ASE-9476 FIS: SOAP and CAMX (IPC-2501)

Outline

- SOAP
  - ✓ SOAP building blocks
  - ✓ SOAP with Attachments
- CAMX
  - ✓ IPC-2501
    - Mechanism for exchanging CAMX messages

FIS and CAMX

- FIS have difficulty in information exchange due to variety of vendors producing various devices
  - ✓ Need of intercommunication between these devices
  - ✓ Delays and difficulties on line changes and modifications
  - ✓ Information systems must be modified accordingly each time
- Response: family of international standards for information interoperability known as CAMX (Computer Aided Manufacturing using XML)
  - ✓ Lower programming costs
  - ✓ Faster production
  - ✓ Greater flexibility

IPC/CAMX Standards

IPC Number/ Function	-xxx1 Generic	-xxx2 Administ	-xxx3 Document	-xxx4 Board Fab	-xxx5 Bare Bd Test	-xxx6 Assembly Manufact	-xxx7 Assy Test/Insp.	-xxx8 Comp. & Mat'ls
IPC-2500 CAMX Framework	IPC-2501		IPC-2503					
IPC-2510 GenCAM Product Data	IPC-2511	IPC-2512A	IPC-2513A	IPC-2514A	IPC-2515A	IPC-2516A	IPC-2517A	IPC-2518A
IPC-2520 Quality Product Data				IPC-2524				
IPC-2530 SRFF Process Data	IPC-2531							
IPC-2540 Shop Floor Communication (CAMX)	IPC-2541					IPC-2546(incl. Am1&2)	IPC-2547	
IPC-2550 Execution Communication (MES)	IPC-2551 (Working Draft)			IPC-2554 (Working Draft)		IPC-2556 (PINS)		
IPC-2560 Enterprise Communication								
IPC-2570 Supply Chain Communication (PDX)	IPC-2571					IPC-2576	IPC-2577	IPC-2578
IPC-2580 Application Specific Data	IPC-2581	IPC-2582	IPC-2583	IPC-2584		IPC-2586		IPC-2588

CAMX short introduction

- CAMX is the framework facilitating interconnection of machines by exchanging standardised XML messages
- The standards are brought by IPC association
- CAMX messages are based on SOAP with Attachments specification

SOAP

- HTTP introduction
- SOAP HTTP binding
- SOAP building blocks
- SOAP with Attachments

SOAP Introduction

- SOAP stands for Simple Object Access Protocol and is used to exchange XML data over the Internet
- It is a transport-independent protocol that uses XML to invoke remote methods, however usually it is used in combination with HTTP
- SOAP is a format for sending messages / communication protocol
  - ✓ between applications
  - ✓ via Internet
- SOAP is based on XML, thus it is:
  - ✓ platform independent
  - ✓ language independent
  - ✓ simple and extensible
- SOAP is a W3C recommendation (24.06.2003)

HTTP

- HTTP communicates over TCP/IP
- An HTTP client connects to an HTTP server using TCP.
- After establishing a connection, the client can send an HTTP request message to the server:

```
POST /item HTTP/1.1
Host: 189.123.345.239
Content-Type: text/plain
Content-Length: 200
```

- The server then processes the request and sends an HTTP response back to the client
- The response contains a status code that indicates the status of the request:

200 OK
Content-Type: text/plain
Content-Length: 200

Standard succes code

400 Bad Request
Content-Length: 0

Request unsuccesfully decoded

SOAP HTTP binding

- SOAP HTTP binding describes the relationship between parts of the SOAP request message and various HTTP headers
- A SOAP request: a HTTP POST or an HTTP GET request.
- The HTTP POST request specifies at least two HTTP headers: Content-Type and Content-Length.
  - ✓ Content-Type header: the MIME type for the message and the character encoding (optional) used for the XML body of the request or response (text/xml or application/xml for SOAP 1.1; application/soap+xml for SOAP 1.2)
    - Example

```
Content-Type: MIMEType; charset=character-encoding
```

```
POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
```

  - ✓ Content-Length header: the number of bytes in the body of the request or response.
    - Example:

```
Content-Length: bytes
```

```
POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 250
```

SOAP HTTP binding example

SOAP Request

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

SOAP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>
</soap:Envelope>
```

SOAP building blocks

- A SOAP message = ordinary XML document
- SOAP building blocks:
  - ✓ An Envelope element (Mandatory): identifies the XML document as a SOAP message
    - defines the start and the end of the message.
  - ✓ A Header element (Optional): contains header information
    - any optional attributes of the message used in processing the message
  - ✓ A Body element (Mandatory): contains call and response information
    - XML data comprising the message being sent.
  - ✓ A Fault element (Optional): errors and status information
- all the elements above are declared in the default namespace for the SOAP envelope:
  - ✓ http://www.w3.org/2001/12/soap-envelope (SOAP 1.2)
  - ✓ http://schemas.xmlsoap.org/soap/envelope/ (SOAP 1.1)
- the default namespace for SOAP encoding and data types is:
  - ✓ http://www.w3.org/2001/12/soap-encoding (SOAP 1.2)
  - ✓ http://schemas.xmlsoap.org/soap/encoding/ (SOAP 1.1)

Syntax Rules

A SOAP message **MUST** be encoded using XML

A SOAP message **MUST** use the SOAP Envelope namespace!

```
<?xml version="1.0"?>
<soap:Envelope

xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

A SOAP message **MUST** use the SOAP Encoding namespace

- A SOAP message **must NOT** contain ***XML Processing Instructions***

SOAP Envelope element (1/2)

- Required
- The root element of a SOAP message
- Defines the XML document as a SOAP message
- The xmlns:soap Namespace
  - ✓ defines the Envelope as a SOAP Envelope.
  - ✓ If a different namespace is used, the application generates an error and discards the message.

```
<?xml version="1.0"?>
<soap:Envelope

xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

Always should have this value!

SOAP Envelope element (2/2)

- The encodingStyle attribute:
  - ✓ to define the data types used in the document
  - ✓ this attribute may appear on any SOAP element
  - ✓ applies to the element's contents and all child elements.
- A SOAP message has no default encoding.

