The information in this document is subject to change without notice.

Nokia Networks’ liability for any errors in the document is limited to the documentary correction of errors. Nokia Networks WILL NOT BE RESPONSIBLE IN ANY EVENT FOR ERRORS IN THIS DOCUMENT OR FOR ANY DAMAGES, INCIDENTAL OR CONSEQUENTIAL (INCLUDING MONETARY LOSSES), that might arise from the use of this document or the information in it.

This document is considered protected by copyright according to the applicable laws.

NOKIA logo is a registered trademark of Nokia Corporation.
Java is a registered trademark of Sun Microsystems, Inc. in the US and other countries.
Other product names mentioned in this document may be trademarks of their respective companies, and they are mentioned for identification purposes only.

Copyright ©2001 Nokia
Specification Lead: Stefan Vaillant Nokia stefan.vaillant@nokia.com
Experts:
  Bret Birnbaum Cygent bret.birnbaum@cygent.com
  Russ Flewelling Digital Fairway rflewelling@digitalfairway.com
  Goran Forsstrom Ericsson goran.forsstrom@emw.ericsson.se
  Dave Raymer Motorola david.raymer@motorola.com
  Shingo Fukui NEC s-fukui@ax.jp.nec.com
  Ben Eng Nortel b.eng@nortelnetworks.com
  Mike Zoratti Nortel m.zoratti@nortelnetworks.com
  Ahmed M Saleh OSI ams1@osi.com
  Christophe Ebro Sun christophe.ebro@france.sun.com
  Alex Buckley Telcordia mbarton@research.telcordia.com
Contents

Introduction .............................................................................................................................................. 9
Service Provisioning Overview .................................................................................................................. 11
Concepts ...................................................................................................................................................... 15
Using the API .............................................................................................................................................. 21
Implementing the API ................................................................................................................................. 37

javax.oss.order ........................................................................................................................................... 41
  OrderManager ............................................................................................................................................ 42
  OrderManagerHome ................................................................................................................................. 54
  OrderValueIterator ................................................................................................................................. 54
  XMLOrderManager ................................................................................................................................. 55
  XMLOrderManagerHome ......................................................................................................................... 66
  XMLOrderValueIterator ......................................................................................................................... 66
  OrderValue ............................................................................................................................................. 67
  OrderKey ................................................................................................................................................ 74
  OrderMessageProperty .......................................................................................................................... 74
  OrderCreateEvent .................................................................................................................................. 75
  OrderStateChangeEvent ............................................................................................................................ 76
  OrderAttributeValueChangeEvent ........................................................................................................... 77
  OrderRemoveEvent ............................................................................................................................... 78
  OrderManagerOption ............................................................................................................................. 78
  OrderPriority .......................................................................................................................................... 80
  OrderState .............................................................................................................................................. 81
  OrderType .............................................................................................................................................. 83
  IllegalStateException ............................................................................................................................. 84
  XMLOrderManager .................................................................................................................................. 55
  XMLOrderManagerHome .......................................................................................................................... 66
  OrderValueIterator .................................................................................................................................. 54
  OrderManagerHome ............................................................................................................................... 54
  OrderManager ....................................................................................................................................... 42

javax.oss.service ..................................................................................................................................... 85
  ServiceValue .......................................................................................................................................... 85
  ServiceKey ............................................................................................................................................. 86
  ServiceState .......................................................................................................................................... 86

javax.oss ...................................................................................................................................................... 89
  Lookup .................................................................................................................................................. 90
  LookupHome .......................................................................................................................................... 97
  LookupResult .......................................................................................................................................... 98
  ManagedEntityKey ............................................................................................................................... 99
  ManagedEntityValue ........................................................................................................................... 101
  Event ..................................................................................................................................................... 105
  QueryValue ........................................................................................................................................... 107
  IllegalArgumentException ......................................................................................................................... 107
  SetException ......................................................................................................................................... 108
  UnsupportedOperationException ............................................................................................................. 109
  XmlSerializable .................................................................................................................................. 109

XML Schema ............................................................................................................................................ 113
Appendix ................................................................................................................................................... 315
Index .......................................................................................................................................................... 319
This document specifies the OSS/J Service Activation API.

It is assumed that the reader of this document has sufficient knowledge of OSS, J2EE and basic knowledge of service provisioning. If you are new to Operation Support Systems or to the OSS/J initiative, please consult http://java.sun.com/products/oss first. If you are new to J2EE, please consult http://java.sun.com/j2ee.

The assumed readership of this document are programmers or software architects who
• want to write clients that call the Service Activation API, or who
• want to implement the API.

Structure of the Specification
This specification consists of four parts:
• The service Provisioning Overview (chapter 2) describes how services are provided by using OSS and what role the OSS/J Service Activation API plays. This chapter is informative.
• The chapters 3 to 5 comprise the user guide of the specification. They introduce the main features and explain how to use the API in typical situations. These chapters are normative.
• The chapters 6 to 8 comprise the reference guide of the specification. The chapters are generated from the java source code by using the javadoc tool. These chapters are normative.
• The chapter 9 specifies the XML schema used in this specification. Note that this chapter will most likely change in the future. This chapter is normative.

All chapters (except chapter 2) define the API in a precise way. It is described what clients of the API can expect from an implementation and it is described what the implementation can expect from the client. If either the client or the implementation violates any statement in this document, the client and the implementation most likely cannot be plugged together. In addition to this specification, one may use the test compatibility kit (TCK) or the reference implementation (RI) to test the implementation or client, respectively.

To improve the precision of the specification, certain words have clearly defined semantics: The words "must", "must not", "required", "recommended", "may", and "optional" in this document are to be interpreted as described in RFC-2119. The word "mandatory" is synonymous to "required".

Example Programs
All generic examples are required to compile and execute with all implementations of the OSS/J Service Activation API. Thus, they are part of the specification. Examples that are marked "service specific" will only run on certain implementations.

To keep the examples short and understandable, the examples do not include exception handling. With the final release of this specification, the source code of all example programs will be available.
Additional Documents
The following documents and software packages are available or will be available in the future:

- General OSS/J documents and presentations from http://java.sun.com/products/oss
- OSS/J Design Guidelines: general guidelines and patterns that apply to all OSS/J APIs (*planned for april 2001*)
- OSS/J Service Activation API White Paper (*planned for summer 2001*)
- OSS/J Service Activation API Reference Implementation (*available for final release*)
- OSS/J Service Activation API Test Compatibility Kit (*available for final release*)
- OSS/J Service Activation API Example Programs (*planned for summer 2001*)

Community Draft Comments
This is the community draft version of the specification. Please send comments to jcp89-eg@mustang.europe.nokia.com

As this is not the final release of the specification, some areas are not as well defined as they should be:

- Example code has not yet been fully compiled. Thus, there might be both syntax errors and logical errors in the examples.
- The Java Value Type access interface has been specified completely. Still we anticipate minor changes to the Java Value Type interface, for example adding additional attributes or adding mandatory queries.
- Only limited time has been spent on the XML access interfaces. There might be some significant changes to this interface, including conceptual ones and changes to the scheme.
Service Provisioning

In Operation Support Systems, the area of Service Provisioning is vast, and complete standards or even de-facto standards are lacking. Several products manage specific parts of Service Provisioning. They can be integrated into an end-to-end solution, for example by using Enterprise Application Integration (EAI) technology. But these solutions are one-shot, service provider specific solutions and, as a consequence, are extremely complex and difficult to achieve, due to the lack of integration standards.

Therefore, the ability to reduce the integration effort via a set of standard, reusable software components to assemble OSS applications in a much shorter time, is an appealing prospect for all players in the OSS marketplace.

A typical service provisioning architecture might look like the one shown in the figure below:

The function of the shown systems is as follows:
Service Provisioning Overview

- **The Customer Care System** is responsible for capturing a subscriber’s request, qualifying the request, and submitting the request in the form of an order to other Service Provisioning Systems, e.g. to the Order Manager.
- A **B2B Gateway** may be an alternative source of orders to the Service Provisioning Systems. A B2B Gateway is responsible for mediating the exchange and execution of orders between trading partners (Service Providers).
- Either the types of Services offered by the Service Provider are hard-coded into the Customer Care System (and the Service Provisioning Systems) or the Services are described by means of a **Service Inventory System**. The Service Inventory API is beyond the scope of this specification, because systems with hard-coded Services do not require this API.
- **The Order Management System** coordinates the execution of a business process to fulfill an order. The activities within the business process may need to interact with the other subsystems to fulfill the order.
- **The Network Inventory System** assigns network resources to Services.
- **The Network Activation System** programs network elements, element management systems, and network management systems to deliver a Service.
- **The Network Elements** and their Element Management Systems (EMS) are systems, which enable network services by providing bandwidth across circuits, trails, and routes.
- Other Operational Support Systems and Business Support Systems include the following types of systems: Billing, Service Assurance, or Workforce Management.

The Service Activation API has been carefully designed in such a way that it can be used in at least two different places:

- As a north-bound interface for the Order Manager.
- As a north-bound interface for the Network Activator (which is then also the south-bound interface of the Order Manager.)

Note that the Service Activation API does not mandate this architecture. Other scenarios are possible as well. In fact, for new 3G services, we assume that there will be a more direct interface between the Customer Care applications and the network. Similar to 2G mobile service provisioning architectures, the API might also be used as is shown in the following figure.

![Service Provisioning Diagram]

The API was designed with this architecture in mind, too. This kind of simplified provisioning process reduces the costs of introducing new services significantly.
API Features

The OSS Service Activation API provides the following features:

- A clean separation between the long living service data and the shorter living order data.
- Support for asynchronous provisioning of the services: By using the Java Messaging Service, clients of the API are informed about the progress and completion of an order.
- Orders can be changed, suspended and cancelled after the order has been started. Orders can be scheduled.
- Support for bulk/mass operations, to support orders from independent or virtual service providers (B2B).
- Powerful, but simple query operations on orders. Vendors might provide additional, vendor specific query operations.
- The API is service agnostic, thus it can be used for all kinds of services, including 3G services or DSL based services. The API has been designed for extensibility: vendors can extend the existing services and orders easily, including the extension of the state models of orders and services.
- The API provides both an EJB level interface and an JMS level interface. By using the JMS level interface, B2B integration is greatly simplified: Incoming XML based orders, for example in an ebXML (www.ebxml.org) envelope, can be forwarded to this API.
- Full integration to J2EE: Because J2EE handles transaction and security issues, the Service Activation API is transactional aware and secure.
- The API follows the OSS/J design guidelines and is structured similar to other OSS/J APIs, thus reducing learning effort for this API significantly.

Service Specific Attributes

This specification does not define concrete attributes for services. I.e. it does not define the attributes for services like GSM voice service or DSL services.

These attribute definitions might be added in later versions of this specification. If they will be added, their definition will be taken from other industry initiatives, like 3GPP or DSL Forum.

Even if it is not specified what attributes a service will have, it is specified how the service attributes have to be modelled in the API: The specification requires that specific design patterns are used to implement services and their attributes. See chapter 5 for further information.

Issue: In addition, we could define an attribute style guide, that specifies individual attributes, their types and constraints. One example could be:
Name: IPAddress
Type: java.net.InetAddress
Constraint: must match the regular expression “d.d.d.d”, where d is [0-9][0-9]?[0-9]?
This would ensure that a) the same attributes in different services would have the same name and syntax and b) generic clients could have built-in logic that support these pre-defined attributes. Comments?

Plug & Work

The ultimate goal of this specification is that OSS/J components that follow the specification can be plugged together easily. But since this specification does not specify attributes for concrete services, plugging together OSS/J components requires special attention.

The current market for products that might want to ask the Service Activation API is split into two segments:
Service Provisioning Overview

- Service-generic products are products that can be customized during deployment time for required service types.
- Service specific products are products that have been optimized for a certain set of services types and can only be used for these.

Obviously, two generic products can always be plugged together if they use the API. On the design level, the products use the service types in a generic way and they may access the service types in a weakly typed way. But it should also be obvious, that you can only plug two service-specific products together, if they support the same service types. On the design level, the products will access the services types in a strongly typed manner.
The Service Abstraction

A Service, an instance of a Service Type, associates something of business value with a specific subscriber. Its attribute values reflect the state of the service, as activated by the Provider. A subscriber can have multiple Service Instances. (syn.: service or subscription)

A Service Type is a description of a type of Service. As a service has attribute values, the service type has attributes that define how a service can (or must) be adopted to individual customers (syn.: Service class or Service definition).

The specification defines that a service must support at least two states: ACTIVE and INACTIVE (see javax.oss.service.ServiceState). An inactive service is a service that is configured in the network, but it is intentionally not usable for the customer.

The common base class for all service types is ServiceValue. For a new service type ST, a newly derived class STValue has to be introduced. STValue defines the attributes of the new service types. This is shown in the following UML diagram:

Note that this specification only defines the base class ServiceValue, not its derived classes, the individual service types.

The Order Abstraction

The second main abstraction of the API is the Order: An order is a request to perform an operation on a service. The API defines four order types: ACTIVATE, DEACTIVATE, MODIFY, and REMOVE, see javax.oss.order.OrderType for further information. An order represents a potentially long-running business pro-
cess, running over multiple systems to ensure that the service wishes of the customer are propagated to the service providers' network and to other service providers.

Order attributes can be divided into two groups:

- The values for **Input Order Attributes** are provided by the client. By using these attributes, the client can adjust the order business process to specific needs. Examples include priority and scheduling information. An implementation may provide additional attributes. Depending on the order business process, the implementation or client may change input attributes while the order is executed: an example is that the implementation may reduce the bandwidth attribute because of limited resources in the network, or the client may change the bandwidth attribute because he has changed his mind.

- The values for **Output Order Attributes** are provided by the implementation. With them, the implementation provides information on how the order business process has been executed. This specification only defines one mandatory output attribute (ActualDeliveryDate), an implementation may add additional ones. Because output attribute values are calculated by the implementation, output attributes may be unpopulated if you retrieve all attributes.

The API defines a state model for all orders, which is shown in the following figure. The states are defined in the class `javax.oss.order.OrderState`, the transitions refer to methods of the interface `javax.oss.order.OrderManager`.

A typical order runs through the following states:

- After an order has been created, it is in the **NOT_STARTED** state. During this state, the client can change all input attributes of an order.

- After an order has been started, it is in the state **RUNNING**. Depending on the complexity of the service and the order, the order may stay for seconds, hours or even weeks in that state. An implementation may provide further substates, that indicate how far the order has proceeded. An implementation may allow an client to modify some input attributes even in the state **RUNNING**. An implementation may provide output attributes in the state **RUNNING**.

- If the order has been completed successfully, the state of the order is **COMPLETED**. This indicates that the service attached to the order has been pushed to the network. If required by the business process, an implementation may fulfill the order only partially. In this case, the end state of the order is also **COMPLETED**. But then additional output attributes should indicate which aspects of the order have not been fulfilled.

A client can control the state of an order. The state diagram above shows which state transitions can be initiated by the client and which are automatically performed by the implementation: transitions labeled with "server" are automatically performed, all others are initiated by the client.

For further details, see `javax.oss.order.OrderState`.
Orders and Services

An order is a request to perform an operation on a service. This implies that there is a certain relationship between orders and services. This relationship is shown in the following figure:

![State model of a service](image)

The figure shows the state model of a service. Each transition in the above figure shows what happens if you execute an order of a certain type on a service instance:

- To create a service, you have to use an order of type ACTIVATE.
- To modify an active or inactive service, you have to use an order of type MODIFY.
- To remove the service from the network, you have to use an order of type REMOVE.
- ...

The transitions in the above figure are orders and therefore, as described, it may take seconds, hours or even days to run through one transition to change the state of a service.

The relationship between Services and Orders on the instance level is not specified. Thus, an implementation may restrict the number of orders that are executed for one service to one or it may allow more than one order for a service. In the latter case, it is not specified how to solve conflicts if concurrently executed orders access the same service.

Managed Entity Values

Both orders and services share some functionality for concepts called population and generic attribute access:

- An attribute can be either populated or unpopulated. Only if an attribute is populated, its value can be accessed. The client may use this concept to improve network utilization by retrieving only a certain set of attributes. Or a client could update only part of an order by populating only a subset of the attributes.
- A client may deal with an OrderValue in a generic way: It can access the attributes without knowing the exact type of the OrderValue nor the exact type of the attributes. A client can ask the OrderValue for a list of attributes, their types and their values.

These basic functionality is provided by the interface javax.oss.ManagedEntityValue.

Access Interfaces: JVT, XVT, and XML/JMS

This API defines three access interfaces to work with services and orders:

- Java Value Type Access Interface (JVT)
- XML Value Type Access Interface (XVT)
Concepts

- JMS Access Interface (XML/JMS)
All access interfaces provide the same functionality, but use different technology. For more details, including guidelines when to use which access interface, please consult "OSS/J Design Guidelines".

The **Java Value Type** Interface (JVT) provides a standard EJB interface. The client calls the implementation by using RMI. In that case objects are transferred as Serialized java objects. In addition, events are sent from the implementation to the client by using JMS Messages containing serialized Java objects. This interface provides maximum performance, strong typing and full support of all J2EE features, like security and transactions. It is the most natural interface for Java clients, for example other EJBs, application clients and web clients.

The **XML Value Type** interface (XVT) provides an EJB interface that uses XML strings for arguments and results. Otherwise the interface is structurally similar to the Java Value Type interface. This interface is the most easy to use from other programming languages, for example from C++ clients or from Java Server Pages or Servlet that make use of XML.

The **XML/JMS** interface provides a request/reply style interface. The Messages contain XML documents. This interface is the most natural for loosely coupled clients, an example being Enterprise Application Integration clients, Business to Business (B2B) clients or clients that need an asynchronous interface.

This specification does not deal with the issue how the interfaces are implemented and what systems are used to implement orders and services. However, an implementation might make use of the freely available reference implementation. The reference implementation reduces the implementation effort of the three access interfaces to just one interface.

**JMS Messages**

All access interfaces use JMS to provide asynchronous communication.

JMS messages are composed of the following parts (the following paragraph is copied from the JMS specification):

- **Header** - All messages support the same set of header fields. Header fields contain values used by both clients and providers to identify and route messages.
• Properties - In addition to the standard header fields, messages provide a built-in facility for adding optional header fields to a message.
• Application-specific properties - In effect, this provides a mechanism for adding application specific header fields to a message.
• Standard properties - JMS defines some standard properties that are, in effect, optional header fields.
• Provider-specific properties - Integrating a JMS client with a JMS provider native client may require the use of provider-specific properties. JMS defines a naming convention for these.
• Body - JMS defines several types of message body which cover the majority of messaging styles currently in use.

The Service Activation API defines the following mandatory application-specific properties for all JMS messages sent by an implementation:

• OSSJ_DOMAIN_PROP_NAME, identifying the system that has published the message.
• ORDER_TYPE, describing the type of order that has changed.
• ORDER_PRIMARY_KEY, identifying the order that has changed.
• OSSJ_EVENT_TYPE_PROP_NAME, describing what kind of event has been sent.
• CLIENT_ID

A client may use this properties to do appropriate filtering to receive only the messages that are relevant for the client. Further details on the contents of these properties can be found in javax.oss.order.OrderMessageProperty.
Concepts
Using the Java Value Type Access Interface

By using the Java Value Type interface, clients perform standard RMI calls on a remote, stateless session bean interface. Order values are exchanged as serialized Java objects. If the state of an order or the attribute value of an order changes, the implementation sends JMS ObjectMessages.

The following diagram shows the interfaces involved when using the Service Activation API:
Using the API

The main interfaces are:

- The stateless session bean remote interface **OrderManager** acts as the main interface to most of the offered functionalities. The main responsibilities of the OrderManager are:
  - Providing a factory for the available service types.
  - Providing a factory for the available order types.
  - Providing methods to control the life cycle of an order from creation to removal.
  - Providing methods to retrieve single or multiple orders.
- If the state of an order changes or if attribute values of an order change, then JMS messages are sent. The payload of this ObjectMessage can be accessed by using the "Event" interfaces.
- Orders live (conceptually) inside the OrderManager and cannot be accessed directly. Instead the client deals with a client-side copy of an order called **OrderValue**. Changing the attributes of the (local) OrderValue object does not change the real order objects. Instead, the client has to call an appropriate method on the OrderManager to transfer the OrderValue object to the server which then makes the changes effective. Therefore the general pattern to change an order is to transfer the order value to the client, change it locally, and then send it back to the implementation. For example:

  ```java
  OrderValue order = orderManager.getOrder(key);
  // do some changes to the order objects
  orderManager.setOrder(order);
  ```

A client may perform the following steps to execute an order:

- locate an OrderManager by using the OSS/J Lookup service.
- register to the relevant Topics by using the OSS/J Lookup service.
- get a new OrderValue object by calling the factory method newOrderValue.
- on the returned OrderValue object, set the parameter of the order.
- get a new ServiceValue object by calling the factory method newServiceValue.
- on the returned ServiceValue object, set the parameter of the server.
- connect the order and the service by calling setServiceValue on the OrderValue object.
- call createOrder to create a new order object in the server.
- call startOrder.
- if supported by the implementation, you may suspend an order by calling suspendOrder.
- a client may abort an order by calling abortOrder.
- the client may call removeOrder in the end.

After a client has created an order, a client may receive JMS Messages:

- incoming JMS messages inform about the state changes of an order or about changes of attributes values.
- the order has been closed when a StateChangeEvent is received with getCurrentState().startsWith(OrderState.CLOSED).

The following paragraphs provide more details.

**Locating the OrderManager**

The home interfaces of the OrderManager can be retrieved by using the OSS/J Lookup Service. The OSS/J Lookup Service maintains a list of all OSS/J API implementations in one OSS. Clients query for implementations by providing attribute values to the Lookup Service. Example attributes are vendor or product version. The Lookup Service then returns all the implementations the attribute values of which match. For further information on the OSS/J Lookup Service, please consult the OSS/J Design Guidelines or see javax.oss.Lookup.

The following example finds all OrderManager implementations in one OSS and prints out their lookup service attributes:
javax.oss.Lookup lookup = ... // see OSS/J Design Guidelines on how to do that.
javax.naming.directory.BasicAttributes attributes = new BasicAttributes();
attributes.put(API_TYPE, OrderManagerHome.class.toString());
NamingEnumeration homeList = lookup.listHomes(attributes);

while (homeList.hasMore()) {
    SearchResult home = (SearchResult) homeList.next();
    NamingEnumeration attrList = home.getAttributes().getAll();
    System.out.println("OrderManager:");
    while (attrList.hasMore()) {
        Attribute attr = (Attribute) attrList.next();
        System.out.println("   " + attr.getID() + " = " + attr.get());
    }
}

Finding all OrderManagers for a given service type sType can be achieved using the following code:

// query OSS/J Lookup for all OrderManagers
javax.naming.directory.BasicAttributes attributes = new BasicAttributes();
attributes.put(API_TYPE, OrderManagerHome.class.toString());
NamingEnumeration homeList = lookup.listHomes(attributes);

// as a result of running the code below, omList is filled with OrderManagers
// some elements of the array may be null.
OrderManager omList[] = new OrderManager[homeList.size()];
int i = 0;
while (homeList.hasMore()) {
    SearchResult home = (SearchResult) homeList.next();
    OrderManagerHome omHome = (OrderManagerHome) PortableRemoteObject.narrow(
        home.getObject(), OrderManager.class);
    OrderManager om = omHome.create();
    String[] serviceTypes = om.getServiceTypes();
    // Search if there is an entry 'sType' in 'serviceTypes'
    Arrays.sort(serviceTypes);
    if (Arrays.binarySearch(serviceTypes, sType) > 0) {
        omList[i] = om;
        i++;
    }
}

Registering for incoming messages

An implementation of the Service Activation API sends JMS Messages in case the order objects are changed.
To receive these messages, a client has to register the relevant JMS topic. Depending on the client, this can be
done in two different ways:

- If the client is a java application, it must use the JMS API to subscribe to the topic. Optionally it might want
to use the OSS/J Lookup service.
- If the client is an EJB-based application, it must be a message-driven bean to receive the messages.

