSDValidationErrorRequest - This is not a syntactically valid request and therefore, the response returned by the mock service is a ElhubSoapFault of type XSD.
2. Double click on any of the requests and click on the top-left icon (play) to send a request.


SoapUI 5.1.3
File Tools Desktop Help
Search Forum
Projects
Elhub-MarketProcesses
Elhub-MeteringValues
Elhub-PollMarketProcesses
Elhub-PollMeteringValues
Elhub-Query
QuerySoapBinding
RequestDataFromElhub
ValidRequest
ValidRequest-System/SecurityFault
XSDValidationErrorRequest
RequestUpfrontMeteringPointCharacteristics
RequestToGridAccessProvider
RequestToElhub
ResponseFromGridAccessProvider
MockService
ElhubSoapFaultMockService
ValidRequest
Submit request to specified endpoint URL (Alt-Enter)
Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Properties (0)
Headers (0) Attachments (0) SSL Info WSS (0) JMS (0)
MockService ElhubSoapFaultMo...
SoapUI log http log jetty log error log wsrm log memory log script log

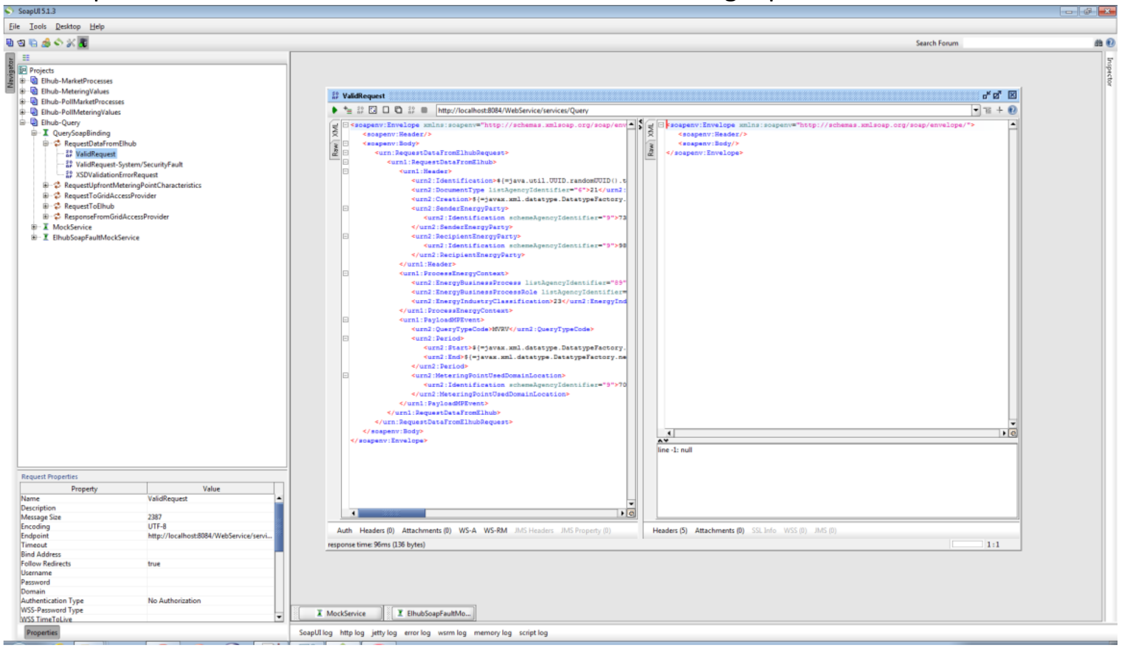
http://localhost:8084/WebService/services/Query

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns
   <soapenv:Header/>
   <soapenv:Body>
      <urn:RequestDataFromElhubRequest>
         <urn1:RequestDataFromElhub>
            <urn1:Header>
               <urn2:Identification>${=java.util.UUID.randomUUID().toString()}</u
               <urn2:DocumentType listAgencyIdentifier="6">21</urn2:DocumentType>
               <urn2:Creation>${=javax.xml.datatype.DatatypeFactory.newInstance()
               <urn2:SenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">7365287653123</
               </urn2:SenderEnergyParty>
               <urn2:RecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">9876543210123</
               </urn2:RecipientEnergyParty>
            </urn1:Header>
            <urn1:ProcessEnergyContext>
               <urn2:EnergyBusinessProcess listAgencyIdentifier="89">BRS-NO-303</
               <urn2:EnergyBusinessProcessRole listAgencyIdentifier="6">DDQ</urn2
               <urn2:EnergyIndustryClassification>23</urn2:EnergyIndustryClassifi
            </urn1:ProcessEnergyContext>
            <urn1:PayloadMPEvent>
               <urn2:QueryTypeCode>MVRV</urn2:QueryTypeCode>
               <urn2:Period>
                  <urn2:Start>${=javax.xml.datatype.DatatypeFactory.newInstance()
                  <urn2:End>${=javax.xml.datatype.DatatypeFactory.newInstance().n
               </urn2:Period>
               <urn2:MeteringPointUsedDomainLocation>
                  <urn2:Identification schemeAgencyIdentifier="9">707057500022939
               </urn2:MeteringPointUsedDomainLocation>
            </urn1:PayloadMPEvent>
         </urn1:RequestDataFromElhub>
      </urn:RequestDataFromElhubRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Request Properties

Property	Value
Name	ValidRequest
Description	
Message Size	2387
Encoding	UTF-8
Endpoint	http://localhost:8084/WebService/servi...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Type	No Authorization
WSS-Password Type	
WSS TimeToLive	

3. The response from the mock service can be viewed on the right pane:


SoapUI 5.1.3
File Tools Desktop Help
Search Forum
ValidRequest
Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)
Headers (5) Attachments (0) SSL Info WSS (0) JMS (0)
MockService ElhubSoapFaultMo...
SoapUI log http log jetty log error log wsrm log memory log script log

