Change History

Version	Date	Summary of amendment
0.0.1	03/28/03	First release.

Table of Contents

1.1 PURPOSE

The purpose of this document is to define a Common Base Event and supporting technologies that define the structure of an event in a consistent and a common format. The purpose of the Common Base Event is to facilitate the effective intercommunication amongst disparate enterprise applications that support logging, management, problem determination, autonomic computing and e-business functions in an enterprise. In this document, baseline aspects are specified which encapsulate properties common to a wide variety of events, including business, autonomic, management, tracing and logging type events. The format of the event is expressed as an XML document in UTF-8 encoding.

It should be noted that this document is not prescriptive regarding how individual applications are to store their data locally; only regarding how messages are to be sent between applications. Therefore, the application requirement is only to be able to generate or render events in this format, not store them. The goal of this effort is to ensure the accuracy, improve the detail and standardize the format of events to assist in designing robust, manageable and deterministic systems. The results are a collection of specifications surrounding a "Common Base Event" definition to be used as a new standard for events amongst management and business enterprise applications.

1.2 Overview

A small event can change things far beyond the seeming initial circumstance. Nowhere is this more true than in today's complex world of e-business where multitudes of interconnected systems must work together to perform many of the simple housekeeping activities which are necessary to keep a computing system healthy. Clearly, in this world, small things can have wide-reaching implications and few things are as small, yet pervasive, in a computing infrastructure than an event. The *event*, which encapsulates message data sent as the result of an occurrence, or situation, represents the very foundation on which these complex systems communicate. Events passed between and amongst applications in complex information technology systems represent the very nervous system that allows these various facets of the system to interoperate, communicate and coordinate their activities. Basic aspects of enterprise management, such as performance monitoring, security and reliability, as well as fundamental portions of e-business communications, such as order tracking, are grounded in the viability and fidelity of these events, in that quality data lends to accurate, deterministic and proper management of the enterprise. Poor fidelity can lead to misguided, potentially harmful results, or even results that are fatal to the system. Consider that even simple things such as the formatting of the date and time specified within an event can render the remaining data in the event useless the format used by the sender is not understood by the receiver beforehand. Clearly effort to ensure the accuracy, improve the detail and standardize the format of these fundamental enterprise building blocks is an imperative towards designing robust, manageable and deterministic systems. We therefore define here the "Common Base Event" as a new standard for events amongst management and business enterprise applications.

The Common Base Event described here lends itself easily to several types of events, in particular: logging, tracing, management and business events. It should not be surprising to note that in all these cases there is a significant need for the data elements and the format of those elements to be consistent, as all of these events need to be correlated with each other. Whether through log files, or through published events to listed subscribers, most products generate data whose interpretation requires the availability of contextual information. Yet this context is frequently maintained only in the minds of developers and analysts intimately familiar with the application generating the event, which makes its interpretation hazardous by any outside application responsible for handling the event, or for systems management, or problem determination purposes. Consider again the basic problem in parsing time stamps. Format and granularity (is it in milliseconds vs. microseconds?) both present needless obfuscation for the receiving application. A more general problem is the lack of consistency in the information presented. What is the hostname of the machine the event occurred on? What physical component failed? Is the component that failed on the same physical machine as the application reporting it? Are there multiple events that should be interpreted as a whole and in a certain order? Obviously, the current lack of standardization creates considerable difficulty for handling situations through automated means. Complexity further increases when the problem occurs in a solution composed of multiple components. Without a standard, data stored as logs or published as events are of little value to autonomic management, or business systems that rely on the completeness and accuracy of data to determine an appropriate course of action to take in response. To alleviate this problem, the Common Base Event definition ensures completeness of the data by providing properties to publish the following general information:

1. The identification of the component that is *reporting* the situation
2. The identification of the component that is *affected* by the situation (which may be the same as the component reporting the situation)
3. The *situation* itself

All properties defined in this model apply to one of these three broad categories, hereafter referred to as 3 tuple, and are described in detail later in this document. In addition, the locale of these components is considered as well. The affected component may not reside on the same physical machine as the application reporting it. This broader scope of information encapsulates enough data such that events can be exchanged and interpreted in a deterministic and appropriate manner across multiple application types without losing fidelity due to serial hops amongst multiple applications.

1.2.1 The scope of this work

One final introductory comment is necessary regarding scope. The goal of this work is to provide more than just an element definition for a common event. In addition, a XML schema definition. Special consideration has been given to how this model would interact with SOAP and AXIS as well as how these properties map to a number of existing event models such as Apache, CIM, JMX, SNMP and others. It is not within the scope of this document to describe how an application is to process this data, only how to format the data and how it is to be sent and received.

The following sections describe in more detail the 3-tuple described above, in the overview section, and the property data that must be collected when a situation occurs. Collectively, the properties include both the data associated with the situation, called situational data, as well as the component identifier.

1.3 Common Base Event and Situation Data

As stated above, when a situation occurs, a 3-tuple must be reported: the identification of the component that is reporting the situation, the identification of the component that is experiencing the situation (may be the same as the component reporting the situation), and the situation itself.

The following table provides a summary of the Common Base Event properties. The entries in the table represent the data that is collected for the 3-tuple. The reporter and the affected component IDs are clearly defined (i.e., reporterComponentId and sourceComponentId) in the table. The other fields in the table make up the 'situation' data.

There are several conventions used in this model. They are:

1.3.1 Notational Convention

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC-2119[1].

1.3.2 Formatted data and String Values

Formatted data is always in human readable form. All formatted data and string values MUST be encoded as UTF-8 string.

1.3.3 Data Types

The Common Base Event will only support the following subset of XML schema data types and the array variation of these types. Description of these data types can be found in the XML Schema specification[2]. The data types are XML schema signed data types.

- byte
- short
- int

[1] RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt

[2] W3C XML Schema Part 2: Datatypes, W3C Recommendation 02 May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

- long
- float
- double
- string
- dateTime
- hexBinary
- boolean

1.3.4 String Length

The maximum string length MUST never exceed 1024 characters. If a longer string is needed then the user MAY choose a hexBinary type.

Based on empirical data concerning known consumers and constraints placed on the processing and storage of events, field lengths have been specified to ensure the broadest possible acceptance and fidelity. However, it is possible that at times a sender may exceed these length restrictions. In this case, it is incumbent upon the consumer of an event to take appropriate actions to either accept the event as-is, massage it to conform to the specification, or discard as invalid. However, in general, preservation of data – even data that is out of conformance with this event specification – is preferable to a total loss of the event. Therefore, we recommend that all efforts be made to preserve as much of the event data as possible.

1.3.5 Case Sensitivity

All names and stings specified in this document are case sensitive.

1.4 Extensibility

The Common Base Event data model described here is built with some level of extensibility to lend itself easily to the needs of a variety of types of events generated by different IT and Business products. In all these cases there is a significant need for the data elements and the format of those elements not only to be consistent, as all of these events need to be correlated with each other, but also flexible to allow for adaptation to product specific requirements.

To include any product specific attributes it is recommended to use the ExtendedDataElement (described later on page 24) that allows for user supplied extensions where you can provide any other attributes not accounted for in the CommonBaseEvent.

In addition, you may include a product specific schema in the CommonBaseEvent XML Schema (described on page 33) to extend it with the product specific requirements in the “any namespace” of the schema. The event consumers or management tools that support the CommonBaseEvent schema should store and forward the part of the extended schema that they do not recognize or have support for. However, there is no implied guarantee that the extended elements are saved and forwarded by the consumer of the event that does not recognize the extension.

Furthermore, the XML Schema provides versioning by appending a version number to the XML schema file name. The current XML file name is therefore, commonbaseevent1_0.xsd. Also, the Common Base Event schema provides "version" token in the schema declaration (CommonBaseEvent XML Schema on page 33).