A java application client can subscribe to incoming messages as is shown in the following example. After exe-
cuting the code below, the method onMessage() of the object will be called for each incoming message.
Using the API

```java
javax.naming.directory.BasicAttributes attributes = new BasicAttributes();
attributes.put(API_TYPE, OrderManagerHome.class.toString());
NamingEnumeration tconList = finder.listConnectionFactories(attributes);
NamingEnumeration topicList = finder.listTopics(attributes);

while (tconList.hasMore()) {
    // extract connection factory and topic from the search results
    SearchResult sres = (SearchResult) tconList.next();
    TopicConnectionFactory tconFactory =
        (TopicConnectionFactory) sres.getObject();
    sres = (SearchResult) topicList.next();
    Topic topic = (Topic) sres.getObject();

    // subscribe
    TopicConnection tcon = tconFactory.createTopicConnection();
    TopicSession tsession = tcon.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
    TopicSubscriber tsubscriber = tsession.createSubscriber(topic);
    tsubscriber.setMessageListener(this);
    tcon.start();
}
```

A message driven bean that wants to subscribe to OrderManager messages has to take the following into account:

- The deployment descriptor of the message-driven bean must refer to the JNDI name of the topic which is used by the OrderManager.
- The JNDI tree of the container running the OrderManager must be federated into the JNDI tree of the container that is running the message driven bean.

Issue: How does the MDB container know about the correct connection factory to use? Does the above example work at all?

**Subscriber orders new Service**

The following example is a service specific example. It assumes that the implementation supports DSL_SERVICE.

The example shows how a complete order is executed. It assumes that a reference to an OrderManager has been retrieved and that onMessage() is called if new JMS messages from an OrderManager arrive. The previous paragraphs explain how to do that.
executeOrder() {
    OrderManager om = // ...
    // create service value and set attributes
    DSLServiceValue service = (DSLServiceValue) om.newServiceValue(ServiceType.DSL_SERVICE);
    service.setSubscriber("John Doe"); // identify the subscriber
    service.setBandwidth(2200000); // 2.2Mb/s DSL
    // create order value and set attributes
    OrderValue order = om.newOrderValue(OrderTypeEnum.ACTIVATE);
    order.setPurchaseOrder("42");
    order.setServiceValue(service);
    // create order and start it
    key = om.createOrder(order);
    order.setKey(key); // make order value consistent inside the client
    om.startOrder(key);
}

public void onMessage(Message msg) {
    ObjectMessage omsg = (ObjectMessage) msg;
    OrderStateChangeEvent orderEvent = (OrderStateChangeEvent) omsg.getObject();
    String newState = orderEvent.getCurrentState();
    System.out.println("The state of the order " + orderEvent.getOrderKey() + " has changed to the state " + newState);
    if (newState.startsWith(OrderState.CLOSED)) {
        System.out.println("ORDER CLOSED");
    }
}

Modifying a Running Order
While an order is running, the implementation may allow modifications to the order and its attached service. This can be done as follows:

public void setBandwidth(OrderValueKey key) {
    OrderValue order = om.getOrder(key);
    DSLServiceValue service = (DSLServiceValue) order.getService();
    try {
        if (order.getState().startsWith(OrderStateEnum.OPEN)) {
            service.setBandwidth(1000000); // set bandwidth to 1 Mbit/s
        }
    } catch (IllegalStateException e) {
        System.out.println("Changing the bandwidth was not possible because:" + e.getMessage());
    }
}

Cancelling a Running Order
An ongoing running order can be aborted by the client. To do so, simply call

    om.abortOrder(key);

 Afterwards, the order is in the state ABORTED_BYCLIENT.

Generic Clients
A generic client is a client that can deal with all kinds of services. The following code demonstrates how this kind of client can make use of the API.
Using the API

```java
OrderManager om = // ...;
// get services offered by this OrderManager and select one
String[] typedSupported = om.getServiceTypes();
String serviceType = askUser("Select one of the following services", typesSupported);
// get orders offered by this OrderManager and let end-user select one
typedSupported = om.getOrderTypes();
String orderType = askUser("What do you want to do?", typesSupported);
// instantiate order and service value from factory method
ServiceValue service = om.newServiceValue(serviceType);
OrderValue order = om.newOrderValue(orderType);
order.setService(service);
// ask for values for all attributes
String[] attributeNames = order.getAttributeNames();
for (int i=0; i<attributeNames.size(); i++) {
    Object value = askUser("Enter Value for Attribute " +
            attributeNames[i] + ":",
            order.getAttributeType(attributeNames[i]));
    order.setValue(attributeNames[i], value);
}
// create order value and create & start the order
key = om.createOrder(order);
order.setKey(key);
om.startOrder(key);
```

Issue: Clarify second level attributes (e.g. service) and the related semantics for setValue/getValue/...

Using the XML/JMS Interface

By using the XML/JMS interface, clients send and receive JMS Text Messages that contain XML instance documents.

The XML/JMS is conceptually similar to the Java Value Type interface. However, since we are dealing with an asynchronous message based interface (as opposed to synchronous Java interfaces), each operation/method from the OrderManager Session Bean has been broken down into three message types:

1. Request message contains the initial request from the client i.e. the arguments from the operation/method of the OrderManager.
2. Response message contains the result (if any) of the request i.e. the return value (if any) from the operation/method of the OrderManager.
3. Exception message which is returned instead of the Response message if an error occurred in processing i.e. if an exception is thrown from the operation/method of the OrderManager.

In addition to these three messages, a fourth type of message models the events:

4. Event Messages are equivalent to the JavaObject Events in the Java Value Type interface, except the messages are in XML format. The XML Events are the payload of a JMS Text Message, which are distributed as multicast messages.

The following are the messages exchanged when using the Service Activation XML Messaging:

1. Client creates a JMS Text Message with the payload being a valid XML Request Message (based on the Service Activation Schema defined in this API). The client then sends the request to a JMS Message Queue and awaits a response from another queue.
2. OrderManager receives the request, processes it, sends a JMS TextMessage containing the response message. Alternatively, it might also send an exception.
3. If necessary, the OrderManager creates an Event message (i.e. if an order was created an OrderCreateEvent must be published).
Using the API

This is visualized in the following figure:

<table>
<thead>
<tr>
<th>XMLClient</th>
<th>JMS Message Queue</th>
<th>Message/Order/ChangeEvent</th>
<th>OrderManager(OL)</th>
<th>JMS Event Topic</th>
</tr>
</thead>
</table>

The XML Schema

The XML/JMS interface is conceptually based on the Java value type interfaces. Exactly the same operations, carried out by the Java value type interfaces can be accomplished using this XML/JMS interfaces. Each data type in the API (i.e. OrderValue, OrderKey, ServiceValue, ServiceKey) has a corresponding complexType in XML. The polymorphic behavior of OrderValue and ServiceValue is preserved in XML, by declaring these types as abstract. An abstract type in XML indicates that the type must be extended and not be used directly. As an example, here is the schema definition of an OrderValue:

```xml
<complexType name="OrderValue" abstract="true">
   <sequence>
      <element ref="sa:BaseState" nullable="true" minOccurs="0"/>
      <element name="OrderKey" type="sa:OrderKey" nullable="true" minOccurs="0"/>
      <element name="ServiceValues" type="sa:ArrayOfServiceValue" minOccurs="0"/>
      <element name="ClientId" type="string" nullable="true" minOccurs="0"/>
      <element ref="sa:Priority" nullable="true" minOccurs="0"/>
      <element name="Description" type="string" nullable="true" minOccurs="0"/>
      <element name="RequestedDeliveryDate" type="timeInstant" nullable="true" minOccurs="0"/>
      <element name="ActualDeliveryDate" type="timeInstant" nullable="true" minOccurs="0"/>
      <element name="OrderDate" type="timeInstant" nullable="true" minOccurs="0"/>
   </sequence>
</complexType>
```

Below you will find an example of how OrderValue would be extended in an XML Schema:
Using the API

```xml
<complexType name="Activate">
  <complexContent>
    <extension base="sa:OrderValue">
      <sequence>
        <element name="AdditionalAttribute" type="string"
          nullable="true" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

**Example: Order Creation**

In order to help understand the format and content of the messages, the example below is provided. It should help illustrate the basic concepts of how to design an XML Service Activation Request.

The fundamental requirement of the API is to allow the creation of an order. To create an order, the client must provide the necessary attributes of that order. To do so with the XML/JMS interface the client creates an OrderCreateByValueRequest XML instance document, which contains an OrderValue instance. Similarly to the OrderCreateEvent example below, this example will use the “Activate” Order (which extends the OrderValue type as illustrated in the Derived Schema above). The client would create a JMS Text Message with this OrderCreateByValueRequest XML document being the payload, and pass it to a JMS Queue.

The example takes into consideration that the client has already made a GetOrderTypesRequest (to determine what order types exist in the system). This is in addition to a NewOrderValueRequest to create a OrderValue instance (this is to get a skeleton OrderValue that the client would populate with values that would be passed to the order creation request).

```xml
<sa:CreateOrderByValueRequest
  xmlns="http://www.somewhere.org/DSLService"
  xmlns:sa="http://www.somewhere.org/ServiceActivation"
  xmlns:co="http://www.somewhere.org/Common"
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
  xsi:schemaLocation="http://www.somewhere.org/DSLService DSLService.xsd">
  <sa:OrderValue xsi:type="Activate">
    <sa:ServiceValues>
      <sa:Item xsi:type="DSL">
        <Bandwidth>1000000</Bandwidth>
      </sa:Item>
    </sa:ServiceValues>
    <sa:ClientId>1</sa:ClientId>
    <sa:Priority>1</sa:Priority>
    <sa:Description>New Customer</sa:Description>
    <AdditionalAttribute>Test</AdditionalAttribute>
  </sa:OrderValue>
</sa:CreateOrderByValueRequest>
```

Once the order is created successfully, the client listening on the JMS Queue would receive the following OrderCreateByValueResponse message which contains the OrderKey of the order just created.

---

28
If the order is not created successfully, then a number of exceptions can be returned. In this example the RemoteException is returned.

Example: Order Create Event

In the Request/Response example an order has been created. Once the order has been created, the OrderManager will publish an OrderCreateEvent on two separate topics (one for XML Events and the other for JavaObject Events). The Order Create example is continued below, with the XML version of the OrderCreateEvent. A client listening on the JMS XML Topic would receive this JMS Text Message, with the payload containing this XML document.
Using the API

```xml
<sa:OrderCreateEvent
  xmlns="http://www.somewhere.org/DLService"
  xmlns:sa="http://www.somewhere.org/ServiceActivation"
  xmlns:co="http://www.somewhere.org/Common"
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
  xsi:schemaLocation="http://www.somewhere.org/DLService DSLService.xsd">
  <sa:Event>
    <co:EventTime>2001-01-01T13:20:00.000-05:00</co:EventTime>
    <co:Domain/>
    <co:NotificationId/>
    <co:SourceIndicator/>
    <sa:OrderValue xsi:type="Activate">
      <sa:State>open.running</sa:State>
      <sa:OrderKey>
        <sa:Domain>"domain"</sa:Domain>
        <sa:PrimaryKey>1</sa:PrimaryKey>
        <sa:Type>"DSL"</sa:Type>
      </sa:OrderKey>
      <sa:ServiceValues>
        <sa:Item xsi:type="DSL">
          <sa:ServiceKey>
            <sa:Domain>"domain"</sa:Domain>
            <sa:PrimaryKey>1</sa:PrimaryKey>
            <sa:Type>"DSL"</sa:Type>
          </sa:ServiceKey>
          <Bandwidth>1000000</Bandwidth>
        </sa:Item>
      </sa:ServiceValues>
      <AdditionalAttribute>This is an DSLActivate Example</AdditionalAttribute>
    </sa:OrderValue>
  </sa:Event>
</sa:OrderCreateEvent>
```

Using the XML Value Type Interface

By using the XML Value Type interface, clients perform standard RMI calls on a remote, stateless session bean interface. Order value are exchanged as XML strings. If the state of an order or the attribute value of an order changes, the implementation sends JMS TextMessages that contain XML strings.

The following diagram shows the interfaces involved when using the XML Value Type interface:
For each method in the interface XMLOrderManager there is a corresponding method in the related Java Value Type interface, OrderManager.

The format of the XML documents representing arguments/return values and exceptions are the same as the XML request/response/exception documents from the XML/JMS interface (for further explanation about the format of these XML instance documents please refer to the XML/JMS interface section). Below is how the method signatures from XMLOrderManager map to the XML/JMS message types.

- Argument list of the methods would contain a single string argument, which contains the XML Request instance document.
- Return Value of the methods are a string, which contains the XML Response instance document.
- Exceptions thrown from the methods are the same Java exceptions from the OrderManager except that these exceptions will contain the XML Exception instance document.

An example is on the createOrderByValue method, the method would take a single string argument (request), the content of the string would contain a CreateOrderByValueRequest XML document. The return value from the method would be a string that would contain CreateOrderByValueResponse XML document and any exception thrown by the method would contain the CreateOrderByValueException XML document. Here is the signature of this method:

```java
String createOrderByValue(String request) throws javax.ejb.CreateException, java.rmi.RemoteException, javax.oss.IllegalArgumentException;
```

### Processing Overview – Composing the XML argument and making the Method Call

The following are the basic steps in using the Service Activation XVT.

- Client creates an XML request document (based on the Service Activation Schema defined in this API). The client then calls the corresponding method in the XMLOrderManager passing the XML document as an argument to the method. The method will either return a XML response document or throw a Java exception.
- XMLOrderManager processes the method call, sends back an XML response or throws a Java Exception (which would contain an XML document exception). If necessary, an event is created (i.e. if an order was created an OrderCreateEvent must be published).
- Client will receive the return value or thrown exception. The client for example might then use XSLT to display the XML response/exception document to the client’s browser.
Using the API

These steps are visualized in the following figure:

In order to help understand the format and content of the Request document, the example below is provided. It should help illustrate the basic concepts of how to design an XML Service Activation Request.

**Example: orderCreateByValue(String request)**

The fundamental requirement of the API is to allow the creation of an order. To create an order, the client must provide the necessary attributes of that order. To do so with the XVT interface, the client creates an OrderCreateByValueRequest XML instance document, which contains an OrderValue instance. This example will use the “Activate” Order. The client will then make a call to createOrderByValue, passing an OrderCreateByValueRequest XML document as a parameter.

The example takes into consideration that the client has already made a getOrderTypes method call (to determine what order types exist in the system). This is in addition to a newOrderValue method call to create an OrderValue instance (this is to get a skeleton OrderValue that the client can populate with values to be passed to the order creation request).
Using the API

Once the order is created successfully, the method call returns an OrderCreateByValueResponse XML instance document which contains the OrderKey of the order just created.

If the order is not created successfully, a Java exception is thrown from the method. The Java exception contains an XML exception instance document. In this example the Java RemoteException is thrown and contained in this exception is the below XML RemoteException instance document.

**Additional Client-Side Notes**

During the development of clients for the API, certain items require special attention to create a fully interoperable client. These items are discussed in the following.

**Preparing for Polymorphic Value Types**

The (dynamic) type of the object returned by newOrderValue() is always a derived type of OrderValue. For example, the newOrderValue(ACTIVATE) might return objects of type VendorXActivateOrder. This has a number of consequences:

- When dealing with multiple implementations, never mix the objects. For example, this code will not work:
Using the API

```java
// WRONG code to move an order form OrderManager omOne to another omTwo!
OrderValue order = omOne.getOrder(key);
omTwo.createOrder(order);
```

- Each OrderManager can only deal with those value types that are returned by its newOrderValue() method. It is not possible to pass an object retrieved from an OrderManager X to an OrderManager Y.
- The client has to know the classes that the implementation uses. There are two ways to achieve that:
  - The OrderManager might be deployed on a J2EE container that supports dynamic class loading of the required ValueType classes. This works as follows: The packet returned by newOrderValue() contains not only the serialized order value object (which includes the class name), but also an URL where to download class files. If the client deserializes the packet and detects that it is confronted with an unknown class, the RMI class loader uses the URL to download the missing class. This alternative is the recommended one. We also recommend to enable SSL for the class download. Note also, that this mechanism avoids class name clashes: If a client uses two OrderManagers that use the same class name for different classes, the client can still deal with this situation.
  - If you connect a new OrderManager to the client, you have to deploy a jar file to the client. The jar file contains all ValueTypeTypes and perhaps required support classes. The jar file should then be part of the CLASSPATH.

Preparing for Substates of an Order and Service

The state model for services and orders can be extended by each implementation. This is done by adding additional (sub-) states to the end of the string. For example a running order might have the additional states “firewall-change”. Then getState() might return “open.running.firewall-change”.

Thus, never write the following code:

```java
if (state.equals(CLOSE)) {  // WRONG
...
}
```

Instead, write

```java
if (state.startsWith(CLOSE)) {  // CORRECT
...
}
```

The same is true for the state attribute of services.

If you violate this rule, your client might run with certain OrderManagers, but definitely not with all.

Preparing for SetException due to Optimistic Locking

The API follows the programming style of optimistic locking.

If an order is modified in the following way

```java
order = getOrder(key);
order.setAttribute(...);
setOrder(order);
```

then getOrder does not lock the order in the OrderManager.

To prevent that one setOrder() call overwrites the previous changes, an implementation may throw a SetException. If the implementation supports this, the following code will throw a SetException, even if no other client is running:
Using the API

```java
order = getOrder(key);
order.setAttribute(...);
setOrder(order);
order.setAttribute(...); // reuse old order value
setOrder(order);         // May throw exception
```

The following code will not throw a SetException if there is only this client running:

```java
order = getOrder(key);
order.setAttribute(...);
setOrder(order);
order = getOrder(key);  // get fresh order value
order.setAttribute(...);
setOrder(order);
```

Of course, in both cases, already the first call to setOrder() may throw a SetException, if

- the implementation supports it
- the other, concurrently running client makes changes to the same order.

Issue: What is the semantics of having getOrder() / setOrder() in one transaction?

Handle Exceptions

The exceptions thrown by the remote interface OrderManager have been designed in such a way that a client can handle them easily. For that, the client has to distinguish two cases:

- `javax.oss.IllegalArgumentException` most likely indicates a logic error (bug) in the client. Thus a client may handle this exception as follows:
  ```java
  try {
      // call OrderManager
  } catch (javax.oss.IllegalArgumentException e) {
      throw java.lang.IllegalArgumentException(e.getMessage());
  }
  ```

  The same is true for `UnsupportedOperationException`, because the client can check beforehand wether a method is implemented or not by calling
  ```java
  om.getSupportedOptions()
  ```

- All other Exceptions indicate that the client is programmed correctly, but that the method cannot be executed. The underlying reason for that may lay in the distributed nature of the API. One example is `IllegalStateException`: If there is only one client using the API, this Exception should never be thrown. But if multiple clients are using the same OrderManager, `IllegalStateExceptions` may be thrown. The same is the case for `SetException`.

  This specification does not deal with the issue how these exceptions should be handled in the client or how the client can prevent the Exceptions from being thrown.

Preparing for different Exceptions for Local and Remote Calls

The methods of `ServiceValue` or `OrderValue` objects may throw `RuntimeExceptions`. One possible exception is `java.lang.IllegalArgumentException` which is thrown by `setAttribute()` in case the value for attribute is incorrect. For example se the method `setPriority` of `OrderValue`.

This behaviour is similar to other local Java APIs.

However, remote objects behave differently. Therefore, an OrderManager does not throw an `java.lang.IllegalArgumentException` in a similar situation, but instead `javax.oss.IllegalArgumentException`, which is inherited from `java.lang.Exception`. In J2EE terms, exceptions derived from `java.lang.Exception` and declared in the throw clause of a method are called business exceptions.

The underlying reason for this behavior is that exception thrown in the J2EE container have special effects on the container. Please consult the EJB 2.0 specification, chapter 17, for further details.
Using the API

As an example, the following local call throws the unchecked java.lang.IllegalArgumentException:

```java
order.setPriority(-1);
```

The following remote call throws a checked javax.oss.IllegalArgumentException:

```java
om.newOrderValue("this types does not exist");
```

**Attribute 'State' is not a Standard Attribute**

Even if the attribute State seems to be a normal attribute of an order, this is not the case.

First, you cannot change the state of an order by calling setOrder(). The state is ignored by this method. Instead, use methods like startOrder(), suspendOrder(), etc.

Second, changes to the state are not notified to the client by an AttributeValueChangeEvent. Instead, a StateChangeEvent is sent.
This section provides guidelines for those who want to implement the API. It explains how to implement new service types with their attributes, new order types with their attributes and state model.

The main source for the implementation is the description of the API, which is described in the next chapters. An implementation must support the functionality described in the next chapters. Any violation, especially of the mentioned postconditions, will lead to problems while plugging together clients and implementations.

This chapter concentrates on two areas:

- It explains some additional requirements, which must be implemented.
- Given an existing product, it gives guidelines how to extend the API. By using numerous possibilities, an implementation may provide more information on the orders and services to the client.

### Registering to the OSS/J Lookup Service

Each OrderManager must be registered in the Lookup service with the following attributes:

- TODO: Add all attributes required by the OSS/J Design Guidelines.

Issue: Should we put the service types to the JNDI tree? E.g. "The value must also be returned by OrderManager#getServiceTypes(). The OSS/J Lookup must have one entry for each service type that is supported by the OrderManager. If an OrderManager supports the service DSL_SERVICE and GSM_VOICE (as returned by OrderManager#getServiceTypes() ), then the OSS/J Lookup must contain two entries, one for each service type. Comments? Can I still use the standard deployment tools of an apps server in this case?"

### Programming Restrictions for the Value Types

If clients access the API by using the Java Value Type interface, the byte code of the value types will be executed in the JVM of the client. The environment of the client is different from the environment implementation, i.e. the container that is running the OrderManager implementation.

Therefore, the implementation of the Value Types (for example OrderValue, OrderKey, ...) must take the following into account:

- Only use APIs that are part of J2SE.
- Follow the programming restrictions of EJB code. This is required in case the client is an EJB. Note that you are not allowed to use the full J2SE APIs.
Implementing the API

Extending Service Information Model

The Service Activation information model is extended by specialization. The Service Activation specification intends for the OrderValue and ServiceValue interfaces to be subclassed by service-specific specifications. These derived value types must follow the standard JavaBean property patterns (i.e., getters and setters).

This specification does not handle the question, how the underlying Order and Service (conceptual) entities are associated with the OrderValue and ServiceValue subclasses respectively. An implementation may use Entity-Beans to model the underlying entities; however, that is not required, nor is an EntityBean implementation visible through the Service Activation API.

For example, the following Java code and XML Schema extend a ServiceValue to be service-specific.

```java
public interface DslServiceValue extends javax.oss.service.ServiceValue {
    public int getSpeed();
    public void setSpeed(int speed);
}

<complexType name="DslServiceValue">
    <extension base="sa:ServiceValue">
        <sequence>
            <element name="Speed" type="unsignedInt"/>
        </sequence>
    </extension>
</complexType>
```

When the extended ServiceValue is delivered as XML, it looks like the following example.

