Canonical Situation Data Format:
The Common Base Event
ACAB.BO0301.1.1

Last Modified: 8/20/2003 5:52:25 PM

David Ogle
Autonomic Computing
daveogle@us.ibm.com

Heather Kreger
Emerging Technologies
kreger@us.ibm.com

Abdi Salahshour
Autonomic Computing
abdis@us.ibm.com

Jason Cornpropst
Tivoli Event Management
jhcornpr@us.ibm.com

Eric Labadie
WSAD PD Tooling
labadie@ca.ibm.com

Mandy Chessell
Business Integration
mandy_chessell@uk.ibm.com

Bill Horn
IBM Research – Yorktown
hornwp@us.ibm.com

John Gerken
Emerging Technologies
john_gerken@us.ibm.com

Verify Version and Completeness Prior to Use

Document Control

Notes
If a hard copy is made it is valid only on the day printed.

Table of Contents

1.1 PURPOSE

This document defines a common base event (CBE) and supporting technologies that define the structure of an event in a consistent and a common format. The purpose of the CBE is to facilitate effective intercommunication among disparate enterprise components that support logging, management, problem determination, autonomic computing and e-business functions in an enterprise. This document specifies baseline that encapsulate properties common to a wide variety of events, including business, autonomic, management, tracing and logging type events. The event format of the event is expressed as an XML document using UTF-8 or 16 encoding.

This document is prescriptive about the format and content of the data that is passed or retrieved from component. However, it is not prescriptive about the ways in which how individual applications are to store their data locally. Therefore, the application requirement is only to be able to generate or render events in this format, not necessarily to store them in this format. 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 that serves as a new standard for events among enterprise management and business applications.

1.2 Overview

For The Want Of A Nail

For the want of a nail, the shoe was lost;
For the want of the shoe, the horse was lost;
For the want of the horse, the rider was lost;
For the want of a rider, the battle was lost;
For the want of the battle, the kingdom was lost;
All for the want of a nail.

---- Unknown

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 that are necessary to keep a computing system healthy. Clearly, in this world, small incidents can have wide-reaching implications and few things are as small, yet pervasive, in a computing infrastructure as 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 among 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. Fundemental aspects of enterprise management and ebusiness communication, 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, Quality event data leads 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. 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, efforts 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. Hence, the "Common Base Event" is defined as a new standard for events used by enterprise management and business applications.

The CBE described here lends itself easily to several types of events, in particular: logging, tracing, management, and business events. In all of these cases, there is a significant need for the data elements and the format of those elements to be consistent, because all of these events need to be correlated with each other. Using log files or events published to subscribers, most IBM and non-IBM 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 who are intimately familiar with the application that generates the event. This lack of context can lead to hazardous situations when other applications that are responsible for handling the event attempt to interpret it, or when the event is used for system management or problem determination purposes. Consider again the basic problem of parsing time stamps. Format and granularity (for example, are the units milliseconds or microseconds?) both present needless obfuscation for the receiving application that receives the time-stamped event. A more general problem is the lack of consistency in the information that is presented. For example:

- What is the hostname of the machine on which the event occurred?
- Which component failed?
- Is the component that failed on the same physical machine as the application that is reporting it?
- Are there multiple events that should be interpreted as a single unit and/or in a certain order?

Obviously, the current lack of standardization creates considerable difficulty for automated situation handling. Complexity increases further when the problem occurs in a solution that is composed of multiple components. Without a standard, data stored in 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 the event. To alleviate this problem, the Common Base Event definition ensures completeness of the data by providing properties to publish the following 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 that is 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 also is considered. The affected component might not reside in the same physical machine as the component that reports it. This broader scope of information encapsulates enough data so that events can be exchanged and interpreted in a deterministic and appropriate manner across multiple management systems that consume the data, without losing fidelity due to serial hops among the multiple management systems.

1.2.1 The scope of this work

The goal of this work is to provide more than just an element definition for a common event. In addition, an XML schema definition is provided. This document's scope is limited to data format and content of the data; how the data is sent and received and how an application processes the data is outside the scope of this document.

The following sections specify in more detail the 3-tuple described earlier (reporting component, affected component, situation), 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]. In total, they represent the "nail" required to "shoe" the application, which can then communicate its status to management and business applications in a meaningful and productive manner, to support the systems infrastructure through improved reliability, availability and security, and in a way in which, it is hoped, will help keep the Rulers of the "Kingdom" happy as well.

1.3 Common Base Event and Situation Data

When a situation occurs, a 3-tuple must be reported:
1. the identification of the component that is reporting the situation,
2. the identification of the component that is experiencing the situation (which might be the same as the component that is reporting the situation), and
3. 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 `reporterComponentId` and `sourceComponentId` table entries define the reporter and the affected component IDs respectively. The remaining fields in the table comprise 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[2].

---

[1] Other groups, such as GGF's Common Management Model team are working on a coordinated component identifier. The proposal included here will be used as input to the other efforts, but may change once the final definition is realized.

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

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 or UTF-16 strings.

1.3.3 Data Types

The Common Base Event only supports 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[3]. The data types are XML schema signed data types.

- byte
- short
- int
- long
- float
- double
- string
- dateTime
- hexBinary
- boolean

1.3.4 String Length

The maximum string length MUST NOT exceed 1024 characters. If a longer string is needed then a hexBinary type MAY be used.

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

---

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

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 particular, the data elements and the format of those elements need to be consistent, as all of these events need to be correlated with each other, and they need to be flexible to allow for adaptation to product-specific requirements.

The use of the ExtendedDataElement (described later on page 26) is RECOMMENDED for including product-specific attributes. This property allows for user-supplied extensions for any other attributes not defined in the CommonBaseEvent.

Product-specific schema MAY be included in the CommonBaseEvent XML Schema (described on page 35) to extend that property 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 unrecognized and unsupported portion of the extended schema. 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.

Multiple versions of the XML Schema are supported by appending a version number to the XML schema file name. The initial (current) XML file name is therefore, commonbaseevent1_0.xsd. Also, the CommonBaseEvent schema provides "version" token in the schema declaration (CommonBaseEvent XML Schema on page 35)

1.5 CommonBaseEvent Description

Table 1 is a summary of the CommonBaseEvent properties; that is the data collected for the situation 3-tuple. A detailed description of the CommonBaseEvent follows the summary table.   As stated earlier, the 3-tuple consists of the ID of the component experiencing the situation, the ID of the component reporting the situation, and the situation itself.

Property Name	Type	Description
sourceComponentId	cbe:ComponentIdentification	Identification of the component that is "affected" or was "impacted" by the event or situation. This is a REQUIRED property for the component that is affected by the situation.
reporterComponentId	cbe:ComponentIdentification	Identification of the component that is the "reporter" of the event or the situation. This is a REQUIRED property if the reporting component is different than the source component. Otherwise this field MUST NOT be present.
situationInformation	xs:SituationInformationType	Information about a situation that is reported by components that produce or report the situation events. For details of the

		situationInformation, refer to the SituationInformationType definition. This is a REQUIRED property.

Table 1: CommonBaseEvent

Detailed description of CommonBaseEvent 3-tuple is described in the following sections:

1.5.1 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 to uniquely identify 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 NOT be present.

1.5.2 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 to uniquely identifying a component.

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

1.5.3 situationInformation

The situationInformation is the data that describes the situation reported by the event. The situation information includes a required set of properties or attributes that are common across products groups and platforms, yet architected and flexible enough to allow for adoption to product-specific requirements.

The data type for this property is a complex type as described by the SituationInformationType type.

This is a REQUIRED property.

1.6 SituationInformationType Description

The SituationInformationType is a collection of data that may be reported. Collections specified here are required set of properties or attributes that are common across products groups and platforms, yet architected and flexible enough to allow for adoption to product-specific requirements.

Table 2 is a summary of the properties for the SituationInformationType. A detailed description of the SituationInformationType properties 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 recommended value is either a 128 bit or 256 bit Globally Unique Id and MUST start with an alphabetic character (i.e., A-Z) .
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:severityType	The perceived 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 contains only general information and is not reporting an error. • **20 Harmless** MUST be used for cases in which 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** MUST be used to indicate that an error occurred, but it is too late to take remedial action. The associated 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:priorityType	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.
situationElement	xs:SituationIdentification	The situationElement specifies the type of the situation that caused the event to be reported. See SituationIdentification definition (described on page 35) for details. This is a REQUIRED property.
contextDataElements	cbe:ContextDataElement[]	An array of contexts that this event is referencing. See the ContextDataElement definition (described on page 28) 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 32) 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 an identical event within 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 was detected.
elapsedTime	xs:long	This is the time interval or the elapsed time during which the number of identical events occurred (as specified by the repeatCount property). This property is expressed in microseconds. If no value (or zero) is specified for repeatCount, then this is an OPTIONAL property with no default value. However, if repeatCount is specified (has a non-zero value), then elapsedTime is REQUIRED.
associatedEvents	cbe:AssociatedEvent[]	This property allows events to be grouped. This property is a complex type that consists of globalInstanceIds that identify the associated events plus a type field that describes 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 indicates any additional elements that are expected to be present within the event. This is an OPTIONAL property. If the value specified is null, then the 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 that allows any other attributes not defined by the CommonBaseEvent. Information placed here is assumed to be product specific data; its interpretation is not specified. This is an OPTIONAL property.

sequenceNumber	xs:long	A source-defined number that allows multiple messages to be sent and processed in logical order that is different than the order in which they arrived at the consumer location (e.g., an event server or management tools). The sequence number helps consumers to sort arrived messages that may arrive out-of-order. This is with respect to the creation time and to the particular reporter of the messages. This is an OPTIONAL property with no default value.