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

line -1: null

response time: 96ms (136 bytes)

15.7.3 Endpoints:

There are two mock services:

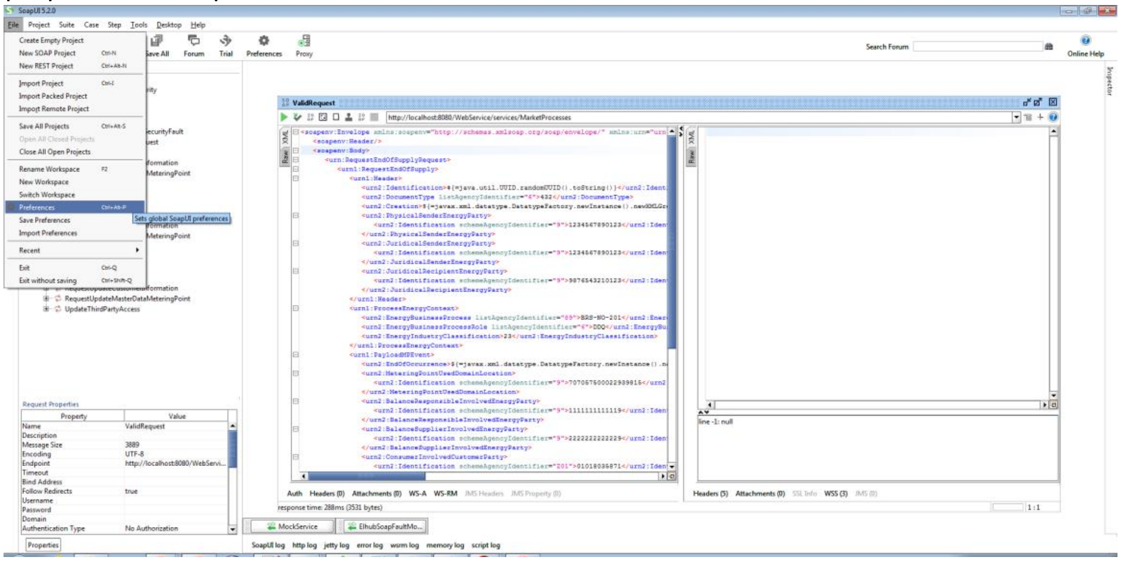
- MockService:
  - Listens on: http://localhost:8084/WebService/services/Query
  - Returns a syntactically correct response
  - Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.
- ElhubSoapFaultMockService
  - Listens on: http://localhost:8094/WebService/services/Query
  - Returns a ElhubSoapFault of type Security or System
  - Or a ElhubSoapFault of type XSD in case the request is not valid according to the XSD.

15.8 Sample - WS-Security

All of the SoapUI-projects with WS-Security are started the same way as the ones without. These instructions apply to all of the WS-Security projects.

15.8.1 Finding the WS-Security header in the request and the response

1. To be able to see the WS-Security header in the Request message you need to change some properties to SoapUI. Go to File -> Preferences