```
<?xml version="1.0"?>
<soap:Envelope

xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

General syntax:

soap:encodingStyle="*URI*"

SOAP Header Element (1/5)

- optional
- application-specific information about the SOAP message
  - ✓ e.g. authentication, payment, etc.
- If the Header element is present, it must be the first child element of the Envelope element
- All immediate child elements of the Header element must be namespace-qualified.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
  soap:mustUnderstand="1">234
  </m:Trans>
</soap:Header>
…
</soap:Envelope>
```

SOAP Header Element (2/5)

- define how a recipient should process the SOAP message
- three attributes defined in the default namespace ("http://www.w3.org/2001/12/soap-envelope"):
  - ✓ mustUnderstand
  - ✓ actor
  - ✓ encodingStyle.

SOAP Header Element (3/5)

soap:mustUnderstand="0|1"

- Is used to indicate whether a header entry is mandatory or optional for the recipient to process

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
  soap:mustUnderstand="1">234
  </m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

the receiver processing the Header **must** recognize the element!
If the receiver does not recognize the element it will fail when processing the Header.

SOAP Header Element (4/5)

soap:actor="*URI*"

- A SOAP message may travel from a sender to a receiver by passing different endpoints along the message path.
  - ✓ not all parts of the SOAP message may be intended for the ultimate endpoint, but instead for one or more of the endpoints on the message path.
- used to address the Header element to a specific endpoint

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
  soap:actor="http://www.w3schools.com/appml/">234
  </m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

SOAP Header Element (5/5)

soap:encodingStyle="*URI*"

- used to define the data types used in the document
- may appear on any SOAP element
  - ✓ applies to that element's contents and all child elements.
- A SOAP message has no default encoding

SOAP Body Element (1/2)

- Required
- Contains the actual SOAP message intended for the ultimate endpoint of the message.
- Immediate child elements of the SOAP Body element may be namespace-qualified.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Application-specific elements (not part of the SOAP namespace)

SOAP Body Element (2/2)

Example SOAP request

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Example SOAP response

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

SOAP Fault Element (1/2)

- Optional
- Used to indicate error messages.
- If a Fault element is present, it must appear as a child element of the Body element.
- A Fault element can only appear once in a SOAP message.
- Sub-elements of the SOAP Fault element:

Sub Element	Description
<faultcode>	A code for *identifying* the fault
<faultstring>	A human readable *explanation* of the fault
<faultactor>	Information about *who caused* the fault to happen
<detail>	Holds *application specific error information* related to the Body element

SOAP Fault Element (2/2)

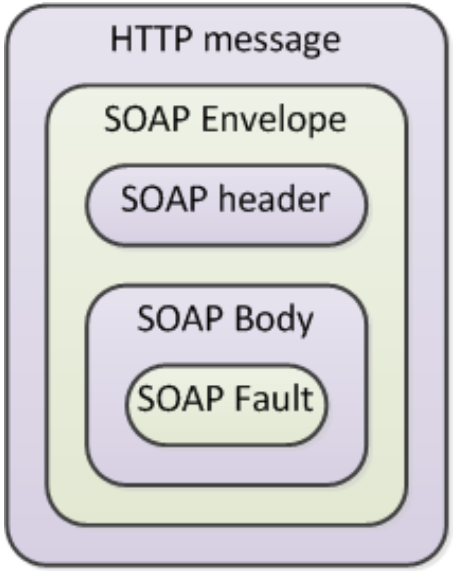
- SOAP Fault Codes
  - ✓ must be used in the faultcode element when describing faults

Error	Description
VersionMismatch	Found an invalid namespace for the SOAP Envelope element
MustUnderstand	An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood
Client	The message was incorrectly formed or contained incorrect information
Server	There was a problem with the server so the message could not proceed

Example of SOAP fault

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Client</faultcode>
            <faultstring>Message does not have necessary info</faultstring>
            <faultactor>http://gizmos.com/order</faultactor>
            <detail>
                <PO:order xmlns:PO="http://gizmos.com/orders/">
                Quantity element does not have a value</PO:order>
                <PO:confirmation xmlns:PO="http://gizmos.com/confirm">
                Incomplete address: no zip code</PO:confirmation>
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP elements summary



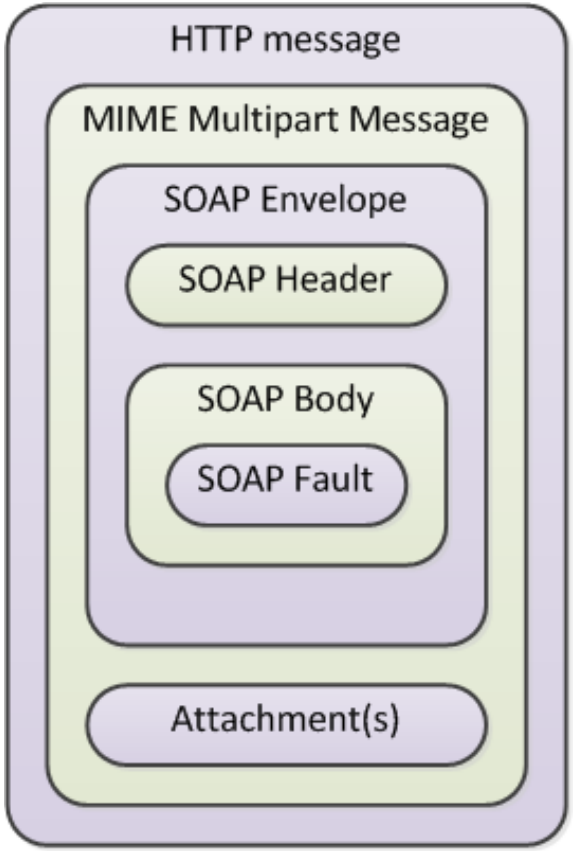
- Envelope element (Mandatory): identifies the XML document as a SOAP message
- Header element (Optional): contains header information
- Body element (Mandatory): contains call and response information
- Fault element (Optional): errors and status information

SOAP with Attachments

- Enables sending additional information together with a SOAP message
- Uses MIME (Multipurpose Internet Mail Extensions) mechanism for attachments
- MIME message is a multipart message with the first part carrying the main message, and subsequent parts carrying the attachments
- MIME boundary separates the parts of the message
  - ✓ Can be any sequence of characters (however is better to choose in a way it does not match with any parts of the message)
  - ✓ Is placed in the beginning, at the end, and in between parts of multipart message
  - ✓ At every instance, the boundary is preceded by two dash characters ”--”

SOAP with Attachments example

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
        start="<claim061400a.xml@claiming-it.com>"
Content-Description: This is the optional message description.

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <claim061400a.xml@claiming-it.com>

<?xml version='1.0' ?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        ..
        <theSignedForm href="cid:claim061400a.tiff@claiming-it.com"/>
        ..
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: image/tiff
Content-Transfer-Encoding: binary
Content-ID: <claim061400a.tiff@claiming-it.com>

...binary TIFF image...
--MIME_boundary--
```