1.5 CommonBaseEvent Description

The following table provides a summary of the CommonBaseEvent properties; the data collected for the situation 3-tuple. A detailed description of the CommonBaseEvent follows the summary table.

Property Name	Type	Description
localInstanceId	xs:string	A source supplied event identifier. There is no guarantee that this value is globally unique. This is an OPTIONAL property. The maximum string length for localInstanceId MUST NOT exceed 128 characters.
globalInstanceId	xs:ID	The primary identifier for the event. This property MUST be globally unique and MAY be used as the primary key for the event. This is an OPTIONAL property. However, once this value is set it MUST never be changed. The recommend value is either a 128 bit or 256 bit Globally Unique Id (represented as hex string).
creationTime	xs:dateTime	The date-time when the event was issued. The value MUST be as defined by the XML Schema dateTime data type. The value of the creationTime MUST provide granularity as precisely as the generating platform allows. This is a REQUIRED property.
Severity	xs:short	The severity of the status the event is describing with respect to the application that reports the event. The predefined severity levels, in order of increasing severity, are as follows: • **0 Unknown** • **10 Information** MUST be used for cases when the event only contains general information and is not reporting an error.

		• **20 Harmless** MUST be used for cases when the error event has no effect on the normal operation of the resource. • **30 Warning** MUST be used when it is appropriate to let the user decide if an action is needed in response to the event. • **40 Minor** MUST be used to indicate that action is needed, but the situation is not serious at this time. • **50 Critical** MUST be used to indicate that an immediate action is needed and the scope is broad (perhaps an imminent outage to a critical resource will result). • **60 Fatal** MUSTbe used to indicate that an error occurred, but it is too late to take remedial action. The values are 0 to 70. The reserved values start at 0 for Unknown and increase by increments of 10 to 60 for Fatal. Other severities MAY be added but MUST NOT exceed 70. This is an OPTIONAL property.
Priority	xs:short	Defines the importance of the event. The predefined priorities are: • **10 Low** • **50 Medium** • **70 High** The values are 0 to 100. The reserved value for Low is 10, for Medium is 50, and for High is 70. Other priorities MAY be added but MUST NOT exceed 100. This is an OPTIONAL property.
reporterComponentId	cbe:ComponentIdentification	Identification of the component that is the "reporter" of the event or the situation. It is a REQUIRED property if the reporting component is different than the source component. Otherwise this field MUST be omitted.
sourceComponentId	cbe:ComponentIdentification	Identification of the component that is "affected" or was "impacted" by the event or the situation. This is a REQUIRED property for the component that is affected by the situation.

situationType	xs:string	The situationType specifies the type of the situation that caused the event to be reported. This is an extensible string value. Proposed reserved keywords are: • START • STOP • FEATURE • DEPENDENCY • REQUEST • CONFIGURE • CONNECT • CREATE • UNKNOWN This is an OPTIONAL property. The maximum string length for situationType MUST NOT exceed 512 characters.
contextDataElements	cbe:ContextDataElement[]	An array of contexts that this event is referencing. See the ContextDataElement definition (described on page 26) for details. This is an OPTIONAL property.
msgDataElement	cbe:MsgDataElement	Identification of the message that this event holds. See the MsgDataElement definition (described on page 29) This is an OPTIONAL property.
msg	xs:string	The text accompanying the event. This is typically the resolved message string in human readable format rendered for a specific locale. This is and OPTIONAL property. The maximum string length for msg MUST NOT exceed 1024 characters.
repeatCount	xs:short	The number of occurrences of a given event for a specific time interval. This is an OPTIONAL property with no default. A value of 0 or no value is indicative of no repeat of the event detected.
elapsedTime	xs:long	This is the time interval or the elapsed time for the number of occurrences of a given event type that is specified by the repeatCount property. This value indicates the duration of the time within which the repeated events were observed. This property is expressed in microseconds. This is an OPTIONAL property with no

		default value. However, if repeatCount is specified you MUST specify a value for elapsedTime.
associatedEvents	cbe:AssociatedEvent[]	This property allows for the grouping of events. This property is a complex type that is made up of globalInstanceIds identifying the associated events along with a type field describing the type of association that is represented by the name of the association. Reserved keyword values include: • Contain • CausedBy • Cleared • MultiPart • Correlated This is an OPTIONAL property.
extensionName	xs:Name	The name of an event class (or element in XML) that this event represents (e.g., CommonBaseEvent). The name is indicative of any additional elements expected to be present within the event. This is a OPTIONAL property. If the value specified is null, then its value is assumed to be "CommonBaseEvent". The maximum string length for extensionName MUST NOT exceed 64 characters.
extendedDataElements	cbe:ExtendedDataElement[]	An array of product specific extensions for extensibility where you can provide any other attributes not accounted for in the CommonBaseEvent. Information placed here is assumed to be product specific data. This is an OPTIONAL property.
sequenceNumber	xs:long	A source defined number that allows for multiple messages to be sent and processed in logical order different from the order in which they arrived at consumer location (e.g., an event server or management tools). The sequence number helps consumers to sort arrived messages. This is with respect to the creation time and to the particular reporter of the messages. This is an OPTIONAL property. There is no default value for this property.

Table 1: CommonBaseEvent

Detailed description of CommonBaseEvent is described in the following sections:

1.5.1 localInstanceId

The localInstanceId is of type string and is used to locally identify instances of an event. There is no implied guarantee that this value is globally unique. However, once it is set it stays constant for the life of the event. The value content of the localInstanceId MAY be a multi-part value, such a timestamp, location, offset, message ID or MAY use other application-defined techniques for providing the content to ensure the uniqueness of the ID. For example, you may set the identifier to the concatenation of a string giving the local host IP address, a string providing the absolute path of the access.log file, a string giving the local fully qualified host name, a string giving the time stamp, and a string representing the sequenceNumber. The resulting String will look as follows:

```
9.27.11.27mycomputer.toronto.ibm.com2002100902534.002000-240
```

This property is not a key. This is an OPTIONAL property that is not mutable, i.e., once it is set it cannot be changed. It MAY be provided by the component issuing the event or assigned by the consumer of the event. The maximum string length for the localInstanceId MUST NOT exceed 128 characters.

1.5.2 globalInstanceId

The globalInstanceId is a complex data type that represents the primary identifier for the event. The property globally and uniquely identifies the event and MAY be used as the primary key for the event. The value MUST be a Globally Unique Id (GUID) that is at least 128 bits in length but not greater than 256 bits. It is incumbent upon the GUID generation algorithm to insure the uniqueness of this value.

One possible standard for constructing a GUID is defined in the Internet draft draft-leach-uuids-guids-01. This value is computed based on using cryptographic quality random information.

This is an OPTIONAL property, however when it is specified it is not mutable, i.e., once it is set, it cannot be changed for the life of the event. The globalInstanceId may be provided either by the component issuing the event or by the consumer of the event.

The globalInstanceId is required to establish associations between events. If globalInstanceId is not specified the association, as described by AssociatedEvent (described on page 26~~29~~), cannot be provided.

1.5.3 creationTime

The date and time the event was created that MUST be specified as defined by the XML Schema dateTime data type[3]. The value of the creationTime MUST provide granularity as precisely as the generating platform allows.

[3] W3C XML Schema Part 2: Datatypes, W3C Recommendation 02 May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

This is a REQUIRED property, which is not mutable and MUST be provided by the component reporting the event. When it is set it cannot be changed for the life of the event.

1.5.4 severity

The severity indicates the severity level of the status the event is describing with respect to the application that reports the event. The meanings of the values the property may contain may be described by an enumeration of common values or qualifiers indicating the severity level of the event. For example, information, warning, or some integers mapping into the intended severity levels are all valid values. This document does not imply any specific implementation, but instead suggests the following values based on prior art with the understanding that users of this field may add additional values as their implementations dictate. This field simply helps define the seriousness of the kind of situation that was encountered and helps to enable administrators to focus on the most severe problems currently occurring in the enterprise.