```xml
<ServiceValue xsi:type="dsl:DslServiceValue">
    <ServiceState> ... </ServiceState>
    <ServiceKey> ... </ServiceKey>
    <Speed>2200000</Speed>
</ServiceValue>
```

Service Bundling

An implementation may provide multiple services, i.e. getServiceTypes() returns more than one service type. If this is the case, a client may want to create an order to activate multiple services with one order.

The recommended way to deal with this situation is: Define a new service type that aggregates the other services. An example interface definition:

```java
public interface ComplexGsmVoice extends ServiceValue {
    public HlrServiceValue getHlrService();
    public void setHlrService(HlrServiceValue svc)
    public VmsServiceValue getVmsService();
    public void setVmsService(VmsServiceValue svc)
    public SmscServiceValue getSmscService();
    public void setSmscService(SmscServiceValue svc)
}
```

Extending the Order to a Business Process

State Model

States are extended by a Design Pattern. Order substates may be added. A substate of a standard state may be added by defining a new string constant, whose value is the standard state’s string constant suffixed by a period followed by the name of the substate. Substates semantically must represent a specialization of (more specific
Implementing the API

than) the parent state, so that clients that do not recognize the substate can treat it the same as the known parent state.

The Service Activation specification expects clients to consider unknown substates equivalent to the known superstate, if the substate’s string constant is prefixed by the known superstate’s string constant. For example, “open.running.compensating” if unrecognized should be treated as “open.running”.

The steps of a business process may be modeled as substates of the state RUNNING. These substates could be used by the client to inquire how far the order provisioning process has proceeded. For example, if the order process requires that a firewall is configured and afterwards a GGSN is provided, it may introduce two substates: open.running.firewall and open.running.ggsn.

An implementation may only introduce additional substates of the following states: NOT_STARTED, RUNNING, SUSPENDED, COMPLETED, ABORTED_BYSERVER, ABORTED_BYCLIENT. An implementation must not introduce additional substates of the following states: OPEN, CLOSED, NOT_RUNNING.

For example, the following Java code extends the Order state model to be service-specific.

```java
public interface GprsAccessOrderState extends javax.oss.order.OrderState {
    public static final String LOOP_QUALIFY = "open.running.firewall";
    public static final String LOOP_QUALIFY = "open.running.ggsn";
}
```

Input Attributes

An implementation may add additional input attributes on the order.

Output Attributes

An implementation may add additional attributes on the order.

Rollback

An implementation may provide support for rollback of the order business process.

The rollback could be initiated when

- the client calls abortOrder()
- the implementation detects that the order cannot be completed.

The implementation may provide further information to the client about a possible rollback by either

- introducing new substates for ABORTED and/or
- introducing additional output attributes for the order.

Extending the Interaction Model

The Service Activation interaction model is fixed. The specification does not intend for the OrderManager interface to be extended with additional operations by derived interfaces. The reason that specialization cannot be supported for extending the OrderManager with additional operations is because the J2EE specification does not allow the OrderManagerHome EJBHome interface to be polymorphic (allowing specializations to inherit from a common base class).
Implementing the API

Extending the Event Model

Events are extended by specialization. Clients that recognize the standard events will be able to understand the extended events.

All events emitted by an implementation must be subclasses of the event types defined in this specification. An implementation must not emit other event types.

For example, the following Java code and XML Schema extend the OrderAttributeValueChangeEvent.

```java
public interface CustomerInfoChangeEvent
    extends javax.oss.order.OrderAttributeValueChangeEvent {
    public String getCustomer();
}
```

```xml
<complexType name="CustomerInfoChangeEventType">
    <extension base="sa:OrderAttributeValueChangeEventType">
        <sequence>
            <element name="Customer" type="string"/>
        </sequence>
    </extension>
</complexType>
```

When the extended event is delivered as an XML message, it looks like the following example.

```xml
<OrderAttributeValueChangeEvent>
    <Event xsi:type="CustomerInfoChangeEventType">
        <OrderValue> ... </OrderValue>
        <Customer>John Doe</Customer>
    </Event>
</OrderAttributeValueChangeEvent>
```

Other Items

Internationalisation

An implementation may return certain strings localized. These strings are:

- Details in exceptions.

Attribute names must not be localized.
CHAPTER 6

Package
javax.oss.order

Description
Provides support for asynchronous execution of orders that effect services.

This package provides an interface for orders. Orders are changes to services that need some time to complete.

The main interfaces in this package are:

- **OrderManager**: A stateless session bean that allows you to create, change and remove orders. Furthermore an order can be started, suspended and aborted.
- **XMLOrderManager**: Similar to the previous interface, except that all arguments are strings containing XML documents.

Class Summary

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderAttributeValueChangeEvent</td>
<td>Event that is sent if any value of an order has changed.</td>
</tr>
<tr>
<td>OrderCreateEvent</td>
<td>Event that is sent if a new order has been created.</td>
</tr>
<tr>
<td>OrderKey</td>
<td>Value type interface: representing an OSS wide unique key to an order.</td>
</tr>
<tr>
<td>OrderManager</td>
<td>EJB remote interface, the central interface of this package to create, manage and remove orders.</td>
</tr>
<tr>
<td>OrderManagerHome</td>
<td>EJB home interface for an OrderManager.</td>
</tr>
<tr>
<td>OrderManagerOption</td>
<td>String constants to indicate which optional parts of the interface are supported by the implementation.</td>
</tr>
<tr>
<td>OrderMessageProperty</td>
<td>String constants that define the predefined JMS properties.</td>
</tr>
<tr>
<td>OrderPriority</td>
<td>Constants defining the possible priorities that can be used with setPriority(int).</td>
</tr>
<tr>
<td>OrderRemoveEvent</td>
<td>Event that is sent if an order has been removed.</td>
</tr>
<tr>
<td>OrderState</td>
<td>String constants that define the predefined states of an order.</td>
</tr>
<tr>
<td>OrderStateChangeEvent</td>
<td>Event that is sent if an order has changed its state.</td>
</tr>
<tr>
<td>OrderType</td>
<td>String Constants that define the predefined order types.</td>
</tr>
<tr>
<td>OrderValue</td>
<td>Value type interface for accessing order values.</td>
</tr>
<tr>
<td>OrderValueIterator</td>
<td>EJB remote interface, returned by OrderManager methods to allow iteration over a lot of orders.</td>
</tr>
<tr>
<td>XMLOrderManager</td>
<td>The XMLOrderManager is a synchronous XML/Java based interface.</td>
</tr>
<tr>
<td>XMLOrderManagerHome</td>
<td>Home EJB interface for an XMLOrderManager.</td>
</tr>
</tbody>
</table>
**OrderManager**

**Class Summary**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLOrderValueIterator</td>
<td>The XMLOrderValueIterator is similar to the OrderValueIterator, except that the returned value on a getNext will return a string.</td>
</tr>
</tbody>
</table>

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IllegalStateException</td>
<td>Exception thrown if the state of an order is not appropriate to execute the method.</td>
</tr>
</tbody>
</table>

**OrderManager**

**Syntax**

```java
public interface OrderManager extends javax.ejb.EJBObject
```

**All Superinterfaces:** `javax.ejb.EJBObject, java.rmi.Remote`

**Description**

EJB remote interface, the central interface of this package to create, manage and remove orders. The methods of this interface fall in one of the following categories:

- Creating order and service values
- State handling of orders
- Changing and retrieving orders
- Querying for orders
- OrderManager information

**Creating order and service values**

A client of this API cannot change orders directly. Instead, most operations make use of an `OrderValue` that is transferred between the client and the implementation. The same is true for a `ServiceValue`.

Use `getOrderTypes()` and `getServiceTypes()` to learn about the orders and services that this `OrderManager` supports. Use `newOrderValue(String)` and `newServiceValue(String)` to instantiate a new order or service.+

Note that these operations only create a client-side object, use `createOrder(OrderValue)` to create the order in the server.

**State handling of orders**

During the lifetime of an order, the order runs through a set of states. A certain set of states is predefined, see `OrderState`. Every implementation may provide additional states which are sub-states of the predefined ones.

The above mentioned states are returned by `getState()`. The state can be changed by using the methods:

- `createOrder(OrderValue)`
- `startOrder(OrderKey)`
- `suspendOrder(OrderKey)`
- `resumeOrder(OrderKey)`
• abortOrder(OrderKey)\textsuperscript{44}, and
• removeOrder(OrderKey)\textsuperscript{50}.

For the create and remove-methods, bulk-operations are also available.

**Changing and retrieving orders**

A single order can be retrieved by using \texttt{getOrder(OrderKey)}\textsuperscript{46}. Multiple orders can be retrieved by using \texttt{getOrders(OrderValue[], String[])}\textsuperscript{47}.

By using the template mechanism described there, all queries that only use an equal operator for comparison can be executed.

Use \texttt{setOrder(OrderValue)}\textsuperscript{51} to modify an order.

**Querying for Orders**

The method \texttt{queryOrders(QueryValue, String[])}\textsuperscript{49} can be used to perform more complex queries. Implementations may, but are not required to support this functionality.

Use \texttt{getQueryTypes()}\textsuperscript{48} to learn about the queries that an implementation provides. Use \texttt{newQueryValue(String)}\textsuperscript{49} to create a query value that can be filled with the query arguments and then passed to \texttt{queryOrders}.

**OrderManager information**

An implementation of an OrderManager might differ from vendor to vendor. For example the Nokia OrderManager might support other orders than the Nortel OrderManager. Use \texttt{getSupportedOperations()}\textsuperscript{48} to find out about the differences.

**Events**

If orders are created, changed or removed the OrderManager sends JMS messages to a topic. The topic can be retrieved by calling \texttt{lookup.findTopic(key.getDomain())}, where key can be retrieved from the order value object.

The following events are sent:

• \texttt{OrderCreateEvent}\textsuperscript{75},
• \texttt{OrderAttributeValueChangeEvent}\textsuperscript{77},
• \texttt{OrderStateChangeEvent}\textsuperscript{76}, and
• \texttt{OrderRemoveEvent}\textsuperscript{78}.

The following is true on all the JMS message \texttt{msg} published:

• \texttt{msg.getStringProperty(OSSJ_DOMAIN_PROP_NAME).equals(result.getDomain())}
• \texttt{msg instanceof ObjectMessage}
• \texttt{msg.getObject().getEventTime() is set.}
• \texttt{msg.getObject().getDomain().equals(result.getDomain())}

Issue: Event type is a string containing the interface name. What is states the Design Guidelines?

**See Also:** \texttt{OrderValue}\textsuperscript{67}, \texttt{OrderManagerHome}\textsuperscript{54}
Abort the order and put it to state ABORTED.

Postcondition:

getOrder(pk).getState().startsWith(ABORTED).

Message Postcondition: The following is true on the JMS message msg published to the topic TODO:

- msg.getStringProperty(OSSJ_EVENT_TYPE_PROP_NAME).equals(OrderStateChangeEvent.class.getName())
- msg.getObject() instanceof OrderStateChangeEvent
- msg.getObject().getKey().equals(pk)
- msg.getObject().getCurrentState().startsWith(ABORTED)

Parameters:

pk - Primary key for an order.
Throws:
   IllegalArgumentException - if violated: The key has been returned by createOrder(OrderValue) of this OrderManager before.
   RemoteException

createOrder(OrderValue)

public OrderKey createOrder(OrderValue value)
   throws javax.oss.IllegalArgumentException, javax.ejb.CreateException

Creates a new order object and returns the key for the new object. The argument value is used to initialize the newly created order: The order is initialized from all populated attributes. The values of the attributes key and state are ignored, even if they are populated. Instead, a new, unique key is created by this method and returned as result. The state is initialized to NOT_STARTED.

Postcondition:
   * result.getType().equals(type), where type is the type used in newOrderType();
   * result.getDomain() can be used with Lookup to find the home interface of this OrderManager.
   * result.getPrimary().toString() is different from all other primary keys of the orders of this OrderManager.

Message Postcondition: When this method is called, a JMS message msg will be published:
   * msg.getStringProperty(OSSJ_EVENT_TYPE_PROP_NAME).equals(OrderCreateEvent.class.getName())
   * msg.getObject() instanceof OrderCreateEvent
   * msg.getObject().getOrderValue() has at least these attributes populated, that where passed to this method. In addition, the attributes key and state must be populated. An implementation may populate further attributes.

Throws:
   IllegalArgumentException - if the some attribute values make it impossible to create the order.
   javax.ejb.CreateException - TODO
   TODO: event.notification id?
   TODO: Do we need event time at all?
   RemoteException

createOrders(OrderValue[])

public OrderKey[] createOrders(OrderValue[] values)

Creates multiple order objects and returns for each a new key. The semantics of this method is equivalent to the semantics of this code:
OrderManager

getOrder(OrderKey)

```java
OrderKey[] result = new OrderKey[values.size()];
for (int i=0; i<values.size(); i++) {
    result[i] = createOrder(values[i]);
}
return result;
```

**Throws:**
RemoteException, CreateException, IllegalArgumentException

getOrder(OrderKey, String[]) throws javax.oss.IllegalArgumentException

Equivalent to `getOrder(OrderKey, String[])` (key, null).

**Throws:**
RemoteException, IllegalArgumentException

getOrders(OrderKey[], String[]) throws javax.oss.IllegalArgumentException

`getOrders(OrderKey[], String[])` throws javax.oss.IllegalArgumentException

Returns values for a list of orders identified by the (unique) keys. The semantics of this method is equivalent to the semantics of this code:
OrderManager

getOrders(OrderValue[], String[])

OrderValue[] result = new OrderValue[keys.size()];
for (int i=0; i<keys.size(); i++) {
    result[i] = getOrder(keys[i], attributeNames);
}
return result;

For detailed description see getOrder(OrderKey, String[]) 46.

Throws:
IllegalArgumentException - as stated in getOrder(OrderKey, String[]) 46 and if pks == null.
RemoteException

getOrders(OrderValue[], String[])

public OrderValueIterator getOrders(OrderValue[] templates,
                                        java.lang.String[] attrNames)
                                throws javax.oss.IllegalArgumentException

Queries all orders and returns the ones that match at least one of the template orders. This method can be used to perform simple queries on the orders. It can handle all queries that need to compare attributes by equality.

The objects that are indirectly returned by the iterator have the following attribute values:

- Every order must matches at least one template order (logical or).
- An order matches a template if all populated attribute values in the template are equal to the order (logical and).

TODO: Add example.

Issue: what is the semantics of passing a unpopulated template? matches all or disallowed?

Issue: Do we need special semantics for state, e.g. match only prefix?

Issue: How to search for all orders on a specific service?

Throws:
IllegalArgumentException - if violated:
- attributeName is a subset of result.getAttributeNames()
- attributeName.size() > 0 (at least one attribute must be populated)
UnsupportedOperationException, RemoteException

getOrderTypes()

public java.lang.String[] getOrderTypes()

Returns a list of Order types, that can be used with newOrderValue(String) 48.

Postcondition:

- result.sie() > 0 At least one order type must be supported.
- result must include ACTIVATE order type. Support for that is mandatory.

Throws:
RemoteException

See Also: OrderType 83
OrderManager

getQueryTypes()

getQueryTypes()

```
public java.lang.String[] getQueryTypes()

Returns all implemented query types.
}
```

Throws:
RemoteException

See Also: newQueryValue(String) 49

getServiceTypes()

getServiceTypes()

```
public java.lang.String[] getServiceTypes()

Returns a list of Service types that can be used with newServiceValue(String) 49.
}
```

Throws:
RemoteException

getSupportedOperations()

getSupportedOperations()

```
public java.lang.String[] getSupportedOperations()

Gives information which methods will not throw UnsupportedOperationException.
Postcondition:
• Every returned string must be one mentioned in OrderManagerOption

Throws:
RemoteException

See Also: OrderManagerOption 78

newOrderValue(String)

newOrderValue(String) 67

```
public OrderValue newOrderValue(java.lang.String typeName)
throws javax.oss.IllegalArgumentException

Returns a new order value for usage in the client. This method does not create a “real” order.

All attributes of an order that are needed to start the order are populated. The values of these populated attributes are implementation-dependent. An implementation may provide useful default values.

The attributes state and key are not populated.

Note that all attributes may be populated. But it is also possible that some attributes are not populated. These attributes represent the “output” attributes of the order process, for example details about the network elements used during the provisioning.

Postcondition:
• result.isPopulated(KEY) == false
• result.isPopulated(STATE) == false

Parameters:
typeName - The type of order and OrderValue that is to be created.
newQueryValue(String)

public QueryValue newQueryValue(java.lang.String queryName)
  throws javax.oss.IllegalArgumentException

Returns a value object that can be used to query for orders. By using the set methods of the returned value object, the parameter values for query can be given and afterwards queryOrders can be called.

Parameters:
  queryName - identifies the type of query.

Throws:
  IllegalArgumentUnexpected107 - if violated: type must be one of the strings returned by getOrderTypes()47
  RemoteException

newServiceValue(String)

public ServiceValue newServiceValue(java.lang.String serviceType)
  throws javax.oss.IllegalArgumentException

Returns a new service value object. In the returned object all attributes are populated and set to valid values such that the object could be passed without modification to createOrder() for an ACTIA VTE order.

Thus, the following code is valid:

    ServiceValue service = newServiceValue(stype);
    OrderValue order = newOrderValue(OrderType.ACTIVATE);
    order.setServiceValue(service);
    createOrder(order);

Postcondition:
  • result.isPopulated(KEY) == false

Parameters:
  serviceType - The type of service value that is to be created.

Throws:
  IllegalArgumentUnexpected107 - if violated: serviceType must be one of the strings returned by getServiceTypes()48
  RemoteException

queryOrders(QueryValue, String[])

public OrderValueIterator queryOrders(QueryValue parameters,
  java.lang.String[] attributeNames)
  throws javax.oss.IllegalArgumentException

Runs a (complex) query and returns the matching orders.
OrderManager

removeOrder(OrderKey)

Parameters:
- `parameters` - must be one of the value objects returned by `newQueryValue`.
- `attributeNames` - indicates which attributes should be populated in the result.

Throws:
- `IllegalArgumentException` - if violated:
  - `attributeNames` is a subset of `result.getAttributeNames()`
  - `attributeNames.size() > 0`
  - parameter has been retrieved by calling `newQueryValue()`
- `RemoteException`

removeOrder(OrderKey)

```java
public void removeOrder(OrderKey key)
    throws javax.oss.order.IllegalStateException, javax.oss.IllegalArgumentException
```

Terminates the lifetime of an order.

The order is removed from the implementation. If this method is not supported, it can be assumed that the implementation takes care of removing the orders by itself (e.g. after a certain time period after the order CLOSED). See also `OrderManagerOption`.

If the order is removed a JMS message `OrderRemoveEvent` is sent. This means:
- If the implementation does not support REMOVE_ORDER and thus removes an order by itself, a message is sent.
- If the implementation does support REMOVE_ORDER and the client calls `removeOrder`, a message is sent.

Message Postcondition: The following is true on the JMS message `msg` published to the topic TODO:
- `msg.getStringProperty(OSSJ_EVENT_TYPE_PROP_NAME).equals(OrderRemoveEvent.class.getName())`
- `msg.getObject() instanceof OrderRemoveEvent`
- `msg.getObject().getOrderValue()` returns the order value

Throws:
- `IllegalStateException` - if violated:
  - `getOrder(pk).getState().startsWith(CLOSED)`
- `IllegalArgumentException` - if violated: The `key` has been returned by `createOrder(OrderValue)` of this `OrderManager` before.
- `RemoteException`, `UnsupportedOperationException`, `RemoveException`

removeOrders(OrderKey[])

```java
public void removeOrders(OrderKey[] keys)
```

Removes multiple order objects. The semantics of this method is equivalent to the semantics of this code:
for (int i=0; i<keys.size(); i++) {
    removeOrder(keys[i]);
}

Throws:
    RemoteException, UnsupportedOperationException, RemoveException, IllegalStateException, IllegalArgumentException

resumeOrder(OrderKey)
public void resumeOrder(OrderKey pk)
    throws javax.oss.order.IllegalArgumentException, javax.oss.IllegalStateException, javax.oss.order.IllegalStateException, javax.oss.IllegalArgumentException

Puts the order from state SUSPENDED back in state RUNNING.

Postcondition:
    • getOrder(pk).getState().startsWith(RUNNING)

Message Postcondition: The following is true on the JMS message msg published:
    • msg.getStringProperty(OSSJ_EVENT_TYPE_PROP_NAME).equals(OrderStateChangeEvent.class.getName())
    • msg.getObject() instanceof OrderStateChangeEvent
    • msg.getObject().getKey().equals(pk)
    • msg.getObject().getCurrentState().startsWith(RUNNING)

Parameters:
    pk - Primary key for an order.

Throws:
    IllegalStateException - if violated: getOrder(pk).getState().startsWith(SUSPENDED)
    IllegalArgumentException - if violated: The key has been returned by createOrder(OrderValue) of this OrderManager before.
    UnsupportedOperationException - if violated: OrderManagerOption.RESUME_ORDER returned by getSupportedOperations()
    RemoteException

setOrder(OrderValue)
public void setOrder(OrderValue order)
    throws javax.oss.order.IllegalArgumentException, javax.oss.order.IllegalStateException, javax.oss.SetException

Changes the attribute values of an order. Only the attributes that are populated are changed. The attribute key identifies the order to change. The attribute state is ignored, even if populated.

Postcondition:
    • getOrder(pk).equals(order)

Message Postcondition: The following is true on the JMS message msg published to the topic TODO:
    • msg.getStringProperty(OSSJ_EVENT_TYPE_PROP_NAME).equals(OrderAttributeValueChangeEvent.class.getName())
**OrderManager**

```java
setOrders(OrderKey[], OrderValue)
```

- `msg.getObject() instanceof OrderAttributeValueChangeEvent`
- `msg.getObject().getOrderValue() contains all the attributes that have been changed.`

**Parameters:**

- **order** - OrderValue.

**Throws:**

- `IllegalArgumentException` - if violated: `order.isPopulated(KEY) && key has been returned by this OrderManager before.`
- `IllegalStateException` - if the order cannot be changed anymore, because the order has proceeded too much. Will be thrown at least in state CLOSED, but also in states RUNNING and SUSPENDED, depending on the concrete order.
- `SetException` - if the order has been changed by another client, since this OrderValue object has been retrieved from the OrderManager.
- `RemoteException`

```java
public void setOrders(OrderKey[] keys, OrderValue value)
```

Changes the attributes of multiple orders to the same values. The semantics of this method is equivalent to the semantics of this code:

```java
for (int i=0; i<keys.size(); i++) {
    value.setOrderKey(keys[i]);
    setOrder(value);
}
```

**Throws:**

- `RemoteException`, `SetException`, `IllegalStateException`, `IllegalArgumentException`

```java
public void setOrders(OrderValue[] values)
```

Changes the attributes of multiple orders. The semantics of this method is equivalent to the semantics of this code:

```java
for (int i=0; i<values.size(); i++) {
    setOrder(values[i]);
}
```

**Throws:**

- `RemoteException`, `SetException`, `IllegalStateException`, `IllegalArgumentException`