http://localhost:8080/WebService/services/MarketProcesses

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn
   <soapenv:Header/>
   <soapenv:Body>
      <urn:RequestEndOfSupplyRequest>
         <urn1:RequestEndOfSupply>
            <urn1:Header>
               <urn2:Identification>${=java.util.UUID.randomUUID().toString()}</urn2:Ident
               <urn2:DocumentType listAgencyIdentifier="6">432</urn2:DocumentType>
               <urn2:Creation>${=javax.xml.datatype.DatatypeFactory.newInstance().newXMLGr
               <urn2:PhysicalSenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">1234567890123</urn2:Iden
               </urn2:PhysicalSenderEnergyParty>
               <urn2:JuridicalSenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">1234567890123</urn2:Iden
               </urn2:JuridicalSenderEnergyParty>
               <urn2:JuridicalRecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">9876543210123</urn2:Iden
               </urn2:JuridicalRecipientEnergyParty>
            </urn1:Header>
            <urn1:ProcessEnergyContext>
               <urn2:EnergyBusinessProcess listAgencyIdentifier="89">BRS-NO-201</urn2:Ener
               <urn2:EnergyBusinessProcessRole listAgencyIdentifier="6">DDQ</urn2:EnergyBu
               <urn2:EnergyIndustryClassification>23</urn2:EnergyIndustryClassification>
            </urn1:ProcessEnergyContext>
            <urn1:PayloadMPEvent>
               <urn2:EndOfOccurrence>${=javax.xml.datatype.DatatypeFactory.newInstance().n
               <urn2:MeteringPointUsedDomainLocation>
                  <urn2:Identification schemeAgencyIdentifier="9">707057500022939815</urn2
               </urn2:MeteringPointUsedDomainLocation>
               <urn2:BalanceResponsibleInvolvedEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">1111111111119</urn2:Iden
               </urn2:BalanceResponsibleInvolvedEnergyParty>
               <urn2:BalanceSupplierInvolvedEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">2222222222229</urn2:Iden
               </urn2:BalanceSupplierInvolvedEnergyParty>
               <urn2:ConsumerInvolvedCustomerParty>
                  <urn2:Identification schemeAgencyIdentifier="201">01018036871</urn2:Iden
```

line -1: null

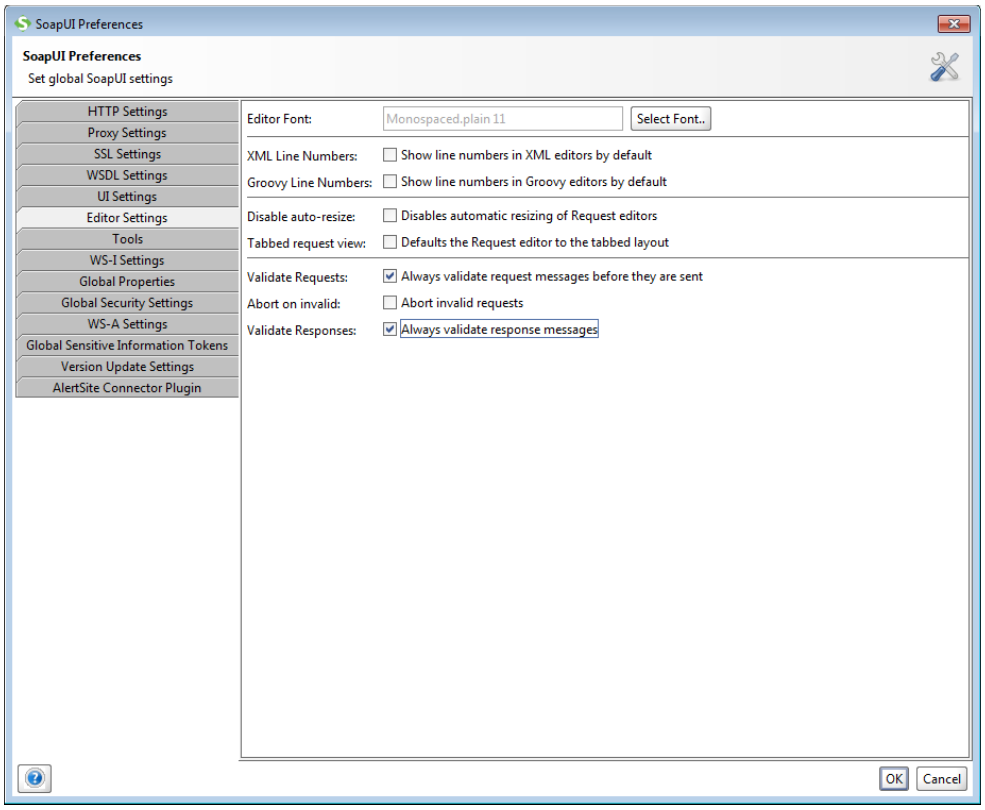
response time: 288ms (3531 bytes)

1 : 1

Request Properties

Property	Value
Name	ValidRequest
Description	
Message Size	3889
Encoding	UTF-8
Endpoint	http://localhost:8080/WebServi...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Type	No Authorization

2. Go to Editor Settings and make sure "Always validate request messages before they are sent" and "Always validate response messages" are selected. Press OK



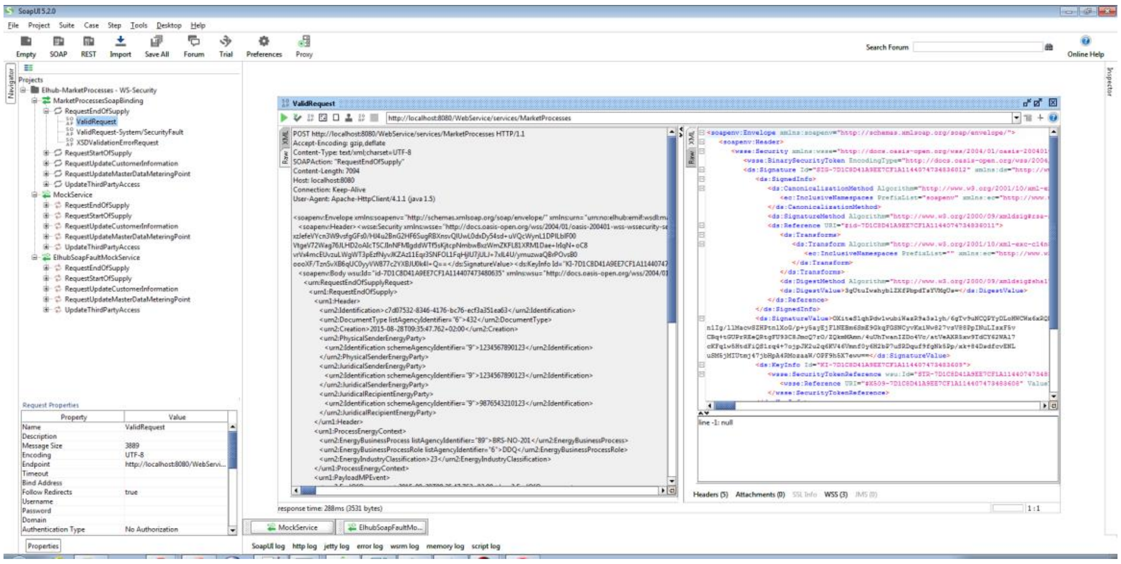
SoapUI Preferences

SoapUI Preferences

Set global SoapUI settings

Editor Font:	Monospaced.plain 11
XML Line Numbers:	☐ Show line numbers in XML editors by default
Groovy Line Numbers:	☐ Show line numbers in Groovy editors by default
Disable auto-resize:	☐ Disables automatic resizing of Request editors
Tabbed request view:	☐ Defaults the Request editor to the tabbed layout
Validate Requests:	☑ Always validate request messages before they are sent
Abort on invalid:	☐ Abort invalid requests
Validate Responses:	☑ Always validate response messages

3. Start the both mock services, and send a request. Press the "Raw" section on the re quest side, and you'll now be able to see the WS-Security header for the request message. For the response you'll be able to see the WS-Security header for in both the "XML" and "Raw" section.



ValidRequest — http://localhost:8080/WebService/services/MarketProcesses