The predefined severity levels, in order of increasing severity, are as follows:

- **0** **Unknown**
- **10** **Information** MUST be used when the event only contains general information and is not reporting a error.
- **20** **Harmless** SMUST be used for cases in which the error has no effect on the normal operation of the resource.
- **30** **Warning** MUST be used when it is appropriate to let the user decide if an action is needed in response to the event.
- **40** **Minor** MUST be used to indicate that action is needed, but the situation is not serious at this time.
- **50** **Critical** MUST be used to indicate that an immediate action is needed and the scope is broad (perhaps an imminent outage to a critical resource will result).
- **60** **Fatal** MUST be used to indicate an error occurred, but it is too late to take remedial action.

The values are 0 to 70. The reserved values start at 0 for Unknown and increase by increments of 10 to 60 for Fatal. Other severities MAY be added but MUST NOT exceed 70. If no value is specified, then this event is interpreted as having no severity.

This is an OPTIONAL property and it is not mutable once it is set. There is no default value for severity.

1.5.5 priority

The priority defines the importance of the event and the relative order in which the records should be processed. The predefined priorities are as follows:

- **10** **Low** - indicative of an event that does not need to be processed immediately.
- **50** **Medium** - indicative of an event of average importance.
- **70** **High** - indicative of an important event that requires attention.

The values are 0 to 100. The reserved value for Low is 10, for Medium is 50, and for High is 70. Other priorities MAY be added but MUST NOT exceed 100.

If no value is specified, then this event is interpreted as having no priority.

The priority property is separate and apart from severity, in that priority is intended to be more of a consumer driven property, where severity dictates the state of the situation as perceived by the affected component. For example, an event with priority HIGH and severity MINOR should be processed before an event with priority LOW and severity CRITICAL.

This is an OPTIONAL property and it is mutable. There is no default value for the priority.

1.5.6 reporterComponentId

The reporterComponentId is the identification of the component that reported the event or situation on behalf of the affected component. The data type for this property is a complex type as described by the ComponentIdentification type that provides the required data for uniquely identifying a component.

It is a REQUIRED property if the reporting component is different from the source component. Otherwise, this field MUST be omitted. This property is not mutable, that is, once it is set, it cannot be changed.

1.5.7 sourceComponentId

The sourceComponentId is the identification of the component that was *affected* or was impacted by the event or situation. The data type for this property is a complex type as described by the ComponentIdentification type that provides the required data for uniquely identifying a component.

This property is REQUIRED and it is not mutable. The producer of the event MUST provide the sourceComponentId. If the reporter and the affected components are the same then the reporterComponentId MUST be omitted.

1.5.8 situationType

The situationType specifies the type of the situation that caused the event to be reported. This is an extensible string value.
Proposed reserved keywords are:

- FEATURE
- DEPENDENCY
- REQUEST
- CONFIGURE
- CONNECT
- CREATE
- UNKNOWN

This is an OPTIONAL property, which it is not mutable.. The maximum string length for situationType MUST NOT exceed 512 characters.

1.5.9 contextDataElements

The contextDataElements is an array of contexts of type ContextDataElement (described on page 24~~28~~) that this event is referencing. This property holds data that is used to assist with correlating messages or events generated along the execution path of a unit of work for problem diagnostics.

This is an OPTIONAL property and mutable. It may be provided by the component issuing the event or may be assigned by the consumer of the event.

1.5.10 msg

The msg property is the text that accompanies the event. This is typically the resolved message string in human readable format rendered for a specific locale.

The locale of the msg property is specified by the msgLocale property of the MsgDataElement type (described on page 28~~32~~). There is no default value for the msg locale

This property is OPTIONAL but RECOMMENDED to have a value if the msgCatalogId and msgCatalog properties of the MsgDataElement do not specify a value.

1.5.11 msgDataElement

The msgDataElement is a property that is referencing a MsgDataElement. This property holds data that is used to specify all the related information associated with the message that this event holds.

This is an OPTIONAL property and non-mutable. It is provided by the component issuing the event.

1.5.12 extensionName

The extensionName property contains the name of an “event class” that this event represents (e.g., Trace, CommonBaseEvent). The event class name is indicative of any additional properties expected to be present within the specific event. If you choose to use ExtendedDataElement, which is described in the next section, it is recommended you specify a value for the extensionName.

This is an OPTIONAL property, which is not mutable and may only be provided by the component reporting the event. If the value specified is null, then the value is assumed to be “CommonBaseEvent”.

1.5.13 extendedDataElements

The extendedDataElements property is a sequence of name elements of type ExtendedDataElement (described on page 22~~29~~). It is to be used for extensibility by providing a location to specify any other attributes not accounted for in the CommonBaseEvent data model. Information placed here is assumed to be product specific data.

This property is user supplied and named elements can be filtered, searched on, or referenced by the correlation rules.

This is an OPTIONAL property that is mutable, i.e., after it is set it may be changed. The value for this property may be provided by the component issuing the event or assigned by the consumer of the event.

1.5.14 associatedEvents

The associatedEvents property allows for but not restricted to the grouping of events or parent-child relationship. This property is a complex type that is made up of globalInstanceIds identifying the AssociatedEvent's (described on page 27).
This is an OPTIONAL property that is mutable, i.e., after it is set it may be changed. The value for this property may be provided by the component issuing the event or assigned by the consumer of the event.

1.5.15 repeatCount

The repeatCount is number of occurrences of a given event for a specific time interval. The time interval is indicated by the elapsedTime property described below. The definition of what makes an event a repeat of a previously issued event is application-specific and therefore not defined by this specification.

This property is OPTIONAL and mutable. The repeatCount may be set by the component issuing the event or the consumer of the event. There is no default value. A value of 0 or no value indicates no repeat of the event.

1.5.16 elapsedTime

The elapsedTime is the time interval or the elapsed time for some number of occurrences of a specific event. The number of occurrences is specified by the value of the repeatCount . This value indicates the duration of the time within which the repeated events were observed.

The value of this property MUST be expressed in microseconds granularity.

This property is OPTIONAL and mutable. However, if the repeatCount is specified then an elapsed time must be present. The elapsedTime MUST be set by the same component that sets the repeatCount. There is no default value for elapsedTime.

1.5.17 sequenceNumber

The sequenceNumber is a source-defined number that allows multiple messages to be sent and processed in a logical order different from the order in which they arrived at the consumer's location (e.g., an event server or management tools). The sequence number helps consumers to sort arrived messages. This is with respect to time and to the particular provider of the event.

This property is OPTIONAL and it is not mutable once it is set. There is no default value.

1.6 ComponentIdentification Description

In problem reporting there are two general categories of components that should be considered for problem diagnosis, the component observing and reporting the situation (reporter), and the actual component that is experiencing the situation (affected). Component Identification provides a collection of attributes required to uniquely identify a component. The same data is used to identify both the component that is reporting an event or situation and the component that is affected or experiencing the situation. In some cases, these components will be the same.

For example, in a typical IT environment, commonly, the activities of applications running in that environment are often monitored using events received or collected from the applications via management agents or adapters.

Example 1: Consider the case where a WebSphere application, called myWebApp, times out on a table query due to a DB2 server problem that is located on a remote system. The web app then issues an event indicating the failure situation. In this case, myWebApp is the “affected” or the “source” component.

Example 2: Consider a case where there is application X running on a Windows server. The application encounters an error and adds an entry to the Widows error log. Then there is a separate application (ie: an adapter) that reads messages from the error log and generates a common base event and submits it. In this case the “affected” or the “source” of the event is the application X and the reporting component is the adapter that generated and submitted the event.

The following table is a summary of the properties in the ComponentIdentification type. A detailed description of the ComponentIdentification properties follows the summary table.