CAMX

- IPC-2501
  - ✓ Terminology
  - ✓ Transaction rules
  - ✓ Messaging patterns

CAMX

- CAMX is the framework facilitating interconnection of machines by exchanging standardised XML messages
- The standards are brought by IPC association

IPC/CAMX Standards

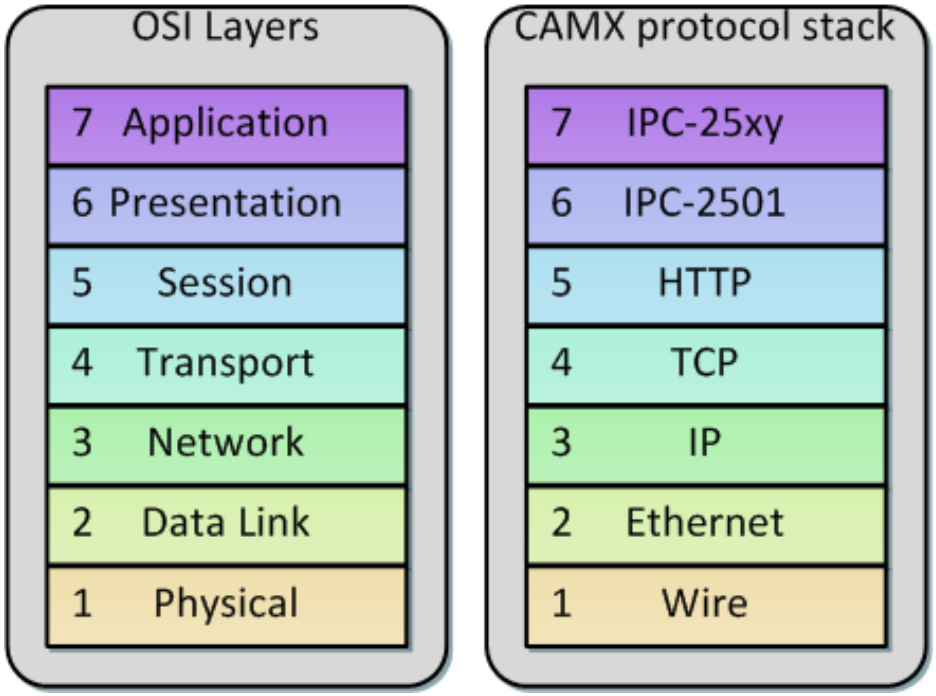
IPC Number/ Function	-xxx1 Generic	-xxx2 Administ	-xxx3 Document	-xxx4 Board Fab	-xxx5 Bare Bd Test	-xxx6 Assembly Manufact	-xxx7 Assy Test/Insp.	-xxx8 Comp. & Mat'ls
IPC-2500 CAMX Framework	IPC-2501		IPC-2503					
IPC-2510 GenCAM Product Data	IPC-2511	IPC-2512A	IPC-2513A	IPC-2514A	IPC-2515A	IPC-2516A	IPC-2517A	IPC-2518A
IPC-2520 Quality Product Data				IPC-2524				
IPC-2530 SRFF Process Data	IPC-2531							
IPC-2540 Shop Floor Communication (CAMX)	IPC-2541					IPC-2546(incl. Am1&2)	IPC-2547	
IPC-2550 Execution Communication (MES)	IPC-2551 (Working Draft)			IPC-2554 (Working Draft)		IPC-2556 (PINS)		
IPC-2560 Enterprise Communication								
IPC-2570 Supply Chain Communication (PDX)	IPC-2571					IPC-2576	IPC-2577	IPC-2578
IPC-2580 Application Specific Data	IPC-2581	IPC-2582	IPC-2583	IPC-2584		IPC-2586		IPC-2588

IPC/CAMX Standards

- IPC-2500 CAMX Framework
  - ✓ IPC-2501:2003 Definition for Web-Based Exchange of XML Data
    - establishes the semantics and an XML based syntax for shop floor communication between electronic assembly equipment and associated software applications
    - outlines the communication architecture and supporting XML messages
- IPC-2540 Shop Floor Communication (CAMX)
  - ✓ IPC-2541: 2001 Generic Requirements for Electronics Manufacturing Shop-Floor Equipment Communication Messages (CAMX)
    - establishes requirements and other considerations for the interchange of information between electronic manufacturing software equipment and factory information systems
  - ✓ IPC-2546 w/Amend 1: 2003 Sectional Requirements for Shop-Floor Equipment Communication Messages (CAMX) for Printed Circuit Board Assembly
  - ✓ IPC-2547: 2002 Sectional Requirements for Shop-Floor Equipment Communication Messages (CAMX) for Printed Circuit Board Test, Inspection and Rework