Table 2: SituationInformationType

Detailed description of SituationInformationType is described in the following sections:

1.6.1 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 but unique within the execution process that generates the event. However, once it is set it MUST remain constant for the lifetime of the event. The value content of the localInstanceId MAY be a multi-part value, such a timestamp, location, offset, or message ID and MAY use other application-defined techniques to ensure the uniqueness of the ID values. For example, the identifier value might be set to the string concatenation of the local host IP address, the absolute path of the access.log file, the local fully qualified host name, a time stamp, and the sequenceNumber. The resulting identifier might look like 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, that is, once it is set it MUST NOT be changed. It MAY be provided by the component that issues the event or MAY be assigned by the consumer of the event. The maximum string length for the localInstanceId MUST NOT exceed 128 characters.

1.6.2 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 and MUST start with an alphabetic character (i.e., A-Z). The GUID generation algorithm MUST ensure the uniqueness of this value.

One standard for constructing a GUID is defined in the Internet draft draft-leach-uuids-guids-01. This value is computed using cryptographic quality random information. This Internet draft does not generate a GUID that starts with an alphabetic character; to use it you MUST append it with a single character.

This is an OPTIONAL property, however, when it is specified it is not mutable, that is, once it is set, it MUST NOT be changed for the lifetime of the event. The globalInstanceId MAY be provided either by the component that issues the event or by the consumer of the event.

The globalInstanceId is required if associations among events are to be established. If globalInstanceId is not specified, then the association specified by AssociatedEvent (described on page 29), cannot be used.

1.6.3 creationTime

The date and time that the event was created that MUST be specified as defined by the XML Schema dateTime data type[4]. The value of the creationTime MUST provide granularity as precisely as the generating platform allows.

This is a REQUIRED property that is not mutable (that is, its value MUST NOT be changed for the lifetime of the event) and MUST be provided by the component that reports the event.

1.6.4 severity

The severity indicates the event status severity level with respect to the component that reports the event. These values are usually provide by a component's domain expert. The meanings of the values that this property may contain MAY be described by an enumeration of common values or qualifiers that indicate the severity level of the event. For example, information, warning, or a set of integers that map to 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 implementation-specific values. This field is intended to define the seriousness of the kind of situation that was encountered so that administrators can focus on the most severe problems.

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

- **0 Unknown**
- **10 Information:** MUST be used when the event contains only general information and is not reporting an 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 that an error occurred, but it is too late to take remedial action.

---

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

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 severity values 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.

1.6.5 priority

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

- **10 Low – for** an event that does not need to be processed immediately.
- **50 Medium -** for an event of average importance.
- **70 High -** for an important event that requires immediate 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 independent of severity, because priority is intended to be used primarily by event consumers, whereas severity indicates 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.

1.6.6 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 to uniquely identifying a component.

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

1.6.7 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 to uniquely identify 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 NOT be present.

1.6.8 situationType

The situationType specifies the type of the situation that caused the event to be reported. The categorization of situations facilitates the building of tools that focus on implementing the analysis and planning functions rather than on product specific data formats.

The data type for this property is a complex type as described by the SituationIdentification type that provides the required data to categorize a situation.

The simplest way to understand the usefulness of categorization is by providing a use case. For example, assume that a problem has been detected with component 'A'. The first step in the root cause analysis might be to check to see if 'x' was actually started, since it is known that 'A' has a dependency on 'x'. One approach to determine if 'x' is running is to check the log file for 'x' to see if it has started. The problem from a programmatic perspective is, there is not standard way to check the log files to see if 'x' has started. 'x' might log "Component 'x' started" or it might say "Change server state from starting to running". The reality is that both of these messages provide the same information, but they provide it using different terminology, making it difficult for a program to use. Simple checks like this would be much easier if all components reported, for example, that they "**started**". Writing code to check dependencies would be much easier and would be, to a large extent, component independent. For example, if product 'A' had dependencies on 'x' and 'y', the code to check the status of 'x' and the code to check the status of 'y' would be the same, in both cases, it would look for a '**started**' message.

This is a REQUIRED property, that once it set it is not mutable, that is it MUST NOT be change.

1.6.9 contextDataElements

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

This is an OPTIONAL property that is not mutable that is, once it is set it MUST NOT be changed. It MAY be provided by the component that issues the event or MAY be assigned by the consumer of the event.

1.6.10 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 32). There is no default value for the msgLocale

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

1.6.11 msgDataElement

The msgDataElement is a property that refers to 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.

1.6.12 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 that you specify a value for the extensionName.

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

1.6.13 extendedDataElements

The extendedDataElements property is a sequence of name elements of type ExtendedDataElement (described on page 26). It offers extensibility by providing a way to specify any other attributes not defined by the CommonBaseEvent data model. Information placed here is assumed to be product specific data.

This property is user-supplied; its named elements can be filtered, searched or referenced by the correlation rules.

This is an OPTIONAL property that is mutable,that is, after it is set it MAY be changed. The value for this property MAY be provided by the component that issues the event or the event consumer MAY assign it.

1.6.14 associatedEvents

The associatedEvents property allows for event grouping or parent-child relationship. This property is a complex type that consists of globalInstanceIds that identify the associated events (described on page 29).
This is an OPTIONAL property that is mutable, that is, after it is set it MAY be changed. The value for this property MAY be provided by the component that issues the event or the event consumer MAY assign it.

1.6.15 repeatCount

The repeatCount specifies the number of occurrences of identical events within a specified time interval. The time interval is specified by the elapsedTime property described next. The definition of "identical events" is application-specific and therefore is not defined by this specification.

This property is OPTIONAL and mutable. The repeatCount MAY be set by the component that issues the event or the event consumer. There is no default value. A value of 0 or no value indicates no repeated occurrences of the event.

1.6.16 elapsedTime

The elapsedTime is the time interval during which some number of identical events occurred. The number of occurrences is specified by the value of the repeatCount . The elapsedTime value indicates the duration of 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.

1.6.17 sequenceNumber

The sequenceNumber is a source-defined number that allows multiple messages to be sent and processed in a logical order that may be different than the order in which they arrived at the consumer's location (for example, with event servers or management tools). The sequence number helps consumers to sort messages when they arrive. 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.

1.7 ComponentIdentification Description

In the area of problem reporting, two general categories of components should be considered for problem diagnosis, the component that observes and reports the situation (the reporter component), and the actual component that experiences the situation (the affected component). Component identification provides a collection of attributes that are required to uniquely identify a component. The same data is used to identify both the reporter and the affected component. In some cases, these components might be the same.

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

Example 1: Consider a case in which a WebSphere application, called myWebApp, times out on a table query because of a problem with a DB2 server on a remote system. The application then issues an event that indicates the failure situation. In this case, myWebApp is the affected component (also called the source component).

Example 2: Consider a case in which an application X is running on a Windows server. The application encounters an error and adds an entry to the Windows error log. Another application (an adapter) reads messages from the error log and generates and sends a CBE formatted event. In this case the affected component (or the source of the event) is the application X and the reporter is the adapter that generated and sent the event.

Table 3 is a summary of the properties for 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 that corresponds to the location of a component. For example, hostname, IP address, MAC address, or VTAM LU. The format of the value of the location is specified by the locationType property. The RECOMMENDED value is a fully qualified hostname. This value should be unique within the network that is capable of raising events to a specified receiver. For example, if an adapter were monitoring a router, this attribute SHOULD 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:location Type	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. If the location value does not have a well-known type then the locationType should contain "Unknown".
application	xs:string	The name of the application (e.g., myWebApp). The

		application version information MAY be appended to the end of the component separated by a # character (e.g., maWebApp#4.5.1). This is an OPTIONAL property. The maximum string length for application 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# 7.1). This value SHOULD be unique within the scope specified by the locationType. The component version information MAY be appended to the end of the component name separated by a # character. This is a REQUIRED property. The maximum string length for the component name MUST NOT exceed 512 characters.
subComponent	xs:string	Specifies a further distinction for the logical component property of the event. It SHOULD contain the identity of the subcomponent of the component property. This property MAY 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, etc.). It SHOULD be the most granular definition specified in the event. The subcomponent version information MAY be appended to the end of the subcomponent name separated by a # character. This is a REQUIRED property. The maximum string length for the subComponent name MUST NOT exceed 512 characters.
componentIdType	xs:Name	Specifies the format and meaning of the component identified by this componentIdentification. The componentIdType is a well-defined name that is used to characterize all instances of a given kind of component. For detailed description of this property refer to the next section following this table. 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 (e.g., Grid Service Handle(GSH)[5] or 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 with 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 processId. This is an OPTIONAL property with no default value. The maximum string length for threadId MUST NOT exceed 64 characters.

Table 3: Component Identification

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

1.7.1 location

The location specifies the physical address that corresponds to the location of a component. Example of locations include: a hostname, IP address, MAC Address, or VTAM LU. The format of the value of the location is specified by the locationType property. The RECOMMENDED value is a fully qualified hostname.

