Canonical Situation Data Format: The Common Base Event ACAB.BO0301.2.0

Last Modified: 10/13/2003 8:33:05 PM

Deleted: 9/10/2003 6:10:42 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

Document Control

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

Table of Contents

Deleted: 26

1.0 *PURPOSE*

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

This document is prescriptive about the format and content of the data that is passed or retrieved from a 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. Quality event data leads to accurate, deterministic and proper management of the enterprise. Poor fidelity can lead to misguided, potentially harmful or fatal results. The results of this effort is a specification for the "Common Base Event" definition that serves as a new standard for events among enterprise management and business applications.

2.0 *Overview*

A small event can change things far beyond the seeming initial circumstance. Nowhere is this truer 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 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. Fundamental aspects of enterprise management and e-business communication, such as performance monitoring, security and reliability management, 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 or fatal results. 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 if the format used by the sender is not understood by the receiver. 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 expensive, hard to create implementations when 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 the basic problem of parsing time stamps. Format and granularity (for example, are the units milliseconds or microseconds?) present needless obfuscation for the application that receives the time-stamped event. A more general problem is the lack of consistency in the information that is presented. For example:

- Which component experienced the situation?
- When did the situation occur?
- Are the component that experienced the situation and the component that is reporting the situation on the same physical machines?
- What is the identifier for the component that experienced the situation?

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, besides providing definitions and requirements for the normal meta-data, 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. A common description of the *situation* that occurred
4. The content that can be used to correlate situations.

All properties defined in this model, except the meta-data, are hereafter referred to as the 4-tuple.

2.1. The scope of this work

The goal of this work is to define the fundamental properties that **must** be reported when a situation occurs. This data will be used by management applications to evaluate what is happening in the system. There are some fields defined in this document that represent the reality that in order for the CBE to be useful today, some legacy data that is used by existing management functions must be made available. We have attempted to call out specifically which of the data elements in this specification are here for legacy or compatibility purposes.

In this document, the format of the CBE is first described in UML so it is easier for the reader to understand the overall structure. The CBE format and content is then rendered as an XML schema for concrete implementations. This document's scope is limited to the 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 4-tuple described earlier.

2.2. Terminology and Notation

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

2.2.1. Notational Convention

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

2.2.2. 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.

2.2.3. 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[2]. The data types are XML schema signed data types.

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

2.2.4. String Length

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

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

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.

2.2.5. Case Sensitivity

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

3.0 *Extensibility*

The Common Base Event data model described here is built with some level of extensibility to lend itself easily to the needs of a variety of types of events generated by different IT and business products. In 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.

If there is a need for product specific data, it is RECOMMENDED that the ExtendedDataElement (described later on page 38) be used to contain that data. This property allows for user-supplied extensions for any attributes not defined in the Common Base Event.

Deleted: 36

The event consumers or management tools that support the Common Base Event 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.

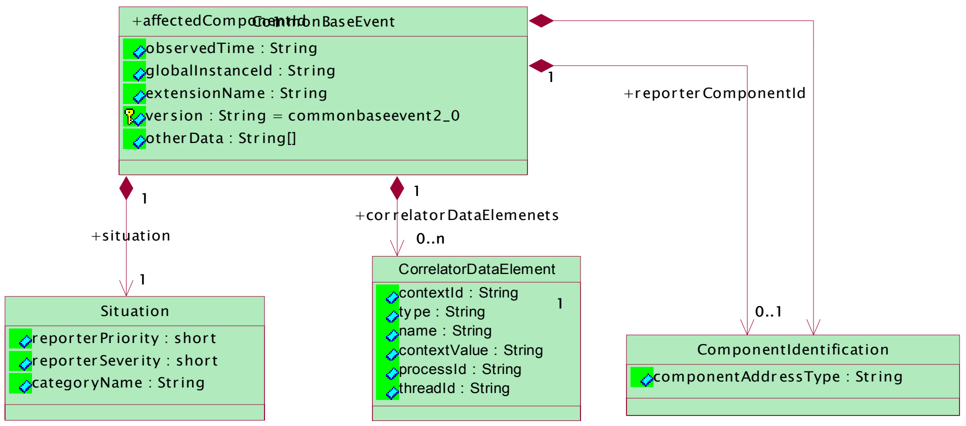
Multiple versions of the XML Schema are supported by appending a version number to the XML schema file name. The initial (current) XML file name is therefore, commonbaseevent1_0.xsd. Also, the CommonBaseEvent schema provides a “version” token in the schema declaration (CommonBaseEvent XML Schema on page 48)

Deleted: 45

4.0 *Common Base Event*

4.1. CommonBaseEvent Description

The common base event, as mentioned above, defines the 4-tuple of data that must be collected when a situation occurs. The UML below describes the data for the 4-tuple. Table 1 is a summary of the CommonBaseEvent properties; that is the 4-tuple that is collected for a situation.. A detailed description of the CommonBaseEvent follows the summary table. The table consists of the meta-data associated with the situation, followed by the 4-tuple.



Property Name	Type	Description
observedTime	xsd:dateTime	This is meta-data. 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 observedTime MUST provide granularity as precisely as the generating platform allows. This is a REQUIRED property.
globalInstanceId	xsd:ID	This is meta-data. This is the primary identifier for the event. This property MUST be globally unique and MAY be used as the primary key for the event. This property is provided for management functions that require events to have an identifier. Once this value is set it MUST NOT 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) This is an OPTIONAL property.
extensionName	xsd:Name	This is meta-data This property is provided for management functions that require events to be named. This

		property is used when a component wants to group a set of events within a situation category. The maximum string length for extensionName MUST NOT exceed 64 characters. This is an OPTIONAL property.
version	xsd:string	A string identifying the version of this event. This is an OPTIONAL property. The maximum string length for version MUST NOT exceed 16 charcters.
affectedComponentId	cbe:ComponentIdentification	Part of the 4-tuple. Identification of the component that is "affected" or is "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.
situation	cbe:Situation	This field provides critical data about the event, including the situation category, which will be used by management functions. This is a REQUIRED property.
correlatorDataElements	cbe:CorrelatorDataElement[]	An array of contexts that this event is referencing. See the CorrelatorDataElement definition (described on page 45) for details. This is an OPTIONAL property.
otherData	xsd:any	

Deleted: 42

Table 1: CommonBaseEvent

Detailed description of the CommonBaseEvent 4-tuple is described in the following sections:

4.1.1. observedTime

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

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.

4.1.2. globalInstanceId

The purpose of the globalInstanceID is to uniquely identify an instance of an event. This field is provided for management functions that require an event to have an instance ID. This field will be filled in either by a component reporting the situation or the consumer of the event. 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.

There are several ways to generate a GUID. 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.

4.1.2.1. extensionName

This field is used for management functions that require events to be named. The extensionName property allows components to group events within a situation category. For example, for the situation category of "AVAILABLE", a component might have need to identify a set of those AVAILABLE situations as representing the event "EIRGP_Neighbor_down".

This is an OPTIONAL property, which is not mutable and may only be provided by the component that reports the event.

4.1.2.2. version

The version attribute is of type string and is used to identify the version of a given event such that it can be recognized and managed appropriately by consumers of the event. In future versions of CBE, this field can be used to assist receivers with parsing and migration of "old" CBEs by providing a marker for backward compatibility processing procedures.

