Error Module

Working Draft 07, 06 November 2005

The Error Module is a sub-component of the ebMS Module, and it must take into account the composed nature of an MSH, which includes relatively independent (SOAP) modules: the Reliability Module, the Security Module, the ebMS Module, and the Addressing Module. The Error Module is also subject to the same connectivity constraints as the exchange of regular messages. This calls for a more comprehensive error model. The Error Module introduces three major concepts: Escalated Error, Error Reporting, and Error Handling. These concepts are defined below. The Error Module specification only specifies when and how errors are raised. The specification considers the error escalation, error reporting, and error handling as orthogonal concepts that are best specified within an agreement (or operation context for Error Module). The Error Module packaging provides extensibility for adding new types of errors to the ebMS Module's set of errors, as well as the possibility to represent errors occurring outside the ebMS Module as errors in the ebMS Module.

6.1 Terminology

Fault: A Fault always means a SOAP Fault. It must be generated and processed according to the SOAP 1.1 or SOAP 1.2 specification.

Error: An error that is not a fault and occurs in either module (ebMS Module, Reliability Module, Security Module, or Addressing Module).

ebMS Error: this is a particular case of an Error. It is an error raised by the ebMS Module.

Reliability Error: this is a particular case of an Error. It is an error raised by the Reliability Module.

Security Error: this is a particular case of an Error. It is an error raised by the Security Module.

Addressing Error: this is particular case of an Error. It is an error raised by the Addressing Module.

Escalated Error: this is a particular case of an ebMS Error. It is an error raised by the ebMS Module not because of a problem occurring inside the ebMS Module, but because an Error was raised by the Security, Reliability, or Addressing Module. In other terms, an Escalated Error is not caused within the ebMS Module itself, but it is merely raised by the ebMS Module (the original cause of the error occurs within a module other than the ebMS Module, such as Security, Reliability, or Addressing Module).

Error Reporting: The act of transferring (communicating) an ebMS Message containing an ebMS Error or an Escalated Error to some other entity (this entity could be the other MSH, the Producer, the Consumer, or any local or remote entity).

Error Handling: describes the processing behavior of a SOAP processor when it detects or receives an Error through Error Reporting.

Message-in-error: A flawed message causing an error or warning of some kind.

6.2 Packaging of ebMS Errors

6.2.1 eb:Error Element

Each error raised by an MSH has the following properties:

- origin (required attribute)
- category (optional attribute)
- errorCode (optional attribute)
- severity (optional attribute)
- description (optional child element)

Example:

```
<eb:Error eb:origin="ebMS" eb:category="Unpackaging"
          eb:errorCode="eb_InvalidHeader" eb:severity="fatal">

    <eb:description> … </eb:description>

</eb:Error>
```

Attribute: eb:Error/@origin

This required attribute identifies the functional module within which the error occurred. This module could be the Reliability Module, the Security Module, the ebMS Module, or the Addressing Module. The possible values for this attribute are: ebMS, security, reliability, addressing.

Attribute: eb:Error/@category

This optional attribute identifies the type of error related to a particular origin. For example: Content, Packaging, UnPackaging, Communication, InternalProcess.

Attribute: eb:Error/@errorCode

This optional attribute is a unique identifier that plays the role of a label for a very particular kind of error in the message in error (much like the driver license number or social security number is a label for a person). This attribute is also used in two ways. First, it is a tool used to profile the errors of the other modules (errors that may rise within reliability, security, and addressing), and map them to ebMS errors. Second, the errorCode attribute could also be used an extensibility mechanism. In other terms, new errors could be added by just defining a new value for the errorCode.

Attribute: eb:Error/@severity

This optional attribute indicates the severity of the error. Valid values are: **warning**, **fatal**.

The warning value indicates that a potentially disabling condition has been detected, but no message processing and/or exchange has failed so far (in particular, if the message was supposed to be delivered to a consumer, it would be delivered even though a warning was issued). Other related messages in the conversation or MEP can be generated and exchanged in spite of this problem.