Property Name	**Type**	**Description**
location	xs:string	Specifies the physical address corresponding to the location of a component. For example, hostname, IP address, or VTAM LU. The format of the value of the location is specified by the locationType property. The preferred value is a fully qualified hostname. This value should be unique within the network capable of raising events to a specified receiver. For example, if an adapter were monitoring a router, this attribute would contain the IP address of the machine where the adapter resides, not the IP address of the router that is having the problem. This is a REQUIRED property. The maximum string length for location MUST NOT exceed 256 characters.
locationType	xs:Name	Specifies the format and meaning of the value in the location property. The well-known reserved keywords for this property are: • **Unknown** • **IPV4** • **IPV6** • **NWA** • **ISDN** • **ICD** • **OID/OSI** • **Dial**

		• **HWA** • **HID** • **X25** • **DCC** • **SNA** • **IPX** • **E.164** • **Hostname** • **FQHostname** • **Devicename** This is a REQUIRED property. The maximum string length for locationType MUST NOT exceed 32 characters. The default value is “Unknown”.
application	xs:string	The name of the application (e.g, myWebApp). This is an optional property. The application version information may be appended to the end of the component separated by a # character. This is an OPTIONAL property. The maximum string length for the application name MUST NOT exceed 256 characters.
executionEnvironment	xs:string	This property identifies the immediate environment that an application is running in. For example, a WebSphere Application Server name: cell:node:server. The executionEnvironment version information may be appended to the end of the component separated by a # character. This is an OPTIONAL property. The maximum string length for executionEnvironment MUST NOT exceed 256 characters.
component	xs:string	Specifies the logical identity of a component. This property MUST contain the name of a particular application, product, or subsystem (e.g., IBM DB2# V7.1). This value SHOULD be unique within the scope specified by the location. The component version information may be appended to the end of the component separated by a # character. This is a REQUIRED property. The maximum string length for the component name MUST NOT exceed 256 characters.
subComponent	xs:string	Specifies a further delineation for the logical component property of the event. It SHOULD contain the identity of the subcomponent of the component property. This property can be one of the various parts of an application or OS resource e.g., a module name, a class name, a class and method name. It should be the most granular definition specified in the event. The subcomponent

		version information may be appended to the end of the subcomponent separated by a # character. This is a REQUIRED property. The maximum string length for the subComponent name MUST NOT exceed 512 characters.
componentIdType	xs:string	Specifies the format and meaning of the component identified by this componentIdentification. The nonexclusive reserved keywords for this property are: • ProductName • DeviceName • SystemName • ServiceName • Process • Application • Unknown This is a REQUIRED property. The maximum string length for componentIdType MUST NOT exceed 32 characters. The default value is “Unknown”.
instanceId	xs:string	Specifies a handle or identifier for the instance of the component that is specified by the component property i.e., Grid Service Handle(GSH)[4] and EJBHandle. This is an OPTIONAL property. The maximum string length for instanceId MUST NOT exceed 128 characters.
processId	xs:string	This property identifies the process ID of the running component or subcomponent that generated the event. This is an OPTIONAL property and there is no default value. The maximum string length for processId MUST NOT exceed 64 characters.
threadId	xs:string	This property identifies the thread ID of the component or subcomponent that generated the event. This value changes with every new thread spawned by the process identified by the processId. This is an OPTIONAL property and there is no default value. The maximum string length for threadId MUST NOT exceed 64 characters.

Table 1: Component Identification

A detailed description of the ComponentIdentification type is described in the following sections:

[4] See Grid Services Specification Draft 4, Global Grid Forum, http://www.gridforum.org/ogsi.wg

1.6.1 location

The location specifies the physical address corresponding to the location of a component. For example, a hostname, IP address, VTAM LU. The format of the value of the location is specified by the locationType property. The preferred value is a fully qualified hostname.

This property is REQUIRED and it is not mutable, i.e., once it is set it cannot be changed. The maximum string length for location MUST NOT exceed 256 characters.

1.6.2 locationType

This property specifies the format and meaning of the value in the location property. The well-known reserved keywords for this property are:

- **Unknown**
- **IPV4**
- **IPV6**
- **NWA**
- **ISDN**
- **ICD**
- **OID/OSI**
- **Dial**
- **HWA**
- **HID**
- **X25**
- **DCC**
- **SNA**
- **IPX**
- **E.164**
- **Hostname**
- **FQHostname**
- **Devicename**

The default value is “Unknown”. This property is REQUIRED and not mutable, i.e., once it is set it cannot be changed. The maximum string length for locationType MUST NOT exceed 32 characters.

1.6.3 application

The application property specifies the user name of application (e.g, myApp). The application version information may be appended to the end of the component separated by a # character. Also, it is recommended to prpend the vendor name to the application name.

This is an OPTIONAL property and it is not mutable, i.e., once it is set it cannot be changed. The maximum string length for the application name MUST NOT exceed 256 characters.

1.6.4 executionEnvironment

The executionEnvironment property identifies the immediate environment that an application is running in. For example, a WebSphere Application Server name: cell:node:server.

The executionEnvironment version information may be appended to the end of the component separated by a # character. The maximum string length for executionEnvironment MUST NOT exceed 256 characters.

This is an OPTIONAL property and it is not mutable. , i.e., once it is set it cannot be changed for the life of the event.

1.6.5 component

The component specifies the logical identity of a component. This property MUST contain the name of a particular application, product, or subsystem (e.g., IBM DB2 V7.1). This value SHOULD be unique within the scope specified by the reporter location.

This property is REQUIRED and it is not mutable, i.e., once it is set it cannot be changed for the life of the event. The maximum string length for the component name MUST NOT exceed 256 characters.

1.6.6 subComponent

The subComponent specifies a further delineation for the logical component property of the event.
It SHOULD contain the identity of the subcomponent of the component property and should be the most granular definition specified in the event. This property can be one of the various parts of an application or OS resource e.g., a module name, a class name, a class and method name.
This property is REQUIRED and it is not mutable, i.e., once it is set it cannot be changed for the life of the event. The maximum string length for the subComponent name MUST NOT exceed 512 characters.

1.6.7 componentIdType

The componentIdType specifies the format and meaning of the component identified by this componentIdentification. The reserved, nonexclusive list of keywords for this property are:

- ProductName - indicates that component represent a specific product.
- DeviceName - indicates that component represent a device.
- SystemName - indicates that component represent a system.
- ServiceName - indicates that component represent a service.
- Process - indicates that component represent a process.
- Application - indicates that component represent a collection of one or more components, where component equals module.component, and subcomponent equals class.method.
- Unknown

This property is REQUIRED and it is not mutable. The maximum string length for componentIdType MUST NOT exceed 32 characters.

1.6.8 instanceId

The instanceId specifies a handle or identifier for the instance of the component that is specified by the component property i.e., Grid Service Handle(GSH)[5], EJBHandle.

[5] See Grid Services Specification Draft 4, Global Grid Forum, http://www.gridforum.org/ogsi.wg

This property is OPTIONAL that is not mutable, i.e., once it is set it cannot be changed. The maximum string length for instanceId MUST NOT exceed 128 characters.

1.6.9 processId

The processId is a string type that identifies the process ID of the "running" component or subcomponent that generated the event.

This is an OPTIONAL property that is not mutable, i.e., once it is set it cannot be changed. The maximum string length for processId MUST NOT exceed 64 characters.

1.6.10 threadId

The threadId property is of type string and identifies the thread ID of the component or subcomponent indicated by the process ID that generated the event. A running process may spawn one or more threads to carry its function and/or incoming requests. Therefore, the thread ID will change accordingly.

This is an OPTIONAL property that is not mutable, i.e., once it is set it cannot be changed. The maximum string length for threadId MUST NOT exceed 64 characters.

1.7 ExtendedDataElement Description

The ExtendedDataElement allows for user-supplied name-type-value collections to be specified for extensibility purposes. This is where a user can include any other attributes not accounted for in the CommonBaseEvent data model. Collections specified here are assumed to be product specific data.