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

1.7.2 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**

---

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

- **OID-OSI**
- **Dial**
- **HWA**
- **HID**
- **X25**
- **DCC**
- **SNA**
- **IPX**
- **E.164**
- **Hostname**
- **FQHostname**
- **Devicename**

This property is REQUIRED and not mutable; that is, once it is set it cannot be changed. The maximum string length for locationType MUST NOT exceed 32 characters. If the location value does not have a well-known type then the locationType MUST contain "Unknown".

1.7.3 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 (e.g., `myApp#3.2`). It is RECOMMENDED that to prepend the vendor name to the application name.

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

1.7.4 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 (e.g., `thiscell:thisnode:thisserver#5.3.2`). The maximum string length for executionEnvironment MUST NOT exceed 256 characters.

This is an OPTIONAL property and it is not mutable; that is, once it is set it MUST NOT be changed for the lifetime of the event.

1.7.5 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#7.1). This value SHOULD be unique within the scope specified by the reporter location.

This property is REQUIRED and it is not mutable; that is, once it is set it MUST NOT be changed for the life of the event. The maximum string length for the component name MUST NOT exceed 512 characters.

1.7.6 subComponent

The subComponent specifies a further distinction for the logical component property of the event. It SHOULD contain the identity of the subcomponent of the component property and it SHOULD be the most granular definition specified in the event. This property MAY be one of the various parts of an application or OS resource such as a module name, class name, class and method name and so on. This property is REQUIRED and it is not mutable; that is, once it is set it MUST NOT be changed for the lifetime of the event. The maximum string length for the subComponent name MUST NOT exceed 512 characters.

1.7.7 componentIdType

The componentIdType specifies the format and meaning of the component identified by this componentIdentification. The componentIdType is a well-defined name that is used to characterize all instances of a given kind of component. A component type is defined as an XML Name that contains a name space qualified name, for example:

http:// www.ibm.com/namespaces/autonomic/WebAppServer_componentTypes

The component type namespace is divided into several name spaces to facilitate delegation to various product groups. The default namespace for component types is described in Appendix Component Type Namespace. Namespaces other than the default namespace will be allowed for component type extensibility.

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

1.7.8 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)[6], EJBHandle and so on.

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

1.7.9 processId

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

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

1.7.10 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

---

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

threads to execute its functions. Therefore, the thread ID will change accordingly. The value is platform-specific.

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

1.8 *ExtendedDataElement Description*

The ExtendedDataElement allows for application-supplied name-type-value collections to be specified for extensibility purposes. This is the mechanism by which other attributes not specified in the CommonBaseEvent data model can be added. Collections specified here are assumed to be product specific data.

Table 4 is a summary of the properties for the `ExtendedDataElement` type. A detailed description of the `ExtendedDataElement` properties follows the summary table

The named properties can be filtered, searched or referenced by the correlation rules. The nameis user defined, however, the nonexclusive reserved keywords are as follows:

RawData - This keyword is indicative of "as is" data that is meaningful only to the producer of that data (typically proprietary 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 ExtendedDataElement's that are defined by 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 at the same level of extendedDataElement hierarchy, however, there may exist a child with the same name at different level or hierarchy. This is a REQUIRED property.
Type	xs:Name	The data type of the values specified in the values property. 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** These are the only valid data types for the ExtendedDataElement type. The default value is string.
values	xs:string[]	The array of values for this extended data element, represented as a string of the specified type, excluding the 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 and the values properties are mutually exclusive. Only one of these properties SHALL 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 4: ExtendedDataElement

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

1.8.1 name

The name property specifies the name of the ExtendedDataElement (including RawData, msgLocale, and EventStatus). This name MUST be unique with respect to all other properties at the same level of extendedDataElement hierarchy, however, there may exist a child with the same name at different level.

This property REQUIRED and it is not mutable.

1.8.2 type

The data type for the values property described next.

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

These data types are the only valid types for the ExtendedDataElement type.

The default value is string.

This property is REQUIRED and is not mutable.

1.8.3 values

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

This property is OPTIONAL and it is mutable. It MUST NOT be specified if the hexValueproperty is specified.

1.8.4 hexValue

The hexValue property is an array of characters that holds the data of type `hexBinary` for all other data types or complexTypes 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 hexValue and values properties are mutually exclusive. Only one of these two properties SHALL be defined.

1.8.5 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.

1.9 ContextDataElement Description

The ContextDataElement type defines the context(s) that this event references. This complex type holds data that is used to assist with problem diagnostics by correlating messages or events generated during execution of a unit of work.

Table 5 provides a summary of the data properties representing a context in the CBE. 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 REQUIRED unless contextId specifies a value.
contextId	xs:IDREF	This property is the reference to the element that contains the context. This is REQUIRED unless contextValue specifies a value.

Table 5: ContextDataElement

A detailed description of the contextDataElement is described in the following sections:

1.9.1 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., PD_LogRecordCorrelator).

This property is REQUIRED and is not mutable.

1.9.2 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.

1.9.3 contextValue

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

This property is REQUIRED unless contextId specifies a value and it is not mutable. It SHOULD NOT be specified if the `contextId` property is specified.

1.9.4 contextId

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

This property is not REQUIRED unless 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 SHALL be defined. However, if contextValue is set, then contextId is ignored.

1.10 AssociatedEvent Description

29

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

Table 6 provides a summary of the data properties representing an event association in the common base event. A detailed description of this `AssociatedEvent` follows the summary table..

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 6: AssociatedEvent

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

1.10.1 associationEngine

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

This property is REQUIRED and is not mutable; that is once it is set MUST NOT be changed.

1.10.2 resolvedEvents

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

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

1.11 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.

Table 7 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 7: AssociationEngine

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

1.11.1 name

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

This property is REQUIRED and is not mutable; that, it MUST NOT be changed.

1.11.2 type

This property SHOULD contain the type of association created by this AssociationEngine. Reserved keyword values include:

- **Contains** : When the association represents containment of other events within a root event
- **CausedBy** : When the association represents a causality in which the associated event can point to the cause of the situation.
- **Cleared** : When the association represents a relationship in which an event points to anther event that corrects the situation or results in the situation becoming irrelevant.
- **MultiPart** : When the association represents a collection of events that 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 is not mutable if it is set for a specific "name" property.

1.11.3 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.

1.12 MsgDataElement Description

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

Table 8 provides a summary of the data properties representing a message in the common base event. A detailed description of this MsgDataElement follows the summary table.

Property Name	Type	Description
msgId	xs:string	Specifies the message identifier of the event. This identifier SHOULD be a unique value string of alphanumeric or numeric characters. It can be as simple as a string of numeric characters that identify a message in a message catalog or a multi-part string of alphanumeric characters (e.g., DBT1234E). This is an 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 msgId 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 nonexclusive reserved keywords include: • IBM* (* is as described above) • JMX • DottedName • Unknown This is an OPTIONAL property. The maximum string length for msgIdType MUST NOT exceed 32 characters.
msgCatalogId	xs:Name	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 an internationalized 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 `msgCatalogTokens` does not need to be specified. This is an OPTIONAL property. The maximum string length for 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 for which this msg property is rendered. Its value is a locale code that conforms to IETF RFC 1766. This is an OPTIONAL property.

Table 8: MsgDataElement

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

1.12.1 msgId

The msgId property specifies the message identifier for the event. This identifier SHOULD be 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 for msgId is specified by the msgIdType property as described in the next section.

This is an OPTIONAL property, which is not mutable; that is, once it is set MJUST NOT be changed. It SHOULD be provided by the component that issues the event. The maximum string length for the msgId property MUST NOT exceed 256 characters.

1.12.2 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 specifies 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 an OPTIONAL property that is not mutable; that is, once it is set it MUST NOT be changed. It SHOULD be provided by the component that issues the event. It must be provided if msgId property is specified. The maximum string length for the msgIdType property MUST NOT exceed 32 characters.

1.12.3 msgLocale

The msgLocale property specifies the locale for which the msg is rendered. Its value is a locale code that conforms to 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; that is , once it is set it MUST NOT be changed. 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.

1.12.4 msgCatalogTokens

The msgCatalogTokens property consists of an array of string values that holds substitution data used to resolve an internationalized 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, defined by msgLocale.

This property is OPTIONAL and it is not mutable; that is, once it is set MUST NOT be changed. 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.

1.12.5 msgCatalogId

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

This property is OPTIONAL and it is not mutable; that is, once it is set it MUST NOT be changed.

1.12.6 msgCatalog

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

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

1.12.7 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.

1.13 SituationIdentification Description

The SituationIdentification type defines the situation type that this event is referencing. This complex type holds data that is used to specify the type of the situation that caused the event to be reported. The categorization of situations facilitates the building of tools that focus on implementing the analysis and planning functions rather than on product, specific data formats.

The data type for this property is a complex type as described by the SituationIdentification type that provides the required data to categorize a situation.

Table 9 provides a summary of the data properties representing a situation in the common base event. A detailed description of this SituationIdentification follows the summary table.