This is an OPTIONAL property that is mutable in the case where a migration procedure has upgraded a CBE to a newer version. The version may be provided either by the component issuing the event or by the consumer of the event.

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

4.1.3. affectedComponentId

This is part of the 4-tuple. The affectedComponentId is the identification of the component that was affected by 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 once it is set it MUST NOT be changed. The producer of the event MUST provide the affectedComponentId. If the reporter and the affected components are the same, then the reporterComponentId MUST NOT be present.

4.1.4. reporterComponentId

This is part of the 4-tuple. 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.

4.1.5. situation

This is part of the 4-tuple. The situation is the data that describes the situation or event 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 to allow for adoption to product-specific requirements.

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

This is a REQUIRED property.

4.1.6. correlatorDataElements

This is part of the 4-tuple. This property holds the contextual data that is used to correlate events or situations. The actual value supplied in this field is defined by the component that fills in the value. The correlatorDataElements is an array of contexts of type CorrelatorDataElement (described on page 45).

Deleted: 42

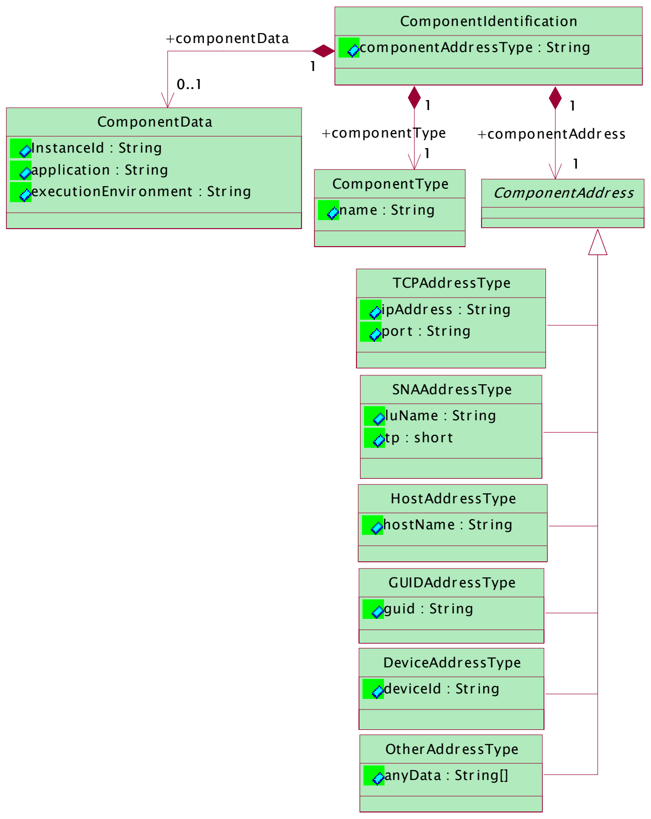
This is an OPTIONAL property and is not mutable that is, once it is set it MUST NOT be changed. It MAY be provided by the component that issues the event or MAY be assigned by the consumer of the event.

4.2. *ComponentIdentification Description*

This section describes the pieces of data that are associated with a component. In the 4-tuple the components that are identified are the component that reports the situation (the reporter component), and the component that experiences the situation (the affected component). In some cases, the reporter and affected components may be the same. The component identification field defines a collection of attributes that are **required** to uniquely identify a component.

In the future, all components will be identified using the same identification mechanism, such as a GUID. In the near term, since components are already identified using differing techniques, the CBE structure must accommodate for, identify, and structure the differing identification techniques. As an example of the differing techniques used to identify components consider two sample components: an Ethernet adapter in a router and an SNA application. The Ethernet adapter is identified by a TCPIP address, whereas an SNA component will be identified using an LU name and a TP name. In both cases, the components are identified using a set of well-known fields: IP address and LU/TP name. This example demonstrates that the fields used to identify a component are 'type' sensitive. The CBE structure used for component identification reflects this 'type' sensitivity.

Table 2 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
componentAddressType	cbe:ComponentAddressType	Specifies the **type** of the componentAddress field. This field defines what the expected values are for the componentAddress field. It is a REQUIRED property.
componentAddress	cbe:ComponentAddress	This field is defined by the componentAddressType field. It contains the specific required fields used to identify the component of type 'componentAddressType'. This is a REQUIRED property.
componentType	cbe:ComponentType	The componentAddressType is a well-defined name that is used to

		characterize all instances of a given kind of component.
componentData	cbe:ComponetData	componentData is a collection of properties that further assist identifying the source or reporter componets of a situation. For detatils refer to the CompontData type. This is an OPTIONAL property, used mainly to carry data that will be used by existing management functions.

Table 2: Component Identification

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

4.2.1. ComponentAddressType

This field defines the type of information that will be found in the componentAddress field. As mentioned above, in the ideal world, there would be one value for componentAddressType, something like typeGUID, and the value for the componentAddress field would be a GUID. In the near-term, there will be a set of well-defined componentAddressType values that allow components to be identified using well known values.

Some well-known componentAddressType values are: (this list should grow based on input from the standards body)

- TCPAddressTypev4
- TCPAddressTypev6
- SNAAddressType
- HostAddressType
- DeviceAddressType
- GUIDAddressType

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

4.2.2. componentAddress

The componentAddress specifies the component address specific data of the event that was reported. The categorization of component address facilitates the identification of the source of event.

The data type for this property is a complex type. The componentAddress are defined below.

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

4.2.3. For componentAddress TCPAddressType

For TCPComponentAddressType the component will be identified using the IP address. This componentAddress is identified by the following two required properties:

- IPAddress (IPV 4 or IPV6), for example 9.27.40.112
- PortNumber, for example 5022

4.2.4. For componentAddress SNAAddressType

For the SNAComponentAddressType the component is identified using the LU name. This componentAddress is identified by the following two required properties:

- luName , for example xxxxx.
- tp, for example xxxx.

4.2.5. For componentAddress HostAddressType

For the HostComponentAddressType the component is identified using the hostname. This componentAddress is identified by the following required property:

- hostname, for example server1.

4.2.6. For componentAddress DeviceAddressType

For the DeviceComponentAddressType the component is identified using the serial number for the device. This componentAddress is identified by the following required property:

- serialnumber, for example X1234ABC72

4.2.7. For componentAddress GUIDAddressType

For the GUIDComponentAddressType the component is identified using a GUID generated by the system. This componentAddress is identified by the following required property:

- GUID, for example A123400AA00389B71

4.2.8. For componentAddress OtherAddressType

For the GUIDComponentAddressType the component is identified using a GUID generated by the system. This componentAddress is identified by the following required property:

- GUID, for example A123400AA00389B71

4.2.9. componentType

Besides having a componentAddress and a componentAddressType, every component has an associated type. A componentType is defined as an XML Name that contains a name space qualified name, for example:

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

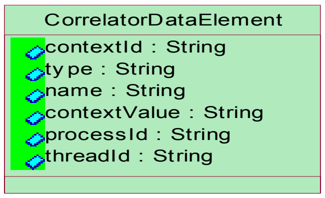
The componentType namespace is divided into several name spaces to facilitate delegation to various product groups. The default namespace for component types is described in Appendix Component Type Namespace. Namespaces other than the default namespace will be allowed for component type extensibility.

