Change Log for OSS Common API version 1.2

OSS through Java™ Initiative

Vincent Perrot, Sun Microsystems, Inc.

COM-API-SPEC_change_log.1.2.5.doc

Executive Summary

This document summarizes the changes to the OSS Common API (JSR 144) specification Version 1.2. The main purpose of this version is

- to add the Core Business Entities (CBE) interfaces for the new JSRs (like Pricing) management purpose,
- Follow the new Draft of the OSS/J Design Guideline v1.2

However, since maintenance release to the specification was taking place, additional modifications to the previously existing Java Value Type interface were also incorporated. All these modifications are coming from the Web Bug tracking system at: http://bugs.sun.com/bugdatabase/index.jsp

There are two lists of changes:

- "proposed" changes are those modifications that are included in OSS Common API version 1.2.
- “rejected” changes are those modification that will not be included in the OSS Common API v 1.2
- "deferred" changes are those modifications that are not included in OSS Common API version 1.2, whether for time reasons or because it was considered that the changes were too significant.

The detailed description of changes in this document is principally of interest to people implementing the OSS Common API specification.

Table of Contents

1 Preface

1.1 Objectives

This document lists all the changes that have been requested for the maintenance release v1.2 version of the OSS Common API, JSR 144.

The changes have been collected through:

- Bug parade: Bug and Request For Evolution (RFE) submitted by Java developers
- OSS/J specification leaders: evolution necessary to incorporate new common objects and to improve the common interfaces and Reference Implementation that will be “inherited” by all maintenance releases of the existing OSS APIs
- OSS/J Architectural Board: The common API needs to reflect the necessary new architectural recommendation (new CBE, etc)

1.2 Audience

This document is used to start a Maintenance Release of the OSS Common API JSR 144.

According to the JCP℠:

The Maintenance Lead (ML) will arrange to have all change items placed into the PROPOSED section of the Change Log (this document) and then send a request to the PMO to initiate a Maintenance Review. The PMO will make a public announcement and begin the review.

1.3 Approval and Distribution

The ML may choose to modify one or more of the proposed changes based on comments received during review.

1.4 Related Information

oss_commmon-1_1-fr-spec.zip: contains the Version 1.1 of the OSS common API, JSR 144, http://java.sun.com/products/oss/start_download.html

1.5 **Revision History**

Date	Version	Author	State	Comments
June 2005	1.2	Vincent Perrot, Sun Microsystems	Initial Draft	• Add CBE for Pricing and order management • Add bug from bug Parade
July 2005	1.2.1	Vincent Perrot, Sun Microsystems	Draft 1	• Add/complete changes after the Product team face to face
August 2005	1.2.2	Vincent Perrot, Sun Microsystems	Draft 2	• Document sent for AB review, comment are due 08-26-2005
August2005	1.2.3	Vincent Perrot, Sun Microsystems	After Review period	• Add *KeyResultIterator in 3.7.2 chapter • Section 3.2 and 3.7.6 still incomplete
August 2005	1.2.4	Vincent Perrot, Sun Microsystems	Proposed changes	• Remove from the list items that not ready for this release.
Oct 2005	1.2.5	Vincent Perrot, Sun Microsystems	Final	• Classify issues

2 Summary of changes

3 Accepted changes

3.1 Bug ID: 6280947 Add CBE components related to Pricing
3.2 Bug ID: 6265157 INVALID EVENT_TYPE_VALUE definition for all alarm events
3.3 Bug ID: 6267986 PartyRoleKey and PartyKey shall be EntityKey instead of managedEntityKey
3.4 Bug ID: 6293854 Remove definition of IRPEvent*
3.5 Bug ID: 6250093 MOC and MOI attribute of the AlarmEvent shall move to the Event interface
3.6 Bug ID: 4753620 Apply OSS/J Design Guidelines v1.2 to the OSS Common API
3.6.1 Fix Design Guidelines implementation in CBE implementation
3.6.2 Deprecate Serializer* and XmlSerializer* interface definitions
3.6.3 Add the new query pattern in the javax.oss.JVTSession interface definition
3.6.4 Add update procedure methods in in the javax.oss.JVTSession interface definition
3.6.5 New exception model
3.6.6 Bug ID: 6293880 Issues in CBE XML Schemas
3.6.7 Split the CBE jar into jars: One jar per package name (under javax.oss.cbe domain).
3.6.8 Bug ID: 6307589: AlarmType enumeration should be integer
3.7 Bug ID: 6294417 Use boolean instead of Boolean in interface AlarmValue and NotifyNewAlarmEvent
3.8 Bug ID 6307648 Re-align the SID and CBE for SLA package
3.9 Bug ID: 6308263 Invalid field description in javax.oss.cbe.product package
3.10 Bug ID 6312124: RFE: replacement of ThresholdInfo with AlarmSpecificInfo
3.11 Bug ID 6267195: PerformanceMonitorValueImpl not initialized with correct attribute name
3.12 Bug ID: 6308261 Move TroubleTicketValue to CBE

4 Rejected changes

4.1 Bug ID 6309694: Missing get/setManagedObjectClass and ManagedObjectInstance in AlarmValue def
4.2 Bug ID 6310400: Breaks Key/Value pair paradigm to reduce the number of unnecessary Key def.
4.3 Deferred Changes from Common v1.1
4.3.1 Relocation of the (JSR 130) activity package
4.3.2 Improve Weakly Typed Arguments

3 Accepted changes

3.1 Bug ID: 6280947 Add CBE components related to Pricing

The JSR 144 has the opportunity to improve the efficiency of API developers and maintain consistency by defining, modeling and implementing these core concepts. This work effort leverages work already in progress being carried out by the TeleManagement Forum's New Generation OSS (NGOSS) Shared Information/Data (SID) Model team.