```
POST http://localhost:8080/WebService/services/MarketProcesses HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "RequestEndOfSupply"
Content-Length: 7094
Host: localhost:8080
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:no:elhub:emif:wsdl:m
  <soapenv:Header><wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-se
xzJefeVYcn3W9vsfgGFs0/HJ4u2BnG2HF6SugRBXnsvQIUwL0dxDy54sd+uVQcWynL1DPILbIF00
VtgeV72Wag76JLHD2oAIcTSCJInNFMIgddWTf5sKjtcpNmbwBxzWmZKFL81XRM1Dae+IrIqN+oC8
vrVx4mcEUvzuLWgWT3pEzfNyvJKZAz11Eqr3SNFOL1FqHjJU7jULJ+7xIL4U/ymuzwaQBrPOvsB0
oooXF/Tzn5vXB6qUC0yyVW877c2YXBJU0k4I+Q==</ds:SignatureValue><ds:KeyInfo Id="KI-7D1C8D41A9EE7CF1A1144074
  <soapenv:Body wsu:Id="id-7D1C8D41A9EE7CF1A114407473480635" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01
    <urn:RequestEndOfSupplyRequest>
      <urn1:RequestEndOfSupply>
        <urn1:Header>
          <urn2:Identification>c7d07532-8346-4176-bc76-ecf3a351ea63</urn2:Identification>
          <urn2:DocumentType listAgencyIdentifier="6">432</urn2:DocumentType>
          <urn2:Creation>2015-08-28T09:35:47.762+02:00</urn2:Creation>
          <urn2:PhysicalSenderEnergyParty>
            <urn2:Identification schemeAgencyIdentifier="9">1234567890123</urn2:Identification>
          </urn2:PhysicalSenderEnergyParty>
          <urn2:JuridicalSenderEnergyParty>
            <urn2:Identification schemeAgencyIdentifier="9">1234567890123</urn2:Identification>
          </urn2:JuridicalSenderEnergyParty>
          <urn2:JuridicalRecipientEnergyParty>
            <urn2:Identification schemeAgencyIdentifier="9">9876543210123</urn2:Identification>
          </urn2:JuridicalRecipientEnergyParty>
        </urn1:Header>
        <urn1:ProcessEnergyContext>
          <urn2:EnergyBusinessProcess listAgencyIdentifier="89">BRS-NO-201</urn2:EnergyBusinessProcess>
          <urn2:EnergyBusinessProcessRole listAgencyIdentifier="6">DDQ</urn2:EnergyBusinessProcessRole>
          <urn2:EnergyIndustryClassification>23</urn2:EnergyIndustryClassification>
        </urn1:ProcessEnergyContext>
        <urn1:PayloadMPEvent>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401
         <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004
         <ds:Signature Id="SIG-7D1C8D41A9EE7CF1A114407473483612" xmlns:ds="http://w
            <ds:SignedInfo>
               <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-e
                  <ec:InclusiveNamespaces PrefixList="soapenv" xmlns:ec="http://www.
               </ds:CanonicalizationMethod>
               <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
               <ds:Reference URI="#id-7D1C8D41A9EE7CF1A114407473483611">
                  <ds:Transforms>
                     <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n
                        <ec:InclusiveNamespaces PrefixList="" xmlns:ec="http://www.w
                     </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1