IPC-2501

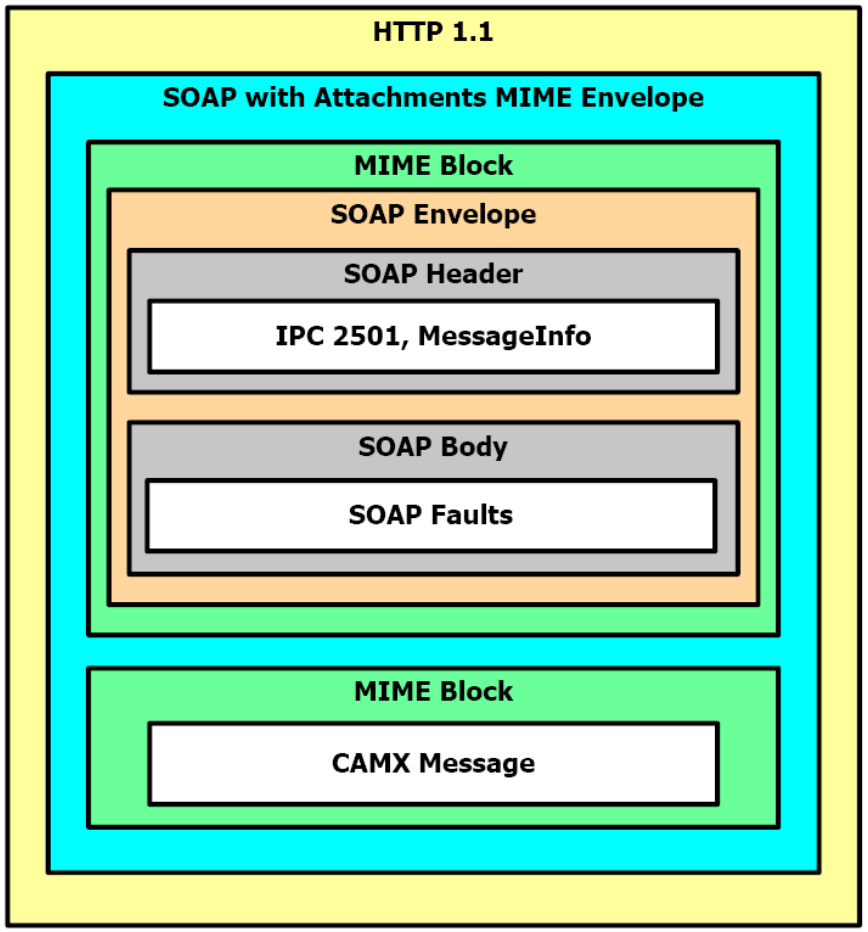
- IPC-2501 focuses on mechanism for exchanging CAMX messages
- The SOAP envelopes are exchanged using HTTP sessions
- To complete the protocol stack, the HTTP sessions are carried over TCP/IP socket connections



IPC-2501 Scope

- domain: electronics assembly manufacturing shop
- up to several hundred machines, each of which is capable of producing tens of messages per second.
- messages relatively small in size (under 20 kilobytes)
- occasionally: some application-specific files of several megabytes
- number of information consumers: relatively small number (<20), typically
- network interruptions are accounted for in the standard

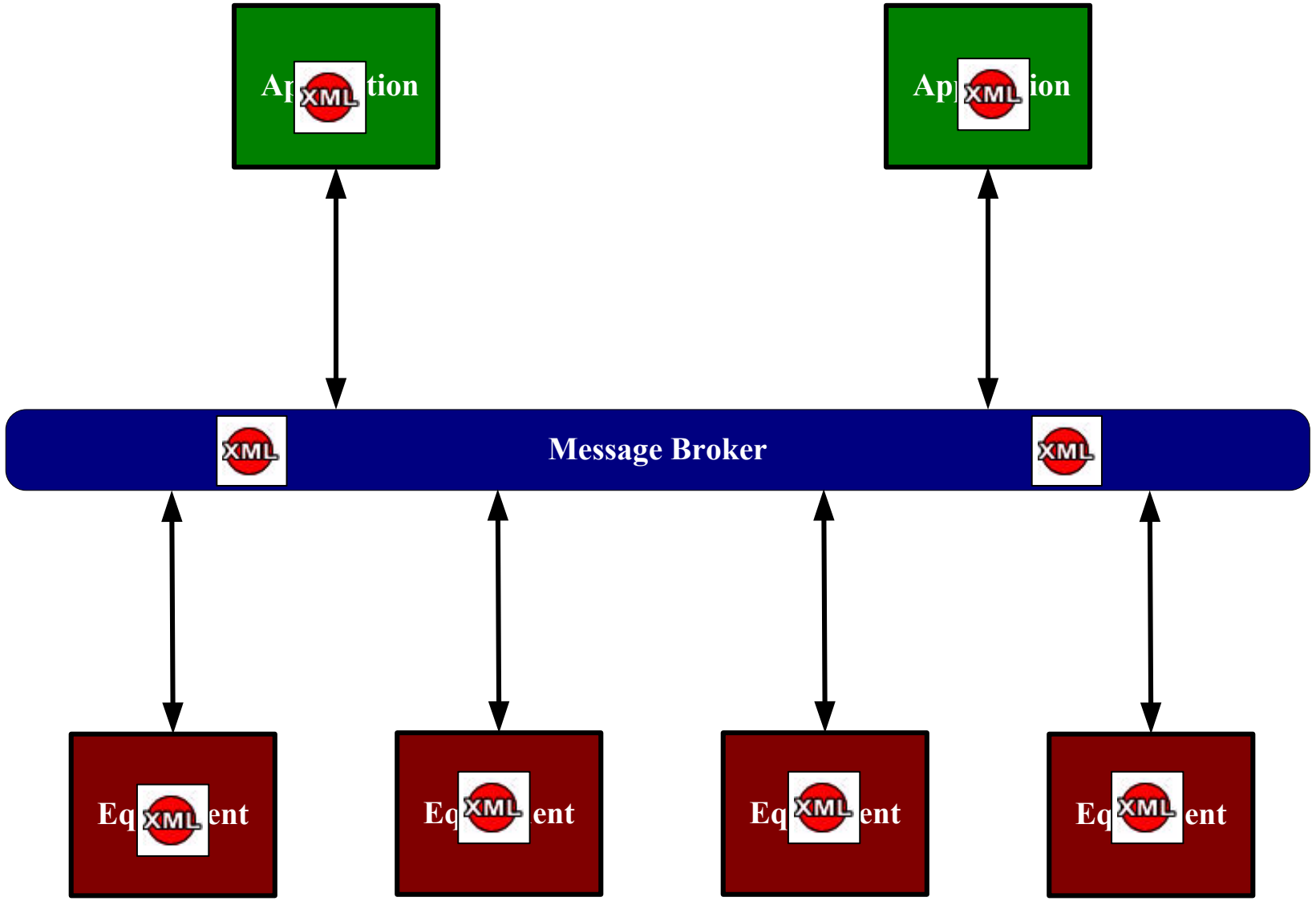
Transmission structure



- Based on SOAP with Attachments specification
  - The main SOAP part carries envelope information
  - The CAMX message is included as an attachment

Terminology

- Client
  - ✓ A generic term for any of the various machines, applications, or devices that may connect to a Message Broker in a Domain.
- Domain
  - ✓ The set of all Clients interested in communicating with each other.
  - ✓ It contains a single logical Message Broker and a configurable number of Clients.
- Domain Configuration
  - ✓ defines publishing capabilities, subscription interests, point-to-point communication privileges, quality of service parameters and general information about the Domain.
- Message Broker (MSB)
  - ✓ the messaging middleware.
  - ✓ responsible for intelligently routing messages among Clients.
  - ✓ Can be seen as a ”Post office” service, acting as intermediary between Clients exchanging messages (no direct communication between Clients is allowed)