```java
startOrder(OrderKey)
```

```java
public void startOrder(OrderKey key)
```

Puts the order in state RUNNING.
suspendOrder(OrderKey)

public void suspendOrder(OrderKey pk)
   throws javax.oss.order.IllegalArgumentException, javax.oss.UnsupportedOperationException
   
Put the order from state RUNNING in the state SUSPENDED.

Postcondition:
   • getOrder(pk).getState().startsWith(SUSPENDED)
   • OrderManagerOption.SUSPENDED_STATE in getSupportedOperations()

Message Postcondition: The following is true on the JMS message msg published to the topic TODO:
   • msg.getStringProperty(OSSJ_EVENT_TYPE_PROP_NAME).equals(OrderStateChangeEvent.class.getName())
   • msg.getObject() instanceof OrderStateChangeEvent
   • msg.getObject().getKey().equals(pk)
   • msg.getObject().getCurrentState().startsWith(SUSPENDED)

Parameters:
   pk - Primary key for an order.

Throws:
   IllegalStateException - if violated:
      getOrder(pk).getState().startsWith(NOT_STARTED)
   IllegalArgumentException - if violated: The key has been returned by
      createOrder(OrderValue) of this OrderManager before.
   UnsupportedOperationException - if violated:
      OrderManagerOption.SUSPEND_ORDER returned by
      getSupportedOperations()
   RemoteException
OrderManagerHome

Syntax
public interface OrderManagerHome extends javax.ejb.EJBHome

All Superinterfaces: javax.ejb.EJBHome, java.rmi.Remote

Description
EJB home interface for an OrderManager.
To locate an OrderManager, consult chapter 3 “Using the API”.

<table>
<thead>
<tr>
<th>Member Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
</tr>
<tr>
<td>OrderManager create()</td>
</tr>
</tbody>
</table>

Methods

create()

    public OrderManager create()

    Throws: RemoteException, EJBException, CreateException

OrderValueIterator

Syntax
public interface OrderValueIterator extends javax.ejb.EJBObject

All Superinterfaces: javax.ejb.EJBObject, java.rmi.Remote

Description
EJB remote interface, returned by OrderManager methods to allow iteration over a lot of orders.
A reference to this object is for example returned by getOrders(OrderKey[], String[]) and queryOrders(QueryValue, String[]). By using it, a client can retrieve the results of the query step by step instead of retrieving all at once. Example:
orderValueIterator = om.getOrders(new OrderValue[] {orderTemplate}, attIds);
orderValues = orderValueIterator.getNext(10);
while(orderValues.length) {
    for (int i = 0; i < orderValues.length; i++) {
        System.out.println(orderValues[i].toString());
    }
    orderValues = orderValueIterator.getNext(10);
}
orderValueIterator.remove();

### Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
<th>OrderValue[] getNext(int)</th>
</tr>
</thead>
</table>

### Methods

**getNext(int)**

```java
public OrderValue[] getNext(int howMany)
    throws javax.oss.IllegalArgumentException
```

Return the next n order values from the result of the query.

An empty result indicates that there are no further values.

An implementation may return less values than requested, for example `getNext(100000)` may only return 256 values. Returning less values than requested does not mean that there are no more values.

If the last call returned an empty array (indicating that there are no further values), all future calls will return an empty array as well.

**Issue:** Implementation may auto-remove instances?

**Postcondition:**

- `result != null || result.length <= howMany`

**Parameters:**

- `howMany` - maximum number of values returned.

**Throws:**

- `IllegalArgumentException` - if violated: `howMany() > 0`
- `RemoteException`

### XMLOrderManager

**Syntax**

```java
public interface XMLOrderManager extends javax.ejb.EJBObject
```

**All Superinterfaces:** `javax.ejb.EJBObject`, `java.rmi.Remote`
XMLOrderManager

abortOrder(String)

Description
The XMLOrderManager is a synchronous XML/Java based interface. The objective of this interface is to allow the clients to interact with the Service Activation (SA) system, using a Session Bean interface that utilizes XML instance documents as arguments/return values and exceptions. This Session Bean interface (XMLOrderManager) contains exactly the same method names as the OrderManager interface, except for the fact that the method signatures take a string as an argument and a return value (i.e the string contents would be the XML documents).

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>String abortOrder(String)</td>
</tr>
<tr>
<td>String createOrderByValue(String)</td>
</tr>
<tr>
<td>String createOrdersByValues(String)</td>
</tr>
<tr>
<td>String getOrderByKey(String)</td>
</tr>
<tr>
<td>String getOrdersByKeys(String)</td>
</tr>
<tr>
<td>XMLOrderValueIterator getOrdersByTemplates(String)</td>
</tr>
<tr>
<td>String getOrderTypes()</td>
</tr>
<tr>
<td>String getQueryTypes()</td>
</tr>
<tr>
<td>String getServiceTypes()</td>
</tr>
<tr>
<td>String getSupportedOperations()</td>
</tr>
<tr>
<td>String newOrderByValue(String)</td>
</tr>
<tr>
<td>String newQueryValue(String)</td>
</tr>
<tr>
<td>String newServiceValue(String)</td>
</tr>
<tr>
<td>XMLOrderValueIterator queryOrders(String)</td>
</tr>
<tr>
<td>String removeOrderByKey(String)</td>
</tr>
<tr>
<td>void removeOrdersByKeys(String)</td>
</tr>
<tr>
<td>String resumeOrder(String)</td>
</tr>
<tr>
<td>String setOrderByValue(String)</td>
</tr>
<tr>
<td>void setOrdersByKeys(String)</td>
</tr>
<tr>
<td>void setOrdersByValues(String)</td>
</tr>
<tr>
<td>String startOrder(String)</td>
</tr>
<tr>
<td>String suspendOrder(String)</td>
</tr>
</tbody>
</table>

Methods

abortOrder(String)

public java.lang.String abortOrder(java.lang.String request)
    throws javax.oss.IllegalArgumentException, java.rmi.RemoteException, javax.oss.order.IllegalStateException

The following request aborts the order and puts the state to ABORTED.

Parameters:
request - An XML document AbortOrderRequest

Returns: An XML Document AbortOrderResponse
XMLOrderManager

createOrderByValue(String)

Throws:

IllegalArgumentException - The IllegalArgumentException exception is returned by the interface to report that the request could not be completed because one of the arguments passed in is invalid.

java.rmi.RemoteException - The RemoteException is returned when an error occurs during any remote object operation.

IllegalStateException - This exception indicates that the order is in not in an appropriate state for this operation.

createOrderByValue(String)

public java.lang.String createOrderByValue(java.lang.String request)

Creates a new order object in the system and returns the key for the new object. A single OrderValue is the only value passed into the request. The state values in OrderValue are ignored, the state is initialized to STARTED by the system.

Parameters:
  request - An XML document CreateOrderByValueRequest.


Throws:

java.rmi.RemoteException - is returned when an error occurs during any remote object operation.

IllegalArgumentException - is thrown to report that the request could not be completed because one of the arguments passed in is invalid.

javax.ejb.CreationException - The exception is used as a standard application-level exception to report a failure to create a managed entity or a collection of managed entities. This exception is thrown when a particular managed entity or group of managed entities cannot be created.

CreateException

createOrdersByValues(String)

public java.lang.String createOrdersByValues(java.lang.String request)

CreateOrdersByValues is the same as createOrderByValue except that it allows the client to create a number of orders in the system. A list of OrderValues is passed into the request. It returns a list of keys, each representing an created order object. The state values for each OrderValue are ignored, the state is initialized to STARTED by the system.

Parameters:
  request - An XML document CreateOrdersByValuesRequest

Returns: An XML Document CreateOrdersByValuesResponse

Throws:

java.rmi.RemoteException - is returned when an error occurs during any remote object operation.
**XMLOrderManager**

**getOrderByKey(String)**

- **IllegalArgumentException** 107: Is thrown to report that the request could not be completed because one of the arguments passed in is invalid.
- **javax.ejb.CreationException**: The exception is used as a standard application-level exception to report a failure in creating a managed entity or a collection of managed entities. This exception is thrown when a particular managed entity or group of managed entities cannot be created.

**getOrderByKey(String)**

```java
public java.lang.String getOrderByKey(java.lang.String request)
    throws java.rmi.RemoteException, javax.oss.IllegalArgumentException
```

Returns values for the order identified by the (unique) key. The value object returned will always have a type that is derived from OrderValue (polymorphism). The attributeName list indicates which attributes are to be returned. If the attributeName is null then all possible attributes are returned.

**Parameters:**
- request - An XML document GetOrderByKeyRequest

**Returns:** An XML Document GetOrderByKeyResponse

**Throws:**
- **java.rmi.RemoteException**: The RemoteException is returned when an error occurs during any remote object operation.
- **IllegalArgumentException**: The IllegalArgumentException exception is returned by the interface to report that the request could not be completed because one of the arguments passed in is invalid.

**getOrdersByKeys(String)**

```java
public java.lang.String getOrdersByKeys(java.lang.String request)
    throws java.rmi.RemoteException, javax.oss.IllegalArgumentException
```

Returns values for a list of orders identified by an XML document containing (unique) keys. Is semantically equivalent to calling getOrderByKey(String) for each element of the pk's individually. Should be used to reduce the number of round trips between client and implementation.

For detailed description see getOrderByKey(String).

**Parameters:**
- request - An XML document GetOrdersByKeysRequest

**Returns:** An XML Document GetOrdersByKeysResponse

**Throws:**
- **java.rmi.RemoteException**: The RemoteException is returned when an error occurs during any remote object operation.
- **IllegalArgumentException**: The IllegalArgumentException exception is returned to indicate that the request could not be completed because one of the arguments passed in is invalid.
public XMLOrderValueIterator getOrdersByTemplates(String request)
throws javax.oss.IllegalArgumentException, javax.oss.UnsupportedOperationException

Similiar to the getOrdersByKeys method except that it takes in a template request and returns a XMLOrderValueIterator

Parameters:
request - An XML document GetOrdersByTemplatesRequest

Returns: An XMLOrderValueIterator that contains GetOrdersByTemplatesResponse XML documents

Throws:
IllegalArgumentException - The IllegalArgumentException exception is returned to indicate that the request could not be completed because one of the arguments passed in, is invalid.

UnsupportedOperationException - The UnsupportedOperationException exception is returned to report that the invoked request could not be answered because it is not implemented.

RemoteException

public java.lang.String getOrderTypes()
throws java.rmi.RemoteException

This method returns an Xml document that contains a string list of available OrderTypes, such as Activate, DeActivate etc. Each one of the strings in the list can be passed as an argument to newOrderValue(factory) to create a corresponding OrderValue instance.


Throws: java.rmi.RemoteException - is returned when an error occurs during any remote object operation

See Also: newOrderValue

getOrderTypes()
**XMLOrderManager**

**get ServiceTypes()**

```java
public java.lang.String getServiceTypes()
    throws java.rmi.RemoteException
```

This method returns an XML document that contains a list of supported service types. The user would determine from the list which service types are supported, such as DSL, GGSN etc..

**Returns:** An XML document GetServiceTypesResponse.

**Throws:**
- `java.rmi.RemoteException` - is returned when an error occurs during any remote object operation

**See Also:** newServiceValue

**get SupportedOperations()**

```java
public java.lang.String getSupportedOperations()
    throws java.rmi.RemoteException
```

This method returns an XML Document that contains a list of implemented optional operations. The user would determine from the list which methods are supported.


**Throws:**
- `java.rmi.RemoteException` - is returned when an error occurs during any remote object operation.

**new OrderValue(String)**

```java
public java.lang.String newOrderValue(java.lang.String request)
    throws javax.oss.IllegalArgumentException, javax.rmi.RemoteException
```

Creates a new OrderValue based on OrderType i.e. this is a factory for the creation of OrderValues. When this method is executed, no actual order, but only an OrderValue is created. The OrderValue would then be passed to createOrder, and that is when an order is created.

**Parameters:**
- `request` - An XML document NewOrderValueRequest contains the type of OrderValue that should be created.

**Returns:** An XML document NewOrderValueResponse which contains the OrderValue created based on order type.

**Throws:**
- `javax.oss.IllegalArgumentException` - is thrown to report that the request could not be completed because one of the arguments passed in is invalid.
- `javax.rmi.RemoteException` - is returned when an error occurs during any remote object operation.
- `RemoteException`
### newQueryValue(String)

```java
public java.lang.String newQueryValue(java.lang.String request)
    throws javax.oss.IllegalArgumentException, javax.rmi.RemoteException
```

This method will return a new QueryValue for usage in a client. The type of QueryValue returned is based on the queryType element populated in this request. It supplies the client with an empty QueryValue (i.e not populated with data).

**Parameters:**
- `request` - An XML document NewQueryValueRequest contains the type of QueryValue that should be created.

**Returns:** An XML document NewQueryValueResponse which contains the QueryValue created based on query type.

**Throws:**
- `IllegalArgumentException` - is thrown to report that the request could not be completed because one of the arguments passed in is invalid.
- `javax.rmi.RemoteException` - is returned when an error occurs during any remote object operation.

### newServiceValue(String)

```java
public java.lang.String newServiceValue(java.lang.String request)
    throws javax.oss.IllegalArgumentException, javax.rmi.RemoteException
```

This method will return a newServiceValue for usage in a client. The type of ServiceValue returned is based on the serviceType element populated in this request. It supplies the client with an empty ServiceValue (i.e not populated with data).

**Parameters:**
- `request` - An XML document NewServiceValueRequest

**Returns:** An XML document NewServiceValueResponse

**Throws:**
- `IllegalArgumentException` - is thrown to report that the request could not be completed because one of the arguments passed in is invalid.
- `javax.rmi.RemoteException` - is returned when an error occurs during any remote object operation.

### queryOrders(String)

```java
public XMLOrderValueIterator queryOrders(java.lang.String request)
    throws java.rmi.RemoteException, javax.oss.IllegalArgumentException
```

The following method takes in an XML document that contains a (complex) query and returns the matching orders. The HowMany element is used to restrict the returning result set. If the result set exceeds the “How-Many” value then consequent response messages are returned.
**XMLOrderManager**

**removeOrderByKey(String)**

**Parameters:**
request - An XML document QueryOrdersRequest.

**Returns:** An XMLOrderValueIterator that contains QueryOrdersResponse documents.

**Throws:**
java.rmi.RemoteException - The RemoteException is returned when an error occurs during any remote object operation.

IllegalArgumentException - The IllegalArgumentException exception is returned to indicate that the request could not be completed because one of the arguments passed in, is invalid.

---

**removeOrderByKey(String)**

```java
public java.lang.String removeOrderByKey(java.lang.String request)
```


The following request will terminate the lifetime or an order. This request indicates that the client(s) do not need the referenced order anymore. The implementation can also remove orders automatically, example batch cleanup of completed orders. If the order is removed successfully an OrderRemoveEvent is published. Exceptions are returned if, for example, the order does not exist or if the order is not in an appropriate state (i.e. RUNNING state). If the order is not in an appropriate state then the user would have to abort the order and then call removeOrder.

**Parameters:**
request - An XML document RemoveOrderByKeyRequest

**Returns:** An XML document RemoveOrderByKeyResponse

**Throws:**
IllegalArgumentException - is thrown to report that the request could not be completed because one of the arguments passed in is invalid.

IllegalStateException - The IllegalStateException exception is returned.

javax.ejb.RemoveException - The RemoveException exception is returned at an attempt to remove a collection of one or more managed entities when the XVT interface does not allow the managed entity to be removed. This exception is returned when a a collection of one or more managed entity cannot be removed.

java.rmi.RemoteException - is returned when an error occurs during any remote object operation.

UnsupportedOperationException - The UnsupportedOperationException exception is returned to report that the invoked request could not be answered because it is not implemented.

---

**removeOrdersByKeys(String)**

```java
public void removeOrdersByKeys(java.lang.String request)
```


This is the plural version of the RemoveOrderByKeyRequest. The only difference is that it takes a list of orders to be removed (indicated by a list of OrderKeys).
Parameters:
request - An XML document RemoveOrdersByKeysRequest

Returns: An XML Document RemoveOrdersByKeysResponse

Throws:
IllegalStateException - The IllegalStateException exception is returned.
IllegalArgumentException - The IllegalArgumentException exception is returned by the interface to report that the request could not be completed because one of the arguments passed in is invalid.
javax.ejb.RemoveException - The RemoveException exception is returned at an attempt to remove a collection of one or more managed entities when the XVT interface does not allow the managed entities to be removed. This exception is returned when a a collection of one or more managed entity cannot be removed
java.rmi.RemoteException - is returned when an error occurs during any remote object operation.

UnsupportedOperationException - The UnsupportedOperationException exception is returned to report that the invoked request could not be answered because it is not implemented.

resumeOrder(String)

public java.lang.String resumeOrder(java.lang.String request)

This request puts the order from a SUSPENDED state back into a RUNNING state

Parameters:
request - An XML document ResumeOrderRequest

Returns: An XML document ResumeOrderResponse

Throws:
java.rmi.RemoteException - The RemoteException is returned when an error occurs during any remote object operation.

IllegalArgumentException - The IllegalArgumentException exception is returned by the interface to report that the request could not be completed because one of the arguments passed in is invalid.

UnsupportedOperationException - The UnsupportedOperationException exception is returned to report that the invoked request could not be answered because it is not implemented.

IllegalStateException - This exception indicates that the order is not in an appropriate state for this operation.

setOrderByValue(String)

public java.lang.String setOrderByValue(java.lang.String request)
throws javax.oss.SetException, javax.oss.order.IllegalStateException, javax.oss.IllegalArgumentException, java.rmi.RemoteException

This request is used to change the attributes of an order. Only the attributes which are populated are changed. The state and key attributes are ignored.
**XMLOrderManager**

**setOrdersByKeys(String)**

**Parameters:**
- request - An XML document SetOrderByValueRequest

**Returns:** An XML document SetOrderByValueResponse

**Throws:**
- SetException\textsubscript{108} - The SetException exception is returned at an attempt to update a collection of one or more managed entities when the XVT interface does not allow the managed entities to be updated. This exception is returned when a collection of one or more managed entities cannot be updated.
- IllegalStateException\textsubscript{84} - This exception indicates that the order is in a state that does not allow updates such as in a RUNNING state.
- IllegalArgumentException\textsubscript{107} - The IllegalArgumentException exception is returned to indicate that the request could not be answered because one of the arguments passed in is invalid.
- java.rmi.RemoteException - The RemoteException is returned when an error occurs during any remote object operation.

**setOrdersByKeys(String)**

```java
public void setOrdersByKeys(java.lang.String request)
    throws javax.oss.SetException, java.rmi.RemoteException, javax.oss.order.IllegalStateException, javax.oss.IllegalArgumentException
```

This is the plural version of the setOrder request. The only difference is that it takes a list of OrderKeys.

**Parameters:**
- request - An XML document SetOrdersByKeysRequest

**Returns:** An XML document SetOrdersByKeysResponse

**Throws:**
- SetException\textsubscript{108} - The SetException exception is returned at an attempt to update a collection of one or more managed entities when the XVT interface does not allow the managed entities to be updated. This exception is returned when a collection of one or more managed entities cannot be updated.
- java.rmi.RemoteException - The RemoteException is returned when an error occurs during any remote object operation.
- IllegalStateException\textsubscript{84} - This exception indicates that the order is in a state that does not allow updates such as in a RUNNING state.
- IllegalArgumentException\textsubscript{107} - The IllegalArgumentException exception is returned to indicate that the request could not be completed because one of the arguments passed in, is invalid.

**setOrdersByValues(String)**

```java
public void setOrdersByValues(java.lang.String request)
    throws javax.oss.SetException, java.rmi.RemoteException, javax.oss.order.IllegalStateException, javax.oss(IllegalArgumentException
```

This is the plural version of the setOrder request. The only difference is that it takes a list of OrderValues.

**Parameters:**
- request - An XML document SetOrdersByValuesRequest
>Returns: An XML document SetOrdersByValuesResponse

.Throws:
  SetException108 - The SetException exception is returned at an attempt to update a collection of one or more managed entities when the XVT interface does not allow the managed entities to be updated. This exception is returned when a collection of one or more managed entities cannot be updated.
  java.rmi.RemoteException - The RemoteException is returned when an error occurs during any remote object operation.
  IllegalStateException84 - This exception indicates that the Order is in a state that does not allow updates such as in a RUNNING state.
  IllegalArgument Exception107 - The IllegalArgument Exception exception is returned to indicate that the request could not be completed because one of the arguments passed in is invalid.

suspendOrder(String)

public java.lang.String suspendOrder(java.lang.String request)
   throws java.rmi.RemoteException, javax.oss.UnsupportedOperationException, javax.oss.order.IllegalStateException

This request puts the order from a RUNNING state into a SUSPENDED state.

.Parameters:
  request - An XML document SuspendOrderRequest

>Returns: An XML document SuspendOrderResponse

.Throws:
  java.rmi.RemoteException - The RemoteException is returned when an error occurs during any remote object operation.
  IllegalStateException84 - This exception indicates that the order is in not in an appropriate state for this operation.

XMLOrderManagerHome

create()

IllegalArgumentException - The IllegalArgumentException exception is returned by the interface to report that the request could not be completed because one of the arguments passed in is invalid.

UnsupportedOperationException - The UnsupportedOperationException exception is returned to report that the invoked request could not be answered because it is not implemented.

IllegalStateException - This exception indicates that the order is not in an appropriate state for this operation.

XMLOrderManagerHome

Syntax
public interface XMLOrderManagerHome extends javax.ejb.EJBHome

All Superinterfaces: javax.ejb.EJBHome, java.rmi.Remote

Description
Home EJB interface for an XMLOrderManager.
To locate an XMLOrderManager, consult chapter 3 “Using the API”.

Methods

**create()**

```
public XMLOrderManager create()
```

**Throws:**

```
RemoteException, EJBException, CreateException
```

XMLOrderValueIterator

Syntax
public interface XMLOrderValueIterator extends javax.ejb.EJBObject

All Superinterfaces: javax.ejb.EJBObject, java.rmi.Remote
Description
The XMLOrderValueIterator is similar to the OrderValueIterator, except that the returned value on a getNext will return a string. This string will be populated with an XML instance document such as a QueryOrders-Response, or a GetOrdersByTemplateResponse.

<table>
<thead>
<tr>
<th>Member Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
</tr>
<tr>
<td>String getNext(int)</td>
</tr>
</tbody>
</table>

Methods

getNext(int)

```java
public java.lang.String getNext(int howMany)
    throws javax.oss.IllegalArgumentException, javax.rmi.RemoteException
```

This method returns an XML Instance Document that contains one or more OrderValues. The number of OrderValues returned is determined by the HowMany argument.

Parameters:
howMany - indicates how many OrderValue instances to return.

Returns: An XML Document QueryOrdersResponse or GetOrdersByTemplateResponse

Throws:
IllegalArgumentException - is thrown to report that the request could not be completed because the argument passed in is invalid.
javax.rmi.RemoteException - is returned when an error occurs during any remote object operation
RemoteException

OrderValue

Syntax
```java
public interface OrderValue extends ManagedEntityValue
```

All Superinterfaces: java.lang.Cloneable, ManagedEntityValue, java.io.Serializable

Description
Value type interface for accessing order values.

Orders are stored inside the OrderManager. A client can retrieve and change orders indirectly by using the methods of an OrderManager. To change the properties of an order, do the following:
OrderValue

getNext(int)

```java
OrderValue value = om.getOrder(pk);
value.setClientId(...);
value.setDescription(...);
om.setOrder(pk);
```

Orders are identified by using keys. See `getOrderKey()` for more information.

OrderValues inherit from ManagedEntityValue and thus all methods to deal with population and generic access of the order value are applicable to the order value, too.

See Also: OrderManager

---

### Member Summary

#### Fields

- `String ACTUAL_DELIVERY_DATE`
- `String CLIENT_ID`
- `String DESCRIPTION`
- `String JEOPARDY_DATE`
- `String ORDER_DATE`
- `String PRIORITY`
- `String PURCHASE_ORDER`
- `String REQUESTED_DELIVERY_DATE`
- `String SERVICES`
- `String STATE`

#### Methods

- `Date getActualDeliveryDate()`  
- `Class getAttributeType(String)`  
- `String getClientId()`  
- `String getDescription()`  
- `Date getJeopardyDate()`  
- `Date getOrderDate()`  
- `OrderKey getOrderKey()`  
- `int getPriority()`  
- `String getPurchaseOrder()`  
- `Date getRequestedDeliveryDate()`  
- `ServiceValue[] getServices()`  
- `String getState()`  
- `void setActualDeliveryDate(Date)`  
- `void setClientId(String)`  
- `void setDescription(String)`  
- `void setJeopardyDate(Date)`  
- `void setOrderDate(Date)`  
- `void setOrderKey(OrderKey)`  
- `void setPriority(int)`  
- `void setPurchaseOrder(String)`  
- `void setRequestedDeliveryDate(Date)`  
- `void setService(ServiceValue)`  
- `void setServices(ServiceValue[])`  
- `void setState(String)`
Fields

**ACTUAL_DELIVERY_DATE**

```java
public static final java.lang.String ACTUAL_DELIVERY_DATE
```

**CLIENT_ID**