The named properties can be filtered, searched on, or referenced by the correlation rules. The "name" is user defined, however, the nonexclusive list of reserved keywords are as follows:

> **RawData** - This keyword is indicative of "as is" data and usually in a format that may be proprietary to the producer of such data. It may be in any form including binary. It is intended to allow the data to be retrieved verbatim, and to support tools that understand the format of the context.
>
> **RootHeader –** This keyword is intended to identify the root ExtendedDataElement for a hierarchy of the ExtendedDataElement's that are defined by the dataRefs.

Property Name	**Type**	**Description**
name	xs:Name	The name of the extended data element. This name MUST be unique with respect to all other fields in the event. This is a REQUIRED property.
type	xs:Name	The data type of the values specified in the values property below. Valid types are as follows: • byte, short, int, long, float, double • string

		• dateTime • byte**Array**, short**Array**, int**Array**, long**Array**, float**Array**, double**Array** • string**Array** • dateTime**Array**, duration**Array** • hexBinary • boolean, boolean**Array** The above data types are the only valid types supported by the ExtendedDataElement type. The default value is "string".
values	xs:string[]	The array of values for this extended data element as a string representation of the type specified above, excluding hexBinary. hexBinary values MUST be defined using the hexValue property. This is an OPTIONAL property.
hexValue	xs:hexBinary	The hexValue is an array of characters that holds the data for any other data type or complexType not in the supported types described above. The hexValue, value, and the values properties are mutually exclusive. Only one of these properties can be defined. This is an OPTIONAL property.
children	cbe:ExtendedData Element	Contains other extendedDataElement(s) to specify a structured list of extendedDataElements. This list allows a reporter to create a hierarchy of extendedDataElements for a specific CommonBaseEvent. This is an OPTIONAL property.

Table 2: ExtendedDataElement

The detailed description of the ExtendedDataElement is described in the following sections:

1.7.1 name

The name property specifies the name of the ExtendedDataElement. This name MUST be unique with respect to all other properties in the event (e.g., RawData, msgLocale, and EventStatus).

This property REQUIRED and it is not mutable.

1.7.2 type

The data type of the values specified in the values property below.

Valid types are as follows:

- byte, short, int, long, float, double
- string

- dateTime
- boolean
- byte**Array**, short**Array**, int**Array**, long**Array**, float**Array**, double**Array**
- string**Array**
- dateTime**Array**
- boolean**Array**
- hexBinary

The above data types are the only valid types supported by the ExtendedDataElement type.

The default value is “string”.

This property is REQUIRED and is not mutable.

1.7.3 values

An array of values for this extended data element of the type defined by the type property described above.

This property is OPTIONAL and it is mutable. It MUST NOT be specified if the “hexValue” property is specified.

1.7.4 hexValue

The hexValues property is an array of characters that holds the data for any other data type or complexType not in the supported list of types defined above.

This property is OPTIONAL and it is not mutable. It MUST NOT be specified if the “values” property is specified.

Note: The hex Value and values properties are mutually exclusive. Only one of these two properties can be defined.

1.7.5 children

The children property refers to other ExtendedDataElement(s) to specify a structured list of ExtendedDataElement’s. This list allows for the creation of a hierarchy of related ExtendedDataElement’s corresponding to a specific group of CommonBaseEvents. Accordingly, this is an efficient and quick way to get access to the list of related ExtendedDataElement’s without having to look through and examine all the ExtendedDataElement’s.

This property is OPTIONAL and it is mutable.

1.8 ContextDataElement Description

The ContextDataElement type defines the context(s) that this event is referencing. This complex type holds data that is used to assist with correlating messages or events generated along the execution path of a unit of work for problem diagnostic purposes.

The following table provides a summary of the data properties representing a context in the common base event. A detailed description of this ContextDataElement follows the summary table.

Property Name	Type	Description
type	xs:Name	The data type of the contextValue property. This is a REQUIRED property.
name	xs:Name	Name of the application that created this context data element. This is a REQUIRED property.
contextValue	xs:string	The value of the context with respect to the implementation of the context. This is not required if contextId specifies a value.
contextId	xs:IDREF	This property is the reference to the element that contains the context. This is not required if contextValue specifies a value.

Table 3: ContextDataElement

A detailed description of the contextDataElement is described in the following sections:
type
This is the data type of the context. This type should allow the consumer of the event to recognize the format of the context value. The type is application specifics (e.g., MyCorrelator).

This property is REQUIRED and not mutable.

1.8.1 name

This is the name of the application that created this context data element (e.g., Correlation engine).

This property is REQUIRED and not mutable.

1.8.2 contextValue

This is the value of the context with respect to the implementation of the context.

This property is not required if contextId specifies a value and it is not mutable.

1.8.3 contextId

This property is the reference to the element that contains a product/user specific context.

This property is not required if contextValue specifies a value; and it is not mutable. It MUST NOT be specified if the contextValue property is specified.

NOTE: The contextValue and the contextId are mutually exclusive, only one of these two properties can be defined. However, if contextValue is set to a value the contextId is ignored.

1.9 AssociatedEvent Description

The AssociatedEvent type allows for the grouping of events. It allows identifying associated events and its associationEngine which is the reference to the application that created the association.

The following table provides a summary of the data properties representing an event association in the common base event.

Property Name	Type	Description
associationEngine	xs:IDREF	Reference to the AssocationEngine that created this AssociatedEvent. This is a REQUIRED property.
resolvedEvents	xs:IDREFS	Array of globalInstanceIds corresponding to the events that are associated with this event. This is a REQUIRED property.

Table 4: AssociatedEvent

The detailed description for the AssociatedEvent is described in the following sections:

1.9.1 associationEngine

This is a reference to the AssocationEngine that identifies the application that created this AssociatedEvent.

This property is REQUIRED and not mutable once it is set.

1.9.2 resolvedEvents

An array of globalInstanceIds corresponding to the events that are associated with this event.

This is a REQUIRED property that is mutable and is provided by the association engine specified by the name property.

1.10 AssociationEngine Description

The AssociationEngine identifies the application that establishes association among related or associated events. In addition, it provides properties to describe the type of the association.
The AssociationEngine is a standalone entity in the XML schema and the AssociatedEvents created by the application that is identified by the AssociationEngine refer to it. This will eliminate the need to repeat the same data in every associated event.

The following table provides a summary of the properties representing an “association” source in the common base event.

Property Name	Type	Description

name	xs:Name	Name of the application that creates the association (e.g,. my correlation engine name). This is a REQUIRED property.
type	xs:Name	This property should contain the type of association created by this AssociationEngine. Some well defined associations are: • Contains • Cleared • CausedBy • MultiPart • Correlated This is a REQUIRED property.
id	xs:ID	The primary identifier for the element. This property MUST be globally unique. The recommend value for this is either a 128 bit or 256 bit Globally Unique Id (represented as hex string). Once this value is set it MUST never be changed. This is a REQUIRED property.

Table 4: AssociationEngine

The detailed description for the AssociationEngine is described in the following sections:

1.10.1 name

Name of the application that will create the association (e.g,. my correlation engine name).

This property is REQUIRED and not mutable once it is set.

1.10.2 type

This property should contain the type of association created by this AssociationEngine.
Some well-defined associations are:
Reserved keyword values include:

- **Contain** : When the association represents containment of other events within a root event
- **CausedBy** : When the association represents a causality allowing the associated event to point to the cause of the situation.
- **Cleared** : When the association represents a relationship where an event points to an event which fixes, or results in the situation to becoming irrelevant.
- **MultiPart** : When the association represents a collection of events together comprise a single event.
- **Correlated** : When the association represents a relationship between a child and parent event based on a correlation algorithm specified in the name of the association.

This property is REQUIRED and not mutable when it is set for a specific “name” property.