The fatal value indicates that the processing of a message did not proceed as expected, and cannot be considered as successful. If in spite of this the message payload is in a state of being delivered, the default behavior is to not deliver it, unless an agreement states otherwise (see OpCtx-ErrorHandling). This error does not presume of the ability of the MSH to process other messages, although the conversation or the MEP instance this message was involved in, is at risk of being invalid.

Attribute: eb:Error/description

This optional element provides a narrative description of the error.

6.2.2 ebMS Error Message

ebMS Errors are packaged as signal messages. Several eb:Error elements may be present under eb:SignalMessage. If this is the case and if eb:RefToMessageId is present as a child of eb:SignalMessage/eb:MessageInfo, then every eb:Error element must report an error on the ebMS message (message-in-error) identified by eb:RefToMessageId.

If the element eb:SignalMessage/eb:MessageInfo does not contain eb:RefToMessageId, then the eb:Error element(s) MUST NOT be related to a particular ebMS message.

Example of an ebMS Error Message :

```
<SOAP:Header …>

  <eb:Message eb:version="3.0" SOAP:mustUnderstand="1" >
     <eb:SignalMessage>
        <eb:MessageInfo>
           <eb:timestamp>2000-07-25T12:19:05</eb:timestamp>
           <eb:MessageId>UUID-2@example.com</eb:MessageId>
           <eb:RefToMessageId> UUID-1@example.com</eb:RefToMessageId>
        </eb:MessageInfo>

        <eb:Error eb:origin="security" category="Processing"
                  eb:errorCode="sec_FailedAuthentication"
                  eb:severity="fatal">
            <eb:Description>Validation of signature failed</eb:Description>
        </eb:Error>
        <eb:Error eb:origin="ebMS" category="Communication"
                  eb:errorCode="eb_EmptyPipe" eb:severity="warning">
          <eb:Description>PullRequest done on an empty pipe</eb:Description>
        </eb:Error>
     </eb:SignalMessage>
  </eb:Message>

</SOAP:Header>
```

6.3 Extensibility of the Error Module

6.3.1 Adding new ebMS Errors

The errorCode attribute (**eb:Message/eb:SignalMessage/eb:Error/@errorCode**) is a unique identifier (globally unique at least within the collection of all modules that are under the MSH orchestration, and this includes Reliability, Security, Addressing, and ebMS Modules). The value of the errorCode attribute is used as an extensibility mechanism to allow adding new ebMS Errors. For example, future versions of this specification may add new ebMS Errors by simply creating new values for the errorCode attribute. Even implementers of this specification may add their own custom ebMS Errors by simply coining new values for the errorCode attribute. A value for the errorCode attribute does not have to be meaningful. A value for the errorCode attribute only needs to be unique and can be as any garbage string as the word “eb_899LF476”. The semantics of the error itself are not carried within the name of the error or within its SOAP packaging. The creator of a new ebMS Error would simply create a new value for the errorCode attribute and then associate some semantics to the code value within this specification (by saying for example that the errorCode eb_899LF476 is used when such and such conditions occur). These semantics won’t be traveling with the ebMS Error packaging.

6.3.2 Escalating Errors as ebMS Errors

Errors occurring within the Reliability, Security, or Addressing Module may sometimes need to be represented as ebMS Errors (this is part of an agreement or operation context for the Error Module). For the Error Module (a sub-component of the ebMS Module) to be able to represent non-ebMS Errors as ebMS Errors, it needs a way to map these non-ebMS Errors to ebMS Errors. This mapping (called also error escalation when it is carried out by the Error Module) would simply consists in creating a new value for the errorCode attribute for each non-ebMS Error that needs to be mapped to a new ebMS Error. In other terms, the semantics (saying that errorCode such and such is used when such non-ebMS Error is raised) would be described in the specification only but not carried within the SOAP packaging of ebMS Errors.

Mapping non-ebMS Errors to new ebMS Errors is a profiling task of the modules under the orchestration of the MSH. This profiling is considered optional in this specification, and if specified, it would only be within an appendix to this specification. Being optional means that an MSH is not required to understand the errorCode attribute if its value is not recognized. However, errorCode values that represent errors occurring within the ebMS Module MUST be understood by an MSH.