```java
public static final java.lang.String CLIENT_ID
```

**DESCRIPTION**

```java
public static final java.lang.String DESCRIPTION
```

**JEOPARDY_DATE**

```java
public static final java.lang.String JEOPARDY_DATE
```

**ORDER_DATE**

```java
public static final java.lang.String ORDER_DATE
```

**PRIORITY**

```java
public static final java.lang.String PRIORITY
```

**PURCHASE_ORDER**

```java
public static final java.lang.String PURCHASE_ORDER
```

**REQUESTED_DELIVERY_DATE**

```java
public static final java.lang.String REQUESTED_DELIVERY_DATE
```

**SERVICES**

```java
public static final java.lang.String SERVICES
```

**STATE**

```java
public static final java.lang.String STATE
```
Methods

getActualDeliveryDate()

public java.util.Date getActualDeliveryDate()

See Also: setActualDeliveryDate(Date) 71

getAttributeType(String)

public java.lang.Class getAttributeType(java.lang.String attributeName)
throws IllegalArgumentException

Return the attribute type for an attribute. Returns the declared argument type of the set method. For example, for setMaxDataRate(int i) Integer.TYPE is returned.

Note: calling getAttributeValue might return null and thus its not possible to infer the type.

Throws:
   IllegalArgumentException - if violated: attributeName must be one of of the strings returned from getAttributeNames()

getClientId()

public java.lang.String getClientId()

See Also: setClientId(String) 72

description()

public java.lang.String getDescription()

Returns detailed description of the order object.

See Also: setDescription(String) 72

getJeopardyDate()

public java.util.Date getJeopardyDate()

Returns: java.util.Date jeopardy date value

See Also: setJeopardyDate(Date) 72

getOrderDate()

public java.util.Date getOrderDate()

See Also: setOrderDate(Date) 72

getOrderKey()

public OrderKey getOrderKey()
Gets the unique identification for this order. Each order value has a key that can be inquired by calling get-OrderKey(). The key of an order is unique amongst OSS/J APIs. A key is assigned to the order object by its OrderManager when the order is created.

See Also: createOrder(OrderValue) 45, setOrderKey(OrderKey) 72

getPriority()

```java
public int getPriority()
   throws UnsupportedOperationException
```

Returns the priority of the order.

Throws:
   UnsupportedOperationException - if priority handling is not supported.

See Also: setPriority(int) 73

getPurchaseOrder()

```java
public java.lang.String getPurchaseOrder()
```

See Also: setPurchaseOrder(String) 73

getRequestedDeliveryDate()

```java
public java.util.Date getRequestedDeliveryDate()
```

See Also: setRequestedDeliveryDate(Date) 73

getServices()

```java
public ServiceValue[] getServices()
```

Returns the service objects that this order wants to change when the order has been completed.

See Also: setService(ServiceValue) 73

getState()

```java
public java.lang.String getState()
```

Returns the state of the order.

See Also: setState(String) 74

setActualDeliveryDate(Date)

```java
public void setActualDeliveryDate(java.util.Date date)
```

Date, when the order reached the state CLOSED.

This is an output attribute, it is set by the implementation. The client cannot change it by calling the OrderManager.

See Also: getActualDeliveryDate() 70
setClientId(String)

public void setClientId(java.lang.String name)

Provides an identification for the client.

It is recommended to use the following strings:

- Use the domain of the client, similar to the value of getOrderKey().getDomain() used by the implementation.
- Use the java class name of the class that drives the interface. This avoids name conflict with other clients, if the class name follows standard java coding principle (i.e. com.nokia.oss.xyz). If there can be multiple client instances, the client has to ensure uniqueness by appending additional ID’s.

See Also: getClientId() 70

setDescription(String)

public void setDescription(java.lang.String text)

Sets detailed description of the order object. The description has no effect on the execution of the order.

See Also: getDescription() 70

setJeopardyDate(Date)

public void setJeopardyDate(java.util.Date date)

Client-definable date for jeopardy management.

Sets the date after which the implementation will put the order into the jeopardy state JEOPARDY, if the order is still in state NOT_STARTED.

Parameters:

java.util.Date - value of jeopardy date

setOrderDate(Date)

public void setOrderDate(java.util.Date date)

Date, when the order was received from the customer.

This is set by the client to the date, when the order has been received by the customer care agent.

See Also: getOrderDate() 70

setOrderKey(OrderKey)

public void setOrderKey(OrderKey key)

Sets the unique identification for this order.

See Also: setOrderKey(OrderKey) 72
**setPriority(int)**

```java
public void setPriority(int prio)
    throws IllegalArgumentException, UnsupportedOperationException
```

Sets the priority of the order.

**Relative priority of the order in the set of all order objects for one order manager.**

**Throws:**
- `IllegalArgumentException` - if violated: `OrderPriority.LOW <= key <= OrderPriority.EXPEDITE`
- `UnsupportedOperationException` - if priority handling is not supported.

**See Also:** `OrderPriority.getPriority()`

**setPurchaseOrder(String)**

```java
public void setPurchaseOrder(java.lang.String str)
```

Order identification as seen by the subscriber.

**See Also:** `getPurchaseOrder()`

**setRequestedDeliveryDate(Date)**

```java
public void setRequestedDeliveryDate(java.util.Date date)
```

Client-definable date of service activation.

**Issue:** Do we require some internal activities in the server? Which?

**See Also:** `getRequestedDeliveryDate()`

**setService(ServiceValue)**

```java
public void setService(ServiceValue service)
```

Changes the value for the service objects.

**Equivalent to the following code:**

```java
setServices( new ServiceValue[] { service } );
```

**See Also:** `setServices(ServiceValue[])`

**setServices(ServiceValue[])**

```java
public void setServices(ServiceValue[] service)
    throws IllegalArgumentException
```

Changes the value for the service objects.

This is the most important aspect of an order, since most orders aim at changing the service data.

A null value is not allowed for service.
setState(String)

public void setState(java.lang.String state)

Sets the state for this order.

Note that some methods in OrderManager ignore this attribute, e.g. setOrders. Instead, to change the state
of an order, explicit methods in the OrderManager have to be called, e.g. startOrder().

This method is used in case you want to query for a certain set of orders by using OrderManager.getOrders.

See Also: OrderState, getState()

OrderKey

Syntax

public interface OrderKey extends ManagedEntityKey

All Superinterfaces: java.lang.Cloneable, ManagedEntityKey, java.io.Serializable

Description

Value type interface: representing an OSS wide unique key to an order.

OrderMessageProperty

Syntax

public interface OrderMessageProperty

Description

String constants that define the predefined JMS properties.

Member Summary

<table>
<thead>
<tr>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>String CLIENT_ID</td>
</tr>
<tr>
<td>String ORDER_PRIMARY_KEY</td>
</tr>
<tr>
<td>String ORDER_TYPE</td>
</tr>
<tr>
<td>String OSSJ_DOMAIN_PROP_NAME</td>
</tr>
<tr>
<td>String OSSJ_EVENT_TYPE_PROP_NAME</td>
</tr>
</tbody>
</table>
OrderCreateEvent

Fields

CLIENT_ID

```java
public static final java.lang.String CLIENT_ID
```

The value of this property is the same as the value returned by orderValue.getClientId().

ORDER_PRIMARY_KEY

```java
public static final java.lang.String ORDER_PRIMARY_KEY
```

The value of this property is the same as the value returned by orderValue.getKey().getPrimary-Key().toString().

ORDER_TYPE

```java
public static final java.lang.String ORDER_TYPE
```

The value of this property is the same as the value returned by orderValue.getKey().getType().

OSSJ_DOMAIN_PROP_NAME

```java
public static final java.lang.String OSSJ_DOMAIN_PROP_NAME
```

The value of this property is the same as the value returned by orderValue.getKey().getDomain().

OSSJ_EVENT_TYPE_PROP_NAME

```java
public static final java.lang.String OSSJ_EVENT_TYPE_PROP_NAME
```

The value of this property is one of:

- javax.oss.order.OrderCreateEvent
- javax.oss.order.OrderAttributeValueChangedEvent
- javax.oss.order.OrderStateChangedEvent
- javax.oss.order.OrderRemoveEvent

OrderCreateEvent

Syntax

```java
public interface OrderCreateEvent extends Event
```

All Superinterfaces: Event, java.io.Serializable

Description

Event that is sent if a new order has been created. TODO: What Attributes are populated?

See Also: createOrder(OrderValue), createOrders(OrderValue[])
OrderStateChangeEvent
gOrderValue()

<table>
<thead>
<tr>
<th>Member Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td>OrderValue</td>
</tr>
</tbody>
</table>

**Methods**

gOrderValue()

```
public OrderValue getOrderValue()
```

**OrderStateChangeEvent**

**Syntax**

```
public interface OrderStateChangeEvent extends Event
```

**All Superinterfaces:** Event, java.io.Serializable

**Description**

Event that is sent if an order has changed its state.

A message is sent when:

- while executing the order, the state of the order has been changed by the implementation. If the state change is also a change in one of the super states defined in OrderState, an implementation must publish an event. If the change is only a change in sub states (i.e. there is no change for the states defined in OrderState), an implementation may publish an event.
- one of the methods listed below have been called.

**See Also:** startOrder(OrderKey), suspendOrder(OrderKey), resumeOrder(OrderKey), abortOrder(OrderKey)

<table>
<thead>
<tr>
<th>Member Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>OrderKey</td>
</tr>
<tr>
<td>String</td>
</tr>
</tbody>
</table>
Methods

getCurrentState()

    public java.lang.String getCurrentState()

getOrderKey()

    public OrderKey getOrderKey()

getReason()

    public java.lang.String getReason()  

OrderAttributeValueChangeEvent

Syntax

public interface OrderAttributeValueChangeEvent extends Event

All Superinterfaces:  Event, java.io.Serializable

Description

Event that is sent if any value of an order has changed.  

A message is send when:  

• setOrder has caused changes to attribute values of an order.  

• some values of an order have been changed during the execution of an order by the implementation. In this case, the implementation might publish a message.  

The message is not published if the state of an order has changed. Only the changed attributes are populated in the OrderValue. Th OrderValue contains the new attribute values.  

See Also:  setOrder(OrderValue), setOrders(OrderValue[])  

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderValue getOrderValue()</td>
</tr>
</tbody>
</table>


OrderRemoveEvent

getOrderValue()

Methods

getOrderValue()

public OrderValue getOrderValue()

OrderRemoveEvent

Syntax

public interface OrderRemoveEvent extends Event

All Superinterfaces: Event, java.io.Serializable

Description

Event that is sent if an order has been removed.

See Also: removeOrder(OrderKey), removeOrders(OrderKey[])

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderValue getOrderValue()</td>
</tr>
</tbody>
</table>

OrderManagerOption

Syntax

public interface OrderManagerOption

Description

String constants to indicate which optional parts of the interface are supported by the implementation. If a string METHOD is returned by getSupportedOperations(), a client can call the corresponding method. Otherwise, UnsupportedOperationException is thrown.
Note: It is recommended that an implementation either returns both SUSPEND_ORDER and RESUME_ORDER, or none. The same is valid for GET_PRIORITY/SET_PRIORITY and REMOVE_ORDER/REMOVE_ORDERS.

<table>
<thead>
<tr>
<th>Member Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fields</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>String GET_ORDERS</td>
</tr>
<tr>
<td>String GET_PRIORITY</td>
</tr>
<tr>
<td>String REMOVE_ORDER</td>
</tr>
<tr>
<td>String REMOVE_ORDERS</td>
</tr>
<tr>
<td>String RESUME_ORDER</td>
</tr>
<tr>
<td>String SET_PRIORITY</td>
</tr>
<tr>
<td>String SUSPEND_ORDER</td>
</tr>
</tbody>
</table>

**GET_ORDERS**

```
public static final java.lang.String GET_ORDERS
Indicates that getOrders(OrderKey[], String[]) is implemented.
```

**GET_PRIORITY**

```
public static final java.lang.String GET_PRIORITY
Indicates that getPriority() is implemented.
```

**REMOVE_ORDER**

```
public static final java.lang.String REMOVE_ORDER
Indicates that removeOrder(OrderKey) is implemented.
```

**REMOVE_ORDERS**

```
public static final java.lang.String REMOVE_ORDERS
Indicates that removeOrders(OrderKey[]) is implemented.
```

**RESUME_ORDER**

```
public static final java.lang.String RESUME_ORDER
Indicates that resumeOrder(OrderKey) is implemented.
```
OrderPriority

**SET_PRIORITY**

```java
public static final java.lang.String SET_PRIORITY
```

Indicates that `setPriority(int)` is implemented.

**SUSPEND_ORDER**

```java
public static final java.lang.String SUSPEND_ORDER
```

Indicates that `suspendOrder(OrderKey)` is implemented.

OrderPriority

**Syntax**

```java
public interface OrderPriority
```

**Description**

Constants defining the possible priorities that can be used with `setPriority(int)`.

Orders with priority EXPEDITE are processed first, Orders with priority LOW are processed last.

<table>
<thead>
<tr>
<th>Member Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fields</strong></td>
</tr>
<tr>
<td>int EXPEDITE</td>
</tr>
<tr>
<td>int HIGH</td>
</tr>
<tr>
<td>int LOW</td>
</tr>
<tr>
<td>int MEDIUM</td>
</tr>
<tr>
<td>int NORMAL</td>
</tr>
</tbody>
</table>

**Fields**

**EXPEDITE**

```java
public static final int EXPEDITE
```

**HIGH**

```java
public static final int HIGH
```

**LOW**

```java
public static final int LOW
```
public static final int MEDIUM

public static final int NORMAL

OrderState

Syntax
public interface OrderState extends java.io.Serializable

All Superinterfaces: java.io.Serializable

Description
String constants that define the predefined states of an order. All states except SUSPENDED and RESUME are mandatory. Besides the states defined here, an implementation may define further substates.

For a complete, graphical state model of an order, see “The Order Abstraction” in chapter “Concepts”.

## Member Summary

### Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>String ABORTED</td>
<td>81</td>
</tr>
<tr>
<td>String ABORTED_BYCLIENT</td>
<td>82</td>
</tr>
<tr>
<td>String ABORTED_BYSERVER</td>
<td>82</td>
</tr>
<tr>
<td>String CLOSED</td>
<td>82</td>
</tr>
<tr>
<td>String COMPLETED</td>
<td>82</td>
</tr>
<tr>
<td>String NOT_RUNNING</td>
<td>82</td>
</tr>
<tr>
<td>String NOT_STARTED</td>
<td>82</td>
</tr>
<tr>
<td>String OPEN</td>
<td>82</td>
</tr>
<tr>
<td>String RUNNING</td>
<td>82</td>
</tr>
<tr>
<td>String SUSPENDED</td>
<td>82</td>
</tr>
</tbody>
</table>

## Fields

**ABORTED**

public static final java.lang.String ABORTED

The order has not been completed successfully.
OrderState
ABORTED_BYCLIENT

public static final java.lang.String ABORTED_BYCLIENT
The order has been aborted by the client.

ABORTED_BYSERVER
public static final java.lang.String ABORTED_BYSERVER
The order has been aborted by the underlying implementation.

CLOSED
public static final java.lang.String CLOSED
The order has reached its final state. The state cannot change anymore.

COMPLETED
public static final java.lang.String COMPLETED
The order has been completed successfully (or at least partially successfully).

NOT_RUNNING
public static final java.lang.String NOT_RUNNING

NOT_STARTED
public static final java.lang.String NOT_STARTED
The order has been created and potentially initialized, but it has not yet been started.

OPEN
public static final java.lang.String OPEN
The order has not reached it final state and thus the state of the order might change in the future.

RUNNING
public static final java.lang.String RUNNING
The order is executing.

SUSPENDED
public static final java.lang.String SUSPENDED
The order is suspended, i.e. it has been running in part, but has been explicitly suspended by the client.
OrderType

Syntax
public interface OrderType extends java.io.Serializable

All Superinterfaces: java.io.Serializable

Description
String Constants that define the predefined order types.

An implementation may support only a subset of these types: ACTIVATE is mandatory, but all other order types are optional. Also an implementation may support additional type, that are not mentioned below.

See Also: getOrderTypes() 47

<table>
<thead>
<tr>
<th>Member Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields</td>
</tr>
<tr>
<td>String ACTIVATE</td>
</tr>
<tr>
<td>String DEACTIVATE</td>
</tr>
<tr>
<td>String MODIFY</td>
</tr>
<tr>
<td>String REMOVE</td>
</tr>
</tbody>
</table>

Fields

ACTIVATE

public static final java.lang.String ACTIVATE

DEACTIVATE

public static final java.lang.String DEACTIVATE

MODIFY

public static final java.lang.String MODIFY

REMOVE

public static final java.lang.String REMOVE
IllegalStateException

IllegalStateException(String)

IllegalStateException

Syntax

public class IllegalStateException extends java.lang.Exception

All Implemented Interfaces: java.io.Serializable

Description

Exception thrown if the state of an order is not appropriate to execute the method.

Member Summary

<table>
<thead>
<tr>
<th>Constructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>IllegalStateException(String)</td>
</tr>
</tbody>
</table>

Constructors

IllegalStateException(String)

public IllegalStateException(java.lang.String details)
Package
javax.oss.service

Class Summary

<table>
<thead>
<tr>
<th>Interfaces</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceKey</td>
<td></td>
</tr>
<tr>
<td>ServiceState</td>
<td></td>
</tr>
<tr>
<td>ServiceValue</td>
<td></td>
</tr>
</tbody>
</table>

String constants that define the predefined service states.

ServiceValue

Syntax

public interface ServiceValue extends ManagedEntityValue

All Superinterfaces: java.lang.Cloneable, ManagedEntityValue, java.io.Serializable

Member Summary

<table>
<thead>
<tr>
<th>Fields</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>String STATE</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>getServiceKey()</td>
<td></td>
</tr>
<tr>
<td>getServiceKey(ServiceKey)</td>
<td></td>
</tr>
<tr>
<td>getState()</td>
<td></td>
</tr>
<tr>
<td>setState(String)</td>
<td></td>
</tr>
<tr>
<td>setState(ServiceKey)</td>
<td></td>
</tr>
</tbody>
</table>

Fields

STATE

public static final java.lang.String STATE
ServiceKey
getAttributeType(String)

Methods

getAttributeType(String)
public java.lang.Class getAttributeType(java.lang.String attributeName)
throws IllegalArgumentException
Return the attribute type for an attribute. Returns the declared argument type of the set method. For example, for setMaxDataRate(int i) Integer.TYPE is returned.
Note: calling getAttributeValue might return null and thus its not possible to infer the type.
Throws:
   IllegalArgumentException - if violated: attributeName must be one of the strings returned from getAttributeNames()

getServiceKey()
public ServiceKey getServiceKey()

getState()
public java.lang.String getState()

setServiceKey(ServiceKey)
public void setServiceKey(ServiceKey key)

setState(String)
public void setState(java.lang.String state)

ServiceKey

Syntax
public interface ServiceKey extends ManagedEntityKey99

All Superinterfaces: java.lang.Cloneable, ManagedEntityKey99, java.io.Serializable

ServiceState

Syntax
public interface ServiceState extends java.io.Serializable

All Superinterfaces: java.io.Serializable
**Description**
String constants that define the predefined service states.

**Member Summary**

<table>
<thead>
<tr>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
</tr>
<tr>
<td>String</td>
</tr>
</tbody>
</table>

**Fields**

**ACTIVE**

```java
public static final java.lang.String ACTIVE
```

**INACTIVE**

```java
public static final java.lang.String INACTIVE
```
ServiceState

INACTIVE
CHAPTER 8

Package
javax.oss

Description
Interfaces shared by all OSS/J APIs.

This package contains the interfaces that are common to all OSS/J API.
The interface have been included here to make it easy to understand the complete API by reading just one document.

It will be assured that the definitions of these interface will always be consistent with all other APIs, for example with the Trouble Ticket API and Quality of Service API.

Class Summary

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>Interface to access the payload of all OSS/J JMS ObjectMessages.</td>
</tr>
<tr>
<td>Lookup</td>
<td>The OSS/J Lookup Service is a centrally managed service used to find the EJB components, JMS destinations (Queues, Topics) and connection factories involved in OSS/J management of multi-domain systems.</td>
</tr>
<tr>
<td>LookupHome</td>
<td>The LookupHome Home interface is used by a client to create a LookupBean Session Bean.</td>
</tr>
<tr>
<td>LookupResult</td>
<td>A LookupResult object is returned when trying to retrieve an object using attribute values.</td>
</tr>
<tr>
<td>ManagedEntityKey</td>
<td>A ManagedEntityKey is a unique identifier for all ManagedEntityValues.</td>
</tr>
<tr>
<td>ManagedEntityValue</td>
<td>The ManagedEntityValue interface is the base interface for all more detailed interfaces which represent any kind of object that is to be managed.</td>
</tr>
<tr>
<td>QueryValue</td>
<td>Constants that define possible result of bulk operations.</td>
</tr>
<tr>
<td>ReturnMode</td>
<td></td>
</tr>
<tr>
<td>XmlSerializable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exceptions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IllegalArgumentException</td>
<td>Similar exception than java.lang.IllegalArgumentException, just that this inherits from Exception, not RuntimeException.</td>
</tr>
<tr>
<td>SetException</td>
<td>Indicates that there was a concurrent access to an object.</td>
</tr>
<tr>
<td>UnsupportedOperationException</td>
<td>Similar exception than java.lang.UnsupportedOperationException, just that this inherits from Exception, not RuntimeException.</td>
</tr>
</tbody>
</table>
Lookup

Syntax
public interface Lookup extends javax.ejb.EJBObject

All Superinterfaces: javax.ejb.EJBObject, java.rmi.Remote

Description
The OSS/J Lookup Service is a centrally managed service used to find the EJB components, JMS destinations (Queues, Topics) and connection factories involved in OSS/J management of multi-domain systems. Application JNDI providers are not exposed directly to the client. Those JNDI providers are grouped in domains. The client is using domain names. A client will create a Lookup Session Bean to access to the set of application JNDI providers made available to it by the Administrator. Using this Session Bean, the client is able to: - list the names of the available domains - know the names of the attributes supported for lookups in a given domain - know the values for a given attribute used to name contexts, subcontexts or objects in a given domain - retrieve Home/Topic/Queue/ConnectionFactory object(s) associated to attribute value(s) - retrieve Home/Topic/Queue/ConnectionFactory object associated to a given JNDI name.

Member Summary

<table>
<thead>
<tr>
<th>Fields</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>String API_TYPE</td>
<td>ConnectionFactory getFactory(String, Name)</td>
</tr>
<tr>
<td>String NAME</td>
<td>EJBHome getHome(String, Name)</td>
</tr>
<tr>
<td>String PRODUCT_NAME</td>
<td>String getNamingAttributes(String)</td>
</tr>
<tr>
<td>String PRODUCT_VERSION</td>
<td>Queue getQueue(String, Name)</td>
</tr>
<tr>
<td>String VENDOR_NAME</td>
<td>Topic getTopic(String, Name)</td>
</tr>
<tr>
<td>String[] listAttributeValues(String, String, Attributes)</td>
<td>String[] listDomains()</td>
</tr>
<tr>
<td>String[] listFactories(String, Attributes)</td>
<td>String[] listHomes(String, Attributes)</td>
</tr>
<tr>
<td>String[] listHomes(String, String, String, String, String, String)</td>
<td>String[] listQueues(String, Attributes)</td>
</tr>
<tr>
<td>String[] listQueues(String, String, String, String, String, String)</td>
<td>String[] listTopics(String, Attributes)</td>
</tr>
<tr>
<td>String[] listTopics(String, String, String, String, String, String)</td>
<td></td>
</tr>
</tbody>
</table>
Fields

API_TYPE

public static final java.lang.String API_TYPE

NAME

public static final java.lang.String NAME

The name attribute can be used if the client is aware of the name of the EJB Home (for example TTSystem-Home)

PRODUCT_NAME

public static final java.lang.String PRODUCT_NAME

PRODUCT_VERSION

public static final java.lang.String PRODUCT_VERSION

VENDOR_NAME

public static final java.lang.String VENDOR_NAME

Methods

getFactory(String, Name)

public javax.jms.ConnectionFactory getFactory(java.lang.String domainName, javax.naming.Name name)

throws NamingException, NotContextException, IllegalArgumentException

Returns the ConnectionFactory with given JNDI name.

Parameters:

domainName -- domain name; null means in any domain

Name -- JNDI name

Throws:

NamingException -- if a naming exception is encountered

NotContextException -- if there is no domain (so no context) registered with specified URL
**getHome(String, Name)**