1.10.3 id

The primary identifier for the element. This property MUST be globally unique. The recommend value for this is either a 128 bit or 256 bit Globally Unique Id (represented as hex string). Once this value is set it MUST never be changed.

This is a REQUIRED property that is not mutable.

1.11 MsgDataElement Description

This MsgDataElement represents the data that is used to specify all the related information associated with the message that this event holds.

The following table provides a summary of the data properties representing a message in the common base event.

Property Name	Type	Description
msgId	xs:string	Specifies the message identifier of the event. This identifier is commonly a unique value string of alphanumeric or numeric characters. It can be as simple as a string of numeric characters identifying a message in a message catalog or a multi-part string of alphanumeric characters (e.g., DBT1234E). This is a OPTIONAL property. The maximum string length for msgId MUST NOT exceed 256 characters.
msgIdType	xs:Name	Specifies the meaning and format of the msgId. If the ID conforms to or represents a standard or a well-known convention, it is named by this property. Examples are: IBM3.4, IBM4.4, IBM3.1.4, IBM3.4.1, IBM4.4.1, and IBM3.1.4.1. The current nonexclusive list of reserved keywords include: • IBM* (* is as described above) • JMX • DottedName • Unknown This is a OPTIONAL property. The maximum string length for msgId MUST NOT exceed 32 characters.
msgCatalogId	xs:string	The index or the identifier for a message that is

		used for resolving the message text from a message catalog. This is an OPTIONAL property.
msgCatalogTokens	string[]	An array of strings used as substitution values for resolving a NLS based message into formatted text. The order of the substitution values is implied by the implicit order of the array elements. If there are no substitution values, then it does not need to be specified. This is an OPTIONAL property. The maximum string length of the msgCatalogTokens property MUST NOT exceed 256 characters per token.
msgCatalog	xs:string	The qualified name of the message catalog that contains the translated message specified by the msgCatalogId. This is an OPTIONAL property. The maximum string length of the msgCatalog MUST NOT exceed 128 characters.
msgCatalogType	xs:Name	The msgCatalogType property specifies the meaning and format of the msgCatalog. The current nonexclusive list of reserved keywords includes: • Java • XPG This property is OPTIONAL and it is not mutable once it is set . The maximum string length for the msgCatalogType property MUST NOT exceed 32 characters.
msgLocale	xs:language	The locale by which this msg property is rendered. Its value is a locale code in conformance with IETF RFC 1766. This is an OPTIONAL property.

Table 5: MsgDataElement

The detailed description of the MsgDataElement is described in the following sections:

1.11.1 msgId

The msgId property specifies the message identifier for the event. This identifier is commonly a unique value string of alphanumeric or numeric characters. It can be as simple as string of numeric characters identifying a message in a message catalog or a multi-part string of alphanumeric characters (e.g., DBT1234E). The format of the content of the msgId is specified by the msgIdType property as described in the next section.

This is a OPTIONAL property, which is not mutable once it is set and should be provided by the component issuing the event. The maximum string length for the msgId property MUST NOT exceed 256 characters.

1.11.2 msgIdType

The msgIdType property specifies the meaning and format of the msgId. If the ID conforms to, or represents a standard or a well-known convention, it is named by this property. For example IBM3.4.1 implies a message ID of a 3 part, 8-character string identifier, consisting of 3 alphabetic characters representing a component, followed by 4 numeric characters, and suffixed with one alphabetic character (e.g., DBT2359I). Other similar reserved keywords are IBM3.4, IBM4.4, IBM3.1.4, IBM3.4.1, IBM4.4.1, and IBM3.1.4.1.

The current nonexclusive list of reserved keywords includes:

IBM* (* is as described above)
JMX
DottedName
Unknown

This is a OPTIONAL property that is not mutable once it is set and should be provided by the component issuing the event. It must be provided if msgId property is specified. The maximum string length for the msgIdType property MUST NOT exceed 32 characters.

1.11.3 msgLocale

The msgLocale property specifies the locale that the msg is rendered in. Its value is a locale code in conformance with the IETF RFC 1766 specifications. For example, en-US is the value for United State English.

This property is OPTIONAL and it is not mutable once it is set. If msgLocale is not specified then it is up to the consumer of the event to decide the locale.

The maximum string length per msgLocale MUST NOT exceed 5 characters.

1.11.4 msgCatalogTokens

The msgCatalogTokens property consists of an array of string values that is for holding substitution data used for resolving a NLS based message into a fully formatted text. The order of the values is implied by the implicit order of the array elements. The Locale of the tokens should be the same as the locale of the message text as defined by msgLocale.

This property is OPTIONAL and it is not mutable once it is set. If there are no substitution values, then this property does not need to be specified. The maximum string length of the msgCatalogTokens property MUST NOT exceed 256 characters per token.

1.11.5 msgCatalogId

The msgCatalogId property is the index or the identifier for a message that is used for resolving the message text from a message catalog.

This property is OPTIONAL and it is not mutable once it is set.

1.11.6 msgCatalog

The msgCatalog property is the qualified name of the message catalog that contains the translated message specified by the msgCatalogId.

This property is OPTIONAL and it is not mutable once it is set. The maximum string length for the msgCatalog property MUST NOT exceed 128 characters.

1.11.7 msgCatalogType

The msgCatalogType property specifies the meaning and format of the msgCatalog. The current nonexclusive list of reserved keywords includes:

- Java
- XPG

This property is OPTIONAL and it is not mutable once it is set and MUST be provided if msgCatalog property is defined. The maximum string length for the msgCatalogType property MUST NOT exceed 32 characters.

CommonBaseEvent XML schema

The following XML Schema is a document that describes the element and the attribute declarations for the Common Base Event data model. This schema must be used to verify that the event XML document is valid according to the defined set of rules.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:cbe="http://www.ibm.com/AC/commonbaseevent1_0"
targetNamespace="http://www.ibm.com/AC/commonbaseevent1_0" version="1.0">
	<xsd:complexType name="CommonBaseEventType">
		<xsd:sequence>
			<xsd:element name="contextDataElements" type="cbe:ContextDataElementType"
minOccurs="0" maxOccurs="unbounded" />
			<xsd:element name="extendedDataElements" type="cbe:ExtendedDataElementType"
minOccurs="0" maxOccurs="unbounded" />
			<xsd:element name="associatedEvents" type="cbe:AssociatedEventType"
minOccurs="0" maxOccurs="unbounded" />
			<xsd:element name="reporterComponentId" type="cbe:ComponentIdentificationType"
minOccurs="0" maxOccurs="1" />
			<xsd:element name="sourceComponentId" type="cbe:ComponentIdentificationType"
minOccurs="1" maxOccurs="1" />
			<xsd:element name="msgDataElement" type="cbe:MsgDataElementType"
minOccurs="0" maxOccurs="1" />
			<xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:attribute name="globalInstanceId" type="xsd:ID" use="optional" />
		<xsd:attribute name="extensionName" use="optional">
			<xsd:simpleType>
				<xsd:restriction base="xsd:Name">
					<xsd:maxLength value="64" />
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
		<xsd:attribute name="localInstanceId" use="optional">
			<xsd:simpleType>
				<xsd:restriction base="xsd:string">
					<xsd:maxLength value="128" />
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
		<xsd:attribute name="creationTime" type="xsd:dateTime" use="required" />
		<xsd:attribute name="severity"  use="optional" >
			<xsd:simpleType>
				<xsd:restriction base="xsd:short">
				   <xsd:minInclusive value="0" />
					<xsd:maxInclusive value="70" />
				</xsd:restriction>
			</xsd:simpleType>
```

```
        </xsd:attribute>
        <xsd:attribute name="msg" use="optional" >
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="1024" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="priority" use="optional">
            <xsd:simpleType>
                <xsd:restriction base="xsd:short">
                  <xsd:minInclusive value="0" />
                    <xsd:maxInclusive value="100" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="sequenceNumber" type="xsd:long" use="optional" />
        <xsd:attribute name="situationType" use="optional">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="512" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="repeatCount" type="xsd:int" use="optional" />
        <xsd:attribute name="elapsedTime" type="xsd:long" use="optional" />