XML
Message Broker
XML
XML

MSB benefits

- Clients only have to maintain single connection to MSB rather than all connections to all clients
  - ✓ Easier configuration for the clients
- MSB is the only place for configuration changes (what connections are allowed, what clients require what kind of messages, etc.)
- The MSB provides Guaranteed Message Delivery (acknowledgement mechanisms are used)
- Placing messaging logic to MSB makes simpler creation of new clients

Transaction rules

- All communication is done through MSB (direct communication between clients is forbidden)
- All clients must be Polling Consumers (meaning, that they always start the communication, even for publish/subscribe patterns)
- The messages for clients are kept in MSB in FIFO queue
  - ✓ clients cannot ask for the specific message
  - ✓ messages will be received by the Client in the sequence that they were received by the Message Broker

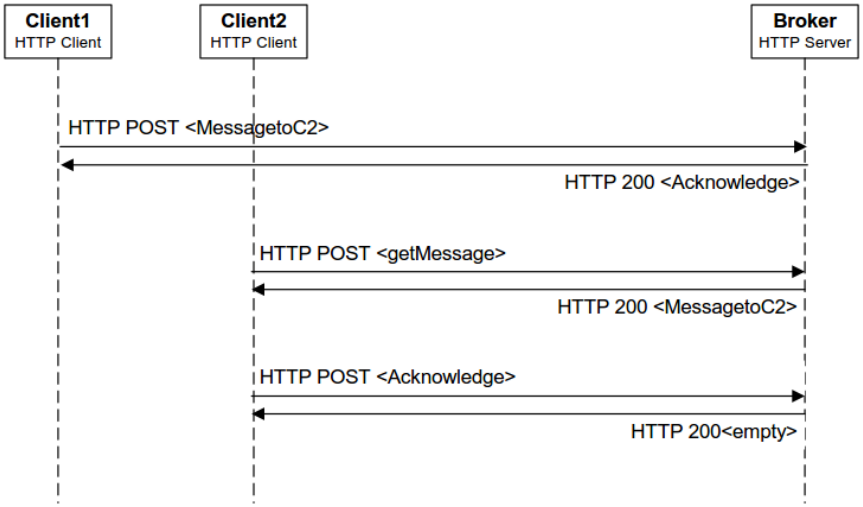
CAMX Messaging Patterns

- Two patterns are distinguished:
  - ✓ Point-to-point
    - One recipient of the message through MSB
  - ✓ Publish/Subscribe
    - The sender is not aware of the recipients

Point-to-point

- The message is sent to specific recipient
- Only one recipient is allowed
- Point-to-point is used to:
  - ✓ Send a command to a machine
  - ✓ Sent a query or request for a parameter value, and to return a response to such a request
  - ✓ Set a parameter, and to return a response
  - ✓ Exchange messages that contain confidential information

Point-to-point



- Client1 transmit the message via an HTTP POST request, in which the message to be transferred is included in the second MIME block.
- The Message Broker respond with an HTTP 200 status code, and an Acknowledge contained in the second MIME block.
- Client2 transmit an IPC-2501 GetMessage via an HTTP POST request, in which the GetMessage is included in the second MIME block.
- The Message Broker respond with an HTTP 200 status code, and the message to be transferred contained in the second MIME block.
- Client2 shall transmit an Acknowledge via an HTTP POST request, in which the Acknowledge is contained in the second MIME block.
- The Message Broker shall respond with an HTTP 200 status code, and an empty SOAP envelope

Publish/Subscribe

- Message can have more than one interested recipients
- Types of the messages in publish/subscribe are called topics
- Publishers send messages associated with topics. Subscribers subscribe to a particular topic(s)
- Publish/subscribe is used to
  - ✓ Publish/anounce events
  - ✓ Broadcast commands

Publish/subscribe

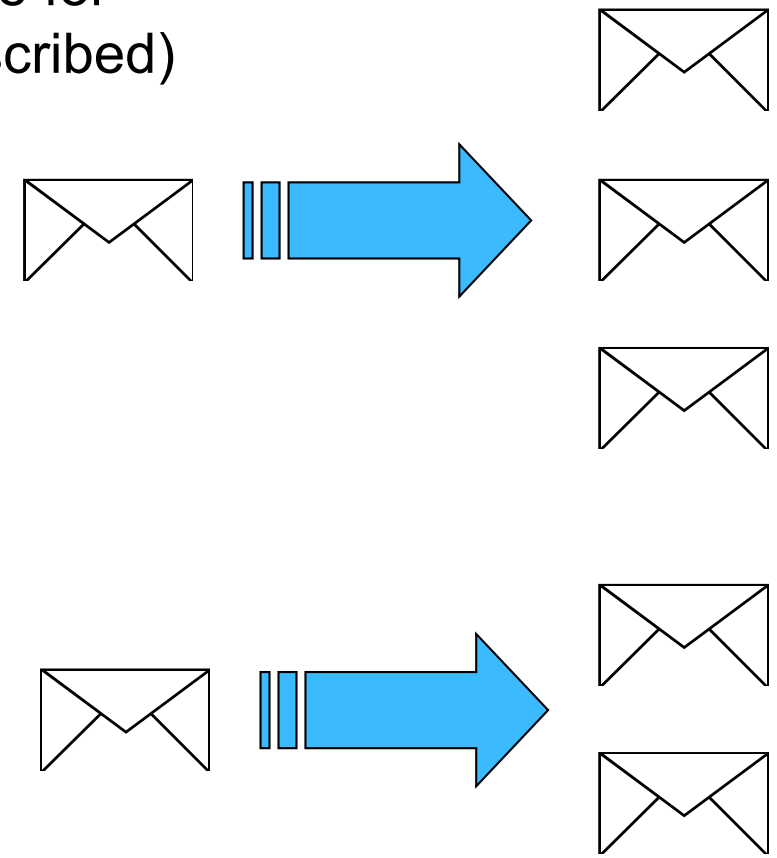
Clients send (publish) messages to the broker.