Property Name	Type	Description
situationName	xs:SituationIdentificationType	This property specifies the type of the situation that caused the event to be reported. The initial set of category names are described following this table. This is a REQUIRED property.
reasoningScope	xsca: SituationReasoningScopeType	This property specifies the scope of the impact of the situation reported. The initial set of values is described following this table. This is a REQUIRED property.

situationData	xs:string	This property allows specifying any situation specific data that can assist to further qualifying the situation. The content of this property is free form and may be gathered from the content of the reported situation. This is an OPTIONAL property.
availabilityDisposition	xs: SituationAvailabilityDispositionType	This property specifies the availability of resource reported by the situation. This is a REQUIRED property for the AVAILABLE and FEATURE categories.
successDisposition	xs: SituationSuccessDispositionType	This property specifies the success disposition of an operation of a situation that caused an event to be reported. The initial set of values is described following this table. This is a REQUIRED property for STAR, STOP, REQUEST, CREATE, and DESTROY.
operationalDispositon	xs: SituationOperationalDispositionType	This property specifies the operation state of the component reported by the situation. The initial set of values is described following this table. This is an OPTIONAL property.
situationQualifier	xs: SituationQualifierType	This property specifies the situation qualifiers that are structured representation of the parameters necessary to describe a situation. The initial set of values is described following this table. This is an OPTIONAL property.
dependencyDisposition	xs: SituationDependencyDispositionType	This property specifies the situation that is indicative of whether dependencies are met or not. The initial set of values is described following this table. This is a REQUIRED property for the DEPENDENCY category.

Table 9: SituationIdentification

A detailed description of the SituationIntification is described in the following sections:

1.13.1 situationName

This property categorize the type of the situation that caused the event to be reported. The initial set of category names are described below:

START	These are messages that deal with the start up process for a component. Messages that indicate that a component has begun the startup process, that it has finished the startup process, or that it has aborted the startup process all fall into this category. Existing messages include words like starting, started, initializing, and initialized. Sample messages; DIA3206I The TCP/IP protocol support was started successfully. DIA3000I "%1S" protocol support was successfully started. DIA3001E "%1S" protocol support was not successfully started. WSVR0037I: Starting EJB jar: {0}
STOP	These are message that deal with the shutdown process for a component. Message that indicate that a component has begun to stop, that it has stopped, or that the stopping process has failed all fall into this category. Existing messages include words like stop, stopping, stopped, completed, and exiting. Sample messages: WSVR0220I: Application stopped: {0} WSVR0102E: An error occurred stopping, {0} MSGS0657I: Stopping the MQJD JMS Provider
CONNECT	These are messages that components use to identify aspects about a connection to another component. Messages that say a connection failed, that a connection was created, or that a connection was ended all fall into this category. Existing messages include words like connection reset, connection failed, and failed to get a connection. Sample messages: DBMN0015W: Failure while creating connection {0} DBMN0033W: connection close failure {0} DBMN0023W: Failed to close a connection {0}
REQUEST	These are messages that a component uses to identify the completion status of a request. Typically, these requests are complex management tasks or transactions that a component undertakes on behalf of a requestor and not the mainline simple requests or transactions. Existing messages include words like configuration synchronization started, and backup procedure completed. Sample message: ADMS0003I: Configuration synchronization completed
CONFIGURE	These are messages that components use to identify their configuration. Any changes that a component makes to its configuration should be logged using this category. Additionally, messages that describe current configuration state fall into this category. Existing message include words like port number is, address is, and process id. Sample message: ADFS0134I: File transfer is configured with host="9.27.11.13", port="9090", securityEnabled="false".
AVAILABLE	This is data that is reported from the component, regarding its operational state and

	availability. This situation provides a context for operations that can be performed on the component by distinguishing if a product is installed, operational and ready to process functional requests, or operational and ready/not ready to process management requests. Existing message include words like those that now ready to take requests, online, and offline. Sample messages: ADMC0013I: SOAP connector available at port 8880 ADMC0026I: RMI Connector available at port 2809
REPORT	This is data that is reported from the component, such as heartbeat or performance information. Data such as current CPU utilization, current memory heap size, etc. would fall into this category. Existing messages include words like utilization value is, buffer size is, and number of threads is. Sample message: IEE890I WTO Buffers in console backup storage = 1024
CREATE	As the name implies, these are messages documenting when a component creates an entity. Messages telling that a document was created, or a file was created, or an EJB was created all fall into this category. Existing message include words like was created, about to create, and now exists. Sample message: ADMR0009I: Document cells/flatfootNetwork/applications/Dynamic Cache Monitor.ear/Dynamic Cache Monitor.ear was created.
DESTROY	These are messages documenting when an entity or component was removed or destroyed. Messages telling that a document was destroyed or a file was deleted all fall into this category. Existing message include words like was created, about to create, and now exists. Sample message: *** Add Example ***
FEATURE	These are messages that announce that a feature of a component is now ready (or not ready) for service requests. Messages that indicate things like services being available and services or features being unavailable fall into this category. Existing messages include words like now available, currently available, and transport is listening on port 123. Sample messages: SRVE0171I: Transport HTTPS is listening on port 9443 MSGS0601I: WebSphere Embedded Messaging has not been installed
DEPENDENCY	These are messages that components produce to say that they cannot find some component or feature that they need. This category includes messages about not finding the 'version' of the component that was expected. Messages that say a resource was not found, or that an application or subsystem that was unavailable, also fall into this category. Existing messages include words like could not find, and no such component. Sample message: WSVR0017E: Error encountered binding the J2EE resource, Pet Store JMS Queue Connection Factory, as jms/queue/QueueConnectionFactory from resources.xml no resource binder found

This property is REQUIRED property and it is not mutable.

1.13.2 reasoningScope

This property specifies the scope of the situation that caused the event to be reported. In other words whether the impact of the situation in on within the product or may have an impact on other entities in

the serving environment. The reasoningScope is of type SituationReasoningScopeType with following initial set of values; and it is extensible using XML union.

INTERNAL | EXTERNAL

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

1.13.3 situationData

This property allows specifying any situation specific data that can assist to further qualifying the situation. The content of the situationData is free form and may be gathered from the content of the reported situation.

This is an OPTIONAL property.

1.13.4 successDisposition

This property specifies the success disposition of an operation of a situation that caused the event to be reported. The successDisposition is of type SituationSuccessDispositionType with following initial set of values; and it is extensible using XML union.

SUCCESSFUL | UNSUCCESSFUL

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

1.13.5 availabilityDisposition

This property specifies the operational status or availability of feature of a component that a situation event reports. The availabilityDisposition is of type SituationAvailabilityDispositionType with following initial set of values; and it is extensible using XML union.

AVAILABLE | NOT AVAILABLE

This is a REQUIRED property for AVAILABLE and FEATURE situation categories; and once it is set it is not mutable.

1.13.6 operationalDisposition

This property specifies the operation state of the component reported by the situation. The operationalDisposition is of type SituationOperationalDispositionType with following initial set of values; and it is extensible using XML union.

STARTABLE | NONSTARTABLE

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

1.13.7 situationQualifier

This property specifies the situation qualifiers that are structured representation of the parameters necessary to describe a situation. The situationQualifier is of type SituationQualifierType with following initial set of values; and it is extensible using XML union.

For START situations
> STARTING | START INITIATED | START COMPLETED | RESTART INITIATED

For STOP situations
> STOPING | STOPED | STOP INITIATED | START COMPLETED | ABORT INITIATED | PAUSE INITIATED

For REQUEST situations
> REQUEST INITIATED | REQUEST COMPLETED

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

1.14 CommonBaseEvent XML schema

The following XML Schema is a document that describes the element and the attribute declarations for the Common Base Event (CBE) 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_1"
targetNamespace="http://www.ibm.com/AC/commonbaseevent1_1" version="1.1">
        <xsd:complexType name="CommonBaseEventType">
                <xsd:sequence>
                        <xsd:element name="sourceComponentId" type="cbe:ComponentIdentificationType"
minOccurs="1" maxOccurs="1" />
                        <xsd:element name="reporterComponentId" type="cbe:ComponentIdentificationType"
minOccurs="0" maxOccurs="1" />
                        <xsd:element name="situationInformation" type="cbe:SituationInformationType"
minOccurs="1" maxOccurs="1" />
                        <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded" />
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="CommonBaseEvent" type="cbe:CommonBaseEventType" />

        <xsd:complexType name="SituationInformationType">
                <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="msgDataElement" type="cbe:MsgDataElementType" minOccurs="0"
maxOccurs="1" />
                        <xsd:element name="situationType" type="cbe:SituationIdentificationType" minOccurs="0"
maxOccurs="1" />
                </xsd:sequence>
                <!-- Added the restriction for V1.1 -->
                <xsd:attribute name="globalInstanceId" use="optional">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:ID">
                                        <xsd:minLength value="16"></xsd:minLength>
                                        <xsd:maxLength value="32"></xsd:maxLength>
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:attribute>
                <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" />
        <!-- Changed the type to severityType V1.1 -->
        <xsd:attribute name="severity" type="cbe:severityType" use="optional" />
        <xsd:attribute name="msg" use="optional">
                <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                                <xsd:maxLength value="1024" />
                        </xsd:restriction>
                </xsd:simpleType>
        </xsd:attribute>
        <!-- Changed type from short to priorityType V1.1 -->
        <xsd:attribute name="priority" use="optional" type="cbe:priorityType" />
        <!-- Added restriction for minValue and maxValue V1.1 -->
        <xsd:attribute name="sequenceNumber" use="optional">
                <xsd:simpleType>
                        <xsd:restriction base="xsd:long">
                                <xsd:minInclusive value="0" />
                                <xsd:maxInclusive value="9223372036854775807" />
                        </xsd:restriction>
                </xsd:simpleType>
        </xsd:attribute>
        <!-- Changed type from int to short and added restriction for minValue and maxValue V1.1 -->
        <xsd:attribute name="repeatCount" use="optional">
                <xsd:simpleType>
                        <xsd:restriction base="xsd:short">
                                <xsd:minInclusive value="0" />
                                <xsd:maxInclusive value="32767" />
                        </xsd:restriction>
                </xsd:simpleType>
        </xsd:attribute>
        <!-- Added restriction for minValue and maxValue V1.1 -->
        <xsd:attribute name="elapsedTime" use="optional">
                <xsd:simpleType>
                        <xsd:restriction base="xsd:long">
                                <xsd:minInclusive value="0" />
                                <xsd:maxInclusive value="9223372036854775807" />
                        </xsd:restriction>
                </xsd:simpleType>
        </xsd:attribute>
</xsd:complexType>