    </xsd:complexType>
    <xsd:element name="CommonBaseEvent" type="cbe:CommonBaseEventType"  />

    <xsd:complexType name="ComponentIdentificationType">
        <xsd:attribute name="component" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="256" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="subComponent" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="512" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="componentIdType" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="32" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="instanceId" use="optional" >
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="128" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="application" use="optional">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="256" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="executionEnvironment" use="optional" >
            <xsd:simpleType>In general, it is 1pm
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="256" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="location" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="256" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="locationType" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:Name">
                    <xsd:maxLength value="32" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="processId" use="optional" >
        <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="64" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="threadId" use="optional" >
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="64" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
```

```
	<xsd:complexType name="MsgDataElementType">
		<xsd:sequence>
			<xsd:element name="msgCatalogTokens" minOccurs="0" maxOccurs="unbounded">
				<xsd:complexType>
					<xsd:attribute name="value" use="required" >
						<xsd:simpleType>
							<xsd:restriction base="xsd:string">
							<xsd:maxLength value="256" />
							</xsd:restriction>
						</xsd:simpleType>
					</xsd:attribute>
				</xsd:complexType>
			</xsd:element>
			<xsd:group ref="cbe:msgIdGroup" minOccurs="0" maxOccurs="1"/>
			<xsd:group ref="cbe:msgCatalogGroup" minOccurs="0" maxOccurs="1"/>
		</xsd:sequence>
		<xsd:attribute name="msgLocale" type="xsd:language" use="optional">
		</xsd:attribute>
	</xsd:complexType>

	<xsd:group name="msgCatalogGroup">
		<xsd:sequence>
			<xsd:element name="msgCatalogId" type="xsd:string" minOccurs="1" maxOccurs="1"
/>
			<xsd:element name="msgCatalogType" minOccurs="1" maxOccurs="1" >
			<xsd:simpleType>
					<xsd:restriction base="xsd:string">
						<xsd:maxLength value="32" />
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:element>
			<xsd:element name="msgCatalog" minOccurs="1" maxOccurs="1">
				<xsd:simpleType>
					<xsd:restriction base="xsd:string">
						<xsd:maxLength value="128" />
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:element>
		</xsd:sequence>
	</xsd:group>

	<xsd:group name="msgIdGroup">
		<xsd:sequence>
		<xsd:element name="msgId" minOccurs="1" maxOccurs="1" >
			<xsd:simpleType>
				<xsd:restriction base="xsd:string">
					<xsd:maxLength value="256" />
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:element>
```

```
        <xsd:element name="msgIdType" minOccurs="1" maxOccurs="1">
            <xsd:simpleType>
                <xsd:restriction base="xsd:Name">
                    <xsd:maxLength value="32" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        </xsd:sequence>
    </xsd:group>

    <xsd:complexType name="AssociatedEventType">
        <!-- This association would contain a serialized version of the GloballyUniqueId for all the
resolvedAssociatedEvent objects -->
        <xsd:attribute name="associationEngine" type="xsd:IDREF" use="required" />
        <xsd:attribute name="resolvedEvents" type="xsd:IDREFS" use="required" />
    </xsd:complexType>

    <xsd:complexType name="AssociationEngineType">
       <!-- This id would contain a serialized version of the GloballyUniqueId for all the
resolvedAssociatedEvent objects -->
       <xsd:attribute name="id" type="xsd:ID" use="required" />
        <xsd:attribute name="type" type="xsd:Name" use="required" />
        <xsd:attribute name="name" type="xsd:Name" use="required" />
    </xsd:complexType>
    <xsd:element name="AssociationEngine" type="cbe:AssociationEngineType" />


    <xsd:complexType name="ExtendedDataElementType">
        <xsd:choice>
            <xsd:element name="values" type="xsd:string" minOccurs="0"
maxOccurs="unbounded" />
            <xsd:element name="hexValue" type="xsd:hexBinary" minOccurs="0" maxOccurs="1"
/>
            <xsd:element name="children" type="cbe:ExtendedDataElementType" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:choice>
        <xsd:attribute name="name" type="xsd:Name" use="required" />
        <xsd:attribute name="type" type="xsd:Name" use="required"  />
    </xsd:complexType>

    <xsd:complexType name="ContextDataElementType">
        <xsd:choice>
            <xsd:element name="contextValue" type="xsd:string" minOccurs="0" maxOccurs="1"
/>
            <xsd:element name="contextId" type="xsd:IDREF" minOccurs="0" maxOccurs="1" />

        </xsd:choice>
        <xsd:attribute name="name" type="xsd:Name" use="required" />
        <xsd:attribute name="type" type="xsd:Name" use="required" />
    </xsd:complexType>
```

```
        <xsd:complexType name="CommonBaseEventsType">
                <xsd:sequence>
                        <xsd:element ref="cbe:AssociationEngine" minOccurs="0" maxOccurs="unbounded" />
                        <xsd:element ref="cbe:CommonBaseEvent" minOccurs="0" maxOccurs="unbounded"
/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="CommonBaseEvents" type="cbe:CommonBaseEventsType" />

</xsd:schema>
```

1.12 Sample of CommonBaseEvents as they may be accommodated in an XML document

1.12.1 Sample 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommonBaseEvent     creationTime="2001-12-31T12:00:00"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:noNamespaceSchemaLocation="commonbaseevent1_0.xsd"
                     globalInstanceId="i0000000000000000000000000000000"
                     msg="Hello World!">
          <sourceComponentId component="myComponent"
                              componentIdType="myComponentIdType"
                              location="myLocation"
                              locationType="myLocationType"
                              subComponent="mySubComponent"/>
</CommonBaseEvent>
```

1.12.2 Sample 2:

```
<?xml version="1.0" encoding="UTF-8"?>
<AssociationEngine    id=" i0000000000000000000000000000044"
                      name="myassociationEngineName"
                      type="Correlated"
                      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xsi:noNamespaceSchemaLocation="commonbaseevent1_0.xsd"/>
```