The following java packages and interfaces are added to the OSS Common API , javax.oss.cbe package to take into account all the base interfaces definitions that will be needed by the Pricing JSR (251) and shared between other JSRs.

Java interface definitions to be added to javax.oss.cbe package

- `javax.oss.cbe.policy`
  The Policy domain defines Policy entities that can be used in managing the behaviour and definition of entities in other domains. Policy takes three primary forms. The first is the definition of how policy is used to manage the definition, change, and configuration of other entities. The second is the definition of how policy itself is managed. The third is how applications use policies to manage entities. All those forms are represented by the based PolicyValue interface definition.

  - `PolicyKey`
  - `PolicyValue`

- `javax.oss.cbe.product.productoffering`
  This package contains only the base interfaces for product offering definition. A product offering is what is externally presented to the market for the markets use. The product offering is primarily the way to position products in the marketplace to create profit.

  - `BundledProductOfferingKey`
  - `BundledProductOfferingValue`
  - `ProductCatalogKey`
  - `ProductCatalogValue`
  - `ProductOfferingKey`
  - `ProductOfferingValue`
  - `SimpleProductOfferingKey`
  - `SimpleProductOfferingValue`

- `javax.oss.cbe.product.productofferingprice`
  This package contains only the base interfaces for product offering price definition. It contains a set of components that can be combined to offer a complete and accurate description of the price charged for an offering.

  - `ProductOfferingPriceKey`
  - `ProductOfferingPriceValue`

  The interfaces listed in this section are subject to change to follow the latest CBE model definition from the OSS/J CBE team.

3.2 Bug ID: 6265157 INVALID EVENT_TYPE_VALUE definition for all alarm events

The EVENT_TYPE_VALUE field in all of *EventPropertyDescriptor interfaces in the javax.oss.cbe.alarm package are not correctly set.

This field shall be set with the name of the interface defining the type of the event.

Example:

public static final String EVENT_TYPE_VALUE = javax.oss.cbe.alarm.NotifyNewAlarmEvent.class.getName();

This impacts the following interface definition:

- `javax.oss.cbe.alarm.AlarmEventPropertyDescriptor`

Note: the Notify* definition from v 1.1 have also been remove from this version as they are not base definition and are info model specific.

3.3 Bug ID: 6267986 PartyRoleKey and PartyKey shall be EntityKey instead of managedEntityKey

The javax.oss.cbe.party.PartyRoleValue and javax.oss.cbe.party.PartyValue are EntityValue, so their respective key definition shall also inherit from EntityKey.

This impacts the following interface definition:

- javax.oss.cbe.party.PartyKey

- javax.oss.cbe.party.PartyRoleKey

3.4 Bug ID: 6293854 Remove definition of IRPEvent*

The javax.oss.util.IRPEvent* definition are specific to the IRP information model. This kind of specification is out of the Scope of the cbe package which shall stay domain agnostic.

The deprecation of this util package impacts the javax.oss.Event and javax.oss.cbe.alarm.AlarmEvent definitions as detail in the following Bug ID 6250093.

Note: javax.oss.util now contains only the InteractionRecord definition mainly used in the quality of service domains.

3.5 Bug ID: 6250093 MOC and MOI attribute of the AlarmEvent shall move to the Event interface

The deprecation of the util package (See above Bug ID 6293854) impacts the javax.oss.Event and javax.oss.cbe.alarm.AlarmEvent definitions as follow:

- The attributes named managedEntityClass and managedEntityInstance move from the IRP definition to the javax.oss.Event interface
- The Attribute NotificationId move to the javax.oss.cbe.alarm.AlarmValue. The Alarm value and key are now the only 2 attribute of the AlarmEvent.

3.6 Bug ID: 4753620 Apply OSS/J Design Guidelines v1.2 to the OSS Common API

3.6.1 Fix Design Guidelines implementation in CBE implementation

All managed entity values (using the naming convention <name>Value) shall get its corresponding key definition (using the naming convention <name>Key.

Example:

```
ManagedEntityValue.java
```

```
ManagedEntityKey.java
```

All managed entity values shall include accessor and mutator for their corresponding key attribute.

All managed entity values shall include a factory method starting with “make” to create the corresponding key definition.