This property is REQUIRED and it is not mutable that is, once it is set it MUST NOT be changed. The maximum string length for componentAddressType MUST NOT exceed 512 characters.

4.3. *ComponentData Description*

The ComponentData property is a collection of properties that may further assist identifying the affected or reporter components of a situation. Most of the data carried in this field is data that is used by existing management functions, and is carried here for compatibility purposes.

Table 3 is a summary of the properties for the ComponentData type. A detailed description of the ComponentData follows the summary table.



Property Name	Type	Description
application	xsd:string	For some legacy systems, the component is associated with an application name. This application name is usually something like “myWebApp” or “flowersByWebsphere:. 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	xsd: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.
instanceId	xsd: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)[4] or EJBHandle). This is an OPTIONAL property. The maximum string length for instanceId MUST NOT exceed 128 characters.

Table 3: Component Identification

4.3.1. application

In many legacy systems, the component is associated with an application name, such as "myWebApp". The application property specifies this user-identified name of a component. 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 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.

4.3.2. 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.

4.3.3. instanceId

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

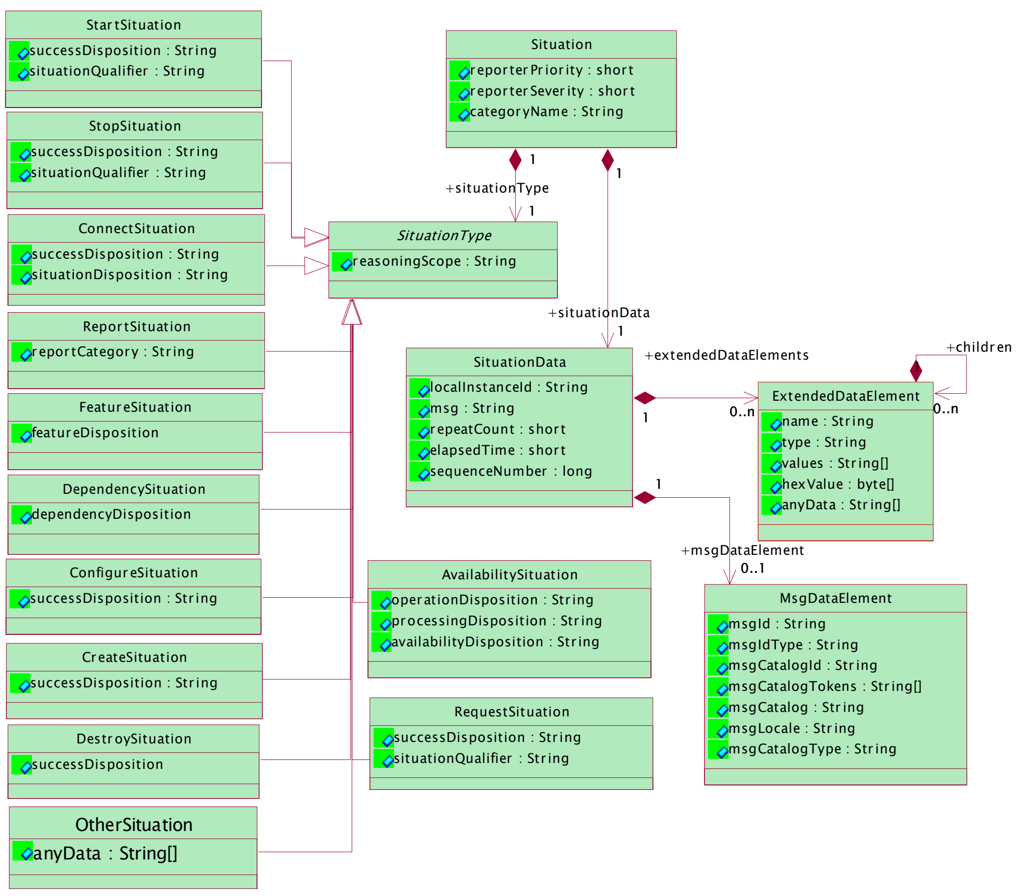
[4] See Grid Services Specification Draft 4, Global Grid Forum, http://www.gridforum.org/ogsi.wg

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

4.4. *Situation Description*

The situation part of the 4-tuple is a collection of data that a component reports for external consumption either by a general management application or by a product-specific manager. A situation represents the categorization of a state change into at least one pre-defined category. The collection specified here are required set of properties or attributes that are common across products groups and platforms, yet architected and flexible to allow for adoption to product-specific requirements.

Table 4 is a summary of the properties for the SituationType. A detailed description of the SituationType properties follows the summary table.



Property Name	Type	Description
situationType	cbe:SituationType	The situationType specifies the type of the situation that caused the event to be reported. See SituationType definition for details. This is a REQUIRED property.
reasoningScope	cbe:ReasoningScopeType	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	cbe:SituationData	This property allows specifying any situation specific data that can assist to further qualifying the situation. This property is provided to accommodate legacy systems that require additional data. See SituationData definition for details. This is an OPTIONAL property.
reporterSeverity	xsd:string	The perceived severity of the status the event is describing with respect to the application that reports the event. This field is provided for management functions that require an event to have a severity. 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.
reporterPriority	xsd:string	This property defines the importance of the event. This field is provided for management functions that require an event to have a priority. 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.
categoryName	cbe:CategoryNameType	This property categorizes the type of the situation that caused the event to be reported. The current initial values are: • StartSituation • StopSituation • ConnectSituation • RequestSituation • ConfigureSituation • FeatureSituation • CreateSituation • DestroySituation • ReportSituation • AvailableSituation • DependencySituation • OtherSituation This is a REQUIRED property and once it

		is set it MUST NOT change.

Table 4: Situation

Detailed description of Situation is described in the following sections:

4.4.1. categoryName

This property categorizes the type of the situation that caused the event to be reported. The current initial values are:

- StartSituation
- StopSituation
- ConnectSituation
- RequestSituation
- ConfigureSituation
- FeatureSituation
- CreateSituation
- DestroySituation
- ReportSituation
- AvailableSituation
- DependencySituation
- OtherSituation

This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2. situationType

The situationType specifies the type or category 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. The situation types or categories are defined below.

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 that 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, largely, 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.

The following sections outline the well-known and acceptable values for the situationType property.