                  <ds:DigestValue>3gUtuIwahyb1ZKfPbpdTsYVMgOs=</ds:DigestValue>
               </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>OXiteSlqhPdw1wubiWazR9a3slyh/6gTv9uNCQPYyDLcHNCHx6xRQ
n1Ig/1lMacwSZHPtnlXoG/p+y6ayEjFlNEBm6SmE9GkqFGSNCyvKmiWw827vsV88PpINuLIsxF6v
CBq+tGUPrRKeQRtgFU93C8JmcQ7rO/ZQkmMAmn/4uUhTwanIZDo4Vc/atVeAXRSxw9TdCY62NA17
cKFq1w5HsdFiQ81rq4+7ojpJK2u2q6KV46Vmnf0y6H2bP7uSRDquf9fgNk5Pp/xk+84DsdfcvENL
uSM6jMIUtmj47jbHpA4RMozaaW/OPF9h5K7ewu==</ds:SignatureValue>
            <ds:KeyInfo Id="KI-7D1C8D41A9EE7CF1A114407473483608">
               <wsse:SecurityTokenReference wsu:Id="STR-7D1C8D41A9EE7CF1A11440747348
                  <wsse:Reference URI="#X509-7D1C8D41A9EE7CF1A114407473483606" Value
               </wsse:SecurityTokenReference>
```

line -1: null

response time: 288ms (3531 bytes)

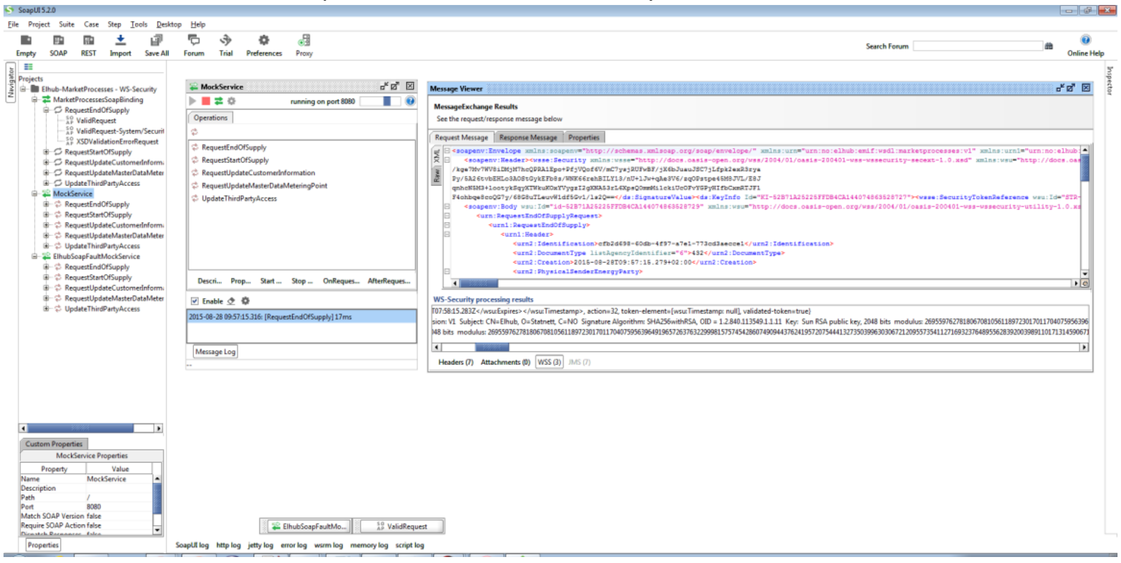
SoapUI 5.2.0
File Project Suite Case Step Tools Desktop Help
Empty SOAP REST Import Save All Forum Trial Preferences Proxy
Search Forum
Online Help
Navigator
Projects
Elhub-MarketProcesses - WS-Security
MarketProcessesSoapBinding
RequestEndOfSupply
ValidRequest
MockService
ElhubSoapFaultMockService
Request Properties
Property
Value
XML
Raw
Headers (5) Attachments (0) SSL Info WSS (3) JMS (0)
MockService
ElhubSoapFaultMo...
SoapUI log http log jetty log error log wsrm log memory log script log

15.8.2 Checking the WS-Security integrity response

15.8.2.1 Valid request and response

1. To see the WS-Security checks for the request message going into the mock open the mock service, and double click on the sample message that was sent in earlier from the mock

service window. There you'll see request that was received into the mock service. Press the WSS section below the request to see the WS-Security checks done.

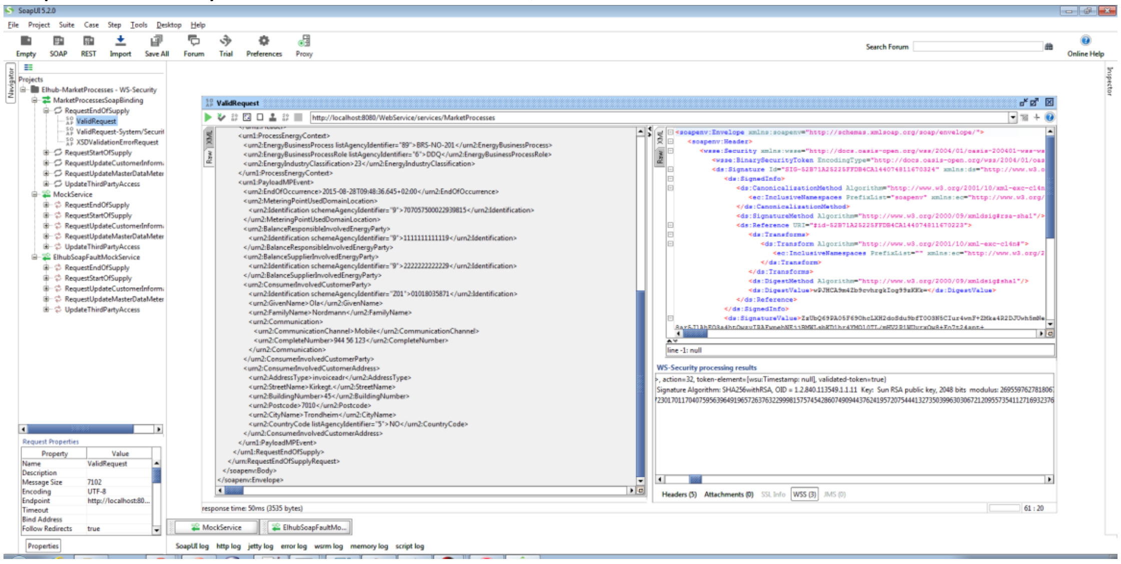


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:no:elhub:emif:wsdl:marketprocesses:v1" xmlns:urn1="urn:no:elhub:
   <soapenv:Header><wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oas
/kge7Mv7WV8iDMjM7hcQPRA1Epo+PfjVQof6V/mC7ysjRUFwBF/jX6bJuauJSC7jLfpk2exR3rya
Py/5A26tvbEHLo3A08tGykEFb8s/WNK66rehBILY13/nU+1Jw+qAe3V6/zqOPstpe45M9JVL/E8J
qnhcNSM3+1ootykSqyXTWkuKOxYVygzI2gXNA&3rL6XpeQ0mmMi1ckiUcOFvYGPyHIfbCxmRTJF1
F4ohbqe8ccQG7y/68G8uTLeuvW1df5Gv1/1s2Q==</ds:SignatureValue><ds:KeyInfo Id="KI-52B71A25225FFDB4CA14407486352872 7"><wsse:SecurityTokenReference wsu:Id="STR-
   <soapenv:Body wsu:Id="id-52B71A25225FFDB4CA144074863528729" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xs
      <urn:RequestEndOfSupplyRequest>
         <urn1:RequestEndOfSupply>
            <urn1:Header>
               <urn2:Identification>cfb2d698-60db-4f97-a7e1-773cd3aecce1</urn2:Identification>
               <urn2:DocumentType listAgencyIdentifier="6">432</urn2:DocumentType>
               <urn2:Creation>2015-08-28T09:57:15.279+02:00</urn2:Creation>
               <urn2:PhysicalSenderEnergyParty>
```