All managed entity values shall declare the attribute names (also including the key attribute) as public final static strings. The naming convention used follow the example below:

```
Public final static string MANAGED_ENTITY_KEY = “managedEntityKey”;
```

Note: it follows the bean convention.

Note: the TCK have been also improved to consolidate this.

3.6.2 Deprecate Serializer* and XmlSerializer* interface definitions

The interface definitions Serializer, XmlSerialer, SerializerFactory, and XmlSerializerEncodingStyles in the javax.oss package have been created to anticipate the java to/from XML marshaling capabilities to implement the XML over JMS integration profile. The java/XML technologies are now mature enough to remove/deprecate these interface definitions from the Common API itself.

In addition with the deprecation of the interfaces, the common interface definitions are impacted as follow:

javax.oss.SerializerFactory is removed from the extension list of the interfaces:

- `javax.oss.cbe.measurement.PerformanceAttributeDescriptor`
- `javax.oss.cbe.report.CurrentResultReport`
- `javax.oss.cbe.report.ReportFormat`
- `javax.oss.Event`
- `javax.oss.ManagedEntityKey`
- `javax.oss.ManagedEntityValue`
- `javax.oss.QueryValue`

3.6.3 Add the new query pattern in the javax.oss.JVTSession interface definition

Named queries are used to implement complex query operations. The result of a named query is named a query response. Usually the template-based JVT operations (like queryManagedEntities ()) are not sufficient to implement such complex query operations.

The corresponding methods in JVT Session are deprecated:

- String[] getQueryTypes()
- QueryValue makeQueryValue(String type)
- ManagedEntityValueIterator queryManagedEntities(QueryValue query, String[] attributeNames)

The implementation of this named query pattern needs the addition of several methods in the JVTSession interface and the creation of the javax.oss.NamedQueryValue and javax.oss.NamedQueryResponse interfaces.

The based definitions javax.oss.ManagedEntityValueIterator and javax.oss.ManagedEntityKeyResultIterator is also extending the NamedQueryResponse.

JVTSession (and JVTLocalSession) contains the following new methods:

```
    /**
     * Query multiple Entities using a NamedQueryValue.
     *
     * @param query a NamedQueryValue object representing the query.
     * @return a NamedQueryResponse used to extract the results of the query.
     * @exception javax.oss.OssIllegalArgumentException unsupported named query value
type.
     * @exception java.rmi.RemoteException
     *
     * @ossj:nillableField name=methodReturnValue value=true
     * @ossj:minOccursField name=methodReturnValue value=0
     * @ossj:minOccursField name=namedQuery value=0
     */
    NamedQueryResponse query(NamedQueryValue namedQuery)
    throws javax.oss.OssIllegalArgumentException, java.rmi.RemoteException;

    /**
     * Get the Named Query type names supported by a JVT Session Bean
     *
     * @return String array which contains the fully qualified names of the leaf
     * node interfaces representing the supported named query value types,
     * i.e., interfaces which extend NamedQueryValue.
     * @exception java.rmi.RemoteException
     */
    String[] getNamedQueryTypes()
    throws java.rmi.RemoteException;

    /**
     * Create a NamedQueryValue Instance matching a Named Query type name.
     * The Session Bean is used as a factory for the creation of
     * named query values.
     *
     * @param type fully qualified name of the leaf node NamedQueryValue interface.
     * @return query value object of the specified type.
     * @exception javax.oss.OssIllegalArgumentException unknown or unsupported
     * named query type.
     * @exception java.rmi.RemoteException
     */
    NamedQueryValue makeNamedQueryValue(String type)
    throws javax.oss.OssIllegalArgumentException,java.rmi.RemoteException;
```

New javax.oss.NamedQueryValue and javax.oss.NamedQueryResponse interface definitions:

```
package javax.oss;

import java.io.Serializable;

/**
 * Named query object is used to implement complex query operations.
 * The result of a named query is named a query response.
 * @see javax.oss.JVTSession
 * @see javax.oss.NamedQueryResponse
 * @see javax.oss.QueryValue
 *
 * @author OSS through Java Initiative, Vincent Perrot Sun Microsystems Inc.
 * @version 1.2
 * @since August 2005
 * @ossj:queryvalue
 * @ossj:abstract
 */
public interface NamedQueryValue extends AttributeAccess, Serializable, Cloneable {
    /** This String defines the type of the named query. <br>
     * This value shall be overloaded according to type, vendor or technology specific
interface defining the new query.
     * The QUERY_TYPE shall correspond to the &lt;interface&gt;.class.getName();
     *  <br>It's value is "javax.oss.QueryValue".
     */
    public static final String QUERY_TYPE = "javax.oss.NamedQueryValue";

    /**
      * Deep copy this query value.
      *
      * @return deep copy of this query value.
      */
    public Object clone();
}
```

```
package javax.oss;

/**
 * Object returned as result of a named query execution using a specified NamedQueryValue.
 * @see java.oss.JVTSession
 * @see java.oss.NamedQueryValue
 *
 * @author OSS through Java Initiative, Vincent Perrot Sun Microsystems Inc.
 * @version 1.2
 * @since August 2005
 * @ossj:complexdata
 */

public interface NamedQueryResponse {

}
```

3.6.4 Add update procedure methods in in the javax.oss.JVTSession interface definition

Named Update Procedures are used to implement complex Update operations.