```java
class Lookup
{
    public javax.ejb.EJBHome getHome(java.lang.String domainName, javax.naming.Name name)
            throws NamingException, NotContextException, IllegalArgumentException
    {
        // Implementation
    }
}
```

Returns the Home with given JNDI name.

**Parameters:**
- `domainName` - domain name; null means in any domain
- `Name` - JNDI name

**Throws:**
- `NamingException` -- if a naming exception is encountered
- `NotContextException` -- if there is no domain (so no context) registered with specified URL
- `IllegalArgumentException` - if empty domain name or empty/null name

---

**getNamingAttributes(String)**

```java
class Lookup
{
    public java.lang.String[] getNamingAttributes(java.lang.String domainName)
            throws NotContextException, IllegalArgumentException
    {
        // Implementation
    }
}
```

Returns names of attributes supported in policy of given domain

**Parameters:**
- `domainName` - name of domain; null means all domains

**Throws:**
- `NotContextException` -- if there is no domain (so no context) registered with specified URL
- `IllegalArgumentException` - if empty domain name

---

**getQueue(String, Name)**

```java
class Lookup
{
    public javax.jms.Queue getQueue(java.lang.String domainName, javax.naming.Name name)
            throws NamingException, NotContextException, IllegalArgumentException
    {
        // Implementation
    }
}
```

Returns the Queue with given JNDI name.

**Parameters:**
- `domainName` - domain name; null means in any domain
- `Name` - JNDI name

**Throws:**
- `NamingException` -- if a naming exception is encountered
- `NotContextException` -- if there is no domain (so no context) registered with specified URL
- `IllegalArgumentException` - if empty domain name or empty/null name

---

**getTopic(String, Name)**

```java
class Lookup
{
    public javax.jms.Topic getTopic(java.lang.String domainName, javax.naming.Name name)
            throws NamingException, NotContextException, IllegalArgumentException
    {
        // Implementation
    }
}
```

Returns the Topic with given JNDI name.

**Parameters:**
- `domainName` - domain name; null means in any domain
- `Name` - JNDI name

**Throws:**
- `NamingException` -- if a naming exception is encountered
- `NotContextException` -- if there is no domain (so no context) registered with specified URL
- `IllegalArgumentException` - if empty domain name or empty/null name
Returns the Topic with given JNDI name.

Parameters:

domainName -- domain name; null means in any domain

Name -- JNDI name

Throws:

NamingException -- if a naming exception is encountered

NotContextException -- if there is no domain (so no context) registered with specified URL

IllegalArgumentException -- if empty domain name or empty/null name

---

listAttributeValues(String, String, Attributes)

```java
public java.lang.String[] listAttributeValues(String domainName,
                                            String att,
                                            javax.naming.directory.Attributes matchingAtts)
```

Throws NamingException, NoSuchAttributeException, IllegalArgumentException

Returns existing values for given attribute name using the given values of other attributes (if any).

Parameters:

domainName -- name of domain; null means all domains

att -- name of attribute for which existing values existing in JNDI providers in given domain have to be returned; null means all attributes

matchingAtts -- set of values of other attributes to refine the search; null means no constraint on other values

Throws:

NamingException -- in case of JNDI problem.

NoSuchAttributeException -- if no attribute with given name

IllegalArgumentException -- if empty domain name, attribute name or matching attribute list

---

listDomains()

```java
public java.lang.String[] listDomains()
```

Returns registered domain names

Returns: String[] An array of String

---

listFactories(String, Attributes)

```java
public javax.naming.NamingEnumeration listFactories(String domainName,
                                                    javax.naming.directory.Attributes matchingAtts)
```

Throws IllegalArgumentException, NamingException

Returns the ConnectionFactory objects associated in given domain to given attribute values. The returned NamingEnumeration is a set of LookupResult instances, each containing the following info: - Object itself (already narrowed to ConnectionFactory) - Attributes (for each attribute, its value associated to the object is provided) - the JNDI name
Lookup

listFactories(String, String, String, String, String, String)

Parameters:
  domainName -- domain name; null means in all domains
  matchingAtts -- attribute values; null means no constraint

Throws:
  IllegalArgumentException -- if empty domain or attribute list
  NamingException -- if a naming exception is encountered

listFactories(String, String, String, String, String, String)

public javax.naming.NamingEnumeration listFactories(java.lang.String domainName,
  java.lang.String apiType, java.lang.String vendorName,
  java.lang.String productName, java.lang.String versionName,
  java.lang.String objName)
throws IllegalArgumentException, NamingException

Returns the ConnectionFactory objects associated in given domain to given attribute values. The returned
NamingEnumeration is a set of LookupResult instances, each containing the following info:
- Object itself (already narrowed to ConnectionFactory)
- Attributes (for each attribute, its value associated to the object is provided)
- The JNDI name

The attributes which can be specified here are the ones recommended to be used in the default naming scheme. Any of them can be null if it is not known.

Parameters:
  domainName -- domain name; null means in all domains
  apiType -- type of the API
  vendorName -- name of vendor
  productName -- name of product
  versionName -- name of version
  objName -- name of object

Throws:
  IllegalArgumentException -- if empty domain or attribute list
  NamingException -- if a naming exception is encountered

listHomes(String, Attributes)

public javax.naming.NamingEnumeration listHomes(java.lang.String domainName,
  javax.naming.directory.Attributes matchingAtts)
throws IllegalArgumentException, NamingException

Returns the EJBHome objects associated in given domain to given attribute values. The returned Naming-Enumeration is a set of LookupResult instances, each containing the following info:
- Object itself (already narrowed to EJBHome)
- Attributes (for each attribute, its value associated to the object is provided)
- The JNDI name

Parameters:
  domainName -- domain name; null means in all domains
  matchingAtts -- attribute values; null means no constraint

Throws:
  IllegalArgumentException -- if empty domain or attribute list
Lookup

listHomes(String, String, String, String, String, String)

public javax.naming.NamingEnumeration listHomes(java.lang.String domainName,
        java.lang.String apiType, java.lang.String vendorName,
        java.lang.String productName, java.lang.String versionName,
        java.lang.String objName)
throws IllegalArgumentException, NamingException

Returns the EJBHome objects associated in given domain to given attribute values. The returned NamingEnumeration is a set of LookupResult instances, each containing the following info: - Object itself (already narrowed to EJBHome) - Attributes (for each attribute, its value associated to the object is provided) - the JNDI name The attributes which can be specified here are the ones recommended to be used in the default naming scheme. Any of them can be null if it is not known.

Parameters:
  domainName -- domain name; null means in all domains
  apiType -- type of the API
  vendorName -- name of vendor
  productName -- name of product
  versionName -- name of version
  objName -- name of object

Throws:
  IllegalArgumentException 107 -- if empty domain or attribute
  NamingException -- if a naming exception is encountered

listQueues(String, Attributes)

public javax.naming.NamingEnumeration listQueues(java.lang.String domainName,
        javax.naming.directory.Attributes matchingAtts)
throws IllegalArgumentException, NamingException

Returns the Queue objects associated in given domain to given attribute values. The returned NamingEnumeration is a set of LookupResult instances, each containing the following info: - Object itself (already narrowed to Queue) - Attributes (for each attribute, its value associated to the object is provided) - the JNDI name

Parameters:
  domainName -- domain name; null means in all domains
  matchingAtts -- attribute values; null means no constraint

Throws:
  IllegalArgumentException 107 -- if empty domain or attribute list
  NamingException -- if a naming exception is encountered
listQueues(String, String, String, String, String, String)

public javax.naming.NamingEnumeration listQueues(java.lang.String domainName,
java.lang.String apiType, java.lang.String vendorName,
java.lang.String productName, java.lang.String versionName,
java.lang.String objName)
throws IllegalArgumentException, NamingException

Returns the Queue objects associated in given domain to given attribute values. The returned Naming-Enumeration is a set of LookupResult instances, each containing the following info: - Object itself (already narrowed to Queue) - Attributes (for each attribute, its value associated to the object is provided) - the JNDI name The attributes which can be specified here are the ones recommended to be used in the default naming scheme. Any of them can be null if it is not known.

Parameters:
  domainName -- domain name; null means in all domains
  apiType -- type of the API
  vendorName -- name of vendor
  productName -- name of product
  versionName -- name of version
  objName -- name of object

Throws:
  IllegalArgumentException -- if empty domain or attribute
  NamingException -- if a naming exception is encountered

listTopics(String, Attributes)

public javax.naming.NamingEnumeration listTopics(java.lang.String domainName,
javax.naming.directory.Attributes matchingAtts)
throws IllegalArgumentException, NamingException

Returns the Topic objects associated in given domain to given attribute values. The returned Naming-Enumeration is a set of LookupResult instances, each containing the following info: - Object itself (already narrowed to Topic) - Attributes (for each attribute, its value associated to the object is provided) - the JNDI name

Parameters:
  domainName -- domain name; null means in all domains
  matchingAtts -- attribute values; null means no constraint

Throws:
  IllegalArgumentException -- if empty domain or attribute list
  NamingException -- if a naming exception is encountered

listTopics(String, String, String, String, String, String)

public javax.naming.NamingEnumeration listTopics(java.lang.String domainName,
java.lang.String apiType, java.lang.String vendorName,
java.lang.String productName, java.lang.String versionName,
Returns the Topic objects associated in given domain to given attribute values. The returned Naming-Enumeration is a set of LookupResult instances, each containing the following info: - Object itself (already narrowed to Topic) - Attributes (for each attribute, its value associated to the object is provided) - the JNDI name The attributes which can be specified here are the ones recommended to be used in the default naming scheme. Any of them can be null if it is not known.

**Parameters:**
- **domainName** -- domain name; null means in all domains
- **apiType** -- type of the API
- **vendorName** -- name of vendor
- **productName** -- name of product
- **versionName** -- name of version
- **objName** -- name of object

**Throws:**
- IllegalArgumentException -- if empty domain or attribute
- NamingException -- if a naming exception is encountered

### Syntax

```java
public interface LookupHome extends javax.ejb.EJBHome
```

### All Superinterfaces:
- javax.ejb.EJBHome, java.rmi.Remote

### Description
The LookupHome Home interface is used by a client to create a LookupBean Session Bean. A Lookup Remote interface is returned. A LookupBean Session Bean created in a Home will automatically retrieve the JndiProvideHandlerHome Home interface located in the same JNDI Provider as itself, i.e. the client JNDI provider. It will then use the JndiProviderHandler Remote interfaces for the JndiProviderHandlerBeans put there by the Administrator for the available application JNDI providers.
LookupResult
create()

Methods

create()

public Lookup90 create()
    throws NamingException

Creates a LookupBean. No parameter, it will automatically use the local JndiProviderHandlerHome.

.Throws:
    NamingException -- no local Home for registered application JNDI providers

LookupResult

Syntax

public interface LookupResult extends java.io.Serializable, java.lang.Cloneable

All Superinterfaces: java.lang.Cloneable, java.io.Serializable

Description

A LookupResult object is returned when trying to retrieve an object using attribute values. It contains as members: - the Object to be returned; it has been narrowed, depending on the type of the Object, to EJBHome, Topic, Queue or ConnectionFactory. So it just has to be downcasted - an Attributes object returning the values of all attributes associated to that Object in the JNDI provider where it is bound. - the JNDI name to be used to directly retrieve that Object

Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Object</td>
</tr>
</tbody>
</table>

Methods

getAttributes()

public javax.naming.directory.Attributes getAttributes()

Returns the set of attributes associated to that Object

getJndiName()

public javax.naming.Name getJndiName()

Returns the JNDI name to retrieve directly the object
**ManagedEntityKey**

### Syntax

```java
public interface ManagedEntityKey extends java.io.Serializable, java.lang.Cloneable
```

**All Known Subinterfaces:** OrderKey, ServiceKey

**All Superinterfaces:** java.lang.Cloneable, java.io.Serializable

### Description

A ManagedEntityKey is a unique identifier for all ManagedEntityValues.

The type and domain attributes are included in ManagedEntityKey, because a primaryKey is only unique within its domain and type, so these attributes are needed to define a globally unique key.

### Member Summary

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getDomain()</td>
</tr>
<tr>
<td>Object getPrimaryKey()</td>
</tr>
<tr>
<td>String getType()</td>
</tr>
<tr>
<td>void setDomain(String)</td>
</tr>
<tr>
<td>void setPrimaryKey(Object)</td>
</tr>
<tr>
<td>void setType(String)</td>
</tr>
</tbody>
</table>

### Methods

#### getDomain()

```java
public java.lang.String getDomain()
```

The Domain name identifies an application’s administrative domain.

An administrative domain maintains a JNDI namespace. This is part of ManagedEntityKey, because primaryKeys can be duplicate between ManagedEntityValue, implemented by the same class, but with different domain.

**Returns:** String domain name
**ManagedEntityKey**

**getPrimaryKey()**

```java
type public java.lang.Object getPrimaryKey()
```

The unique identifier for the Managed Entity.

In case the implementation uses Entity Beans, an implementation may map the primary key to the EJB primary key.

**Returns:** the primary key

**getType()**

```java
type public java.lang.String getType()
```

Gets the type of ManagedEntityValue within ManagedEntityKey.

This is part of ManagedEntityKey, because primaryKey can be duplicate between ManagedEntityValue, implemented by the same class, but from different type (not java type).

**Returns:** String entity's type name

**setDomain(String)**

```java
void public void setDomain(java.lang.String domain)
  throws java.lang.IllegalArgumentException
```

Sets a new value for domain. There are no assumptions about it's format, yet.

**Parameters:**
  - domain - the new value for domain

**Throws:**
  - java.lang.IllegalArgumentException - to be defined in subinterfaces / classes.

**setPrimaryKey(Object)**

```java
void public void setPrimaryKey(java.lang.Object key)
  throws java.lang.IllegalArgumentException
```

Sets a new value for key.

**Parameters:**
  - key - the new value for key

**Throws:**
  - java.lang.IllegalArgumentException - to be defined in subinterfaces / classes.

**setType(String)**

```java
void public void setType(java.lang.String type)
  throws java.lang.IllegalArgumentException
```

Sets a new value for type.

**Parameters:**
  - type - the new value for type
ManagedEntityValue

Syntax
public interface ManagedEntityValue extends java.io.Serializable, java.lang.Cloneable

All Known Subinterfaces: OrderValue, ServiceValue

All Superinterfaces: java.lang.Cloneable, java.io.Serializable

Description
The ManagedEntityValue interface is the base interface for all more detailed interfaces which represent any kind of object that is to be managed.

Classes which implement ManagedEntityValue or a sub interface of ManagedEntityValue are also called value class.

All classes implementing a value interface provide several ways to access the attributes:

- Attributes can be accessed through standard JavaBeans get/set (is/set) methods.
- Attributes can be accessed through the generic methods
  public Object getAttributeValue(String attributeName)
  public void setAttributeValue(String attributeName, Object newValue)

A Client needs to know which attributes exists in order to provide correct attributeNames. It can get these information with a call to public String[] getAttributeNames().

Attributes in a value class need not to be present. If an attribute contains some useful data it is called populated. If a client wants to read an attribute, it should first check if it is populated.

<table>
<thead>
<tr>
<th>Member Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fields</strong></td>
</tr>
<tr>
<td>String KEY</td>
</tr>
<tr>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td>String[] getAttributeNames()</td>
</tr>
<tr>
<td>Object getAttributeValue(String)</td>
</tr>
<tr>
<td>ManagedEntityKey getManagedEntityKey()</td>
</tr>
<tr>
<td>String[] getPopulatedAttributeNames()</td>
</tr>
<tr>
<td>String[] getSettableAttributeNames()</td>
</tr>
<tr>
<td>boolean isFullyPopulated()</td>
</tr>
<tr>
<td>boolean isPopulated(String)</td>
</tr>
<tr>
<td>void setAttributeValue(String, Object)</td>
</tr>
<tr>
<td>void setFullyPopulated()</td>
</tr>
<tr>
<td>void setManagedEntityKey(ManagedEntityKey)</td>
</tr>
<tr>
<td>void unpopulateAllAttributes()</td>
</tr>
<tr>
<td>void unpopulateAttribute(String)</td>
</tr>
</tbody>
</table>
ManagedEntityValue
KEY

Fields

KEY

public static final java.lang.String KEY

This String defines the attribute name for the ManagedEntityKey attribute.

Methods

getAttributeNames()

public java.lang.String[] getAttributeNames()

Returns all attribute names, which are available in this value object.

Use one of the returned names to access the generic methods getAttributeValue(...) and setAttributeValue(...).

This method may be used by generic clients to obtain information on the attributes. It does not say anything about the state of an attribute, i.e. if it is populated or not.

Returns: the array contains all attribute names in no particular order.

getAttributeValue(String)

public java.lang.Object getAttributeValue(java.lang.String attributeName)
throws java.lang.IllegalArgumentException

This method returns the value of the specified attribute.

Parameters:
attributeName - the attribute’s name

Returns: The attribute’s value. Primitive types are wrapped in their respective classes.

Throws:
java.lang.IllegalArgumentException - An IllegalArgumentException is thrown, when
• there is no attribute with this name
• the attribute is not populated

getManagedEntityKey()

public ManagedEntityKey getManagedEntityKey()
throws java.lang.IllegalArgumentException

Gets the key for this object. The key is unique over all objects.

Returns: returns the key for this value object

Throws:
java.lang.IllegalArgumentException - in case the key attribute is not populated

102
getPopulatedAttributeNames()

```java
public java.lang.String[] getPopulatedAttributeNames()
```

Gets all attribute names, which attribute values contain something meaningful.

Although an attribute is populated, it can be null!

**Returns:** all names of attributes, which contain some data. When no attributes are populated an **empty array** is returned. It is required to return a subset of the array returned by `getAttributeNames()`.

getSettableAttributeNames()

```java
public java.lang.String[] getSettableAttributeNames()
```

Gets all attributes which can be set in the server implementation.

It is required to return a subset of the array returned by `getAttributeNames()`.

isFullyPopulated()

```java
public boolean isFullyPopulated()
```

Returns true, if all attributes in this value object are populated.

**Returns:** true, if all attributes are populated

**See Also:** `isPopulated(String)`

isPopulated(String)

```java
public boolean isPopulated(java.lang.String name)
```

Checks if a specific attribute is populated. If the value object is fully populated, i.e. `isFullyPopulated()` returns true, this method returns true;

**Parameters:**

- name - the name of the attribute which is to be checked for population

**Returns:** true, if this attribute contains some data, false otherwise

**Throws:**

- `java.lang.IllegalArgumentException` - this exception is thrown, when there is no attribute with this name

**See Also:** `isFullyPopulated()`

setAttributeValue(String, Object)

```java
public void setAttributeValue(java.lang.String attributeName, java.lang.Object Value)
```

Assings a new value to an attribute.
ManagedEntityValue

setFullyPopulated()

Even though some attributes may be readonly in the server implementation, they can be set here nonetheless. This is because value objects are also used as templates for a “query by template”. To see which attributes can be set in the server implementation, the client needs to call getSettableAttributeNames().

Parameters:
- attributeName - The attribute’s name which shall be changed
- Value - The attribute’s new value. This can either be:
  - An Object which can be casted to the real type of attributesName
  - A wrapper class for primitive types, i.e. Integer instead of int. In any other case an exception is thrown.

Throws:
- java.lang.IllegalArgumentException - This Exception is thrown, when either
  - There is no attribute with this name
  - The value is out-of-range.

setFullyPopulated()

public void setFullyPopulated()

Defines this value object as fully populated.

setManagedEntityKey(ManagedEntityKey)

public void setManagedEntityKey(ManagedEntityKey key)

Throws: java.lang.IllegalArgumentException - an exception is thrown when the given key is not of correct type. Typically, subinterfaces of ManagedEntityValue have an corresponding subinterface of ManagedEntityKey. An implementing type of this sub key class might be expected as a parameter.

unpopulateAllAttributes()

public void unpopulateAllAttributes()

Reset all the attributes to unpopulated.

unpopulateAttribute(String)

public void unpopulateAttribute(java.lang.String attr_name)

Throws java.lang.IllegalArgumentException

Parameters:
- attr_name - The attribute which shall be unpopulated.
Event

Syntax
public interface Event extends java.io.Serializable

All Known Subinterfaces:  
- OrderAttributeValueChangeEvent
- OrderCreateEvent
- OrderRemoveEvent
- OrderStateChangeEvent

All Superinterfaces:  
- java.io.Serializable

Description
Interface to access the payload of all OSS/J JMS ObjectMessages. In OSS/J, java value type interfaces send JMS Messages, that are ObjectMessages. This interface is the base interface for each ObjectMessage that is sent.

If onMessage is subscribed to receive a message, the following code will run:

```java
public void onMessage(Message msg) {
    ObjectMessage omsg = (ObjectMessage) msg;
    Event event = (Event) omsg.getObject();
    System.out.println("New event received from " + event.getDomain());
}
```

Throw:
java.lang.IllegalArgumentException - thrown, if this is not a valid attribute name

See Also: unpopulateAllAttributes()

### Member Summary

#### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getDomain()</td>
<td></td>
</tr>
<tr>
<td>Date getEventTime()</td>
<td></td>
</tr>
<tr>
<td>String getEventType()</td>
<td></td>
</tr>
<tr>
<td>String getNotificationId()</td>
<td></td>
</tr>
<tr>
<td>String getSourceIndicator()</td>
<td></td>
</tr>
<tr>
<td>void setDomain(String)</td>
<td></td>
</tr>
<tr>
<td>void setEventTime(Date)</td>
<td></td>
</tr>
<tr>
<td>void setEventType(String)</td>
<td></td>
</tr>
<tr>
<td>void setNotificationId(String)</td>
<td></td>
</tr>
<tr>
<td>void setSourceIndicator(String)</td>
<td></td>
</tr>
</tbody>
</table>

### Methods

getDomain()
Event
gEventTime()

Returns the admin domain of the system that has published this message.
By using the lookup service, you can get the home interface that was “responsible” for sending the message.
See Also: Lookup

gEventTime()

public java.util.Date getEventTime()

Returns the time when the event was published by the other system.

gEventType()

public java.lang.String getEventType()

getNotificationId()

public java.lang.String getNotificationId()

getSourceIndicator()

public java.lang.String getSourceIndicator()

setDomain(String)

public void setDomain(java.lang.String nid)

Throws:
    IllegalArgumentException

setEventTime(Date)

public void setEventTime(java.util.Date time)

Throws:
    IllegalArgumentException

setEventType(String)

public java.lang.String setEventType(java.lang.String event_type)

Throws:
    IllegalArgumentException

setNotificationId(String)

public void setNotificationId(java.lang.String nid)

Throws:
    IllegalArgumentException
setQueryValue

setQueryValue(String)

public void setQueryValue(String value)

Throws:
   IllegalArgumentException

QueryValue

Syntax
public interface QueryValue extends java.io.Serializable, java.lang.Cloneable

All Superinterfaces: java.lang.Cloneable, java.io.Serializable

Member Summary

Methods
   void reset()

IllegalArgumentException

Syntax
public class IllegalArgumentException extends java.lang.Exception

All Implemented Interfaces: java.io.Serializable

Description
Similar exception than java.lang.IllegalArgumentException, just that this inherits from Exception, not RuntimeException.

This exception is thrown if the argument values of a remote method are out-of-range.

This new class is required because J2EE container deal with RuntimeException in a special way, see EJB specification.
SetException
IllegalArgumentException()

Constructor Summary

IllegalArgumentException()

public IllegalArgumentException()

IllegalArgumentException(String)

public IllegalArgumentException(java.lang.String message)

SetException

Syntax
public class SetException extends java.lang.Exception

All Implemented Interfaces:
java.io.Serializable

Description
Indicates that there was a concurrent access to an object.
This exception is thrown by setMEV() methods, if one client would overwrite the changes done by another cli-

cient.

Constructor Summary

SetException()

public SetException()

SetException(String)

public SetException(java.lang.String message)
UnsupportedOperationException

Syntax
public class UnsupportedOperationException extends java.lang.Exception

All Implemented Interfaces: java.io.Serializable

Description
Similar exception than java.lang.UnsupportedOperationException, just that this inherits from Exception, not RuntimeException.

This exception is thrown if a method is not implemented (and the throw clause states that this exception may be thrown). Note that in this case, also the return value of getSupportedOperations() of the EJB should indicates that the method is not implemented.

This new exception required because J2EE container deal with RuntimeException in a special way, see EJB specification.

<table>
<thead>
<tr>
<th>Member Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructors</td>
</tr>
<tr>
<td>UnsupportedOperationException()</td>
</tr>
<tr>
<td>public UnsupportedOperationException()</td>
</tr>
<tr>
<td>UnsupportedOperationException(String)</td>
</tr>
<tr>
<td>public UnsupportedOperationException(java.lang.String message)</td>
</tr>
</tbody>
</table>

XmlSerializable

Syntax
public interface XmlSerializable
### Methods

**fromXml(Element)**

```java
public void fromXml(org.w3c.dom.Element element)
throws org.w3c.dom.DOMException
```

Deserializes the XML DOM element subtree into the Java XmlSerializable object.

**Parameters:**
- `element` - org.w3c.dom.Element the subtree representing the serialized object

**Throws:**
- org.w3c.dom.DOMException

**getRootName()**

```java
public java.lang.String getRootName()
```

Get the XML element name at the root of this document type, where it exists as a standalone XML document instance.

**Returns:** String XML element name

**getXmlHeader()**

```java
public java.lang.String getXmlHeader()
```

Get the XML doctype declaration for a document instance of this type.

**Returns:** String XML doctype declaration

**toXml()**

```java
public java.lang.String toXml()
```

Serialize this Java object as an XML document instance with an XML doctype declaration.

**Returns:** String XML document instance

**toXml(String)**

```java
public java.lang.String toXml(java.lang.String elementName)
```
Serializes this Java object as an XML element. No doctype declaration is generated, in order to allow this string to be embedded as an element of a larger XML document instance.

**Parameters:**
- `elementName` - String, the name of the element for this value

**Returns:** String XML element
XmlSerializable

toXml(String)
This chapter defines the XML schema, which is used by the XML Value Type interface and for XML/JMS interface.

The schema is splitted into two parts:

- The first schema is the common schema for all OSS/J APIs
- The second part is the Service Activation specific part.

The relationship of the following XML schema to XML standards like ebXML (www.ebxml.org) or SOAP and related JCPs 67 and 101 is described in the OSS/J Common Guidelines.

Please note that this is an early draft.
**OSS Common XML Schema**

**complexType** `ArrayOfString`

- **namespace**: http://www.somewhere.org/Common
- **children**: `Item`
- **annotation**:
  - **documentation**: This is a representation of a String array.