WS-Security processing results

```
T07:58:15.283Z</wsu:Expires></wsu:Timestamp>, action=32, token-element=[wsu:Timestamp: null], validated-token=true}
sion: V1  Subject: CN=Elhub, O=Statnett, C=NO  Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11  Key:  Sun RSA public key, 2048 bits  modulus: 2695597627818067081056118972301701170407595639
48 bits  modulus: 2695597627818067081056118972301701170407595639649196572637632299981575745428607490944376241957207544413273503996303067212095573541127169323764895562839200398911017131459067
```

2. To see the WS-Security checks for the response message press the WSS section below the response. There you'll see there is no errors, and that the tokens are all valid.



```
http://localhost:8080/WebService/services/MarketProcesses

      <urn1:ProcessEnergyContext>
        <urn2:EnergyBusinessProcess listAgencyIdentifier="89">BRS-NO-201</urn2:EnergyBusinessProcess>
        <urn2:EnergyBusinessProcessRole listAgencyIdentifier="6">DDQ</urn2:EnergyBusinessProcessRole>
        <urn2:EnergyIndustryClassification>23</urn2:EnergyIndustryClassification>
      </urn1:ProcessEnergyContext>
      <urn1:PayloadMPEvent>
        <urn2:EndOfOccurrence>2015-08-28T09:48:36.645+02:00</urn2:EndOfOccurrence>
        <urn2:MeteringPointUsedDomainLocation>
          <urn2:Identification schemeAgencyIdentifier="9">707057500022939815</urn2:Identification>
        </urn2:MeteringPointUsedDomainLocation>
        <urn2:BalanceResponsibleInvolvedEnergyParty>
          <urn2:Identification schemeAgencyIdentifier="9">1111111111119</urn2:Identification>
        </urn2:BalanceResponsibleInvolvedEnergyParty>
        <urn2:BalanceSupplierInvolvedEnergyParty>
          <urn2:Identification schemeAgencyIdentifier="9">2222222222229</urn2:Identification>
        </urn2:BalanceSupplierInvolvedEnergyParty>
        <urn2:ConsumerInvolvedCustomerParty>
          <urn2:Identification schemeAgencyIdentifier="Z01">01018035871</urn2:Identification>
          <urn2:GivenName>Ola</urn2:GivenName>
          <urn2:FamilyName>Nordmann</urn2:FamilyName>
          <urn2:Communication>
            <urn2:CommunicationChannel>Mobile</urn2:CommunicationChannel>
            <urn2:CompleteNumber>944 56 123</urn2:CompleteNumber>
          </urn2:Communication>
        </urn2:ConsumerInvolvedCustomerParty>
        <urn2:ConsumerInvolvedCustomerAddress>
          <urn2:AddressType>invoiceadr</urn2:AddressType>
          <urn2:StreetName>Kirkegt.</urn2:StreetName>
          <urn2:BuildingNumber>45</urn2:BuildingNumber>
          <urn2:Postcode>7010</urn2:Postcode>
          <urn2:CityName>Trondheim</urn2:CityName>
          <urn2:CountryCode listAgencyIdentifier="5">NO</urn2:CountryCode>
        </urn2:ConsumerInvolvedCustomerAddress>
      </urn1:PayloadMPEvent>
    </urn1:RequestEndOfSupply>
  </urn:RequestEndOfSupplyRequest>
 </soapenv:Body>
</soapenv:Envelope>

response time: 50ms (3535 bytes)
```

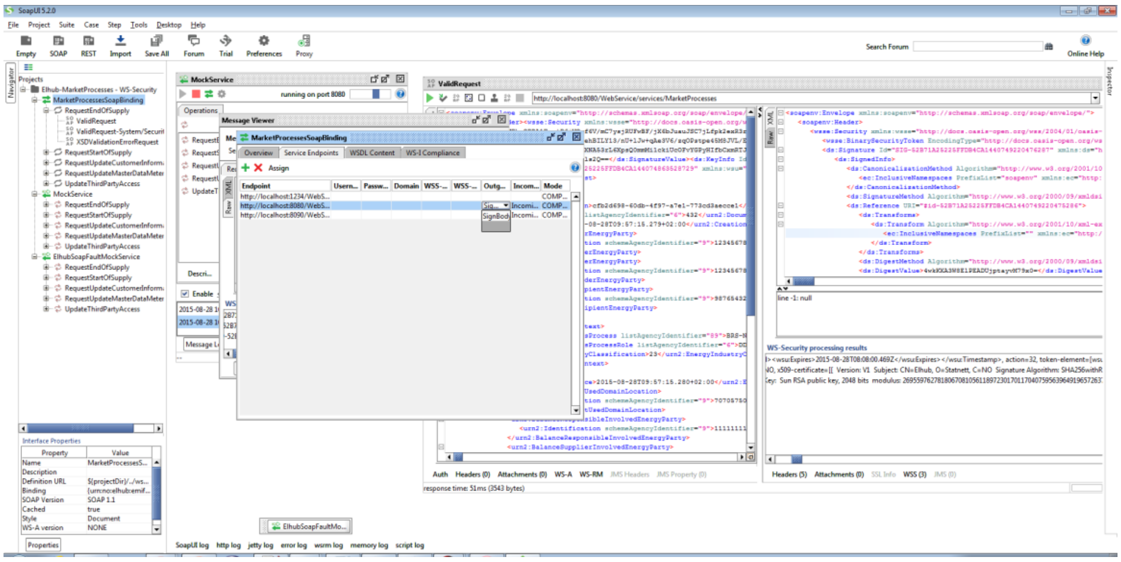
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-ws
         <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oas
         <ds:Signature Id="SIG-52B71A25225FFDB4CA144074811670324" xmlns:ds="http://www.w3.o
            <ds:SignedInfo>
               <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n
                  <ec:InclusiveNamespaces PrefixList="soapenv" xmlns:ec="http://www.w3.org/
               </ds:CanonicalizationMethod>
               <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
               <ds:Reference URI="#id-52B71A25225FFDB4CA144074811670223">
                  <ds:Transforms>
                     <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces PrefixList="" xmlns:ec="http://www.w3.org/2
                     </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>wPJHCA9m42b9evhrgkIog99zKKk=</ds:DigestValue>
               </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>ZzUbQ69PA05F69OhcLXH2do8du9bfTOO3N6CIur4wnF+ZMka4R2DJUwh5mNe