Named update procedures are similar to named queries and allow implementing complex atomic update operations. The result of the execution of a named update procedure will typically be the creation,

removal or update of a collection of managed entity values As in the case of named queries, the template-based JVT operations are not sufficient to implement such complex operations.

Even if equivalent functionality can be achieved by template-based JVT operations, named update procedures may still be the preferred approach when there is a need to expose a business oriented method for update purposes instead of a generic (template-based) operation. In general better performance should be expected for named update procedures than for specification-based update operations.

Similarly to the named queries, named update procedures allow extending the functionality of the javax.oss.JVTSession interface in order to support new update operations and the creation of the javax.oss.UpdateProcedureValue and javax.oss.UpdateProcedureResponse interfaces. The result of an update using an UpdateProcedureValue is an UpdateProcedureResponse.

JVTSession (and JVTLocalSession) contains the following new methods:

```
    /**
     * used to execute named update procedures
     * Execute the given update procedure.
     *
     * @param updateValue a UpdateProcedureValue object representing the Update Procedure
to be performed.
     * @return a UpdateProcedureResponse resulting from the Update Procedure execution.
     * @exception javax.oss.OssIllegalArgumentException unsupported Update Procedure value
type.
     * @exception java.rmi.RemoteException
     *
     * @ossj:nillableField name=methodReturnValue value=true
     * @ossj:minOccursField name=methodReturnValue value=0
     * @ossj:minOccursField name=updateValue value=0
     */
    UpdateProcedureResponse update(UpdateProcedureValue updateValue)
    throws javax.oss.OssIllegalArgumentException ,java.rmi.RemoteException;

    /**
     * Create a UpdateProcedureValue Instance matching a Update Procedure type name.
     * The Session Bean is used as a factory for the creation of
     * Update Procedure values.
     *
     * @param type fully qualified name of the leaf node UpdateProcedureValue interface.
     * @return Update Procedure value object of the specified type.
     * @exception javax.oss.OssIllegalArgumentException unknown or unsupported
     * Update Procedure type.
     * @exception java.rmi.RemoteException
     */
    UpdateProcedureValue makeUpdateProcedureValue(String type)
    throws javax.oss.OssIllegalArgumentException ,java.rmi.RemoteException;

    /**
     * Get the UpdateProcedure type names supported by a JVT Session Bean
     *
     * @return String array which contains the fully qualified names of the leaf
     * node interfaces representing the supported update procedure value types,
     * i.e., interfaces which extend UpdateProcedureValue.
     * @exception java.rmi.RemoteException
     */

    String[] getUpdateProcedureTypes()
```

```
    throws java.rmi.RemoteException;
```

New javax.oss.UpdateProcedureValue and javax.oss.UpdateProcedureResponse interfaces:

```
package javax.oss;

import java.io.Serializable;

/**
 * Named Update Procedures are used to implement complex Update operations.
 * <p>
 * Named update procedures are similar to named queries and allow implementing
 * complex atomic update operations. The result of the execution of a named update
 * procedure will typically be the creation, removal or update of  a collection
 * of managed entity values. As in the case of named queries, the template-based
 * JVT operations are not sufficient to implement such complex operations.
 * <p>
 * @see javax.oss.JVTSession
 *
 * @author OSS through Java Initiative, Vincent Perrot Sun Microsystems Inc.
 * @version 1.2
 * @since August 2005
 * @ossj:complexdata
 */

public interface UpdateProcedureValue extends AttributeAccess, Serializable, Cloneable {

    /**
     * This String defines the type of update procedure. <br>
     * This value shall be overloaded according to type, vendor or technology specific
interface
     * defining the new update procedure.
     * The UPDATE_TYPE shall correspond to the &lt;interface&gt;.class.getName();
     *  <br>It's value is "javax.oss.UpdateProcedureValue".
     */
    public static final String UPDATE_TYPE = "javax.oss.UpdateProcedureValue";

    /**
     * Deep copy this update procedure value.
     *
     * @return deep copy of this update procedure value.
     */
    public Object clone();

}
```

```
package javax.oss;

/**
 * Object returned as result of an update execution using a specified
UpdateProcedureValue.
 * @see java.oss.JVTSession#update
 * @see java.oss.UpdateProcedureValue
 *
 * @author OSS through Java Initiative, Vincent Perrot Sun Microsystems Inc.
 * @version 1.2
 * @since August 2005
 * @ossj:complexdata
 */

public interface UpdateProcedureResponse {

    /**
     * Flag indicating that the update procedure is still in progress
     */
```

```
    public final static int IN_PROGRESS = 1;

    /**
     * Flag indicating that the update procedure was completed successfully.
     */
    public final static int COMPLETE = 8;

    /**
     * Flag indicating that the update procedure was aborted.
     */
    public final static int ABORTED = 2;

    /**
     * Flag indicating that the update procedure was completed successfully.
     */
    public final static int ERRORED = 4;

    /**
     * Flag indicating that the update procedure was either completed
     * or aborted or encoutered an error.
     */
    public final static int DONE = (ABORTED | ERRORED | COMPLETE);

    /**
     * Return the execution status of the update procedure
     */
    public int getStatus();

    /**
     * Return true is the update procedure was completed sucessfully.
     * @return completion
     */

    public boolean isSuccessful();
}
```

3.6.5 New exception model

It is recommended to reuse as much as possible application Exceptions already defined in Java. In particular, the exceptions as defined in the EJB Specification should be reused as much as possible and their semantic should be preserved while applying them to JVTSession Bean operations on managed entities.