<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>
            <!-- Change the type from string to componentIdTypeType V1.1 -->
            <xsd:attribute name="componentIdType" use="required" type="cbe:componentIdTypeType" />
            <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>
                            <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>
            <!-- Changed the type from Name to locationTypeType V1.1 -->
            <xsd:attribute name="locationType" use="required" type="cbe:locationTypeType" />
            <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>
```

```xml
<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>
<!-- Changed the type tp language and added restriction V1.1 -->
<xsd:attribute name="msgLocale" use="optional">
        <xsd:simpleType>
                <xsd:restriction base="xsd:language">
                        <xsd:maxLength value="11"></xsd:maxLength>
                </xsd:restriction>
        </xsd:simpleType>
</xsd:attribute>
</xsd:complexType>

<xsd:group name="msgCatalogGroup">
        <xsd:sequence>
                <!-- Added restriction to 128 and changed the type to Name V1.1 -->
                <xsd:element name="msgCatalogId" minOccurs="1" maxOccurs="1">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="128" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:element>
                <!-- Changed the type from Name to msgCatalogTypeType V1.1 -->
                <xsd:element name="msgCatalogType" minOccurs="1" maxOccurs="1"
type="cbe:msgCatalogTypeType" />
                <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" type="cbe:msgIdTypeType"
/>
                </xsd:sequence>
        </xsd:group>

        <!-- Added the locationType and the reserved keywords V1.1  -->
        <xsd:simpleType name="locationTypeType">
                <xsd:union>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="32" />
                                        <xsd:enumeration value="UNKNOWN" />
                                        <xsd:enumeration value="IPV4" />
                                        <xsd:enumeration value="IPV6" />
                                        <xsd:enumeration value="NWA" />
                                        <xsd:enumeration value="ISDN" />
                                        <xsd:enumeration value="ICD" />
                                        <!-- Changed the / to - V1.1 -->
                                        <xsd:enumeration value="OID-OSI" />
                                        <xsd:enumeration value="DIAL" />
                                        <xsd:enumeration value="HWA" />
                                        <xsd:enumeration value="HID" />
                                        <xsd:enumeration value="X25" />
                                        <xsd:enumeration value="DCC" />
                                        <xsd:enumeration value="SNA" />
                                        <xsd:enumeration value="IPX" />
                                        <xsd:enumeration value="E.164" />
                                        <xsd:enumeration value="Hostname" />
                                        <xsd:enumeration value="FQHostname" />
                                        <xsd:enumeration value="DeviceName" />
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="32" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>

        <xsd:simpleType name="componentIdTypeType">
                <xsd:restriction base="xsd:Name">
                        <xsd:maxLength value="32" />
                </xsd:restriction>
        </xsd:simpleType>

        <!-- Added the msgIdTypeType, the reserved value and changed type from Name to Name V1.1 -->
        <xsd:simpleType name="msgIdTypeType">
                <xsd:union>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="32" />
                                        <xsd:enumeration value="IBM3.4" />
                                        <xsd:enumeration value="IBM4.4" />
                                        <xsd:enumeration value="IBM3.1.4" />
                                        <xsd:enumeration value="IBM3.4.1" />
```

```
                              <xsd:enumeration value="IBM4.4.1" />
                              <xsd:enumeration value="IBM3.1.4.1" />
                              <xsd:enumeration value="JMX" />
                              <xsd:enumeration value="DottedName" />
                              <xsd:enumeration value="Unknown" />
                      </xsd:restriction>
              </xsd:simpleType>
              <xsd:simpleType>
                      <xsd:restriction base="xsd:Name">
                              <xsd:maxLength value="32" />
                      </xsd:restriction>
              </xsd:simpleType>
      </xsd:union>
</xsd:simpleType>

<!-- Added the msgCatalogTypeType, the reserved value and changed type from Name to Name V1.1 -->
<xsd:simpleType name="msgCatalogTypeType">
      <xsd:union>
              <xsd:simpleType>
                      <xsd:restriction base="xsd:Name">
                              <xsd:maxLength value="32" />
                              <xsd:enumeration value="JAVA" />
                              <xsd:enumeration value="XPG" />
                      </xsd:restriction>
              </xsd:simpleType>
              <xsd:simpleType>
                      <xsd:restriction base="xsd:Name">
                              <xsd:maxLength value="32" />
                      </xsd:restriction>
              </xsd:simpleType>
      </xsd:union>
</xsd:simpleType>

<!-- Added the severityType and the reserved value and restrictions  V1.1 -->
<xsd:simpleType name="severityType">
      <xsd:union>
              <xsd:simpleType>
                      <xsd:restriction base="xsd:short">
                              <xsd:enumeration value="0" /><!-- Unknown -->
                              <xsd:enumeration value="10" /><!-- Information -->
                              <xsd:enumeration value="20" /><!-- Harmless -->
                              <xsd:enumeration value="30" /><!-- Warning -->
                              <xsd:enumeration value="40" /><!-- Minor -->
                              <xsd:enumeration value="50" /><!-- Critical -->
                              <xsd:enumeration value="60" /><!-- Fatal -->
                              <xsd:minInclusive value="0" />
                              <xsd:maxInclusive value="70" />
                      </xsd:restriction>
              </xsd:simpleType>
              <xsd:simpleType>
                      <xsd:restriction base="xsd:short">
                              <xsd:minInclusive value="0" />
                              <xsd:maxInclusive value="70" />
                      </xsd:restriction>
              </xsd:simpleType>
      </xsd:union>
```

```
        </xsd:simpleType>

        <!-- Added the priorityType and the reserved value V1.1 -->
        <xsd:simpleType name="priorityType">
                <xsd:union>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:short">
                                        <xsd:enumeration value="10" /><!-- Low -->
                                        <xsd:enumeration value="50" /><!-- Medium -->
                                        <xsd:enumeration value="70" /><!-- High -->
                                        <xsd:minInclusive value="0" />
                                        <xsd:maxInclusive value="100" />
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:short">
                                        <xsd:minInclusive value="0" />
                                        <xsd:maxInclusive value="100" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>

        <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="optional" />
                <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 -->
                <!-- Added restrictions V1.1 -->
                <xsd:attribute name="id" use="required">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:ID">
                                        <xsd:minLength value="16"></xsd:minLength>
                                        <xsd:maxLength value="32"></xsd:maxLength>
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:attribute>
                <!-- Changed the type from string to AssociationEnginetypeType V1.1 -->
                <xsd:attribute name="type" use="required" type="cbe:AssociationEngineTypeType" />
                <!-- Changed the type to Name and added restriction V1.1 -->
                <xsd:attribute name="name" use="required">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:attribute>
        </xsd:complexType>
        <xsd:element name="AssociationEngine" type="cbe:AssociationEngineType" />
```

```
<!-- Added the AssociationEngineTypeType, the reserved value and -->
<!-- changed the type to Name and added maxLength restriction V1.1 -->
<xsd:simpleType name="AssociationEngineTypeType">
        <xsd:union>
                <xsd:simpleType>
                        <xsd:restriction base="xsd:Name">
                                <xsd:enumeration value="Contain"></xsd:enumeration>
                                <xsd:enumeration value="CausedBy"></xsd:enumeration>
                                <xsd:enumeration value="Cleared"></xsd:enumeration>
                                <!-- Removed the multi-part keyword V1.1 -->
                                <xsd:enumeration value="MultiPart"></xsd:enumeration>
                                <xsd:enumeration value="Correlated"></xsd:enumeration>
                                <xsd:maxLength value="64" />
                        </xsd:restriction>
                </xsd:simpleType>
                <xsd:simpleType>
                        <xsd:restriction base="xsd:Name">
                                <xsd:maxLength value="64" />
                        </xsd:restriction>
                </xsd:simpleType>
        </xsd:union>
</xsd:simpleType>