line -1: null
```

WS-Security processing results

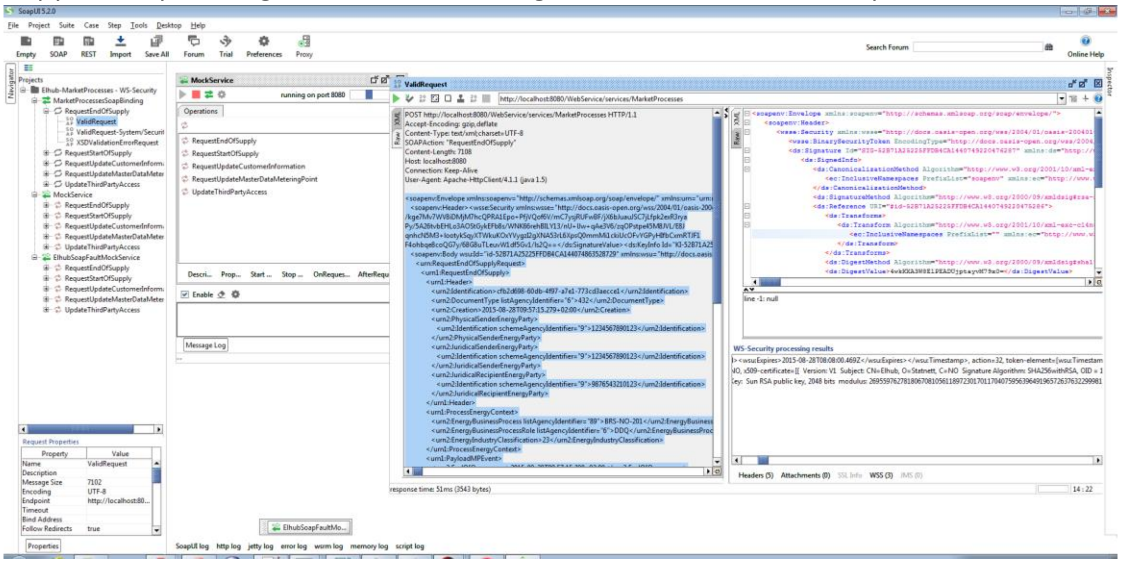
```
>, action=32, token-element=[wsu:Timestamp: null], validated-token=true}
Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11  Key:  Sun RSA public key, 2048 bits  modulus: 269559762781806
7230170117040759563964919657263763229998157574542860749094437624195720754441327350399630306721209557354112716932376
```

15.8.2.2 Invalid request

1. If you want to send an expired message you first need to disable WS-Security for the request message. Double click on the "MarketProcessesSoapBinding" and "Services Endpoints" and deselect "SignBody" for the "Outgoing WSS".

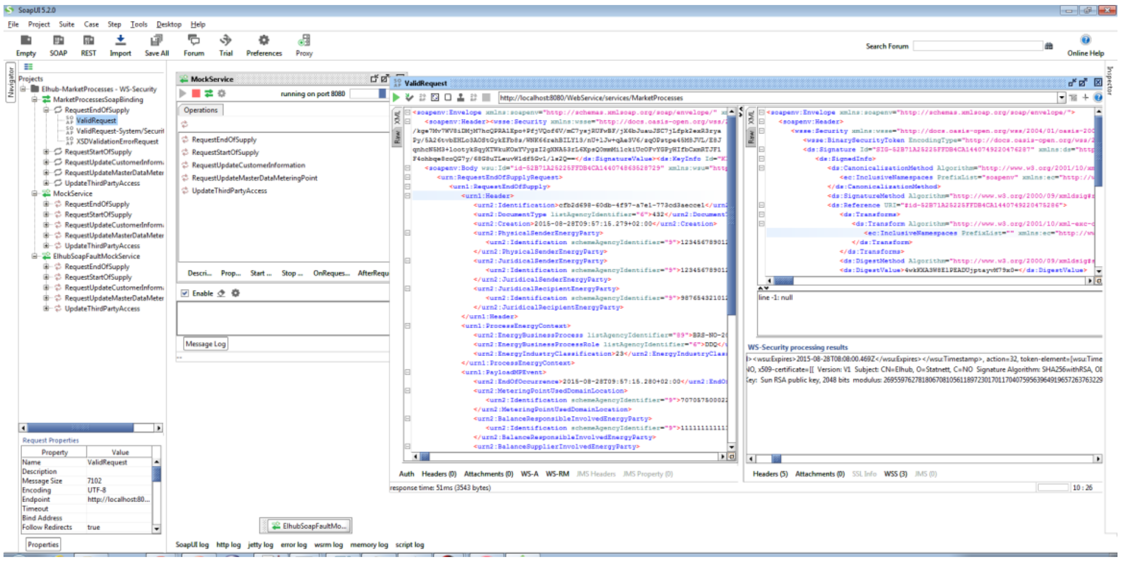

SoapUI 5.2.0
File Project Suite Case Step Tools Desktop Help
Empty SOAP REST Import Save All Forum Trial Preferences Proxy
Search Forum
Online Help
Navigator
Projects
Elhub-MarketProcesses - WS-Security
MarketProcessesSoapBinding
MockService
running on port 8080
Message Viewer
MarketProcessesSoapBinding
Overview Service Endpoints WSDL Content WS-I Compliance
Assign
Endpoint Usern... Passw... Domain WSS-... WSS-... Outg... Incom... Mode
ValidRequest
http://localhost:8080/WebService/services/MarketProcesses
WS-Security processing results
Interface Properties
Property Value
response time: 51ms (3543 bytes)
SoapUI log http log jetty log error log wsrm log memory log script log

2. Copy the soap message from the raw message from an earlier sent in request

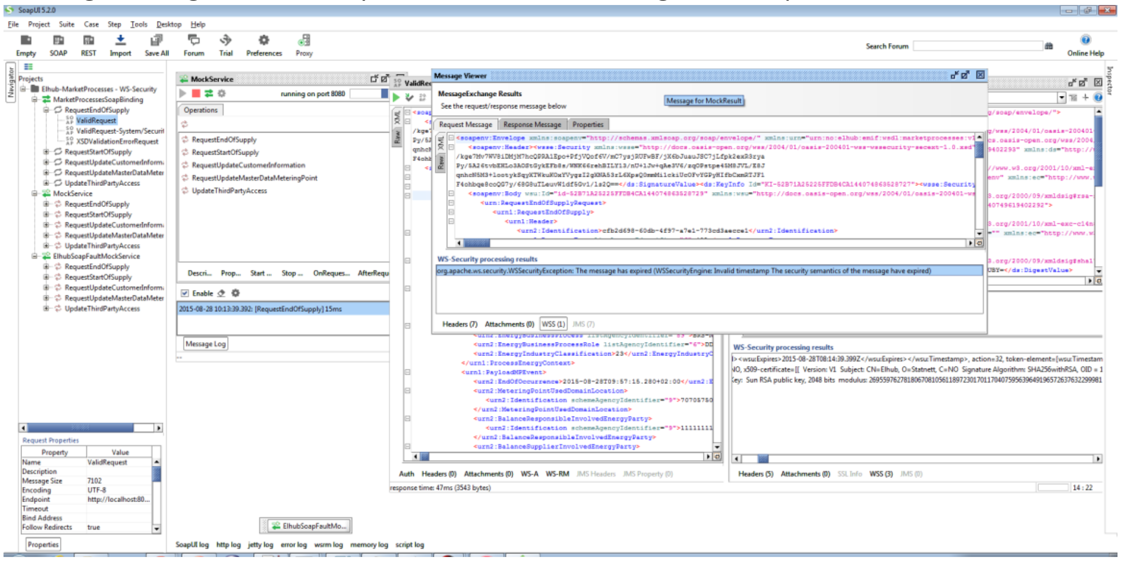

SoapUI 5.2.0
File Project Suite Case Step Tools Desktop Help
Empty SOAP REST Import Save All Forum Trial Preferences Proxy
Projects
Elhub-MarketProcesses - WS-Security
MarketProcessesSoapBinding
RequestEndOfSupply
ValidRequest
ValidRequest-System/Security
XSDValidationErrorRequest
RequestStartOfSupply
RequestUpdateCustomerInformation
RequestUpdateMasterDataMeteringPoint
UpdateThirdPartyAccess
MockService
ElhubSoapFaultMockService
ValidRequest
Request Properties
Property Value
Name ValidRequest
Message Size 7102
Encoding UTF-8
Endpoint http://localhost:80...
Follow Redirects true
WS-Security processing results
response time: 51ms (3543 bytes)