Since RuntimeException exceptions are meant to be caught by the container and will not be thrown to the user they should not be used as application exceptions. Some RuntimeException names defined in the java.lang domain have been redefined (extending java.lang.Exception) in the javax.oss domain using the same exception names. Even if this was valid regarding the java language definition, this caused several major issues especially in automatically generated code when deploying applications or when using “import javax.oss.*;” in source code.

So the following exception definitions in the javax.oss domain have been deprecated and replaced by prefixing their name with the string ”Oss”.

- javax.oss.IllegalArgumentException replaced by javax.oss.OssIllegalArgumentException

- javax.oss.IllegalAttributeValueException replaced by javax.oss.OssIllegalAttributeValueException
- javax.oss.IllegalStateException replaced by javax.oss.OssIllegalStateException
- javax.oss.ResyncRequiredException replaced by javax.oss.OssResyncRequiredException
- javax.oss.SetException replaced by javax.oss.OssSetException
- javax.oss.UnsupportedAttributeException replaced by javax.oss.OssUnsupportedAttributeException
- javax.oss.UnsupportedOperationException replaced by javax.oss.OssUnsupportedOperationException

3.6.6 Bug ID: 6293880 Issues in CBE XML Schemas

The XML schema present in the COM-API-SPEC_PART4_XML_SCHEMA.1.2.zip are automatically generated from the doclet information present in the java interface definitions.

The following issues are fixed either by improving the XML schema generator, or fixing the generation properties or finally fixing the doclet information (“@ossj”) present in the javadoc sections of the interfaces.

Note also that the new XML schema naming convention has been applied.

1) XmlCommonSchema renamed Common/v1-2/OSSJ-Common-v1-2.xsd

The missing definition of the RuntimeException is added:

```
<complexType name="RuntimeException">
        <annotation>
                <documentation>
                        RuntimeException is the superclass of those exceptions that
                        can be thrown during the normal operation of the Java
                        Virtual Machine.
                </documentation>
        </annotation>
        <complexContent>
                <extension base="co-v1-2:BaseException">
                        <sequence/>
                </extension>
        </complexContent>
</complexType>
```

2) XmlCBEAlarmSchema renamed Common-CBEAlarm/v1-2/OSSJ-Common-CBEAlarm-v1-2.xsd

The invalid references to IRPEvent* from the javax.oss.util package disappear automatically as the IRPEvent* are deprecated and not used anymore.

In javax.oss.cbe.alarm.AlarmValue and javax.oss.cbe.alarm.NotifyNewAlarmEventType, the attribute named backedUpStatus moves from the java.lang.Boolean to boolean. Then the generator sets the correct mapping. (see chapter 3.7 Bug ID: 6294417 Use boolean instead of Boolean in interface AlarmValue and NotifyNewAlarmEvent)

3) XmlCBEPartySchema renamed Common-CBEParty/v1-2/OSSJ-Common-CBEParty-v1-2.xsd

In the doclet.properties used for generation, the invalid string “DataTypes” is replaced by the valid “Datatypes” string.

4) XmlCBEServiceSchema renamed Common-CBEService/v1-2/OSSJ-Common-CBEService-v1-2.xsd

The missing Datatypes dependent schema is added to doclet.properties.

3.6.7 Split the CBE jar into jars: One jar per package name (under javax.oss.cbe domain).

To ease the adoption and usage of the CBE definitions, one jar per package name under the javax.oss.cbe domain have been created.

The following jar naming convention has been used:

oss_cbe_<package name>_spec-<version>.jar

Example:

oss_cbe_location_spec-1.2.2.jar

Note: The java definitions present in the javax.oss.cbe package are included into the oss_common_spec-1.2.jar

3.6.8 Bug ID: 6307589: AlarmType enumeration should be integer

Following the Design Guidelines the enumeration types shall be map to int to reduce the memory usage.

The same integer values already used in the existing standards have been used. A default value named “UNKNOWN_ALARM_TYPE “ has also been created at the same time.