<!-- Added restrictions and type definitions as well as new types V1.1 -->
<xsd:complexType name="ExtendedDataElementType">
        <xsd:sequence>
                <xsd:choice>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="values">
                                <xsd:simpleType>
                                        <xsd:restriction base="xsd:string">
                                                <xsd:maxLength value="1024"></xsd:maxLength>
                                        </xsd:restriction>
                                </xsd:simpleType>
                        </xsd:element>
                        <xsd:element maxOccurs="1" minOccurs="0" name="hexValue"
type="xsd:hexBinary" />
                        <!-- Added the following new types. V1.1 -->
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="intValues"
type="xsd:int"></xsd:element>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="longValues"
type="xsd:long"></xsd:element>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="shortValues"
type="xsd:short"></xsd:element>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="dateTimeValues"
type="xsd:dateTime"></xsd:element>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="byteValues"
type="xsd:byte"></xsd:element>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="booleanValues"
type="xsd:boolean"></xsd:element>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="floatValues"
type="xsd:float"></xsd:element>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="doubleValues"
type="xsd:double"></xsd:element>
                        <xsd:element maxOccurs="1" minOccurs="0" name="largeStringValue"
type="xsd:string"></xsd:element>
                </xsd:choice>
```

```
<!-- Moved the children outside of the choice so we can specify a value and a children V1.1 -->
<xsd:element maxOccurs="unbounded" minOccurs="0" name="children"
type="cbe:ExtendedDataElementType" />
</xsd:sequence>
<!-- Added length restriction V1.1 -->
<xsd:attribute name="name" use="required">
<xsd:simpleType>
<xsd:restriction base="xsd:Name">
<xsd:maxLength value="64" />
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<!-- Should consider to group the type and value as specific groups for V2.0 -->
<xsd:attribute name="type" use="required">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<!-- Added type noValue when only a children is specified.  -->
<!-- With a new type noValue, so the value is ignored and only the children is
considered. V1.1 -->
<xsd:enumeration value="noValue"></xsd:enumeration>
<xsd:enumeration value="byte"></xsd:enumeration>
<xsd:enumeration value="short"></xsd:enumeration>
<xsd:enumeration value="int"></xsd:enumeration>
<xsd:enumeration value="long"></xsd:enumeration>
<xsd:enumeration value="float"></xsd:enumeration>
<xsd:enumeration value="double"></xsd:enumeration>
<xsd:enumeration value="string"></xsd:enumeration>
<xsd:enumeration value="dateTime"></xsd:enumeration>
<xsd:enumeration value="boolean"></xsd:enumeration>
<xsd:enumeration value="byteArray"></xsd:enumeration>
<xsd:enumeration value="shortArray"></xsd:enumeration>
<xsd:enumeration value="intArray"></xsd:enumeration>
<xsd:enumeration value="longArray"></xsd:enumeration>
<xsd:enumeration value="floatArray"></xsd:enumeration>
<xsd:enumeration value="doubleArray"></xsd:enumeration>
<xsd:enumeration value="stringArray"></xsd:enumeration>
<xsd:enumeration value="dateTimeArray"></xsd:enumeration>
<xsd:enumeration value="booleanArray"></xsd:enumeration>
<xsd:enumeration value="hexBinary"></xsd:enumeration>
<xsd:enumeration value="largeStringValue"></xsd:enumeration>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>


<!-- Added restrictions V1.1 -->
<xsd:complexType name="ContextDataElementType">
<xsd:sequence>
<xsd:choice>
<xsd:element maxOccurs="1" minOccurs="1" name="contextValue">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="1024"></xsd:maxLength>
</xsd:restriction>
```

```
                </xsd:simpleType>
            </xsd:element>
            <xsd:element maxOccurs="1" minOccurs="1" name="contextId" type="xsd:IDREF"
/>
        </xsd:choice>
        <!-- Added children element to support structured contexts V1.1 -->
        <xsd:element maxOccurs="unbounded" minOccurs="0" name="children"
type="cbe:ContextDataElementType" />
    </xsd:sequence>
    <!-- changed the type to Name-->
    <xsd:attribute name="name" use="required">
        <xsd:simpleType>
            <xsd:restriction base="xsd:Name">
                <xsd:maxLength value="64"></xsd:maxLength>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
    <!-- changed the type to Name-->
    <xsd:attribute name="type" use="required">
        <xsd:simpleType>
            <xsd:restriction base="xsd:Name">
                <xsd:maxLength value="64"></xsd:maxLength>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>
<!-- Added new SituationType V1.1 -->
<xsd:complexType name="SituationIdentificationType">
    <xsd:attribute name="category" type="cbe:SituationCategoryType" use="required" />
    <xsd:attribute name="successDisposition" type="cbe:SituationSuccessDispositionType" use="required"
/>
    <xsd:attribute name="situationQualifier" type="cbe:SituationQualifierType" use="required" />
    <xsd:attribute name="operationDisposition" type="cbe:SituationOperationDispositionType"
use="optional" />
    <xsd:attribute name="reasoningScope" type="cbe:SituationReasoningScopeType" use="required" />
    <xsd:attribute name="connectDisposition" type="cbe:SituationConnectDispositionType" use="optional"
/>
    <xsd:attribute name="availabilityDisposition" type="cbe:SituationAvailabilityDispositionType"
use="optional" />
    <xsd:attribute name="situationData" use="optional">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:maxLength value="1024" />
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>
<!-- Added SituationCategoryType V1.1 -->
<xsd:simpleType name="SituationCategoryType">
    <xsd:union>
        <xsd:simpleType>
            <xsd:restriction base="xsd:Name">
                <xsd:enumeration value="START" />
                <xsd:enumeration value="STOP" />
                <xsd:enumeration value="FEATURE" />
                <xsd:enumeration value="DEPENDENCY" />
```

```
                                        <xsd:enumeration value="REQUEST" />
                                        <xsd:enumeration value="CONFIGURE" />
                                        <xsd:enumeration value="CONNECT" />
                                        <xsd:enumeration value="CREATE" />
                                        <xsd:enumeration value="REPORT" />
                                        <xsd:enumeration value="AVAILABLE" />
                                        <xsd:enumeration value="UNKNOWN" />
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <!-- Added new SituationSuccessDispositionType V1.1 -->
        <xsd:simpleType name="SituationSuccessDispositionType">
                <xsd:union>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:enumeration value="SUCCESSFUL" />
                                        <xsd:enumeration value="UNSUCCESSFUL" />
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <!-- Added new SituationQualifierType V1.1-->
        <xsd:simpleType name="SituationQualifierType">
                <xsd:union>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value="start initiated" />
                                        <xsd:enumeration value="start completed" />
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <!-- Added new SituationOperationDispositionType V1.1 -->
        <xsd:simpleType name="SituationOperationDispositionType">
                <xsd:union>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
```

```
                                        <xsd:enumeration value="startable" />
                                        <xsd:enumeration value="nonstartable" />
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <!-- Added new SituationReasoningScopeType V1.1 -->
        <xsd:simpleType name="SituationReasoningScopeType">
                <xsd:union>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:enumeration value="Infrastructure" />
                                        <xsd:enumeration value="Application" />
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <!-- Added new SituationConnectDispositionType V1.1 -->
        <xsd:simpleType name="SituationConnectDispositionType">
                <xsd:union>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:enumeration value="External" />
                                        <xsd:enumeration value="Internal" />
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <!-- Added new SituationAvailabilityDispositionType V1.1 -->
        <xsd:simpleType name="SituationAvailabilityDispositionType">
                <xsd:union>
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:Name">
                                        <xsd:enumeration value="blank" />
                                        <xsd:maxLength value="64" />
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType>
```

```
                              <xsd:restriction base="xsd:Name">
                                    <xsd:maxLength value="64" />
                              </xsd:restriction>
                        </xsd:simpleType>
                  </xsd:union>
            </xsd:simpleType>
            <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>
```

1.15 Sample CommonBaseEvent XML

1.15.1 Sample 1:
```
<?xml version="1.0" encoding="UTF-8"?>
<cbe:CommonBaseEvent xmlns:cbe="http://www.ibm.com/AC/commonbaseevent1_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/AC/commonbaseevent1_1 commonbaseevent1_1___.xsd ">
        <sourceComponentId component="myComponent"
                                    componentIdType="Name"
                                    location="121.0.0.1"
                                    locationType="IPV4"
                                    subComponent="mySubComponent" />
        <situationInformation creationTime="2001-12-31T12:00:00"
                                            globalInstanceId="A00000000000000000000000000000000"
                                            msg="Hello World!"/>
</cbe:CommonBaseEvent>
```

1.15.2 Sample 2:
```
<?xml version="1.0" encoding="UTF-8"?>
<cbe:CommonBaseEvents xmlns:cbe="http://www.ibm.com/AC/commonbaseevent1_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/AC/commonbaseevent1_1 commonbaseevent1_1___.xsd ">

 <cbe:AssociationEngine        id="a00000000000000000000000000000000"
                                            name="myassociationEngineName"
                                            type="Correlated" />


 <cbe:CommonBaseEvent>
        <sourceComponentId        application="myApplication1"
                                            component="myComponent1"
                                            componentIdType="myComponentIdType1"
                                            executionEnvironment="myExec1"
                                            instanceId="myInstanceId1"
                                            location="myLocation1"
                                            locationType="myLocationType1"
                                            processId="102"
                                            subComponent="mySubComponent1"
                                            threadId="123" />

        <reporterComponentId    application="myApplication"
                                            component="myComponent"
                                            componentIdType="myComponentIdType"
                                            executionEnvironment="myExec"
                                            instanceId="myInstanceId"
                                            location="myLocation"
                                            locationType="myLocationType"
                                            processId="100"
                                            subComponent="mySubComponent"
                                            threadId="122" />
```

