SOAP FEATURES AND MODULES

SOAP Modules

The combined syntax and semantics of a set of SOAP headers are known as a SOAP module. A SOAP module realizes one or more SOAP features. This enables us to specify a more general-purpose concept such as a secure purchase order, with a combination of one or more features, including the purchase order MEP as described above, the security feature, and more.

So far we have discussed message packaging, transmission, and processing in a transport in a rather independent manner. These SOAP messages can be transported through different transport bindings, such as HTTP, TCP, and BEEP. The SOAP defined some protocol bindings for its transport methods . The most notable one is SOAP over HTTP utilizing the GET and POST operations.

The SOAP Processing Model

The processing of a SOAP message is dependent on the role assumed by the processor. As we have mentioned above, the SOAP headers are targeted using a "role" attribute. If the SOAP intermediary plays the role as defined by the SOAP message, it can then process the message.

There are two options related to processing. If the SOAP header is targeted to this node and specifies a "mustUnderstand" flag set to "true," then the processing node must process that header. If there is no such requirement (i.e., mustUnderstand flag is not set), it is up to the processing node to decide on the processing of the message. Once the processing is completed, the message will be directed to the next node. The decision on the next node selection is not specified by the SOAP specification. Therefore, it is now the choice of the processing node to make such a decision. However, there are some standards that exist to specify common routing mechanisms, such as WS-Routing [15] and WS-Addressing. [16]

Another interesting aspect of this message-forwarding paradigm is the concept of relaying SOAP headers. A header can have a "relay" attribute value (i.e., true or false) to indicate that

nonprocessed headers get forwarded to the next node. The default value is "false." This indicates a SOAP node, which is targeted by this header, will not forward this header to the next node.
If we refer back to the SOAP 1.1 specification (SOAP1.1), we could see that there is nostandard processing rules for SOAP extensions (header) processing. This causes a number of interoperability problems. However, the SOAP 1.2 specification introduces the following core SOAP constructs in order to support SOAP extensions.

SOAP Features

A SOAP feature is an extension to the SOAP messaging framework. These features are common in distributed computing such as reliability, security, correlation, routing, and message exchange patterns such as request/response, one-way, and peer-to-peer conversations. Readers must be aware that this is an abstract concept with the indication that there is some processing needs to be done in the SOAP nodes but it does not specify how this processing is done.
A SOAP feature has the following characteristics:

1. **A** unique name **used to identify the feature and its properties. This enables us to identify whether a SOAP node supports a specific feature. For example, if we have a feature called "secure-ssl-channel," then we can ask the SOAP nodes, including the ultimate receiver, whether they support that feature or not.**
2. A set of properties associated with a feature that can be used to control, constrain, or identify a feature. For example, we can see in the SOAP request “response message exchange pattern there are properties for accessing the inbound or outbound messages, the immediate sender, and next destination.

It is important to understand that SOAP provides two mechanisms for implementing these features:

1. **SOAP header blocks. In this kind of implementation SOAP header blocks are used to specify a feature. These headers are processed by the SOAP nodes. As we have seen, the SOAP processing model defines the behaviors of a single processing SOAP node in order to process an individual message. The most common example of such a feature is the security features as defined by WS-Security specifications.**
2. SOAP binding protocol. In this case the features are directly implemented in the protocol binding level. For example, a binding extension to support the SOAP over SSL protocol.

As we have noted, in the first case it is more protocol independent and flexible, but this may cause unnecessary processing overhead. The second case is protocol dependent and the flexibility is limited.

In addition to the extensions as specified in the above cases, the SOAP specification defined a standard feature for the message exchange pattern called MEP. Let us now explore in further detail exactly how this is defined.

Message Exchange Pattern

One special type of SOAP feature is the MEP. A SOAP MEP is a template that establishes a pattern for the exchange of messages between SOAP nodes. Some examples of MEPs include request/response, one-way, peer-to-peer conversation, and so on. To clarify further, we could consider a special MEP, a request for proposal (RFP) MEP, where we could define a message exchange pattern for the RFP such as submitting a request to the service provider, getting a response from the provider at a later period, and other subpatterns. We can envision building these kinds of message exchange patterns.

MEP, similar to other features, is implemented either as headers or using protocol bindings. A MEP may be supported by one or more underlying protocol binding instances either directly or indirectly with support from the software. This software implements the required processing to support the SOAP feature, expressed as a SOAP module.