Find below the new definitions:

```
package javax.oss.cbe.alarm;

import java.io.Serializable;
```

```
/**
 * This interface identifies all 3G TS 32.111-2 [5] defined alarm event
 * types used by this API. Their semantics are defined by 3GPP. Their
 * encodings for this API are defined here.
 * @author OSS through Java Initiative, Vincent Perrot Sun Microsystems Inc.
 * @version 1.2.2
 * @since March 2005
 * @ossj:enumeration
 */
public interface AlarmType extends Serializable {
    ////////////////////////////////////
    //attributes
        /**
         * Default Alarm type
         */
        public static final int UNKNOWN_ALARM_TYPE = Integer.MAX_VALUE;

    /**
     * An alarm of this type is associated with the procedure and/or process required
conveying
    information from one point to another
     */

    //    CR6307589:
    //    public static final String COMMUNICATIONS_ALARM = "communicationsAlarm";
    public static final int COMMUNICATIONS_ALARM = 1;

    /**
     * An alarm of this type is associated with a software or processing fault
     */

    //    CR6307589:
    //    public static final String PROCESSING_ERROR_ALARM = "processingErrorAlarm";
    public static final int PROCESSING_ERROR_ALARM = 2;

    /**
    * An alarm of this type is associated with a condition related to an enclosure in
which the equipment
    resides
    */

    //    CR6307589:
    //    public static final String ENVIRONMENTAL_ALARM = "environmentalAlarm";
    public static final int ENVIRONMENTAL_ALARM = 3;

    /**
     * An alarm of this type is associated with degradation in the quality of a service
     */

    //           CR6307589:
    //    public static final String QUALITY_OF_SERVICE_ALARM = "qualityOfServiceAlarm";
    public static final int QUALITY_OF_SERVICE_ALARM = 4;

    /**
    * An alarm of this type is associated with an equipment fault
    */

    //    CR6307589:
    //    public static final String EQUIPMENT_ALARM = "equipmentAlarm";
    public static final int EQUIPMENT_ALARM = 5;

    /**
     * An attempt to alter or destroy data or executable content that is inconsistent with
the sensor's surveillance policy
     */

    //    CR6307589:
    //    public static final String INTEGRITY_VIOLATION = "integrityViolation";
    public static final int INTEGRITY_VIOLATION = 6;

    /**
     * Represents ...
```

```
     */

    //    CR6307589:
    //    public static final String OPERATIONAL_VIOLATION = "operationalViolation";
    public static final int OPERATIONAL_VIOLATION = 7;

    /**
     * Represents ...
     */

    //    CR6307589:
    //    public static final String PHYSICAL_VIOLATION = "physicalViolation";
    public static final int PHYSICAL_VIOLATION = 8;

    /**
     * Represents ...
     * It is equals to SECURITY_SERVICE_OR_MECHANISM_VIOLATION as named in
     * Rel-5 CR 32.111-3 (Fault Management; Alarm IRP CORBA solution set)
     */

    //    CR6307589:
    //    public static final String SECURITY_VIOLATION = "securityViolation";
    public static final int SECURITY_VIOLATION = 9;

    /**
     * Represents ...
     */

    //    CR6307589:
    //    public static final String TIME_DOMAIN_VIOLATION = "timeDomainViolation";
    public static final int TIME_DOMAIN_VIOLATION = 10;

} // end AlarmType
```

3.7 Bug ID: 6294417 Use boolean instead of Boolean in interface AlarmValue and NotifyNewAlarmEvent

javax.oss.cbe.alarm.AlarmValue and javax.oss.cbe.alarm.NotifyNewAlarmEvent both contain the attribute backedUpStatus which is of type java.lang.Boolean. This attribute shall be boolean instead.

This issue was caused when generating the corresponding XML schema definition where this attribute shall be map to a simple nillable boolean in the xml declaration.

3.8 Bug ID 6307648 Re-align the SID and CBE for SLA package

javax.oss.cbe.sla.KeyQualityIndicatorParam and KeyPerformanceIndicatorSlsParm move to javax.oss.cbe.service package. The interface definitions follow the SID name (to explain the typo in name “Parm” instead of “Param”).

javax.oss.cbe.sla.KeyQualityIndicatorParamIterator is removed. It is not used anymore in the interface.

javax.oss.cbe.sla.Query* interface are removed. It is a service that shall be provided by the interface itself rather than as a component in the cbe package.

javax.oss.cbe.sla.ServiceLevelObjective* move to the javax.oss.cbe.service package. This follows the SID model service definitions.

javax.oss.cbe.sla.TransformationAlgorithm move to the javax.oss.cbe.service package. This interface is used only by components from the service package.

The javax.oss.cbe.sla.* interface definitions are removed and replaced by the following one:

- `javax.oss.cbe.sla.ServiceLevelAgreementItemKey`
- `javax.oss.cbe.sla.ServiceLevelAgreementItemKeyResult`
- `javax.oss.cbe.sla.ServiceLevelAgreementItemValue`
- `javax.oss.cbe.sla.ServiceLevelAgreementItemValueIterator`
- `javax.oss.cbe.sla.ServiceLevelAgreementKey`
- `javax.oss.cbe.sla.ServiceLevelAgreementKeyResult`
- `javax.oss.cbe.sla.ServiceLevelAgreementValue`
- `javax.oss.cbe.sla.ServiceLevelAgreementValueIterator`

This follows the SID component names.