4.4.2.1. StartSituation

The StartSituation deals 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, for example:

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}

The StartSituation includes the following properties:

Property Name	Type	Description
successDisposition	cbe:SuccessDispositionType	This property specifies the success disposition of an operation of a situation that caused the situation to be reported. The successDisposition is of type SuccessDispositionType with the following set of values: • SUCCESSFUL • UNSUCESSFUL This is a REQUIRED property and once it is set it MUST NOT change.
situationQualifier	cbe:StartSituationQaulifierType	This property specifies the situation qualifiers that are representation of the parameters necessary to describe the situation. The situationQualifier is of type StartSituationQaulifierType with the following set of values: • START INITIATED • RESTART INITIATED • START COMPLETED This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.2. StopSituation

The StopSituation deals 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, for example:

WSVR0220I: Application stopped: {0}
WSVR0102E: An error occurred stopping, {0}
MSGS0657I: Stopping the MQJD JMS Provider

The StopSituation includes the following properties:

Property Name	Type	Description
successDisposition	cbe:SuccessDispositionType	This property specifies the success disposition of an operation of a situation that caused the situation to be reported. The successDisposition is of type SuccessDispositionType with the following set of values: • SUCCESSFUL • UNSUCESSFUL This is a REQUIRED property and once it is set it MUST NOT change.
situationQualifier	cbe:StopSituationQaulifierType	This property specifies the situation qualifiers that are representation of the parameters necessary to describe the situation. The situationQualifier is of type StopSituationQaulifierType with the following set of values: • STOP INITIATED • ABORT INITIATED • PAUSE INITIATED • STOP COMPLETED This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.3. ConnectSituation

The ConnectSituation deals with the situations that 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, for example:

DBMN0015W: Failure while creating connection {0}
DBMN0033W: connection close failure {0}
DBMN0023W: Failed to close a connection {0}

The ConnetSituation includes the following properties:

Property Name	Type	Description
successDisposition	cbe:SuccessDispositionType	This property specifies the success disposition of an operation of a situation that caused the situation to be reported. The successDisposition is of type SuccessDispositionType with the following set of values: • SUCCESSFUL • UNSUCESSFUL This is a REQUIRED property and once it is set it MUST NOT change.
situationDisposition	cbe:ConnectSituationDispositionType	This property specifies the situation disposition that is representation of the parameters necessary to describe the situation. The situationQualifier is of type ConnectSituationDispositionType with the following set of values: • INUSE • FREED • CLOSED • AVAILABLE This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.4. RequestSituation

The RequestSituation deals with the sitautions 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 complete, for example:

ADMS0003I: Configuration synchronization completed

The RequestSituation includes the following properties:

Property Name	Type	Description
successDisposition	cbe:SuccessDispositionType	This property specifies the success disposition of an operation of a situation that caused the situation to be reported. The successDisposition is of type SuccessDispositionType with the following set of values: • SUCCESSFUL • UNSUCESSFUL This is a REQUIRED property and once it is set it MUST NOT change.
situationQualifier	cbe:RequestSituationQualifier Type	This property specifies the request qualifiers that are representation of the parameters necessary to describe the situation. The situationQualifier is of type RequestSituationQualifierType with the following set of values: • REEQUEST INITIATED • REEQUEST COMPLETED This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.5. ConfigureSituation

The ConfigureSituation deals with the components identifying 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, for example:

ADFS0134I: File transfer is configured with host="9.27.11.13", port="9090", securityEnabled="false".

The ConfigureSituation includes the following properties:

Property Name	Type	Description
successDisposition	cbe:SuccessDispositionType	This property specifies the success disposition of an operation of a situation that caused the situation to be reported. The successDisposition is of type SuccessDispositionType with the following set of values:

		• SUCCESSFUL • UNSUCESSFUL This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.6. AvailableSituation

The AvailableSituation deals with the situations 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, for example:

ADMC0013I: SOAP connector available at port 8880
ADMC0026I: RMI Connector available at port 2809

The ConfigureSituation includes the following properties:

Property Name	Type	Description
operationDisposition	cbe: OperationDispositionType	This property specifies the operation state of the component reported by the situation. The operationalDisposition is of type OperationDispositionType with the following set of values: • STARTABLE • NONSTARTABLE This is an REQUIRED property and once it is set it MUST NOT change.
availablityDisposition	cbe:AvailabilityDispositionType	This property specifies the availability disposition of an entity or component that caused the situation to be reported. The availableDisposition is of type AvailabilityDispositionType with the following set of values: • AVAILABLE • NOT AVAILABLE This is a REQUIRED property and once it is set it MUST NOT change.
processingDisposition	cbe:ProcessingDispositionType	This property specifies the processing disposition of a component opertation that caused the situation to be reported.

		The processingDisposition is of type ProcessingDispositionType with the following set of values: • FUNCTION_PROCESS • FUNCTION_BLOCK • MGMTTASK_PROCESS • MGMTTASK_BLOCKED This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.7. ReportSituation

The ReportSituation deals with the situations 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, for example:

IEE890I WTO Buffers in console backup storage = 1024

The ReportSituation includes the following properties:

Property Name	Type	Description
reportCategory	cbe:ReportCategoryType	This property specifies the category of the reported situation. The reportCategory is of type ReportCategoryType with the following set of values: • PERFORMANCE • SECURITY • HEARTBEAT • STATUS This is an REQUIRED property and once it is set it MUST NOT change.

4.4.2.8. CreateSituation

The CreateSituation deals with the situations 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, for example:

ADMR0009I: Document cells/flatfootNetwork/applications/Dynamic Cache Monitor.ear/Dynamic Cache Monitor.ear was created

The CreateSituation includes the following properties:

Property Name	Type	Description
successDisposition	cbe:SuccessDispositionType	This property specifies the success disposition of an operation of a situation that caused the situation to be reported. The successDisposition is of type SuccessDispositionType with the following set of values: • SUCCESSFUL • UNSUCESSFUL This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.9. DestroySituation

The DestroySituation deals with the situations 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, for example:

CONM6007I: The connection pool was destroyed for data source (UDDI.Datasource.techs8.server1).

The FeatureSituation includes the following properties:

Property Name	Type	Description
successDisposition	cbe:SuccessDispositionType	This property specifies the success disposition of an operation of a situation that caused the situation to be reported. The successDisposition is of type SuccessDispositionType with the following set of values: • SUCCESSFUL • UNSUCESSFUL This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.10. FeatureSituation

The FeatureSituation deals with the situations that announce that a feature of a component is now ready (or not ready) for service requests. Situations that indicate things like services being available and services or features being unavailable fall into this category. Existing situations include words like now available, currently available, and transport is listening on port 123, for example:

SRVE0171I: Transport HTTPS is listening on port 9443
MSGS0601I: WebSphere Embedded Messaging has not been installed

The FeatureSituation includes the following properties:

Property Name	Type	Description
featureDisposition	cbe:FeatureDispositionType	This property specifies the availability disposition of a feature of a component that caused the situation to be reported. The featureDisposition is of type FeatureDispositionType with the following set of values: • AVAILABLE • NOT AVAILABLE This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.11. DependencySituation

The DependencySituation deals with the situations 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, for example:

WSVR0017E: Error encountered binding the J2EE resource, Pet Store JMS Queue Connection Factory, as jms/queue/QueueConnectionFactory from resources.xml no resource binder found

The DependencySituation includes the following properties:

Property Name	Type	Description
dependencyDisposition	cbe:DependencyDispositionType	This property specifies the dependency disposition of a feature of a component that caused the situation to be reported. The featureDisposition is of type DependencyDispositionType with the following set of values: • MET • NOT MET This is a REQUIRED property and once it is set it MUST NOT change.

4.4.2.12. OtherSituation
The OtherSituation category is to provide support for the situation that is product specific requirement other than the existing categories.

4.4.3. reasoningScope
This property specifies the scope of the situation. The reasoningScope defines whether this situation has an internal only impact or has a potential external impact. The reasoningScope is of type ReasoningScopeType and has the following initial set of values.

- INTERNAL
- EXTERNAL

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

4.4.4. situationData
This property allows specifying any situation specific data that can assist to further qualifying the situation. This property is provided for legacy management functions that need additional data. See SituationData definition. for details.