1.12.3 Sample 3:

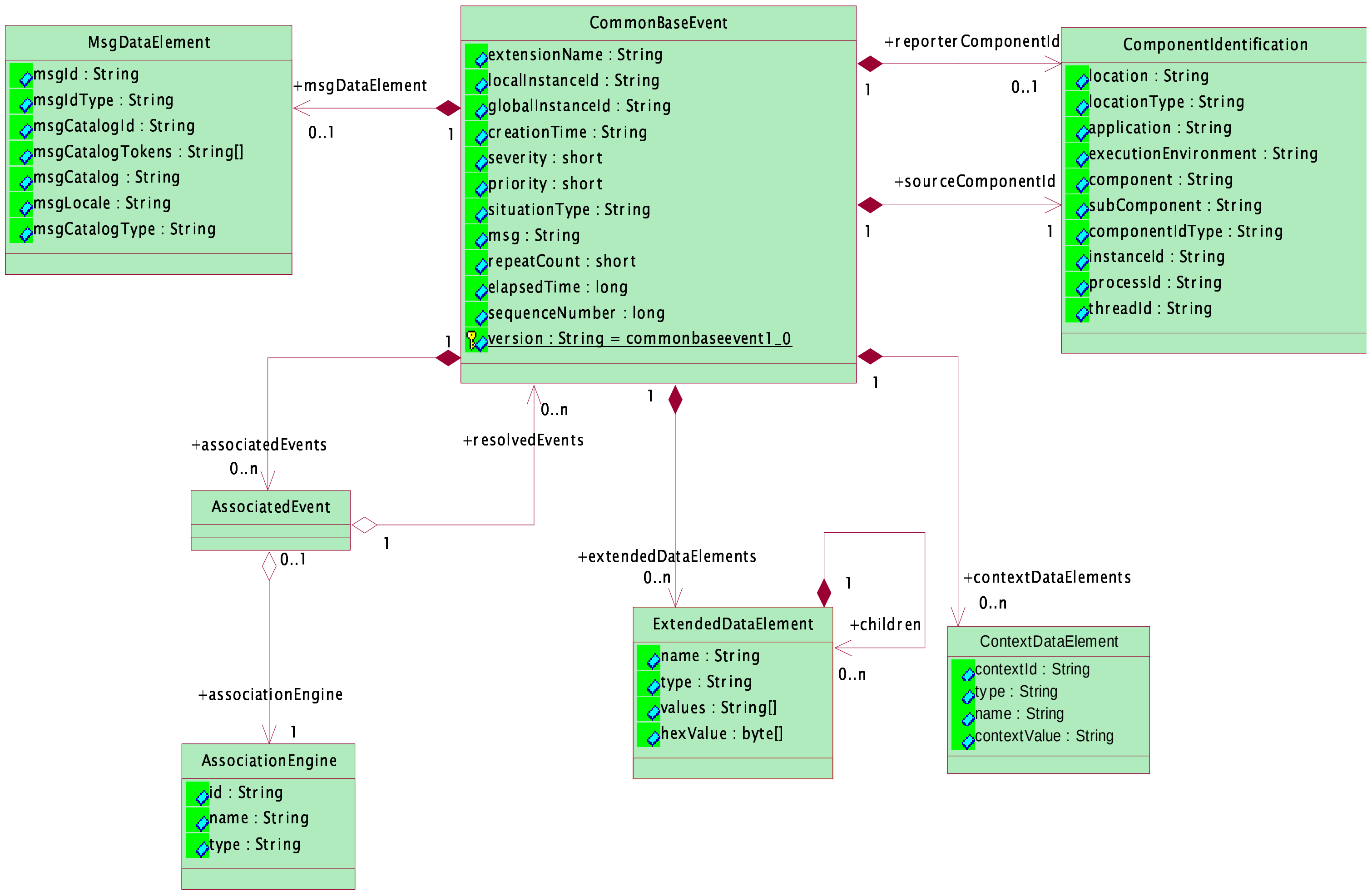
```
<?xml version="1.0" encoding="UTF-8"?>
<CommonBaseEvents                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                      xsi:noNamespaceSchemaLocation="commonbaseevent1_0.xsd">
      <AssociationEngine         id="a0000000000000000000000000000000"
                                           name="myassociationEngineName"
                                           type="Correlated" />
      <CommonBaseEvent                creationTime="2001-12-31T12:00:00"
                                           elapsedTime="0"
                                           extensionName="CommonBaseEvent"
                                           globalInstanceId="i0000000000000000000000000000000"
                                           localInstanceId="myLocalInstanceId"
                                           priority="0"
                                           repeatCount="0"
                                           sequenceNumber="0"
                                           severity="0"
```

```
                                                        situationType="mySituation">
        <contextDataElements      name="Name"
                                                type="myContextType">
                                                        <contextValue>contextValue</contextValue>
        </contextDataElements>
        <extendedDataElements     name="z"
                                                        type="Integer">
                                                        <values>1</values>
        </extendedDataElements>
        <associatedEvents                 associationEngine="a0000000000000000000000000000000"
                                                        resolvedEvents="i0000000000000000000000000000001" />
         <reporterComponentId     application="myApplication"
                                                        component="myComponent"
                                                        componentIdType="myComponentIdType"
                                                        executionEnvironment="myExec"
                                                        instanceId="myInstanceId"
                                                        location="myLocation"
                                                        locationType="myLocationType"
                                                        processId="100"
                                                        subComponent="mySubComponent"
                                                        threadId="122" />
        <sourceComponentId
                                                        application="myApplication1"
                                                        component="myComponent1"
                                                        componentIdType="myComponentIdType1"
                                                        executionEnvironment="myExec1"
                                                        instanceId="myInstanceId1"
                                                        location="myLocation1"
                                                        locationType="myLocationType1"
                                                        processId="102"
                                                        subComponent="mySubComponent1"
                                                        threadId="123" />
        <msgDataElement msgLocale="en-US">
                                                       <msgCatalogTokens value="2" />
                                                       <msgId>myMsgId2</msgId>
                                                       <msgIdType>myMsgIdType</msgIdType>
                                                       <msgCatalogId>myMsgCatalogId</msgCatalogId>
                                                       <msgCatalogType>myMsgCatalogType</msgCatalogType>
                                                       <msgCatalog>myMsgCatalog</msgCatalog>
        </msgDataElement>
</CommonBaseEvent>
<CommonBaseEvent                          creationTime="2001-12-31T12:00:01"
                                                        elapsedTime="0"
                                                        extensionName="CommonBaseEvent"
                                                        globalInstanceId="i0000000000000000000000000000001"
                                                        localInstanceId="myLocalInstanceId"
                                                        priority="0"
                                                        repeatCount="0"
                                                        sequenceNumber="0"
                                                        severity="0"
                                                        situationType="mySituation">
        <contextDataElements      name="Name"
                                                type="myContextType">
                                                        <contextValue>contextValue</contextValue>
        </contextDataElements>
        <extendedDataElements     name="z"
```

```
                                                        type="Integer">
                                                        <values>1</values>
      </extendedDataElements>
       <associatedEvents              associationEngine="a0000000000000000000000000000000"
                                                        resolvedEvents="i0000000000000000000000000000000" />
       <reporterComponentId     application="myApplication"
                                                        component="myComponent"
                                                        componentIdType="myComponentIdType"
                                                        executionEnvironment="myExec"
                                                        instanceId="myInstanceId"
                                                        location="myLocation"
                                                        locationType="myLocationType"
                                                        processId="100"
                                                        subComponent="mySubComponent"
                                                        threadId="122" />
      <sourceComponentId
                                                        application="myApplication1"
                                                        component="myComponent1"
                                                        componentIdType="myComponentIdType1"
                                                        executionEnvironment="myExec1"
                                                        instanceId="myInstanceId1"
                                                        location="myLocation1"
                                                        locationType="myLocationType1"
                                                        processId="102"
                                                        subComponent="mySubComponent1"
                                                        threadId="123" />
      <msgDataElement msgLocale="en-US">
                                                       <msgCatalogTokens value="1" />
                                                       <msgId>myMsgId1</msgId>
                                                       <msgIdType>myMsgIdType</msgIdType>
                                                       <msgCatalogId>myMsgCatalogId</msgCatalogId>
                                                       <msgCatalogType>myMsgCatalogType</msgCatalogType>
                                                       <msgCatalog>myMsgCatalog</msgCatalog>
      </msgDataElement>
 </CommonBaseEvent>
</CommonBaseEvents>
```

1.13 CommonBaseEvent Class Hierarchy

The following figure depicts the CommonBaseEvent schema using UML (Unified Modeling Language™) notation. We have used this graphical language to simplify the visualization of the Common Base Event schema. For detailed descriptions of the CommonBaseEvent and its properties, please refer to the CommonBaseEvent Description Section (described on page 9).



References

- *XML Schema Part 0: Primer*
- draft-leach-uuids-guids-01
- DMTF - Standards - Common Information Model (CIM) Specification Version 2.2
- COMMON INFORMATION MODEL (CIM) INDICATIONS (FINAL DRAFT)

Issues/Concerns

1. Define more granular semantics for each of the componentID type values.
2. Have “type” for each token msgCatalogTokens.
3. Have a field that captures the "confidentiality" of an event may be an interesting (optional) dimension when rendering the events in a role based dashboard. e.g. Sys admins may not be allowed to see highly confidential business level events, or a trader may not be allowed to see the events generated by other traders.
4. Define situationType as a complex type to allow for more granular situation taxonomy.