Message Broker

Publish/subscribe

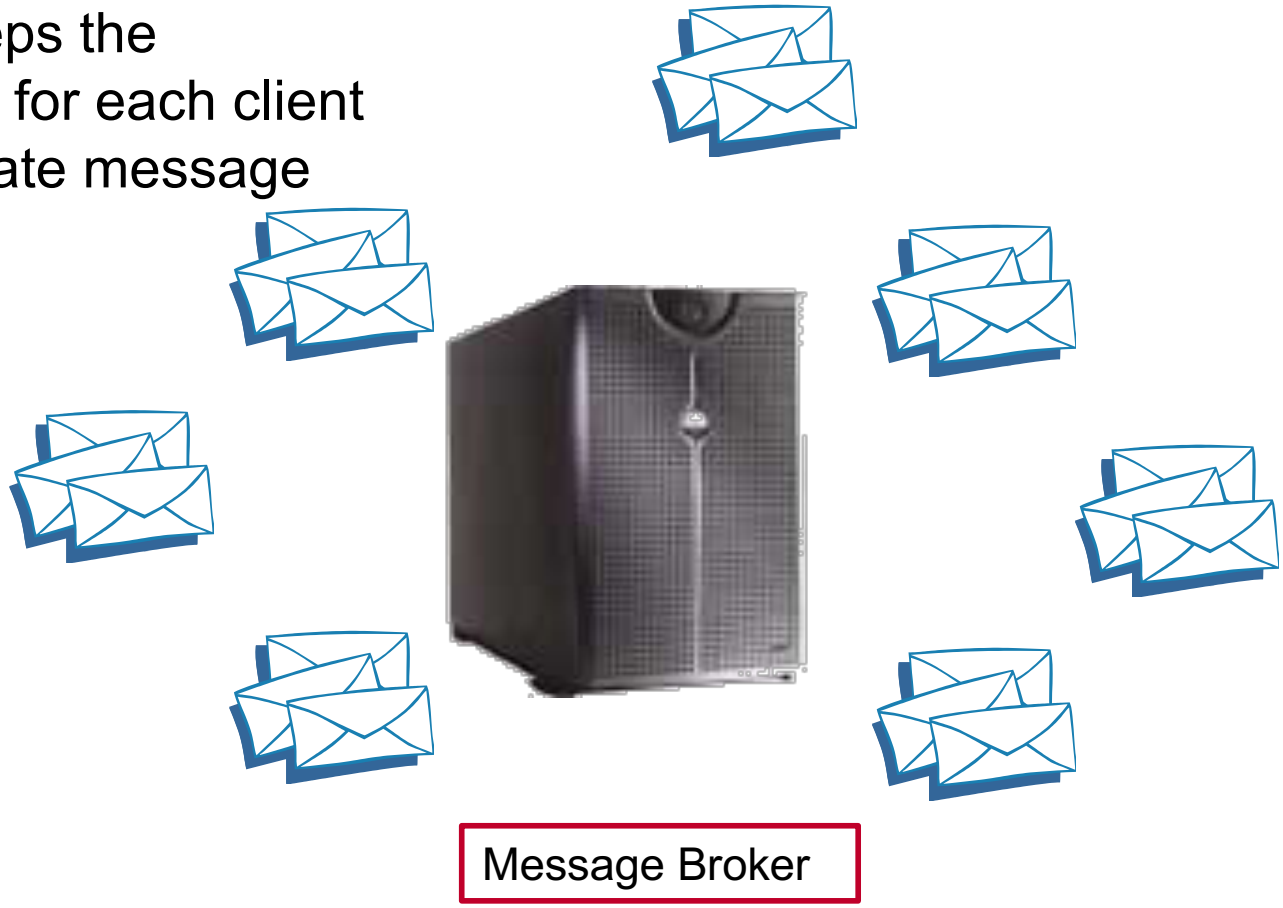
The broker then creates a copy of each message for each interested (subscribed) client...

Message Broker



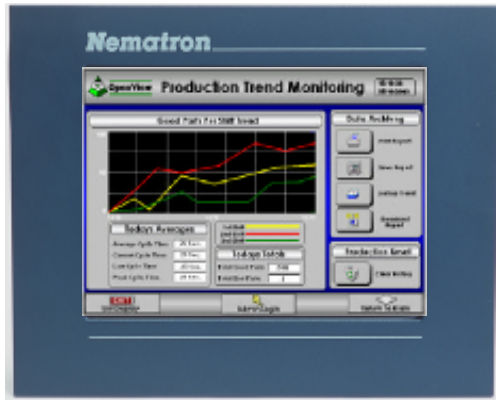
Publish/subscribe

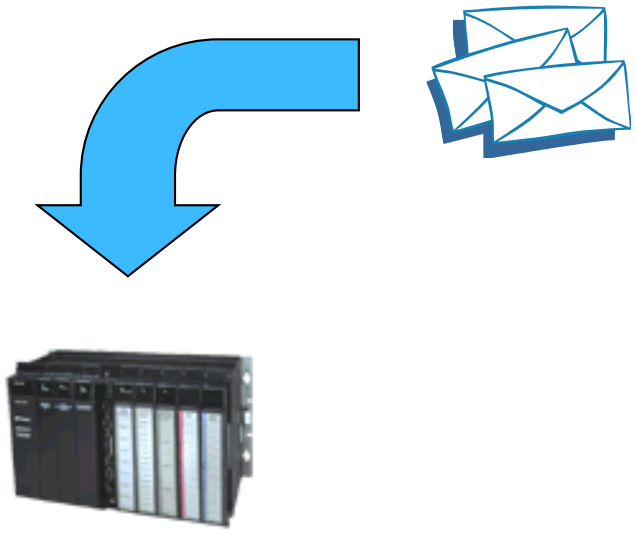
... and keeps the messages for each client in a separate message box.



Publish/subscribe

Clients then retrieve messages from *their queue* in the broker.





Message Broker

CAMX Messaging- Example Scenario (1/4)

Manufacturing Equipment 1

1

- Assembly line finishes work on an item
- posts a message with data (item ID, batch No., and components' S.N...)

Message Broker

CAMX Messaging - Example Scenario (2/4)

Manufacturing Equipment 1

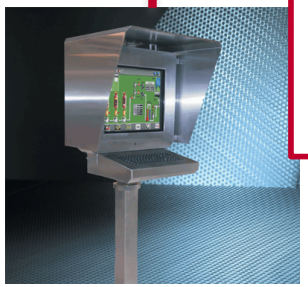
1

- finishes work on an item
- posts a message with data (item ID, batch No., and components' S.N...)