This is an OPTIONAL property.

1.1.1 reporterSeverity
The reporterSeverity indicates the event status severity level with respect to the component that reports the event. This property is provided for compatibility with management functions that require events to have a severity. 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 reporterSeverity 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.

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.

4.4.5. reporterPriority

The reporterPriority defines the importance of the event and the relative order in which the event records SHOULD be processed. This field is provided for compatibility with management functions that require events to have a priority. 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.

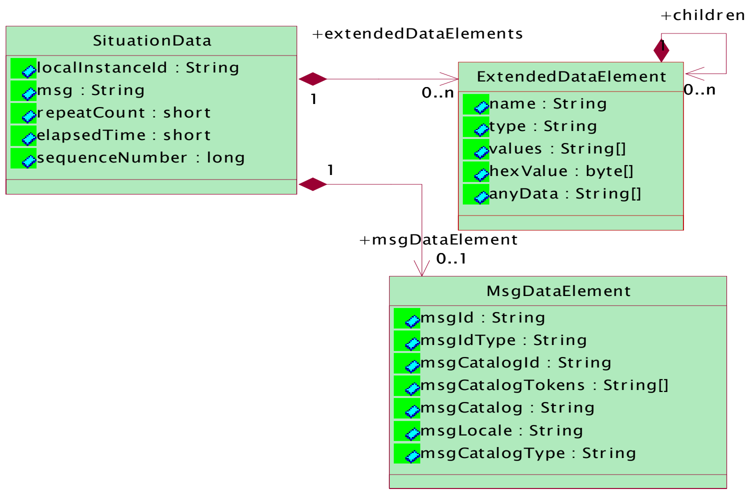
The reporterPriority property is independent of severity, because priority is intended to be used primarily by event consumers, whereas severity indicates the state of the situation as perceived by the affected component. For example, an event with reporterPriority HIGH and severity MINOR should be processed before an event with reporterPriority LOW and severity CRITICAL.

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

4.5. *SitautionData Description*

The SituationData allows describing a collection of data that a component reports for external consumption either by a general management application or by a product-specific manager. This property is provided to allow compatibility with existing management functions

Table 5 is a summary of the properties for the SituationData. A detailed description of the SitauationDat properties follows the summary table.



Property Name	Type	Description
localInstanceId	xsd: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.
msg	xsd: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.
msgDataElement	cbe:MsgDataElement	Identification of the message that this event holds. See the MsgDataElement definition. This is an OPTIONAL property.
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.
repeatCount	xsd: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	xsd: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.
sequenceNumber	xsd: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 5: SituationData

Detailed description of SituationData is described in the following sections:

4.5.1. localInstanceId

The localInstanceId is of type string and is used to locally identify instances of an event. There is no implied guarantee that this value is globally unique 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.

4.5.2. 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. 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.

4.5.3. 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.

4.5.4. extendedDataElements

The extendedDataElements property is a sequence of name elements of type ExtendedDataElement (described on page 38). 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.

Deleted: 36

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.

4.5.5. 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 field is provided for compatibility with management functions that require a repeatCount.

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.

4.5.6. 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. This property is provided for compatibility with management functions that require repeatCount and elapsedTime.

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.

4.5.7. 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 is provided for compatibility with components that generate multiple related events and currently allow for those events to be sequenced.
This property is OPTIONAL and it is not mutable once it is set. There is no default value.

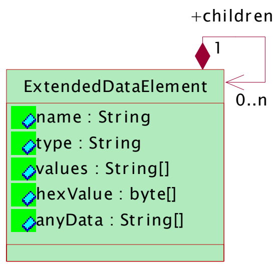
4.6. *ExtendedDataElement Description*

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. This field is provided both for extensibility and for compatibility with existing management functions.

Table 6 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	xsd: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	xsd: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** • **any** These are the only valid data types for the ExtendedDataElement type. The default value is string.
values	xsd: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	xsd: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.
anyData	xsd:any	Any data type value required by product specific requirements.

Table 6: ExtendedDataElement

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

4.6.1. 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.

4.6.2. 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
- any

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.

4.6.3. 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.

Note: The hexValue, values, anyData properties are mutually exclusive. Only one of these three properties SHALL be defined.

4.6.4. 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, values, anyData properties are mutually exclusive. Only one of these three properties SHALL be defined.

4.6.5. 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.

4.6.6. anyData

The anyData property provide support for the other product specific element. The anyData MUST be specified when the *type* property is set to “any”.

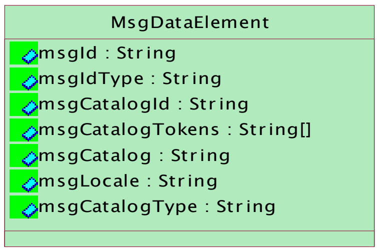
This property is OPTIONAL and it is not mutable. It MUST NOT be specified if the “values” property is specified.

Note: The hexValue, values, anyData properties are mutually exclusive. Only one of these three properties SHALL be defined.

4.7. *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. This field is provided for compatibility with existing management functions.

Table 7 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	xsd: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	xsd: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	xsd: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	xsd: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	xsd: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	xsd: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 7: MsgDataElement

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

4.7.1. 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.

4.7.2. 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.

4.7.3. 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.

4.7.4. 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.

4.7.5. 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.

4.7.6. 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.

4.7.7. msgCatalogType

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

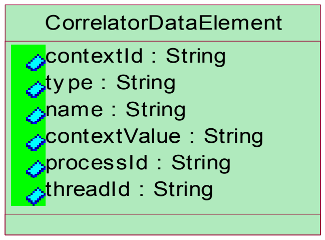
- Java
- XPG

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.

4.8. *CorrelatorDataElement Description*

The correlatortDataElement 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 8 provides a summary of the data properties representing a context in the CBE. A detailed description of this CorrelatorDataElement follows the summary table.



Property Name	Type	Description
type	xsd:Name	The data type of the contextValue property. This is a REQUIRED property.
name	xsd:Name	Name of the application that created this context data element. This is a REQUIRED property.
contextValue	xsd:string	The value of the context with respect to the implementation of the context. This is REQUIRED unless contextId specifies a value.
contextId	xsd:IDREF	This property is the reference to the element that contains the context. This is REQUIRED unless contextValue specifies a value.
processId	xsd: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	xsd: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 8: CorrelatorDataElement

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

4.8.1. 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

4.8.2. 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.

4.8.3. 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.

4.8.4. 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.

4.8.5. 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.

4.8.6. threadId

The threadId property is of type string and identifies the thread ID of the component or subcomponent indicated by the process ID that generated the event. A running process may spawn one or more threads to 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.