6.4 Generating ebMS Errors

This specification identifies key ebMS Errors as well as the conditions under which these must be generated. Some of these error-raising conditions include the escalation as ebMS Errors of either Faults or Errors generated by Reliability, Addressing, and Security modules. These modules could be those contained in the MSH raising the Error, or those contained in a remote MSH communicating with the MSH raising the Error. Error escalation policies are left to an agreement between users, the representation of which is abstracted in the operation context of an MSH (OpCtx-ErrorHandling).

Whether module-specific Errors must be escalated as ebMS Errors or not is case-dependent. For example, some Errors raised by a receiving Reliability module are intended to the sending Reliability module, which is able to handle these appropriately (e.g., a reliable message has been assigned to an invalid message sequence). If the sending module cannot handle them in a satisfying way, it may escalate them as ebMS Errors (e.g., in case the Reliability sending module relies on its container entity – here the MSH - for assigning valid messages to sequences). The general recommendation is to not interfere with module-specific Errors that are exchanged between a sending / receiving pair of a functional module.

Other Reliability Errors are raised with out-of-band notification intent, e.g. a failure to successfully resend a reliable message will be notified by a sending Reliability module to the containing MSH, as required by the reliability specification. An MSH MUST be capable of capturing such Errors generated within MSH scope, and MUST be able to generate ebMS Errors from these. The actual decision to do so at run-time, as well as the way to report such errors, may be left to the operation context (OpCtx-ErrorHandling), e.g. as resulting from an agreement.

6.5 Error Reporting

There are three primary means for Error Reporting: Reporting with **Fault Sending**, Reporting with **Notification**, and Reporting with **Error Signal.**

Reporting with Fault Sending: An MSH may generate a SOAP Fault when a certain type of ebMS Error is raised. It is not recommended to use the Fault reporting action if other actions are sufficient.

Reporting with Notification: An out-of-band transfer of error information from MSH to some entity (message producer, consumer, or any other entity, be it local or remote). In case of notification to the message Producer or Consumer, such error reporting is abstracted by the “Notify” operation in the messaging model.

Reporting with Error Signal: An ebMS signal message sent from one MSH to another, which contains at least one eb:Error element. Such error reporting is modeled by Send and Receive abstract operations over such a message.

Example of different options in reporting errors raised on a Sending MSH: Some error detected on a submitted message and before it is even packaged, would normally be locally notified to the message Producer, and not even reported to the destination MSH. However, in case this message was part of a larger exchange that is holding its state waiting for completion on the receiving side, the preferred policy could state that the message-in-error be also reported (using an error message) to the Receiving MSH. If a Pull-mode of transfer is established so that the Receiving MSH is getting its messages as responses to PullRequest signals, such ebMS errors could be pulled in the same way. If a Push-mode is active from sender to receiver, it could be decided that errors generated on sender side will be pushed like any regular message. It could also be decided that they will be submitted to a separate message pipe in Pull-mode, so that the Receiving MSH has the option to get such sending-side errors only on demand.

Example of different options in reporting errors raised on a Receiving MSH: If a Receiving MSH detects an error in a received message, the reporting policy may vary depending on the context and the ability of parties to process such errors. For example, the error-raising Receiving MSH may just notify its own Consumer party, or send back an error message to the Sending MSH, or both. The usual common requirement in all these cases, is that the error be reported somehow, and complies with the eb:Error element structure.

6.6 Standard ebMS Errors

This section describes the Errors that are occurring within the ebMS Module itself (namely the ebMS Errors that are not Escalating Errors).

errorCode value	Recommended severity	category value	Description or Semantics
eb_ValueNotRecognized	fatal	Content	
eb_FeatureNotSupported	warning	Content	
eb_ValueInconsistent	fatal	Content	
eb_Other	fatal	Content	
eb_ConnectionFailure	fatal	Communication	
eb_EmptyPipe	warning	Communication	
eb_MimeInconsistency	fatal	Unpackaging	
eb_FeatureNotSupported	fatal	Unpackaging	
eb_InvalidHeader	fatal	Unpackaging	