Some of these components extend the new definitions from the agreement package:

- `javax.oss.cbe.agreement.AgreementItemKey`
- `javax.oss.cbe.agreement.AgreementItemValue`
- `javax.oss.cbe.agreement.AgreementKey`
- `javax.oss.cbe.agreement.AgreementValue`

Themselves extending the based business interaction definitions (see also chapter 3.12 Bug ID: 6308261 Move TroubleTicketValue to CBE)

3.9 Bug ID: 6308263 Invalid field description in javax.oss.cbe.product package

The typo in javax.oss.cbe.product.ProductValue is fixed as follow:

- The final static string name is renamed from "DESCRPTION" to “DESCRIPTION”

The typo in javax.oss.cbe.product.ProductSpecificationValue is fixed as follow:

- The final static string name is renamed from "DSECRPTION" to “DESCRIPTION”

Note: The javadoc comment in the interface javax.oss.cbe.service.ServiceSpecificationValue component is also fixed using “description” instead of "desciption".

3.10 Bug ID 6312124: RFE: replacement of ThresholdInfo with AlarmSpecificInfo

The javax.oss.cbe.alarm.ThresholdInfo components were handled from the AlarmValue and NotifyNewAlarmEvent of the same package. The alarm definition shall be more generic and shall handle more kind of alarm information components.

The component javax.oss.cbe.alarm.AlarmSpecificInfo is replacing ThresholdInfo as the based interface definition for alarm specific information. The factory pattern will be used to implements AlarmSpecificInfo.

The ThresholdInfo component now extends AlarmSpecificInfo, and so shall be a supported as a valid AlarmSpecificInfo type.

Find below the interface changes that are applied to AlarmValue:

```
/**
 * Return the Alarm Specific Info
 *
 * @return alarm information, defined in interface AlarmSpecificInfo.
 * @throws IllegalStateException
 * @throws UnsupportedOperationException
 */
public AlarmSpecificInfo getAlarmSpecificInfo()
throws IllegalStateException, UnsupportedOperationException;

/**
 * Changes the Alarm Specific Info
 *
 * @param value
 * @throws IllegalStateException
 * @throws UnsupportedOperationException
 * @throws IllegalArgumentException
 */
public void setAlarmSpecificInfo(AlarmSpecificInfo value)
throws IllegalStateException, UnsupportedOperationException, IllegalArgumentException;
```

Note: The factory methods for getting and creating new AlarmSpecificInfo types will be handle by the JVT session of the domain specific API(s) to simplify the AlarmValue definition and implementation size.

3.11 Bug ID 6267195: PerformanceMonitorValueImpl not initialized with correct attribute name

The PerformanceMonitorValue interface definition shall use a consistent attribute naming convention. The interface definition javax.oss.cbe.measurement.PerformanceMonitorValue is modified as follow:

```
//Fix to 6267195: rename attribute definition using "name" instead of "measurementName"
public static final String NAME = "name";
```

3.12 Bug ID: 6308261 Move TroubleTicketValue to CBE

The JSR 144 has the opportunity to improve the efficiency of API developers and maintain consistency by defining, modeling and implementing these core concepts. This work effort leverages work already in progress being carried out by the TeleManagement Forum's New Generation OSS (NGOSS) Shared Information/Data (SID) Model team.

The following java packages and interfaces are added to the OSS Common API , javax.oss.cbe package to take into account all the basic interfaces definitions that will be needed by the OSS Trouble Ticket JSR 91 and shared between other JSRs.

Java interface definitions to be added to javax.oss.cbe package

- javax.oss.cbe.trouble
- javax.oss.cbe.bi (for business interaction) (see also chapter 3.8Bug ID 6307648 Re-align the SID and CBE for SLA package)

The classes listed in this section are subject to change to follow the latest CBE model definition from the OSS/J CBE team.

4 Rejected changes

4.1 Bug ID 6309694: Missing get/setManagedObjectClass and ManagedObjectInstance in AlarmValue def

The javax.oss.cbe.alarm.AlarmValue declared the attributes managedObjectClass and managedObjectInstance but didn't implement the corresponding accessor and mutator methods.

The following method declarations are added to the AlarmValue interface:

```
     /**
     * Gets the class name of the object instance.
     *
     * @return String The class name object instance.
     * @exception IllegalStateException Is thrown if the attribute is supported,
     * and the attribute has not been populated.
     * @see #setManagedObjectClass
     */
    //CR6309694
    String getManagedObjectClass() throws IllegalStateException;

    /**
     * Sets the class name of the object instance.
     *
     * @param moc The class name of the object instance.
     * @exception IllegalArgumentException Is thrown to report that
     * a bad argument was provided to the method.
     * @see #getManagedObjectClass
     */
    //CR6309694
    void setManagedObjectClass( String moc) throws IllegalArgumentException;

    /**
     * Gets the distinguished name of the object instance.
     *
     * @return String The distinguished name object instance.
     * @exception IllegalStateException Is thrown if the attribute is supported,
     * and the attribute has not been populated.
     * @see #setManagedObjectInstance
     */
    //CR6309694
    String getManagedObjectInstance() throws IllegalStateException;

    /**
     * Sets the distinguished name of the object instance.
     *
     * @param moi The distinguished name of the object instance.
     * @exception IllegalArgumentException Is thrown to report that
     * a bad argument was provided to the method.
     * @see #getManagedObjectInstance
     */
    //CR6309694
    void setManagedObjectInstance( String moi) throws IllegalArgumentException;
```

Reason:

The managedObjectClass and managedObjectInstance shall only be present in the Event definition, else they will be redundant if present also in AlarmValue.