5.0 *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/commonbaseevent2_0"
targetNamespace="http://www.ibm.com/AC/commonbaseevent2_0" version="2.0">
        <xsd:complexType name="CommonBaseEvent">
                <xsd:sequence>
                        <xsd:element name="affectedComponentId" type="cbe:ComponentIdentification" minOccurs="1"
                        maxOccurs="1" />
                        <xsd:element name="reporterComponentId" type="cbe:ComponentIdentification" minOccurs="0"
                        maxOccurs="1" />
                        <xsd:element name="correlatorDataElements" type="cbe:CorrelatorDataElement" minOccurs="0"
                        maxOccurs="unbounded" />
                        <xsd:element name="situation" type="cbe:Situation" minOccurs="1" maxOccurs="1" />
                        <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded" />
                </xsd:sequence>

                <xsd:attribute name="globalInstanceId" use="optional">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:ID">
                                        <xsd:minLength value="32"></xsd:minLength>
                                        <xsd:maxLength value="64"></xsd:maxLength>
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="observerTime" type="xsd:dateTime" use="required" />
                <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="version" type="xsd:string" use="optional" />
        </xsd:complexType>
        <xsd:element name="CommonBaseEvent" type="cbe:CommonBaseEvent" />

        <xsd:complexType name="Situation">
                <xsd:sequence>
                        <xsd:element name="situationData" type="cbe:SituationData" minOccurs="1" maxOccurs="1" />
                        <xsd:element name="situationType" type="cbe:SituationType" minOccurs="1" maxOccurs="1" />

                </xsd:sequence>
                <xsd:attribute name="reporterSeverity" type="cbe:SeverityType" use="optional" />
                <xsd:attribute name="reporterPriority" use="optional" type="cbe:PriorityType" />
                <xsd:attribute name="categoryName" type="cbe:CategoryNameType" use="required"/>
        </xsd:complexType>

        <xsd:complexType name="SituationData">
                <xsd:sequence>
                        <xsd:element name="extendedDataElements" type="cbe:ExtendedDataElementType" minOccurs="0"
                        maxOccurs="unbounded" />
                        <xsd:element name="msgDataElement" type="cbe:MsgDataElementType" minOccurs="0"
                        maxOccurs="1" />
                </xsd:sequence>
                <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="msg" use="optional">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="1024" />
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>

                <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>

                <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>

                <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="SituationType" abstract="true">
                <xsd:attribute name="reasoningScope" type="cbe:ReasoningScopeType" use="required"/>
            </xsd:complexType>

            <xsd:complexType name="StartSituation" abstract="false">
                <xsd:complexContent>
                    <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                            <xsd:element name="successDisposition" type="cbe:SuccessDispositionType"
                        minOccurs="1" maxOccurs="1" />
                        <xsd:element name="situationQualifier" type="cbe:StartSituationQualifierType" minOccurs="1"
                        maxOccurs="1" />
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>

            <xsd:complexType name="StopSituation" abstract="false">
                <xsd:complexContent>
                    <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                            <xsd:element name="successDisposition" type="cbe:SuccessDispositionType"
                            minOccurs="1" maxOccurs="1" />
                            <xsd:element name="situationQualifier" type="cbe:StopSituationQualifierType"
                            minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>

            <xsd:complexType name="ConnectSituation" abstract="false">
                <xsd:complexContent>
                    <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                            <xsd:element name="successDisposition" type="cbe:SuccessDispositionType"
                            minOccurs="1" maxOccurs="1" />
                            <xsd:element name="situationDisposition"
                            type="cbe:ConnectSituationDispositionType" minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>

            <xsd:complexType name="ReportSituation" abstract="false">
                <xsd:complexContent>
                    <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                            <xsd:element name="reportCategory" type="cbe:ReportCategoryType"
                            minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>

            <xsd:complexType name="FeatureSituation" abstract="false">
                <xsd:complexContent>
                    <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                            <xsd:element name="featureDisposition" type="cbe:FeatureDispositionType"
                            minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>

        <xsd:complexType name="DependencySituation" abstract="false">
                <xsd:complexContent>
                    <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                            <xsd:element name="dependencyDisposition"
                            type="cbe:DependencyDispositionType" minOccurs="1" maxOccurs="1" />

                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>

        <xsd:complexType name="ConfigureSituation" abstract="false">
                <xsd:complexContent>
                    <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
```

```
                                <xsd:element name="successDisposition" type="cbe:SuccessDispositionType"
                                minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="CreateSituation" abstract="false">
        <xsd:complexContent>
                <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                                <xsd:element name="successDisposition" type="cbe:SuccessDispositionType"
                                minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="DestroySituation" abstract="false">
        <xsd:complexContent>
                <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                                <xsd:element name="successDisposition" type="cbe:SuccessDispositionType"
                                minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="AvailableSituation" abstract="false">
        <xsd:complexContent>
                <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                                <xsd:element name="operationDisposition" type="cbe:OperationDispositionType"
                                minOccurs="1" maxOccurs="1" />
                                <xsd:element name="processingDisposition" type="cbe:ProcessingDispositionType"
                                minOccurs="1" maxOccurs="1" />
                                <xsd:element name="availabilityDisposition" type="cbe:AvailabilityDispositionType"
                                minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="RequestSituation" abstract="false">
        <xsd:complexContent>
                <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                                <xsd:element name="successDisposition" type="cbe:SuccessDispositionType"
                                minOccurs="1" maxOccurs="1" />
                                <xsd:element name="situationQualifier" type="cbe:RequestSituationQualifierType"
                                minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="OtherSituation" abstract="false">
        <xsd:complexContent>
                <xsd:extension base="cbe:SituationType">
                        <xsd:sequence>
                                <xsd:any namespace="##any" minOccurs="1" maxOccurs="unbounded" />
```

```
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="ComponentIdentification">
            <xsd:sequence>
                        <xsd:element name="componentData" type="cbe:ComponentData" minOccurs="0"
                        maxOccurs="1" />
                        <xsd:element name="componentType" type="cbe:ComponentType" minOccurs="1"
                        maxOccurs="1" />
                        <xsd:element name="componentAddress" type="cbe:ComponentAddress"
                        minOccurs="1" maxOccurs="1" />
            </xsd:sequence>
            <xsd:attribute name="componentAddressType" type="cbe:ComponentAddressType" use="required" />

        </xsd:complexType>

        <xsd:complexType name="ComponentAddress" abstract="true">
        </xsd:complexType>

        <xsd:complexType name="ComponentType">
            <xsd:attribute name="name" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="512" />
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
        </xsd:complexType>

        <xsd:complexType name="TCPAddressType" abstract="false">
            <xsd:complexContent>
                <xsd:extension base="cbe:ComponentAddress">
                    <xsd:sequence>
                        <xsd:element name="ipAddress" minOccurs="1" maxOccurs="1" >
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                    <xsd:maxLength value="23" />
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:element>
                        <xsd:element name="port" minOccurs="1" maxOccurs="1" >

                            <xsd:simpleType>
                                <xsd:restriction base="xsd:string">

                                    <xsd:maxLength value="64" />
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>

 <xsd:complexType name="SNAAddressType" abstract="false">
            <xsd:complexContent>
                <xsd:extension base="cbe:ComponentAddress">
                    <xsd:sequence>
                        <xsd:element name="luName" minOccurs="1" maxOccurs="1" >
                            <xsd:simpleType>
```

```
                                                    <xsd:restriction base="xsd:string">
                                                            <xsd:maxLength value="64" />
                                                    </xsd:restriction>
                                            </xsd:simpleType>
                                    </xsd:element>
                                    <xsd:element name="tp" minOccurs="1" maxOccurs="1" >