```xml
<complexType name="ArrayOfString">
  <annotation>
    <documentation>This is a representation of a String array.</documentation>
  </annotation>
  <sequence>
    <element name="Item" type="string" nullable="true" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

**element** `ArrayOfString/Item`

- **namespace**: http://www.somewhere.org/Common
- **type**: string
- **source**:
  ```xml
  <element name="Item" type="string" nullable="true" maxOccurs="unbounded"/>
  ```

**complexType** `BaseEventType`
XML Schema

```
<complexType name="BaseEventType" abstract="true">
  <annotation>
    <documentation>Base Event</documentation>
  </annotation>
  <sequence>
    <element name="EventType" type="string" nullable="false"/>
    <element name="EventTime" type="timeInstant" nullable="false"/>
    <element name="Domain" type="string" nullable="false"/>
    <element name="NotificationId" type="string" nullable="false"/>
    <element name="SourceIndicator" type="string" nullable="false"/>
  </sequence>
</complexType>
```

---

element BaseEventType/EventType

```
<complexType name="BaseEventType">
  <sequence>
    <element name="EventType" type="string" nullable="false"/>
  </sequence>
</complexType>
```

element BaseEventType/EventTime

```
<complexType name="BaseEventType">
  <sequence>
    <element name="EventTime" type="string" nullable="false"/>
  </sequence>
</complexType>
```
### XML Schema

<table>
<thead>
<tr>
<th>diagram</th>
<th>name</th>
<th>namespace</th>
<th>type</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>EventTime</code></td>
<td><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></td>
<td>timeInstant</td>
<td><code>&lt;element name=&quot;EventTime&quot; type=&quot;timeInstant&quot; nullable=&quot;false&quot;/&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>Domain</code></td>
<td><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></td>
<td>string</td>
<td><code>&lt;element name=&quot;Domain&quot; type=&quot;string&quot; nullable=&quot;false&quot;/&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>NotificationId</code></td>
<td><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></td>
<td>string</td>
<td><code>&lt;element name=&quot;NotificationId&quot; type=&quot;string&quot; nullable=&quot;false&quot;/&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>SourceIndicator</code></td>
<td><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></td>
<td>string</td>
<td><code>&lt;element name=&quot;SourceIndicator&quot; type=&quot;string&quot; nullable=&quot;false&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

#### element **BaseEventType/Domain**

- **Diagram:** `Domain`
- **Namespace:** http://www.somewhere.org/Common
- **Type:** string
- **Source:** `<element name="Domain" type="string" nullable="false"/>`

#### element **BaseEventType/NotificationId**

- **Diagram:** `NotificationId`
- **Namespace:** http://www.somewhere.org/Common
- **Type:** string
- **Source:** `<element name="NotificationId" type="string" nullable="false"/>`

#### element **BaseEventType/SourceIndicator**

- **Diagram:** `SourceIndicator`
- **Namespace:** http://www.somewhere.org/Common
- **Type:** string
- **Source:** `<element name="SourceIndicator" type="string" nullable="false"/>`

#### complexType **BaseException**
The BaseException is the parent complexType of the Exceptions.

The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.

**namespace**
http://www.somewhere.org/Common

**children**
- Message

**used by**
- CreateException
- DuplicateKeyException
- FinderException
- IllegalArgumentException
- IllegalStateException
- ObjectNotFoundException
- RemoteException
- RemoveException
- SetException
- UnsupportedOperationException

**annotation**
The BaseException is the parent complexType of the Exceptions.

**source**
```
<complexType name="BaseException">
  <annotation>
    <documentation>The BaseException is the parent complexType of the Exceptions.</documentation>
  </annotation>
  <sequence>
    <element name="Message" type="string">
      <annotation>
        <documentation>The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.</documentation>
      </annotation>
    </element>
  </sequence>
</complexType>
```

**element** BaseException/Message
The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.

```
<element name="Message" type="string">
  <annotation>
    <documentation>The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.</documentation>
  </annotation>
</element>
```
**complexType CreateException**

The `CreateException` exception can be returned by all `create(...)` requests defined in the OSS through Java XML/JMS interface. The exception is used as a standard application-level exception to report a failure to create a managed entity or a collection of managed entities. This exception is thrown when a particular managed entity or group of managed entities cannot be created.

**namespace** http://www.somewhere.org/Common

**type** extension of `co:BaseException`

**children** `Message`

**source**
```xml
<complexType name="CreateException">
  <annotation>
    <documentation> The CreateException exception can be returned by all create(...) requests defined in the OSS through Java XML/JMS interface. The exception is used as a standard application-level exception to report a failure to create a managed entity or a collection of managed entities. This exception is thrown when a particular managed entity or group of managed entities cannot be created. </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```

**complexType DuplicateKeyException**

The `DuplicateKeyException` exception is likely the result from a `Exception.getMessage()` call.
The DuplicateKeyException exception is returned if a managed entity cannot be created because an object with the same key already exists. This exception is only used when a managed entity key is provided in a create(...) request and when client controlled naming is used. This exception is returned by the create requests defined in the OSS through Java XML/JMS interface.

```xml
<complexType name="DuplicateKeyException">
  <annotation>
    <documentation>
The DuplicateKeyException exception is returned if a managed entity cannot be created because an object with the same key already exists. This exception is only used when a managed entity key is provided in a create(...) request and when client controlled naming is used. This exception is returned by the create requests defined in the OSS through Java XML/JMS interface. </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence />
    </extension>
  </complexContent>
</complexType>
```
The exception is used as a standard application-level exception to report a failure to find the requested managed entities. This exception is returned when a collection of one or more entity cannot be found. This exception should not be returned by requests that return a collection of managed entities using an associative lookup approach (they should return a null list instead).

```xml
<complexType name="FinderException">
  <annotation>
    <documentation> The exception is used as a standard application-level exception to report a failure to find the requested managed entities. This exception is returned when a collection of one or more entity cannot be found. This exception should not be returned by requests that return a collection of managed entities using an associative lookup approach (they should return a null list instead). </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```

**NAMESPACE**

<table>
<thead>
<tr>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>FinderException</td>
</tr>
<tr>
<td>Message</td>
</tr>
</tbody>
</table>

The `Message` element indicates the error message from the Exception. This is most likely the result from a `Exception.getMessage()` call.

**SOURCE**

```xml
<complexType name="FinderException">
  <annotation>
    <documentation> The exception is used as a standard application-level exception to report a failure to find the requested managed entities. This exception is returned when a collection of one or more entity cannot be found. This exception should not be returned by requests that return a collection of managed entities using an associative lookup approach (they should return a null list instead). </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```

**COMPLEX TYPE**

`IllegalArgument Exception`
The `IllegalArgumentException` exception is returned by the OSS through Java XML/JMS interface to report that the request could not be completed because one of the arguments passed in is invalid.

```xml
<complexType name="IllegalArgumentException">
  <annotation>
    <documentation>
      The `IllegalArgumentException` exception is returned by the OSS through Java XML/JMS interface to report that the request could not be completed because one of the arguments passed in is invalid.
    </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```
The IllegalArgumentException exception is returned by the OSS through Java XML/JMS interface to report that the request could not be completed because one of the arguments passed in is invalid.

The IllegalArgumentException exception is returned by the OSS through Java XML/JMS interface to report that the request could not be completed because one of the arguments passed in is invalid.

```xml
<complexType name="IllegalStateException">
  <annotation>
    <documentation>The IllegalArgumentException exception is returned by the OSS through Java XML/JMS interface to report that the request could not be completed because one of the arguments passed in is invalid.</documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```
This is a representation of an Iterator Request.

The HowMany element indicates how many result instances should be returned in the response message. If the value supplied is greater than the complete result set, then just the result set is returned in the response message, and EndOfResult will set to 'true'. If the HowMany element exceeds a preset application maximum, the number of instances returned will equal the preset application maximum. Not specifying this element will return the entire result set.
<table>
<thead>
<tr>
<th>&lt;complexType name=&quot;IteratorRequest&quot;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;annotation&gt;</td>
</tr>
<tr>
<td>&lt;documentation&gt;This is a representation of an Iterator Request.&lt;/documentation&gt;</td>
</tr>
<tr>
<td>&lt;/annotation&gt;</td>
</tr>
<tr>
<td>&lt;sequence&gt;</td>
</tr>
<tr>
<td>&lt;element name=&quot;HowMany&quot; type=&quot;positiveInteger&quot; nullable=&quot;true&quot; minOccurs=&quot;0&quot;&gt;</td>
</tr>
<tr>
<td>&lt;annotation&gt;</td>
</tr>
<tr>
<td>&lt;documentation&gt;The HowMany element indicates how many result instances should be returned in the Response message. If the value supplied is greater than the complete result set then just the result set is returned in the response message, and End-Of-Reply element will be set to 'true'. If the HowMany element exceeds a preset maximum (for performance reasons) then the number of instances returned will equal the preset application maximum. Not specifying this element will return the entire result set.&lt;/documentation&gt;</td>
</tr>
<tr>
<td>&lt;/annotation&gt;</td>
</tr>
<tr>
<td>&lt;/element&gt;</td>
</tr>
<tr>
<td>&lt;/sequence&gt;</td>
</tr>
<tr>
<td>&lt;/complexType&gt;</td>
</tr>
</tbody>
</table>
### element IteratorRequest/HowMany

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>The HowMany element indicates how many result instances should be returned in the Response message. If the value supplied is greater than the complete result set then just the result set is returned in the response message, and EndOfReply element will be set to 'true'. If the HowMany element exceeds a preset application maximum (for performance reasons) then the number of instances returned will equal the preset application maximum. Not specifying this element will return the entire result set.</td>
</tr>
</tbody>
</table>

| namespace | http://www.somewhere.org/Common |
| type | positiveInteger |

<table>
<thead>
<tr>
<th>annotation</th>
<th>documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The HowMany element indicates how many result instances should be returned in the Response message. If the value supplied is greater than the complete result set then just the result set is returned in the response message, and EndOfReply element will be set to 'true'. If the HowMany element exceeds a preset application maximum (for performance reasons) then the number of instances returned will equal the preset application maximum. Not specifying this element will return the entire result set.</td>
<td></td>
</tr>
</tbody>
</table>
The HowMany element indicates how many result instances should be returned in the Response message. If the value supplied is greater than the complete result set then just the result set is returned in the response message, and End-Of-Reply element will be set to 'true'. If the HowMany element exceeds a preset maximum (for performance reasons) then the number of instances returned will equal the preset application maximum. Not specifying this element will return the entire result set.
### complexType `IteratorResponse`

**Diagram:**

- **IteratorResponse**: This is a representation of an Iterator Response.
- **Sequence**: The `Sequence` number indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to ensure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.
- **EndOfReply**: The `EndOfReply` indicates whether there are any more result sets being returned.

**Namespace:**

`http://www.somewhere.org/Common`

**Children:**

- `Sequence`
- `EndOfReply`

**Annotation:**

This is a representation of an Iterator Response.

**Source Code:**

```xml
<complexType name="IteratorResponse">
  <annotation>
    <documentation> This is a representation of an Iterator Response. </documentation>
  </annotation>
  <sequence>
    <element name="Sequence" type="positiveInteger">
      <annotation>
        <documentation> The Sequence number indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to ensure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned. </documentation>
      </annotation>
    </element>
    <element name="EndOfReply" type="boolean">
      <annotation>
        <documentation> The EndOfReply indicates whether there are any more result sets being returned. </documentation>
      </annotation>
    </element>
  </sequence>
</complexType>
```
**element** IteratorResponse/Sequence

![Diagram](diagram.png)

- **namespace**: http://www.somewhere.org/Common
- **type**: positiveInteger
- **annotation**: documentation

  The Sequence number indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.

**source**

```xml
<element name="Sequence" type="positiveInteger">
  <annotation>
    <documentation>
      The Sequence number indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.
    </documentation>
  </annotation>
</element>
```

**element** IteratorResponse/EndOfReply

- **namespace**: http://www.somewhere.org/Common
- **type**: boolean
- **annotation**: documentation

  The EndOfReply indicates whether there are any more result sets being returned.
XML Schema

<table>
<thead>
<tr>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;element name=&quot;EndOfReply&quot; type=&quot;boolean&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;annotation&gt;</code></td>
</tr>
<tr>
<td><code>&lt;documentation&gt;The EndOfReply indicates whether there are any more result sets being returned. &lt;/documentation&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/annotation&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/element&gt;</code></td>
</tr>
</tbody>
</table>
complexType **ObjectNotFoundException**

The ObjectNotFoundException exception is returned by an OSS through Java request to indicate that the specified managed entity does not exist. Only the request that are declared to return a single managed entity use this exception. This exception should not be returned by methods that return a collection of managed entities. This exception is returned when a singular managed entity cannot be found.

```xml
<complexType name="ObjectNotFoundException">
  <annotation>
    <documentation> The ObjectNotFoundException exception is returned by an OSS through Java request to indicate that the specified managed entity does not exist. Only the request that are declared to return a single managed entity use this exception. This exception should not be returned by methods that return a collection of managed entities. This exception is returned when a singular managed entity cannot be found. </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence />
    </extension>
  </complexContent>
</complexType>
```

complexType **RemoteException**
**XML Schema**

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Diagram of RemoteException and Message]</td>
</tr>
</tbody>
</table>

- **namespace**: http://www.somewhere.org/Common
- **type**: extension of co:BaseException
- **children**: `Message`
- **annotation documentation**: The RemoteException is returned when an error occurs during any remote object operation.

```xml
<complexType name="RemoteException">
  <annotation>
    <documentation>The RemoteException is returned when an errors occurs during any remote object operation.</documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```

**complexType** RemoteException
The RemoveException exception is returned at an attempt to remove a collection of one or more managed entity when the XML/JMS interface does not allow the managed entity to be removed. This exception is returned when a collection of one or more managed entity cannot be removed.

```
<complexType name="RemoveException">
  <annotation>
    <documentation>
The RemoveException exception is returned at an attempt to remove a collection of one or more managed entity when the XML/JMS interface does not allow the managed entity to be removed. This exception is returned when a collection of one or more managed entity cannot be removed</documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```
The SetException exception is returned at an attempt to update a collection of one or more managed entity when the XML/JMS interface does not allow the managed entity to be updated. This exception is returned when a collection of one or more managed entity cannot be updated.

[source]
<complexType name="SetException">
  <annotation>
    <documentation> The SetException exception is returned at an attempt to update a collection of one or more managed entity when the XML/JMS interface does not allow the managed entity to be updated. This exception is returned when a collection of one or more managed entity cannot be updated. </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
The UnsupportedOperationException exception is returned by the OSS through Java XML/JMS interface to report that the invoked request could not be completed because it is not implemented.

```xml
<complexType name="UnsupportedOperationException">
  <annotation>
    <documentation>
      The UnsupportedOperationException exception is returned by the OSS through Java XML/JMS interface to report that the invoked request could not be completed because it is not implemented.
    </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence />
    </extension>
  </complexContent>
</complexType>
```

OSS Service Activation XML Schema

**element** `AbortOrder.Exception`

The following exceptions are returned if an error occurs:

- `IllegalArgumentException`
- `IllegalStateException`
- `RemoteException`
The following exceptions are returned if an error occurs.

```xml
<element name="AbortOrder.Exception">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="IllegalStateException" type="co:IllegalStateException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```
element **AbortOrder.Exception/IllegalArgumentException**

```
namespace http://www.somewhere.org/ServiceActivation
type co:IllegalArgumentException
children Message
source &lt;element name="IllegalArgumentException" type="co:IllegalArgumentException"/&gt;
```

---

element **AbortOrder.Exception/IllegalStateException**

```
namespace http://www.somewhere.org/ServiceActivation
type co:IllegalStateException
children Message
source &lt;element name="IllegalStateException" type="co:IllegalStateException"/&gt;
```

---

element **AbortOrder.Exception/RemoteException**
### XML Schema

<table>
<thead>
<tr>
<th>Diagram</th>
<th>co:RemoteException</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>co:RemoteException</td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
</tbody>
</table>
| source | `<element name="RemoteException" type="co:RemoteException"/>

**element** AbortOrderRequest

<table>
<thead>
<tr>
<th>Diagram</th>
<th>AbortOrderRequest</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>children</td>
<td>OrderKey</td>
</tr>
<tr>
<td>annotation</td>
<td>documentation</td>
</tr>
</tbody>
</table>
| source | `<element name="AbortOrderRequest">
<annotation>
  <documentation>The following request aborts the order and puts the state to ABORTED.</documentation>
</annotation>
<complexType>
  <sequence>
    <element name="OrderKey" type="sa:OrderKey"/>
  </sequence>
</complexType>
</element>`

**element** AbortOrderRequest/OrderKey

---

The following request aborts the order and puts the state to ABORTED.

The `RemoteException` element indicates the error message from the Exception. This is most likely the result from a `Exception.getMessage()` call.
element `AbortOrderResponse`

```
<element name="AbortOrderResponse">
  <annotation>
    <documentation>Only the message header is returned. This is sufficient to indicate that the order has been aborted. </documentation>
  </annotation>
  <complexType>
  </complexType>
</element>
```

**Source**

```
<element name="OrderKey" type="sa:OrderKey"/>
```
### XML Schema

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>used by</td>
<td>complexType OrderValue</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;BaseState&quot; type=&quot;string&quot;/&gt;</code></td>
</tr>
</tbody>
</table>
element CreateOrderByValueException

namespace http://www.somewhere.org/ServiceActivation

children CreateException IllegalArgumentException RemoteException

annotation documentation
This is returned if an exception occurs.

source
<element name="CreateOrderByValueException">
  <annotation>
    <documentation>This is returned if an exception occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="CreateException" type="co:CreateException"/>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>

element CreateOrderByValueException/CreateException

namespace http://www.somewhere.org/ServiceActivation

type co:CreateException

children Message

The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.
<element name="CreateException" type="co:CreateException"/>
element CreateOrderByValueException/IllegalArgumentException

```
namespace http://www.somewhere.org/ServiceActivation
type co:IllegalArgumentException
children Message
source <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
```

element CreateOrderByValueException/RemoteException

```
namespace http://www.somewhere.org/ServiceActivation
type co:RemoteException
children Message
source <element name="RemoteException" type="co:RemoteException"/>
```

element CreateOrderByValueRequest
**XML Schema**

**CreateOrderByValueRequest/OrderValue**

- **namespace**: `http://www.somewhere.org/ServiceActivation`
- **children**: `OrderValue`
- **annotation** documentation:
  Creates a new order object in the system and returns the key for the new object. A single `OrderValue` is the only value passed into the request. The state values in `OrderValue` are ignored, the state is initialized to `STARTED` by the system.

**Source code**:

```
<element name="CreateOrderByValueRequest">
  <annotation>
    <documentation>Creates a new order object in the system and returns the key for the new object. A single OrderValue is the only value passed into the request. The state values in OrderValue are ignored, the state is initialized to STARTED by the system.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderValue" type="sa:OrderValue"/>
    </sequence>
  </complexType>
</element>
```
### XML Schema

**namespace**

http://www.somewhere.org/ServiceActivation

**type**

`sa:OrderValue`

**children**

- `sa:BaseState`
- `OrderKey`
- `ServiceValues`
- `ClientId`
- `sa:Priority`
- `Description`
- `RequestedDeliveryDate`
- `ActualDeliveryDate`
- `OrderDate`

**source**

```xml
<element name="OrderValue" type="sa:OrderValue"/>
```

**element CreateOrderByValueResponse**

**namespace**

http://www.somewhere.org/ServiceActivation

**children**

- `OrderKey`

**annotation**

This returns the OrderKey corresponding to the order which was created in the system.
<table>
<thead>
<tr>
<th>source</th>
</tr>
</thead>
</table>
| <element name="CreateOrderByValueResponse">
|   <annotation>
|     <documentation>This returns the OrderKey corresponding to the order which was created in the system</documentation>
|   </annotation>
|   <complexType>
|     <sequence>
|       <element name="OrderKey" type="sa:OrderKey"/>
|     </sequence>
|   </complexType>
| </element> |
### element `CreateOrderByValueResponse/OrderKey`

This is returned if an exception occurs.

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td><code>sa:OrderKey</code></td>
</tr>
<tr>
<td>children</td>
<td>Domain, PrimaryKey, Type</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;OrderKey&quot; type=&quot;sa:OrderKey&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

### element `CreateOrdersByValuesException`

This is returned if an exception occurs.