4.2 Bug ID 6310400: Breaks Key/Value pair paradigm to reduce the number of unnecessary Key def.

The proposal is to keep only the definitions for the EntityKey, EntitySpecification and AssociationKey. The other CBE Keys do not add more value (except the strong typing).

Reason:

This would represent a deviation from the existing DG that is not “easy” explained. That is the fact that only “concrete” model should provide keys is self-contradictory with the fact that the CBE model itself contains and need to contain Entity, Association, Specification keys (otherwise we need to revisit a number of specification).

See also 3.6.1 Fix Design Guidelines implementation in CBE implementation

4.3 Deferred Changes from Common v1.1

4.3.1 Relocation of the (JSR 130) *activity* package

In order to share the activity concept with other OSSJ JSRs, the classes in the *activity* package from the JSR 130 specification shall move to the JSR 144. The location of the *activity* package is *javax.oss.cfi*. The following classes are impacted:

javax.oss.cfi.activity.ActivityCapability
javax.oss.cfi.activity.ActivityControlException
javax.oss.cfi.activity.ActivityControlParams
javax.oss.cfi.activity.ActivityController
javax.oss.cfi.activity.ActivityCreationEvent
javax.oss.cfi.activity.ActivityCreationEventPropertyDescriptor
javax.oss.cfi.activity.ActivityEvent
javax.oss.cfi.activity.ActivityEventPropertyDescriptor
javax.oss.cfi.activity.ActivityExecParams
javax.oss.cfi.activity.ActivityKey
javax.oss.cfi.activity.ActivityKeyResult
javax.oss.cfi.activity.ActivityKeyResultIterator
javax.oss.cfi.activity.ActivityPrimaryKey
javax.oss.cfi.activity.ActivityRemovalEvent
javax.oss.cfi.activity.ActivityRemovalEventPropertyDescriptor

javax.oss.cfi.activity.ActivityReportAvailableEvent
javax.oss.cfi.activity.ActivityReportAvailableEventPropertyDescriptor
javax.oss.cfi.activity.ActivityReportDataEvent
javax.oss.cfi.activity.ActivityReportDataEventPropertyDescriptor
javax.oss.cfi.activity.ActivityReportParams
javax.oss.cfi.activity.ActivityResumeEvent
javax.oss.cfi.activity.ActivityResumeEventPropertyDescriptor
javax.oss.cfi.activity.ActivityState
javax.oss.cfi.activity.ActivitySuspendEvent
javax.oss.cfi.activity.ActivitySuspendEventPropertyDescriptor
javax.oss.cfi.activity.ActivityValue
javax.oss.cfi.activity.ActivityValueIterator
javax.oss.cfi.activity.AttributeDescriptor
javax.oss.cfi.activity.DailyScheduleInfo
javax.oss.cfi.activity.QueryActivityReportData
javax.oss.cfi.activity.QueryActivityValue
javax.oss.cfi.activity.RecordDescriptor
javax.oss.cfi.activity.ReportFormat
javax.oss.cfi.activity.ReportInfo
javax.oss.cfi.activity.ReportInfoIterator
javax.oss.cfi.activity.ReportIterator
javax.oss.cfi.activity.ReportMode
javax.oss.cfi.activity.ReportRecord
javax.oss.cfi.activity.Schedule
javax.oss.cfi.activity.SubscriptionFilter
javax.oss.cfi.activity.SubscriptionParams
javax.oss.cfi.activity.WeeklyScheduleInfo

Reason:
These component have already more or less been defines in the CBE using the definitions from the SID. It looks like the JSR 130 will need a maintenance release to realign its API and components definitions with the common API V 1.2.

4.3.2 Improve Weakly Typed Arguments

Replace static final constants with J2SE 1.5 enums. Update the design guidelines and Interfaces definitions accordingly. Update all OSS/J APIs in the following releases.

```
public enum OrderState { OPEN, OPEN.NOT_RUNNING, OPEN.NOT_RUNNING.NOT_STARTED,
OPEN.NOT_RUNNING.SUSPENDED, RUNNING, CLOSED, COMPLETED, ABORTED, ABORTED_BYCLIENT,
ABORTED_BYSERVER };
```

Reason:

In J2SE 5 the enum can not be extended. The base pattern/design guideline for the common API is the extention. So enum will not adopted.

I also looks like that most of the deployed applications are still using J2SE 1.4. And the Common API definitions shall still be integrated/implemented by those applications.