```
POST http://localhost:8080/WebService/services/MarketProcesses HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "RequestEndOfSupply"
Content-Length: 7108
Host: localhost:8080
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

3. Paste it in into the "XML" section


SoapUI 5.2.0
File Project Suite Case Step Tools Desktop Help
Empty SOAP REST Import Save All Forum Trial Preferences Proxy
MockService
running on port 8080
ValidRequest
http://localhost:8080/WebService/services/MarketProcesses
XML
Raw
WS-Security processing results
Request Properties
Name ValidRequest
Message Size 7102
Encoding UTF-8
response time: 51ms (3543 bytes)
SoapUI log http log jetty log error log wsrm log memory log script log

4. Call the mock service with this new request. Open up the mock service, and double click on the message and select the "WSS" section below the request message. There you'll see a message stating: "The security semantics of the message have expired"



WS-Security processing results

```
org.apache.ws.security.WSSecurityException: The message has expired (WSSecurityEngine: Invalid timestamp The security semantics of the message have expired)
```

15.9 Connecting to the SoapUI Package using .Net

This gives some brief information of how to connect to the WSS enabled SoapUI services. This is not a comprehensive tutorial, neither a recommendation for how to connect to Elhub. It is only intended to aid in connecting to the WSS enabled SoapUI package using .Net. This uses dummy certificates and uses more lax security than to use when communicating with Elhub.

Generate a dummy certificate for client use, import it (pfx file) into Windows certificate manager and set it as ClientCertificate on ClientCredentials:

- openssl genrsa -out privatekey.pem 1024
- openssl req -new -x509 -key privatekey.pem -out publickey.cer -days 365
- openssl pkcs12 -export -out public_privatekey.pfx -inkey privatekey.pem -in publickey.cer

Extract server certificate from the SoapUI keystore, import it to Windows certificate manager and set it as ServiceCertificate in ClientCredentials with CertificateValidationMode = X509CertificateValidationMode.None:

- keytool -export -keystore <ref to soapui folder in provided package>\keystore.jks -file <outfile>.cer -alias "elhub test"
- Password: test
- keytool is part of the Java SDK

When creating the channel, set "elhub" as the EndpointIdentity (EndpointIdentity.CreateDnsIdentity("elhub")), or define "elhub" in your hosts file and refer to the computer running the SoapUI package and use "elhub" as dns name when communicating with the SoapUI package.

Add allowSerializedSigningTokenOnReply="true" and securityHeaderLayout="LaxTimestampLast" to the bindings.

16 Questions

Questions related to the interface can be directed to post@elhub.no.