Message Broker

2

- subscribed to Equipment 1 published messages
- periodically checks with the Broker to see if it has any messages in queue.
- Status Monitor, MES, QAS, …



Monitoring Equipment

CAMX Messaging - Example Scenario (3/4)

Manufacturing Equipment 1

1

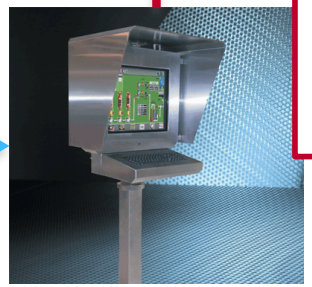
- finishes work on an item
- posts a message with data (item ID, batch No., and components' S.N...)

Message Broker

3 A copy of the assembly line published message is delivered to the subscriber

2 subscribed to Equipment 1 published messages

- periodically checks with the Broker to see if it has any messages in queue.
- Status Monitor, MES, QAS, …



Monitoring Equipment

CAMX Messaging - Example Scenario (4/4)

Manufacturing Equipment 1

1

- finishes work on an item
- posts a message with data (item ID, batch No., and components' S.N...)

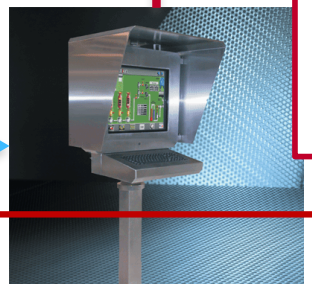
Message Broker

3 A copy of the assembly line published message is delivered to the subscriber

2 subscribed to Equipment 1 published messages

- periodically checks with the Broker to see if it has any messages in queue.
- Status Monitor, MES, QAS, …

Manufacturing Equipment 2



Monitoring Equipment

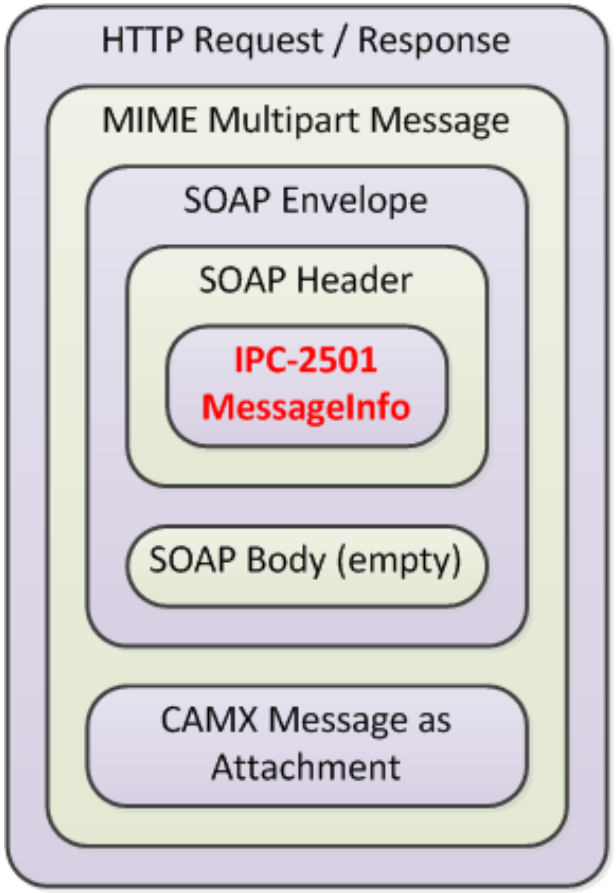
Subscriber to the same message from Equipment 1

CAMX IPC-2501 Transactions

- MessageInfo element
- Publishing messages to MSB
- Getting messages from MSB

MessageInfo SOAP Element

- Every transmission carries IPC-2501 MessageInfo element in the SOAP Header element
- SOAP body element is left empty
- CAMX message is carried as MIME attachment
  - At most one CAMX message is allowed as an attachment



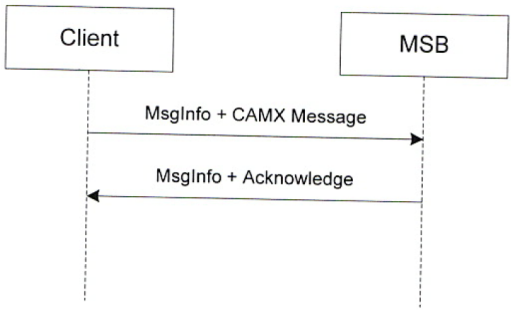
MessageInfo attributes

- sender: the Unique Resource Identifier (URI) identifying the sender of the message
- destination: the URI identifying the destination of the message
  - ✓ URI of the machine in point-to-point messages
  - ✓ URI of the MSB in publish/subscribe messages
- dateTime: date and timestamp for the transaction
  - ✓ Is formatted according to W3C recommendations
- messageSchema: the type of message being transported
  - ✓ MSB does not check for message content, but using it as a topic in publish/subscribe message exchange
- messageId: a unique identifier of the message being transported
  - ✓ Is free to assign, however usually ID is a concatenation of the sender URI, messageSchema, the timestamp, and a random number

MessageInfo example

```
------=_Part_0_2086370.1060164663859
Content-Type: text/xml
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope
          xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-env:Header>
    <IPC2501MsgInfo:MessageInfo
    xmlns:IPC2501MsgInfo="http://webstds.ipc.org/2501/MessageInfo.xsd"
      sender="robot3.line2.company.org"
      destination="broker.company.org"
      dateTime="2004-12-24T16:01:00.00+02:00"
      messageSchema="http://webstds.ipc.org/2541/EquipmentAlarm.xsd"
      messageId="robot3.line2.company.org|2004-12-24T16:01:00.00+02:00|0001"/>
  </soap-env:Header>
  <soap-env:Body/>
</soap-env:Envelope>

------=_Part_0_2086370.1060164663859
Content-Type: text/xml
Content-Transfer-Encoding: 8bit
Content-ID: robot3.line2.company.org|2004-12-24T16:01:00.00+02:00|0001
Content-Length: 431

<?xml version="1.0" encoding="UTF-8"?>
<IPC2541EqAlrm:EquipmentAlarm
  xmlns:IPC2541EqAlrm="http://webstds.ipc.org/2541/EquipmentAlarm.xsd"
    dateTime="2004-12-24T16:01:00.00+02:00"
    alarmId="012"
    alarmInstanceId="000132"
    alarmType="ITEMSAFETY"
    laneList="1"
    zoneList="1"/>

------=_Part_0_2086370.1060164663859--
```