```
<situationInformation     creationTime="2001-12-31T12:00:00"
                                    elapsedTime="0"
                                    extensionName="CommonBaseEvent"
                                    globalInstanceId="i0000000000000000000000000000000"
                                    localInstanceId="myLocalInstanceId"
                                    priority="0"
                                    repeatCount="0"

                                    sequenceNumber="0"
                                    severity="0"  >

    <contextDataElements     name="Name"
                                    type="myContextType">
                                        <contextValue>contextValue</contextValue>
    </contextDataElements>
<extendedDataElements          name="z"

                                    type="int">
                                    <values>1</values>
    </extendedDataElements>
<associatedEvents              associationEngine="a0000000000000000000000000000000"
                                        resolvedEvents="i0000000000000000000000000000000"
/>

    <msgDataElement msgLocale="en-US">
                                        <msgCatalogTokens value="2" />
                                        <msgId>myMsgId2</msgId>
                                        <msgIdType>myMsgIdType</msgIdType>
                                        <msgCatalogId>myMsgCatalogId</msgCatalogId>

    <msgCatalogType>myMsgCatalogType</msgCatalogType>
                                        <msgCatalog>myMsgCatalog</msgCatalog>

    </msgDataElement>

    <situationType             availabilityDisposition="AVAILABLE"
                                    category="START"
                                    reasoningScope="INTERNAL"
                                    situationData="trade3"
                                    situationQualifier="START_COMPLETED"
                                    successDisposition="SUCCESSFUL" />
    </situationInformation>
</cbe:CommonBaseEvent>
</cbe:CommonBaseEvents>
```
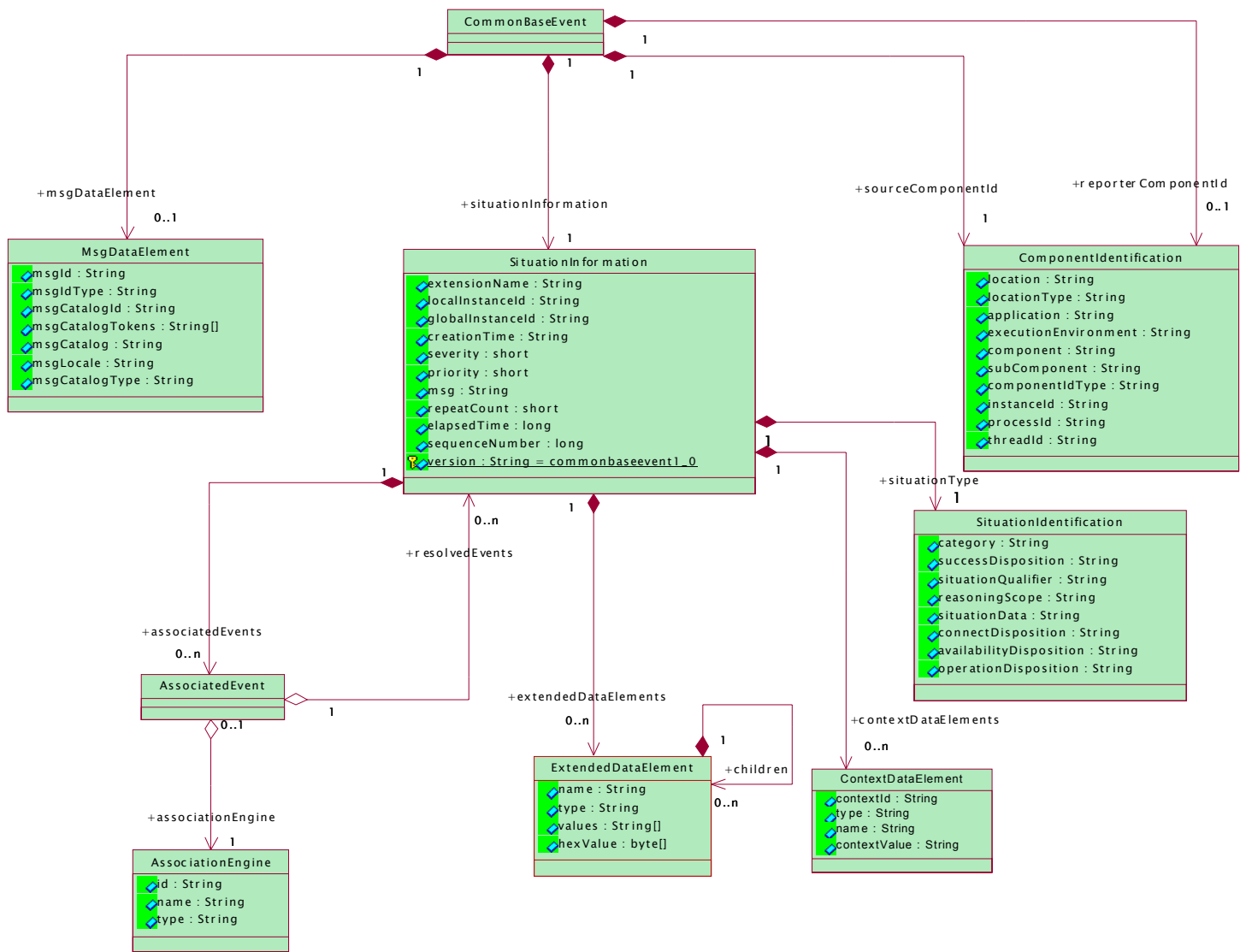
1.16 CommonBaseEvent Class Hierarchy

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 10).

Canonical Situation Data Format: The Common Base Event

CommonBaseEvent

MsgDataElement
- msgId : String
- msgIdType : String
- msgCatalogId : String
- msgCatalogTokens : String[]
- msgCatalog : String
- msgLocale : String
- msgCatalogType : String

+msgDataElement 0..1

+situationInformation 1

+sourceComponentId 1

+reporterComponentId 0..1

SituationInformation
- extensionName : String
- localInstanceId : String
- globalInstanceId : String
- creationTime : String
- severity : short
- priority : short
- msg : String
- repeatCount : short
- elapsedTime : long
- sequenceNumber : long
- version : String = commonbaseevent1_0

ComponentIdentification
- location : String
- locationType : String
- application : String
- executionEnvironment : String
- component : String
- subComponent : String
- componentIdType : String
- instanceId : String
- processId : String
- threadId : String

+situationType 1

SituationIdentification
- category : String
- successDisposition : String
- situationQualifier : String
- reasoningScope : String
- situationData : String
- connectDisposition : String
- availabilityDisposition : String
- operationDisposition : String

+resolvedEvents 0..n

+associatedEvents 0..n

AssociatedEvent

+extendedDataElements 0..n

ExtendedDataElement
- name : String
- type : String
- values : String[]
- hexValue : byte[]

+children 0..n 1

+contextDataElements 0..n

ContextDataElement
- contextId : String
- type : String
- name : String
- contextValue : String

+associationEngine 1

AssociationEngine
- id : String
- name : String
- type : String

Legend:

Rectangle Box: Represents a class, which is a group of methods and properties that have similar attributes and behavior. Class name is specified on top part of the box followed by the attributes and their type. Operations are typically shown in the bottom portion of the box but are not specified here.

Straight line: Indicates an association or relationship between classes. The direction is indicated by an arrow head (">").

Solid Diamond: Depicts a composite aggregation (by-value) or "has" relationship. The diamond is placed on the target end of the aggregation next to the target class, indicating that a class is embedded within another class. Cardinality, the number of objects, is indicated by numbers on either end of the association line.

Hollow Diamond: Depicts an aggregate association relationship (by-reference) between two classes to show that each instance of one class has a pointer or reference within it to an instance of another class. The hollow diamond is placed on the target end of the aggregation next to the target class.

Appendix

Component Type Namespace

A component type is a well-defined name (syntax and semantics) that is used to characterize all instances of a given kind of component. This appendix provides enumeration of standardized values for component types that will be used by IBM products. The component type namespace is divided into several name spaces to facilitate delegation to various product groups. However, it is expected that the initial list of component types are globally unique to facilitate their use without a qualifier.

Overall namespace:
http:// www.ibm.com/namespaces/autonomic/common_componentTypes

Delegated namespaces:

WebAppServer Components
http:// www.ibm.com/namespaces/autonomic/WebAppServer_componentTypes

DataManagement Components
http:// www.ibm.com/namespaces/autonomic/DM_componentTypes

Lotus Components
http:// www.ibm.com/namespaces/autonomic/Lotus_componentTypes

Tivoli Components
http:// www.ibm.com/namespaces/autonomic/Tivoli_componentTypes

OperatingSystem components
http:// www.ibm.com/namespaces/autonomic/OS_componentTypes

The Operating System Hosting Environment

<u>Namespace:</u>

> http:// www.ibm.com/namespaces/autonomic/OS_componentTypes

<u>Hosting Environment:</u>

> *Operating_System*

<u>Hosting Environment that hosts this Hosting Environment:</u>

> Operating_System_Container

<u>Component Type Enumeration Rule for this Hosting Environment:</u>

\<Operating System Product Name\>_\<Optional OS Family\>_\<Optional OS Role\>
where,

OS_Role can be *Server, Workstation, AdvancedServer,Home,Professional*

Component Types for this Hosting Environment based on Enumeration Rule:

- *RedHatLinux*
- *SuSELinux*
- *UnitedLinux*
- *SunSolaris*
- *IBMAIX*
- *HPUX*
- *NovellNetware*
- *IBMMVS*
- *IBMOS400*

- *MicrosoftWindows_98*
- *MicrosoftWindows_ME*
- *MicrosoftWindows_NT_Workstation*
- *MicrosoftWindows_NT_Server*
- *MicrosoftWindows_2000_Workstation*
- *MicrosoftWindows_2000_Server*
- *MicrosoftWindows_2000_AdvancedServer*
- *MicrosoftWindows_XP_Home*

- *MicrosoftWindows_NT_Server*
- *MicrosoftWindows_XP_Professional*
- *MicrosoftWindows_2003_Server*