                                            <xsd:simpleType>
                                                    <xsd:restriction base="xsd:string">
                                                            <xsd:maxLength value="64" />
                                                    </xsd:restriction>
                                            </xsd:simpleType>
                                    </xsd:element>
                            </xsd:sequence>
                    </xsd:extension>
            </xsd:complexContent>
    </xsd:complexType>

<xsd:complexType name="HostAddressType" abstract="false">
            <xsd:complexContent>
                    <xsd:extension base="cbe:ComponentAddress">
                            <xsd:sequence>
                                    <xsd:element name="hostname" minOccurs="1" maxOccurs="1" >
                                            <xsd:simpleType>
                                                    <xsd:restriction base="xsd:string">
                                                            <xsd:maxLength value="512" />
                                                    </xsd:restriction>
                                            </xsd:simpleType>
                                    </xsd:element>
                            </xsd:sequence>
                    </xsd:extension>
            </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="GUIDAddressType" abstract="false">
            <xsd:complexContent>
                    <xsd:extension base="cbe:ComponentAddress">
                            <xsd:sequence>
                                    <xsd:element name="guid" minOccurs="1" maxOccurs="1" >
                                            <xsd:simpleType>
                                                    <xsd:restriction base="xsd:string">
                                                            <xsd:minLength value="32" />
                                                            <xsd:maxLength value="64" />
                                                    </xsd:restriction>
                                            </xsd:simpleType>
                                    </xsd:element>
                            </xsd:sequence>
                    </xsd:extension>
            </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="DeviceAddressType" abstract="false">
            <xsd:complexContent>
                    <xsd:extension base="cbe:ComponentAddress">
                            <xsd:sequence>
                                    <xsd:element name="deviceId" minOccurs="1" maxOccurs="1" >
                                            <xsd:simpleType>
                                                    <xsd:restriction base="xsd:string">

                                                            <xsd:maxLength value="128" />
                                                    </xsd:restriction>
                                            </xsd:simpleType>
```

```
                                </xsd:element>
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="OtherAddressType" abstract="false">
        <xsd:complexContent>
                <xsd:extension base="cbe:ComponentAddress">
                        <xsd:sequence>
                                <xsd:any namespace="##any" minOccurs="0" maxOccurs="unbounded" />

                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="ComponentData">
        <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:complexType>

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

	<xsd:simpleType name="ComponentAddressType">
		<xsd:union>
			<xsd:simpleType>
				<xsd:restriction base="xsd:Name">
					<xsd:maxLength value="32" />
					<xsd:enumeration value="TCPAddressTypeV4" />
					<xsd:enumeration value="TCPAddressTypeV6" />
					<xsd:enumeration value="SNAAddressType" />
					<xsd:enumeration value="HostAddressType" />
					<xsd:enumeration value="FQHostAddressType" />
					<xsd:enumeration value="DeviceAddressType" />

					<xsd:enumeration value="GUIDAddressType" />
		</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="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>

<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>

<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>

		<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="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" />

					<xsd:any namespace="##any" minOccurs="0" maxOccurs="unbounded" />
				</xsd:choice>

				<xsd:element maxOccurs="unbounded" minOccurs="0" name="children"
				type="cbe:ExtendedDataElementType" />
			</xsd:sequence>

			<xsd:attribute name="name" use="required">
				<xsd:simpleType>
					<xsd:restriction base="xsd:Name">
						<xsd:maxLength value="64" />
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:attribute name="type" use="required">
			<xsd:simpleType>
								<xsd:restriction base="xsd:string">
									<xsd:maxLength value="64"></xsd:maxLength>
									<xsd:enumeration alue="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:enumerationvalue="dateTime"></xsd:enumeration>
                                                                        <xsd:enumeration value="boolean"></xsd:enumeration>

                                                              <xsd:enumerationvalue="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="any"></xsd:enumeration>
                              </xsd:restriction>
          </xsd:simpleType>
          </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="CorrelatorDataElement">
          <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:NMTOKEN" />
                    </xsd:choice>
                    <xsd:element maxOccurs="unbounded" minOccurs="0" name="children"
                    type="cbe:CorrelatorDataElement" />
          </xsd:sequence>

          <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>

          <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: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:simpleType name="CategoryNameType">
                    <xsd:restriction base="xsd:Name">
                            <xsd:enumeration value="StartSituation" />
                            <xsd:enumeration value="StopSituation" />
                            <xsd:enumeration value="FeatureSituation" />
                            <xsd:enumeration value="DependencySituation" />
                            <xsd:enumeration value="RequestSituation" />
                            <xsd:enumeration value="ConfigureSituation" />
                            <xsd:enumeration value="ConnectSituation" />
                            <xsd:enumeration value="CreateSituation" />
                            <xsd:enumeration value="DestroySituation" />
                            <xsd:enumeration value="ReportSituation" />
                            <xsd:enumeration value="AvailableSituation" />
                            <xsd:enumeration value="OtherSituation" />

                            <xsd:maxLength value="64" />
                        </xsd:restriction>
            </xsd:simpleType>

            <xsd:simpleType name="SuccessDispositionType">
                <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>

            <xsd:simpleType name="StartSituationQualifierType">
                <xsd:union>
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="START INITIATED" />
                            <xsd:enumeration value="RESTART 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>
            <xsd:simpleType name="StopSituationQualifierType">
                <xsd:union>
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="STOP INITIATED" />
                            <xsd:enumeration value="ABORT INITIATED" />
                            <xsd:enumeration value="PAUSE INITIATED" />
                            <xsd:enumeration value="STOP 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>
            <xsd:simpleType name="ConnectSituationDispositionType">
                <xsd:union>
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="IN USE" />
                            <xsd:enumeration value="FREED" />
                            <xsd:enumeration value="CLOSED" />
                            <xsd:enumeration value="AVAILABLE" />
                            <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>