Publishing messages to MSB



- Sequence of SOAP interactions
  - Client sends SOAP request message with MessageInfo and CAMX message as an attachment
  - If MSB processes request successfully, it returns response with new MessageInfo and Acknowledge type of message as CAMX attachment
- The sequence for publishing is the same for point-to-point and publish/subscribe
  - The difference is in the MessageInfo attribute values

CAMX Acknowledge message

- IPC-2501 specifies Acknowledge message. It contains two attributes:
  - ✓ dateTime of the message creation
  - ✓ messageId, which identifies the message being acknowledged. It must be the copy of the messageId sent to the MSB

Publish message to MSB example

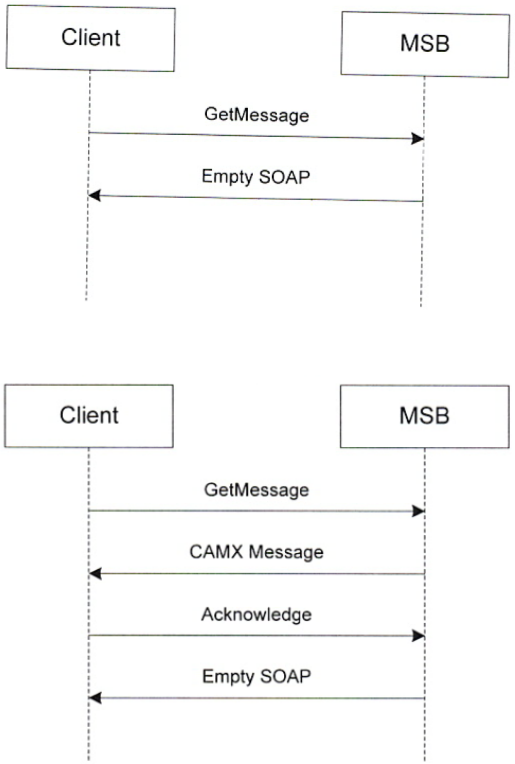
CAMX message

```
------=_Part_0_2086370.1060164663859
Content-Type: text/xml
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope
        xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-env:Header>
    <IPC2501MsgInfo:MessageInfo
    xmlns:IPC2501MsgInfo="http://webstds.ipc.org/2501/MessageInfo.xsd"
      sender="robot3.line2.company.org"
      destination="broker.company.org"
      dateTime="2004-12-24T16:01:00.00+02:00"
      messageSchema="http://webstds.ipc.org/2541/EquipmentAlarm.xsd"
      messageId="robot3.line2.company.org|2004-12-24T16:01:00.00+02:00|0001"/>
  </soap-env:Header>
  <soap-env:Body/>
</soap-env:Envelope>

------=_Part_0_2086370.1060164663859
Content-Type: text/xml
Content-Transfer-Encoding: 8bit
Content-ID: robot3.line2.company.org|2004-12-24T16:01:00.00+02:00|0001
Content-Length: 431

<?xml version="1.0" encoding="UTF-8"?>
<IPC2541EqAlrm:EquipmentAlarm
  xmlns:IPC2541EqAlrm="http://webstds.ipc.org/2541/EquipmentAlarm.xsd"
    dateTime="2004-12-24T16:01:00.00+02:00"
    alarmId="012"
    alarmInstanceId="000132"
    alarmType="ITEMSAFETY"
    laneList="1"
    zoneList="1"/>

------=_Part_0_2086370.1060164663859--
```

Corresponding acknowledge

```
------=_Part_1_2496732.1066370346739
Content-Type: text/xml
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope
        xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-env:Header>
    <IPC2501MsgInfo:MessageInfo
     xmlns:IPC2501MsgInfo="http://webstds.ipc.org/2501/MessageInfo.xsd"
      sender="broker.company.org"
      destination="robot3.line2.company.org"
      dateTime="2004-12-24T16:01:01.00+02:00"
      messageSchema="http://webstds.ipc.org/2501/Acknowledge.xsd"
      messageId="”broker.company.org|2004-12-24T16:01:02.00+02:00|0001”/>
  </soap-env:Header>
  <soap-env:Body/>
</soap-env:Envelope>

------=_Part_1_2496732.1066370346739
Content-Type: text/xml
Content-Transfer-Encoding: 8bit
Content-ID: robot3.line2.company.org|2004-12-24T16:01:01.00+02:00|0001
Content-Length: 431

<?xml version="1.0" encoding="UTF-8"?>
<IPC2501Ack:Acknowledge
  xmlns:IPC2501Ack="http://webstds.ipc.org/2501/Acknowledge.xsd"
    dateTime="2004-12-24T16:01:01.00+02:00"
    MessageId="robot3.line2.company.org|2004-12-24T16:01:00.00+02:00|0001"
/>

------=_Part_1_2496732.1066370346739--
```

Getting messages from MSB



- When messages are sent to MSB, they are placed into message boxes for corresponding clients in FIFO order. Then clients should request messages
- Sequence of SOAP interactions for getting messages from MSB
  - Client sends SOAP request message with MessageInfo and CAMX GetMessage message as an attachment
  - If MSB does not have messages for Client
    - MSB returns Empty SOAP message for Client
  - If MSB has messages for Client
    - MSB sends corresponding SOAP message with MessageInfo and CAMX message as an attachment
    - The client sends back an Acknowledge message to MSB
    - The MSB removes message and sends to Client Empty SOAP message

GetMessage and Empty SOAP messages

- GetMessage has only 1 attribute
  - ✓ dateTime is the time of the message creation
- Empty SOAP message does not have contents for header and body

```
<?xml version=”1.0” encoding=”UTF-8”?>
<soap:Envelope
xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
>
    <soap:Header/>
    <soap:Body/>
</soap:Envelope>
```

Summary

- CAMX is the framework facilitating interconnection of machines by exchanging standardised XML messages
  - ✓ IPC created a series of standards for machine communication
- IPC-2501 focuses on mechanism for exchanging CAMX messages
  - ✓ Defines communication stack
  - ✓ Defines rules for exchanging messages between clients and MSB
  - ✓ Defines XML configuration of message broker (Domain Configuration)