- *MicrosoftWindows_2003_AdvancedServer*

Component Types hosted in this Hosting Environment:

Components of this Hosting Environment

- *Language_Runtime*
- *Device_Driver*
- *Software*
- *Process*
- *Thread*
- *TCPIP_port*

Components Hosted by this Hosting Environment

- *Relational_Database*
- *Web_Application_Server*
- *Messaging*

Catgegorization (Grouping) Component Types:

- *Windows*
- *Windows-Win32*
- *UNIX*
- *POSIX*
- *Linux*

- *Operating_System*
- *MicrosoftWindows_NT*
- *MicrosoftWindows_2000*
- *MicrosoftWindows_XP*
- *MicrosoftWindows_2003*

The Web_Application_Server Hosting Environment

Namespace:
http:// www.ibm.com/namespaces/autonomic/WebAppServer_componentTypes

59

<u>Hosting Environment:</u>

> *Web_Application_Server*

<u>Hosting Environment that hosts this Hosting Environment:</u>

> *Operating_System*

<u>Component Type Enumeration Rule for this Hosting Environment:</u>

<Web Application Server Product Name>

<u>Component Types for this Hosting Environment based on Enumeration Rule:</u>

- *WebSphereApplicationServer*
- *WebLogicApplicationServer*

- *OracleWebApplicationServer*
- *SunONEWebApplicationServer*
- *TomCatWebApplicationServer*
- *JBossWebApplicationServer*

<u>Component Types hosted in this Hosting Environment:</u>

Components of this Hosting Environment	**Components Hosted by this Hosting Environment**
• *EAR_File* • *WAR_File* • *J2EE_Application* • *Web_Module* • *EJB_Module* • *Transport* • *Resource_Provider* • *Virtual_Host* • *Alias_Name* • *WebAppServer_Mail_Provider* • *WebAppServer_URL_Provider* • *WebAppServer_JDBC_Provider* • *WebAppServer_JMS_Provider* • *WebAppServer_Resource_Adapter* • *WebAppServer_DataSource* • *WebAppServer_Authentication_Mechanism* • *WebAppServer_User_Registry* • *Web_Application*	• *Portal_Server* • *Workflow_Engine* • *Commerce_Server* • *WebAppServer_Deployment_Manager* • *WebAppServer_JMS_Server* • *WebAppServer_Node* • *WebAppServer_Web_Container* • *WebAppServer_EJB_Container*

<u>Catgegorization (Grouping) Component Types:</u>

- *WebApplicationServer_Cluster*
- *J2EE_Application_Server*
- *WebSphere_Cell*

- *WebApplicationServer*

1.16.1.1 3.1.3 The Relational_Database Hosting Environment

Namespace:

http:// www.ibm.com/namespaces/autonomic/DM_componentTypes

Hosting Environment:

Relational_Database

Hosting Environment that hosts this Hosting Environment:

Operating_System

Component Type Enumeration Rule for this Hosting Environment:

<Relational Database Product Name>_<Optional Relational Database Family>_<Optional Relational Database Role>

where,

Family is   Enterprise, Workgroup, Personal, Express (DB2)

DynamicServer, OnlineExtendedEdition, SE (Informix)

Role is Client,Server

Component Types for this Hosting Environment based on Enumeration Rule:

- *IBMDB2UDB*
- *Informix*

- *Sybase*
- *Oracle*
- *MicrosoftSQL*
- *LDAP(??)*

Component Types hosted in this Hosting Environment:

Component Types that do NOT explicitly provide a Hosting Environment	Components Hosted by this Hosting Environment
- *RelationalDatabase_Application* - *RelationalDatabase_Connection* - *RelationalDatabase_Instance* - *RelationalDatabase_Table* - *RelationalDatabase_Tablespace*	- *RelationalDatabase_Node* - *RelationalDatabase_APPC_Node* - *RelationalDatabase_APPCLU_node* - *RelationalDatabase_APPN_Node* - *RelationalDatabase_NETBIOS_Node*

- *RelationalDatabase_Tablespace_Container*
- *RelationalDatabase_TCPIP_Node*
- *RelationalDatabase_LDAP_Node*
- *RelationalDatabase_Local_Node*

Catgegorization (Grouping) Component Types:

- *RelationalDatabase-NodeGroup*

1.174.0 List of Component Type Values by Namespace

4.1 Web Application Server Component Types
(http://www.ibm.com/namespaces/autonomic/WebAppServer_componentTypes)

- *WebSphereApplicationServer*
- *WebLogicApplicationServer*

-

- *EAR_File*
- *WAR_File*
- *J2EE_Application*
- *Web_Module*
- *EJB_Module*
- *Transport*
- *Resource_Provider*
- *Virtual_Host*
- *Alias_Name*
- *WebAppServer_Mail_Provider*
- *WebAppServer_URL_Provider*
- *WebAppServer_JDBC_Provider*
- *WebAppServer_JMS_Provider*
- *WebAppServer_Resource_Adapter*
- *WebAppServer_DataSource*
- *WebAppServer_Authentication_Mechanism*
- *WebAppServer_User_Registry*
- *Web_Application*

- *WebApplicationServer_Cluster*
- *J2EE_Application_Server*
- *WebSphere_Cell*

- *OracleWebApplicationServer*
- *SunONEWebApplicationServer*
- *TomCatWebApplicationServer*
- *JBossWebApplicationServer*

- *Portal_Server*
- *Workflow_Engine*
- *Commerce_Server*
- *WebAppServer_Deployment_Manager*
- *WebAppServer_JMS_Server*
- *WebAppServer_Node*
- *WebAppServer_Web_Container*
- *WebAppServer_EJB_Container*

- *WebApplicationServer*

4.2 DataManagement Component Types
(http:// www.ibm.com/namespaces/autonomic/DM_componentTypes)

62

• *IBMDB2UDB*	• *Sybase*
•	• *Oracle*
•	• *MicrosoftSQL*
•	• *LDAP*
•	
• *Informix_DynamicServer*	
• *Informix_OnlineExtendedEdition*	
• *Informix_SE*	
• *RelationalDatabase_Application*	• *RelationalDatabase_Node*
• *RelationalDatabase_Connection*	• *RelationalDatabase_APPC_Node*
• *RelationalDatabase_Instance*	• *RelationalDatabase_APPCLU_node*
• *RelationalDatabase_Table*	• *RelationalDatabase_APPN_Node*
• *RelationalDatabase_Tablespace*	• *RelationalDatabase_NETBIOS_Node*
• *RelationalDatabase_Tablespace_Container*	• *RelationalDatabase_TCPIP_Node*
	• *RelationalDatabase_LDAP_Node*
	• *RelationalDatabase_Local_Node*
• *RelationalDatabase-NodeGroup*	

4.3 OperatingSystem Component Types

(http:// www.ibm.com/namespaces/autonomic/OS_componentTypes)

• *RedHatLinux*	• *MicrosoftWindows_98*	• *MicrosoftWindows_NT_Server*
• *SuSELinux*	• *MicrosoftWindows_ME*	• *MicrosoftWindows_XP_Professional*
• *UnitedLinux*	• *MicrosoftWindows_NT_Workstation*	• *MicrosoftWindows_2003_Server*
• *SunSolaris*	• *MicrosoftWindows_NT_Server*	• *MicrosoftWindows_2003_AdvancedServer*
• *IBMAIX*	• *MicrosoftWindows_2000_Workstation*	
• *HPUX*	• *MicrosoftWindows_2000_Server*	
• *NovellNetware*	• *MicrosoftWindows_2000_AdvancedServer*	
• *IBMMVS*	• *MicrosoftWindows_XP_Home*	
• *IBMOS400*		
• *Language_Runtime*	• *Relational_Database*	
• *Device_Driver*	• *Web_Application_Server*	
• *Software*	• *Messaging*	
• *Process*		
• *Thread*		
• *TCPIP_port*		
• *Windows*	• *Operating_System*	
• *Windows-Win32*	• *MicrosoftWindows_NT*	
• *UNIX*	• *MicrosoftWindows_2000*	

• *POSIX*	• *MicrosoftWindows_XP*	
• *Linux*	• *MicrosoftWindows_2003*	

References

[RFC2119] *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt

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

XML Schema Part 0: Primer

[GUID] Leach and Salz, Internet Engineering Task Force, *UUIDs and GUIDs*, Internet draft `draft-leach-uuids-guids-01.txt`, http://www.opengroup.org/dce/info/draft-leach-uuids-guids-01.txt

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

DMTF - Standards - Common Information Model (CIM) Specification Version 2.2
COMMON INFORMATION MODEL (CIM) INDICATIONS (FINAL DRAFT)