            <xsd:simpleType name="ReportCategoryType">
                <xsd:union>
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="PERFORMANCE" />
                            <xsd:enumeration value="SECURITY" />
                            <xsd:enumeration value="HEART BEAT" />
                            <xsd:enumeration value="STATUS" />
                            <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>
```

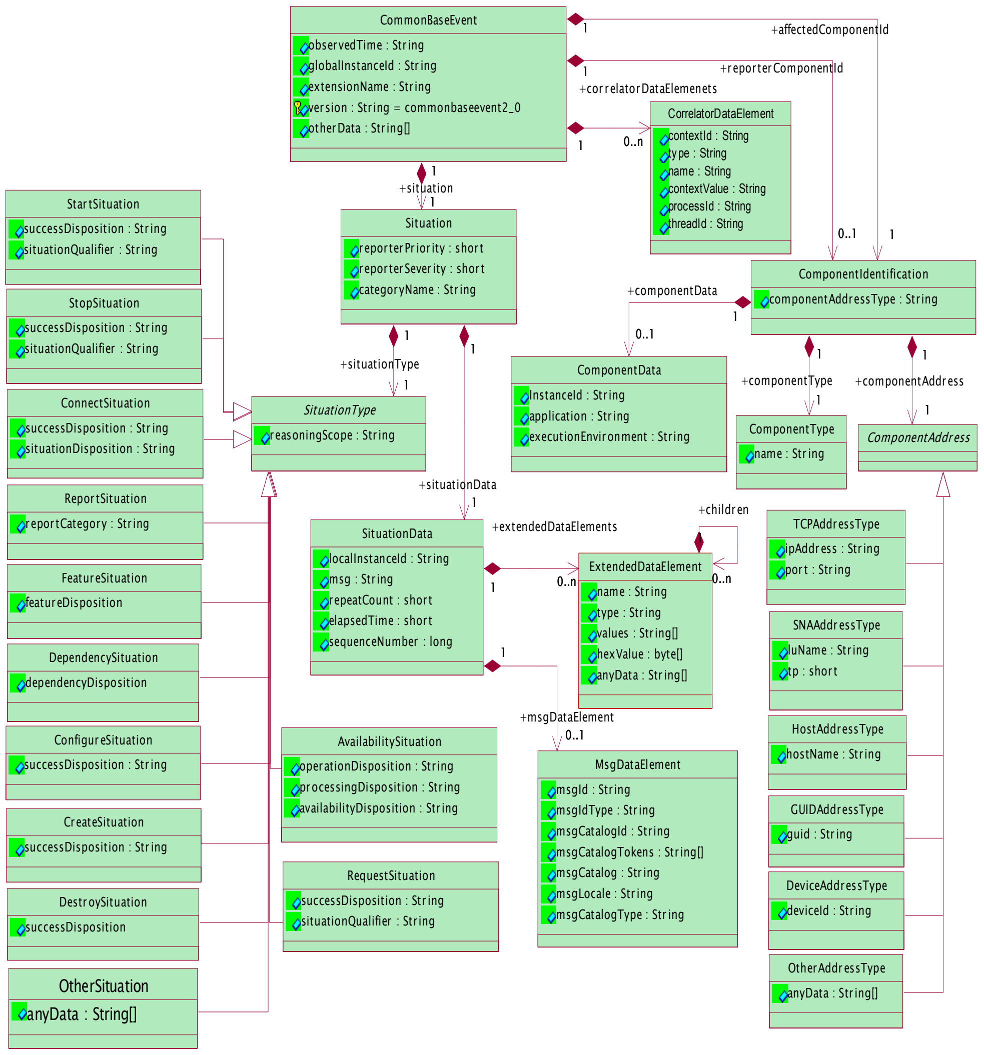
```
<xsd:simpleType name="FeatureDispositionType">
	<xsd:union>
		<xsd:simpleType>
			<xsd:restriction base="xsd:string">
				<xsd:enumeration value="AVAILABLE" />
				<xsd:enumeration value="NOT AVAILABLE" />
				<xsd:maxLength value="64" />
			</xsd:restriction>
		</xsd:simpleType>
		<xsd:simpleType>
			<xsd:restriction base="xsd:string">
				<xsd:maxLength value="64" />
			</xsd:restriction>
		</xsd:simpleType>
	</xsd:union>
</xsd:simpleType>

<xsd:simpleType name="DependencyDispositionType">
	<xsd:union>
		<xsd:simpleType>
			<xsd:restriction base="xsd:string">
				<xsd:enumeration value="MET" />
				<xsd:enumeration value="NOT MET" />
				<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>

<xsd:simpleType name="ConfigureDispositionType">
	<xsd:union>
		<xsd:simpleType>
			<xsd:restriction base="xsd:string">
				<xsd:enumeration value="SUCCESSFUL" />
				<xsd:enumeration value="UNSUCCESSFUL" />
				<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>

<xsd:simpleType name="CreateDispositionType">
	<xsd:union>
		<xsd:simpleType>
			<xsd:restriction base="xsd:string">
				<xsd:enumeration value="SUCCESSFUL" />
				<xsd:enumeration value="UNSUCCESSFUL" />
				<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>

        <xsd:simpleType name="DestroyDispositionType">
            <xsd:union>
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="SUCCESSFUL" />
                        <xsd:enumeration value="UNSUCCESSFUL" />
                        <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>

        <xsd:simpleType name="AvailabilityDispositionType">
            <xsd:union>
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="STARTABLE" />
                        <xsd:enumeration value="NONSTARTABLE" />
                        <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>

        <xsd:simpleType name="ProcessingDispositionType">
            <xsd:union>
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="FUNCTION_PROCESSED" />
                        <xsd:enumeration value="FUNCTION_BLOCKED" />
                        <xsd:enumeration value="MGMT_TASK_PROCESSED" />
                        <xsd:enumeration value="MGMT_TASK_BLOCKED" />
                        <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>

        <xsd:simpleType name="RequestSituationQualifierType">
            <xsd:union>
                <xsd:simpleType>
```

```
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="REQUEST INITIATED" />
                            <xsd:enumeration value="REQUEST 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>


            <xsd:simpleType name="OperationDispositionType">
                <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>

            <xsd:simpleType name="ReasoningScopeType">
                <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>

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

        </xsd:schema>
```

6.0 *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).



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.

7.0 *Appendix*

A. 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.

A.1 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

B.1 The Operating System Hosting Environment

Namespace:

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

Hosting Environment:

Operating_System

Hosting Environment that hosts this Hosting Environment:

Operating_System_Container

Component Type Enumeration Rule for this Hosting Environment:

<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	Components Hosted by this Hosting Environment
• *Language_Runtime*	• *Relational_Database*
• *Device_Driver*	• *Web_Application_Server*
• *Software*	• *Messaging*
• *Process*	
• *Thread*	
• *TCPIP_port*	
•	

Catgegorization (Grouping) Component Types:

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

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

C.1 The Web_Application_Server Hosting Environment

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

Hosting Environment:

Web_Application_Server

Hosting Environment that hosts this Hosting Environment:

Operating_System

Component Type Enumeration Rule for this Hosting Environment:

<Web Application Server Product Name>

Component Types for this Hosting Environment based on Enumeration Rule:

- *WebSphereApplicationServer*
- *WebLogicApplicationServer*
- *OracleWebApplicationServer*
- *SunONEWebApplicationServer*
- *TomCatWebApplicationServer*
- *JBossWebApplicationServer*

Component Types hosted in this Hosting Environment:

Components of 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*

Components Hosted by this Hosting Environment

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

Catgegorization (Grouping) Component Types:

- *WebApplicationServer_Cluster*
- *J2EE_Application_Server*
- *WebApplicationServer*

- *WebSphere_Cell*

D.1 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

- *RelationalDatabase_Application*
- *RelationalDatabase_Connection*
- *RelationalDatabase_Instance*
- *RelationalDatabase_Table*
- *RelationalDatabase_Tablespace*
- *RelationalDatabase_Tablespace_Container*

Components Hosted by this Hosting Environment

- *RelationalDatabase_Node*
- *RelationalDatabase_APPC_Node*
- *RelationalDatabase_APPCLU_node*
- *RelationalDatabase_APPN_Node*
- *RelationalDatabase_NETBIOS_Node*
- *RelationalDatabase_TCPIP_Node*
- *RelationalDatabase_LDAP_Node*
- *RelationalDatabase_Local_Node*

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

- *RelationalDatabase-NodeGroup*

E.1 List of Component Type Values by Namespace

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

- *WebSphereApplicationServer*
- *WebLogicApplicationServer*
-
- *OracleWebApplicationServer*
- *SunONEWebApplicationServer*
- *TomCatWebApplicationServer*
- *JBossWebApplicationServer*

- *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*

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

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

• *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*	

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*	

B. 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)