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>children</td>
<td>CreateException, RemoteException</td>
</tr>
<tr>
<td>annotation</td>
<td>This is returned if an exception occurs.</td>
</tr>
</tbody>
</table>
<element name="CreateOrdersByValuesException">
  <annotation>
    <documentation>This is returned if an exception occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="CreateException" type="co:CreateException"/>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
XML Schema

**element** CreateOrdersByValuesException/CreateException

```
<element name="CreateException" type="co:CreateException"/>
```

**element** CreateOrdersByValuesException/IllegalArgumentException

```
<element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
```

**element** CreateOrdersByValuesException/RemoteException
XML Schema

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
</table>

```xml
<element name="RemoteException" type="co:RemoteException"/>
```

diagram

namespace http://www.somewhere.org/ServiceActivation

type co:RemoteException

children Message

source

```
This is the same as createOrder except it allows the client to create a number of orders in the system. A list OrderValues is passed into the request, it returns a list of keys, each representing each order object created. The state values for each OrderValue are ignored, the state is initialized to STARTED by the system.
```

documentation

```
This is the same as createOrder except it allows the client to create a number of orders in the system. A list OrderValues is passed into the request, it returns a list of keys, each representing each order object created. The state values for each OrderValue are ignored, the state is initialized to STARTED by the system.
```

<element name="CreateOrdersByValuesRequest">
  <annotation>
    <documentation>This is the same as createOrder except it allows the client to create a number of orders in the system. A list OrderValues is passed into the request. It returns a list of keys, each representing each order object created. The state values for each OrderValue are ignored, the state is initialized to STARTED by the system.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderValue" type="sa:ArrayOfOrderValue"/>
    </sequence>
  </complexType>
</element>
### XML Schema

**element CreateOrdersByValuesRequest/OrderValue**

<table>
<thead>
<tr>
<th>diagram</th>
<th>namespace</th>
<th>type</th>
<th>children</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
<td>sa:ArrayOfOrderValue</td>
<td>Item</td>
<td><code>&lt;element name=&quot;OrderValue&quot; type=&quot;sa:ArrayOfOrderValue&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element CreateOrdersByValuesResponse**

<table>
<thead>
<tr>
<th>diagram</th>
<th>namespace</th>
<th>children</th>
<th>annotation</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2.png" alt="Diagram" /></td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
<td>OrderKey</td>
<td>This returns a list of OrderKeys corresponding to each order which was created in the system</td>
<td><code>&lt;element name=&quot;CreateOrdersByValuesResponse&quot;&gt;</code>&lt;annotation&gt;&lt;documentation&gt;This returns a list of OrderKeys corresponding to each order which was created in the system&lt;/documentation&gt;<code>&lt;complexType&gt;&lt;sequence&gt;&lt;element name=&quot;OrderKey&quot; type=&quot;sa:ArrayOfOrderKey&quot;/&gt;&lt;/sequence&gt;&lt;/complexType&gt;</code>&lt;element&gt;`</td>
</tr>
</tbody>
</table>

**element CreateOrdersByValuesResponse/OrderKey**
The following exceptions are returned if an error occurs.

```
<element name="GetOrderByKeyException">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```

```
<element name="GetOrderByKeyException/IllegalArgumentException">
  <complexType>
    <choice>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
    </choice>
  </complexType>
</element>
```

```
<element name="GetOrderByKeyException/RemoteException">
  <complexType>
    <choice>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```
XML Schema

namespace http://www.somewhere.org/ServiceActivation

type co:IllegalArgumentException

children Message

source
<element name="IllegalArgumentException" type="co:IllegalArgumentException"/>

element GetOrderByKeyException/RemoteException

namespace http://www.somewhere.org/ServiceActivation

type co:RemoteException

children Message

source
<element name="RemoteException" type="co:RemoteException"/>

element GetOrderByKeyRequest


**XML Schema**

**namespace**  
http://www.somewhere.org/ServiceActivation

**children**  
OrderKey, AttributeName

**annotation**  
Returns values for the order identified by the (unique) key. The value object returned will always have a type that is derived from OrderValue(polymorphism). The attributeName list indicates which attributes are to be returned. If the AttributeName is null then all possible attributes are returned.

**source**  
```xml
<element name="GetOrderByKeyRequest">
    <annotation>
        <documentation>Returns values for the order identified by the (unique) key. The value object returned will always have a type that is derived from OrderValue(polymorphism). The attributeName list indicates which attributes are to be returned. If the AttributeName is null then all possible attributes are returned.</documentation>
    </annotation>
    <complexType>
        <sequence>
            <element name="OrderKey" type="sa:OrderKey"/>
            <element name="AttributeName" type="co:ArrayOfString" nullable="true" minOccurs="0"/>
        </sequence>
    </complexType>
</element>
```

**element**  
GetOrderByKeyRequest/OrderKey
**XML Schema**

<table>
<thead>
<tr>
<th>element</th>
<th>GetOrderByKeyRequest/AttributeName</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>co:ArrayOfString</td>
</tr>
<tr>
<td>children</td>
<td>Item</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;AttributeName&quot; type=&quot;co:ArrayOfString&quot; nullable=&quot;true&quot; minOccurs=&quot;0&quot;/&gt;</td>
</tr>
</tbody>
</table>

**element GetOrderByKeyResponse**

<table>
<thead>
<tr>
<th>annotation</th>
<th>The value for the order is returned.</th>
</tr>
</thead>
<tbody>
<tr>
<td>documentation</td>
<td>The value for the order is returned.</td>
</tr>
</tbody>
</table>
<element name="GetOrderByKeyResponse">
  <annotation>
    <documentation> The value for the order is returned. </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderValue" type="sa:OrderValue"/>
    </sequence>
  </complexType>
</element>
**XML Schema**

**element** GetOrderByKeyResponse/OrderValue

```xml
<element name="OrderValue" type="sa:OrderValue"/>
```

**namespace** http://www.somewhere.org/ServiceActivation

**type** sa:OrderValue

**children** sa:BaseState OrderKey ServiceValues ClientId sa:Priority Description RequestedDeliveryDate ActualDeliveryDate OrderDate

**source**

```
GetOrderByKeyResponse

namespace http://www.somewhere.org/ServiceActivation

type sa:OrderValue

children sa:BaseState OrderKey ServiceValues ClientId sa:Priority Description RequestedDeliveryDate ActualDeliveryDate OrderDate

source <element name="OrderValue" type="sa:OrderValue"/>

**element** GetOrdersByKeysException

```xml
GetOrdersByKeysException

 namespace http://www.somewhere.org/ServiceActivation

children IllegalArgumentException RemoteException
```

The following exceptions are returned if an error occurs.
The following exceptions are returned if an error occurs.

<table>
<thead>
<tr>
<th>annotation</th>
<th>documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;element name=&quot;GetOrdersByKeysException&quot;&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;annotation&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;documentation&gt;The following exceptions are returned if an error occurs.&lt;/documentation&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/annotation&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;complexType&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;choice&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;element name=&quot;IllegalArgumentException&quot; type=&quot;co:IllegalArgumentException&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;element name=&quot;RemoteException&quot; type=&quot;co:RemoteException&quot;/&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/choice&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/complexType&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/element&gt;</td>
<td></td>
</tr>
</tbody>
</table>
element GetOrdersByKeysException/IllegalArgumentException

```
namespace http://www.somewhere.org/ServiceActivation
type co:IllegalArgumentException
children Message
source
<element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
```

element GetOrdersByKeysException/RemoteException

```
namespace http://www.somewhere.org/ServiceActivation
type co:RemoteException
children Message
source
<element name="RemoteException" type="co:RemoteException"/>
```

element GetOrdersByKeysRequest
**GetOrdersByKeysRequest**

This is the plural version of GetOrder. It takes in a list of order keys.

```xml
<element name="GetOrdersByKeysRequest">
  <annotation>
    <documentation>This is the plural version of GetOrder. It takes in a list of order keys.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderKey" type="sa:ArrayOfOrderKey"/>
      <element name="AttributeNames" type="co:ArrayOfString" nullable="true" minOccurs="0"/>
    </sequence>
  </complexType>
</element>
```

**OrderKey**

```xml
<element name="OrderKey" type="sa:ArrayOfOrderKey"/>
```

**AttributeNames**

```xml
<element name="AttributeNames" type="co:ArrayOfString" nullable="true" minOccurs="0"/>
```
**XML Schema**

**element** GetOrdersByKeysResponse

```
<element name="GetOrdersByKeysResponse">  
  <annotation>  
    <documentation>A list of order values is returned.</documentation>  
  </annotation>  
  <complexType>  
    <sequence>  
      <element name="OrderValue" type="sa:ArrayOfOrderValue"/>  
    </sequence>  
  </complexType>  
</element>
```

**element** GetOrdersByKeysResponse/OrderValue

```
<element name="OrderValue" type="sa:ArrayOfOrderValue"/>  
```
### XML Schema

**Namespace**

http://www.somewhere.org/ServiceActivation

**Type**

`sa:ArrayOfOrderValue`

**Children**

- `Item`

**Source**

```xml
<element name="OrderValue" type="sa:ArrayOfOrderValue"/>
```

### Element `GetOrdersByTemplatesException`

**Namespace**

http://www.somewhere.org/ServiceActivation

**Children**

- `UnsupportedOperationException`
- `IllegalArgumentException`
- `RemoteException`

**Annotation**

The following exceptions are returned if an error occurs.

**Source**

```xml
<element name="GetOrdersByTemplatesException">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="UnsupportedOperationException" type="co:UnsupportedOperation-Exception"/>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```
### Element GetOrdersByTemplatesException/UnsupportedOperationException

```
<element name="UnsupportedOperationException" type="co:UnsupportedOperationException"/>
```

<table>
<thead>
<tr>
<th>Diagram</th>
<th><code>co:UnsupportedOperationException</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td><code>co:UnsupportedOperationException</code></td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;UnsupportedOperationException&quot; type=&quot;co:UnsupportedOperationException&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

### Element GetOrdersByTemplatesException/IllegalArgumentException

```
<element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
```

<table>
<thead>
<tr>
<th>Diagram</th>
<th><code>co:IllegalArgumentException</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td><code>co:IllegalArgumentException</code></td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;IllegalArgumentException&quot; type=&quot;co:IllegalArgumentException&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

### Element GetOrdersByTemplatesException/RemoteException
element GetOrdersByTemplatesRequest

This is an associative lookup which will return values for the order, identified by the OrderValue template. The value object returned will always have a type that is derived from OrderValue (polymorphism).

The AttributeName list indicates which attributes are to be returned. If the attributeName is null then all possible attributes are returned. The HowMany element (from IterateRequest) is used to restrict the returning result set. If the result set exceeds the "HowMany" value then consequent response messages are returned.

The HowMany element indicates how many result instances should be returned in the Response message. If the value supplied is greater than the complete result set then just the result set is returned in the response message, and EndOfReply element will be set to 'true'. If the HowMany element exceeds a preset application maximum (for performance reasons) then the number of instances returned will equal the preset application maximum. Not specifying this element will return the entire result set.
### XML Schema

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>extension of co:IteratorRequest</td>
</tr>
<tr>
<td>children</td>
<td>HowMany Template AttributeNames</td>
</tr>
<tr>
<td>annotation</td>
<td>This is an associative lookup which will return values for the order, identified by the OrderValue template. The value object returned will always have a type that is derived from OrderValue (polymorphism). The AttributeName list indicates which attributes are to be returned. If the attributeName is null then all possible attributes are returned. The HowMany element (from IteratorRequest) is used to restrict the returning result set. If the result set exceeds the &quot;HowMany&quot; value than consequent response messages are returned.</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;GetOrdersByTemplatesRequest&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;annotation&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;documentation&gt;This is an associative lookup which will return values for the order, identified by the OrderValue template. The value object returned will always have a type that is derived from OrderValue (polymorphism). The AttributeName list indicates which attributes are to be returned. If the attributeName is null then all possible attributes are returned. The HowMany element (from IteratorRequest) is used to restrict the returning result set. If the result set exceeds the &quot;HowMany&quot; value than consequent response messages are returned.&lt;/documentation&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/annotation&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;complexType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;complexContent&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;extension base=&quot;co:IteratorRequest&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;sequence&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;element name=&quot;Template&quot; type=&quot;sa:ArrayOfOrderValue&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;element name=&quot;AttributeNames&quot; type=&quot;co:ArrayOfString&quot; nullable=&quot;true&quot; minOccurs=&quot;0&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/sequence&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/extension&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/complexContent&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/complexType&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/element&gt;</td>
</tr>
</tbody>
</table>
**element GetOrdersByTemplatesRequest/Template**

<table>
<thead>
<tr>
<th>Diagram</th>
<th>sa:ArrayOfOrderValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>sa:ArrayOfOrderValue</td>
</tr>
<tr>
<td>children</td>
<td>Item</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;Template&quot; type=&quot;sa:ArrayOfOrderValue&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element GetOrdersByTemplatesRequest/AttributeNames**

<table>
<thead>
<tr>
<th>Diagram</th>
<th>co:ArrayOfString</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>co:ArrayOfString</td>
</tr>
<tr>
<td>children</td>
<td>Item</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;AttributeNames&quot; type=&quot;co:ArrayOfString&quot; nullable=&quot;true&quot; minOccurs=&quot;0&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element GetOrdersByTemplatesResponse**
GetOrdersByTemplatesResponse

The result set is a list (the amount in the list is determined by the HowMany element in the IteratorRequest type) of orderValues. The EndOfReply (from IteratorResponse) indicates whether there are any more result sets being returned. The Sequence number (from IteratorResponse) indicates the result set order, since there can be a number of response messages generated and there is no mechanism to ensure the responses are sent sequentially, there needs to be a method of preserving the order of the response messages.

namespace http://www.somewhere.org/ServiceActivation

type extension of co:IteratorResponse

children Sequence EndOfReply OrderValue

annotation documentation

The result set is a list (the amount in the list is determined by the HowMany element in the IteratorRequest type) of orderValues. The EndOfReply (from IteratorResponse) indicates whether there are any more result sets being returned. The Sequence number (from IteratorResponse) indicates the result set order, since there can be a number of response messages generated and there is no mechanism to ensure the responses are sent sequentially, there needs to be a method of preserving the order of the response messages.
<element name="GetOrdersByTemplatesResponse">
  <annotation>
    <documentation>The result set is a list (the amount in the list is determined by the How-Many element in the IteratorRequest type) of orderValues. The EndOfReply (from IteratorResponse) indicates whether there are any more result sets being returned. The Sequence number (from IteratorResponse) indicates the result set order, since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of preserving the order of the response messages.</documentation>
  </annotation>
  <complexType>
    <complexContent>
      <extension base="co:IteratorResponse">
        <sequence>
          <element name="OrderValue" type="sa:ArrayOfOrderValue" nullable="true" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
element `GetOrdersByTemplatesResponse/OrderValue`

<table>
<thead>
<tr>
<th>diagram</th>
<th>sa:ArrayOfOrderValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>sa:ArrayOfOrderValue</td>
</tr>
<tr>
<td>children</td>
<td>Item</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;OrderValue&quot; type=&quot;sa:ArrayOfOrderValue&quot; nullable=&quot;true&quot; minOccurs=&quot;0&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element `GetOrderTypesException`**

<table>
<thead>
<tr>
<th>diagram</th>
<th>RemoteException</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>children</td>
<td>RemoteException</td>
</tr>
<tr>
<td>annotation</td>
<td>documentation</td>
</tr>
</tbody>
</table>
| source | `<element name="GetOrderTypesException"> 
  <annotation>
  <documentation>The GetOrderException is returned if an exception occurred.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>` |
element `GetOrderTypesException/RemoteException`

```
<element name="RemoteException" type="co:RemoteException"/>
```

**Diagram:**

- `co:RemoteException` is used to return a string list of available OrderTypes. Each one of the strings in the list can be passed as an argument to `newOrderValue(factory)` to create a corresponding `OrderValue` instance.

**Namespace:**

http://www.somewhere.org/ServiceActivation

**Type:**

`co:RemoteException`

**Children:**

- `Message`

**Source:**

```
<element name="RemoteException" type="co:RemoteException"/>
```

---

**element `GetOrderTypesRequest`**

```
<element name="GetOrderTypesRequest" type="co:RemoteException"/>
```

**Diagram:**

- `GetOrderTypesRequest` is used to return a string list of available OrderTypes. Each one of the strings in the list can be passed as an argument to `newOrderValue(factory)` to create a corresponding `OrderValue` instance.

**Namespace:**

http://www.somewhere.org/ServiceActivation

**Annotation Documentation:**

- The `GetOrderTypesRequest` is used to return a string list of available OrderTypes. Each one of the strings in the list can be passed as an argument to `newOrderValue(factory)` to create a corresponding `OrderValue` instance.
<element name="GetOrderTypesRequest">
  <annotation>
    <documentation>The GetOrderTypesRequest is used to return a string list of available OrderTypes. Each one of the strings in the list can be passed as an argument to newOrderValue(factory) to create a corresponding OrderValue instance.</documentation>
  </annotation>
  <complexType>
    <sequence/>
  </complexType>
</element>
element GetOrderTypesResponse

```
<element name="GetOrderTypesResponse">
  <annotation>
    <documentation>The GetOrderTypesResponse will return a list of available OrderTypes.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderType" type="co:ArrayOfString" minOccurs="0"/>
    </sequence>
  </complexType>
</element>
```

element GetOrderTypesResponse/OrderType

```
<element name="OrderType" type="co:ArrayOfString" minOccurs="0"/>
```

element GetQueryTypesException
**XML Schema**

<table>
<thead>
<tr>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

The GetQueryTypesException is returned if an exception occurred.

**Namespace**

- `http://www.somewhere.org/ServiceActivation`

**Children**

- `RemoteException`

**Annotation**

- The GetQueryTypesException is returned if an exception occurred.

**Source**

```
<element name="GetQueryTypesException">
  <annotation>
    <documentation>The GetQueryTypesException is returned if an exception occurred.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```

**Element GetQueryTypesException/RemoteException**

<table>
<thead>
<tr>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

The Message element indicates the error message from the Exception. This is most likely the result from a `Exception.getMessage()` call.

**Namespace**

- `http://www.somewhere.org/ServiceActivation`

**Type**

- `co:RemoteException`

**Children**

- `Message`

**Source**

```
<element name="RemoteException" type="co:RemoteException"/>
```
The GetQueryTypesRequest will return a string list of available query types. Each one of the Query Types can then be passed to the newQueryValue(factory) to create a corresponding QueryValue instance.

```
<element name="GetQueryTypesRequest">
  <annotation>
    <documentation>The GetQueryTypesRequest will return a string list of available query types. Each one of the Query Types can then be passed to the newQueryValue(factory) to create a corresponding QueryValue instance.</documentation>
  </annotation>
  <complexType>
    <sequence/>
  </complexType>
</element>
```

The GetQueryTypesResponse will contain the list of available Query Types.

```
<element name="GetQueryTypesResponse">
  <complexType>
    <sequence>
      <element name="QueryType"/>
    </sequence>
  </complexType>
</element>
```
<element name="GetQueryTypesResponse">
  <annotation>
    <documentation>The GetQueryTypesResponse will contain the list of available Query Types.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="QueryType" type="co:ArrayOfString" minOccurs="0"/>
    </sequence>
  </complexType>
</element>
**element GetQueryTypesResponse/QueryType**

```
<element name="QueryType" type="co:ArrayOfString" minOccurs="0"/>
```

**element GetServiceTypes.Response**

```
<element name="GetServiceTypes.Response">
  <annotation>
    <documentation>The GetServiceTypesResponse will contain the list of available Service Types.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="ServiceType" type="co:ArrayOfString" minOccurs="0"/>
    </sequence>
  </complexType>
</element>
```
**XML Schema**

**element** `GetServiceTypes.Response/ServiceType`

<table>
<thead>
<tr>
<th>diagram</th>
<th><code>ServiceType</code></th>
<th><code>Item</code></th>
<th><code>1..*</code></th>
</tr>
</thead>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| type      | `co:ArrayOfString` |
| children  | Item |
| source    | `<element name="ServiceType" type="co:ArrayOfString" minOccurs="0"/>` |

**element** `GetServiceTypesException`

<table>
<thead>
<tr>
<th>diagram</th>
<th><code>GetServiceTypesException</code></th>
<th><code>RemoteException</code></th>
</tr>
</thead>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| children  | `RemoteException` |
| annotation | The `GetServiceTypesException` is returned if an exception occurred. |
| source    | `<element name="GetServiceTypesException">
  <annotation>
    <documentation>The `GetServiceTypesException` is returned if an exception occurred.</documentation>
  </annotation>
  `<complexType>
    <choice>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
  </element>` |

**element** `GetServiceTypesException/RemoteException`
The GetServiceTypesRequest is used to return a string list of available Service Types. Each one of the Service Types returned can then be passed as an argument to newServiceValue(factory) to create a corresponding ServiceValue.

The Message element indicates the error message from the Exception. This is most likely the result from a Exception.getMessage() call.

```xml
<element name="RemoteException" type="co:RemoteException"/>
```
<element name="GetServiceTypesRequest">
  <annotation>
    <documentation>The GetServiceTypesRequest is used to return a string list of available Service Types. Each one of the Service Types returned can then be passed as an argument to newServiceValue(factory) to create a corresponding ServiceValue.</documentation>
  </annotation>
  <complexType>
    <sequence/>
  </complexType>
</element>
element **GetSupportedOperationsException**

```
<element name="GetSupportedOperationsException">
  <annotation>
    <documentation>
      The GetSupportedOperationsException is returned if an exception occurred during the request. The only exception that is thrown is a RemoteException.
    </documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```

**namespace**  http://www.somewhere.org/ServiceActivation

**children**  RemoteException

**annotation**

The GetSupportedOperationsException is returned if an exception occurred during the request. The only exception that is thrown is a RemoteException.

---

element **GetSupportedOperationsException/RemoteException**

```
RemoteException
```

**diagram**

The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.

---

```
<complexType>
  <sequence>
    <element name="Message" type="co:Message"/>
  </sequence>
</complexType>
```

```
<complexType>
  <simpleContent>
    <extension base="co:RemoteException">
      <attribute name="message" type="co:Message"/>
    </extension>
  </simpleContent>
</complexType>
```
<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>co:RemoteException</td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;RemoteException&quot; type=&quot;co:RemoteException&quot;/&gt;</td>
</tr>
</tbody>
</table>
element GetSupportedOperationsRequest

- namespace http://www.somewhere.org/ServiceActivation
- annotation
  - documentation
    The GetSupportedOperationsRequest is used to return a list of the implemented optional operations. It will return a subset of the OrderManagerOption enumeration, indicating which of the optional operations are implemented.
    - Postcondition: Every returned string in the GetSupportedOperationsResponse Message, can be one mentioned in the OrderManagerOption.

source

```
<element name="GetSupportedOperationsRequest">
  <annotation>
    <documentation> The GetSupportedOperationsRequest is used to return a list of the implemented optional operations. It will return a subset of the OrderManagerOption enumeration, indicating which of the optional operations are implemented.
    - Postcondition: Every returned string in the GetSupportedOperationsResponse Message, can be one mentioned in the OrderManagerOption. </documentation>
  </annotation>
  <complexType>
    <sequence/>
  </complexType>
</element>
```

element GetSupportedOperationsResponse

- namespace http://www.somewhere.org/ServiceActivation

- diagram
  - The GetSupportedOperationsResponse will return a list of implemented optional operations.
The GetSupportedOperationsResponse will return a list of implemented optional operations.

```xml
<element name="GetSupportedOperationsResponse">
  <annotation>
    <documentation>The GetSupportedOperationsResponse will return a list of implemented optional operations</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderManagerOption" type="co:ArrayOfString" minOccurs="0"/>
    </sequence>
  </complexType>
</element>
```
element GetSupportedOperationsResponse/OrderManagerOption

```
<element name="OrderManagerOption" type="co:ArrayOfString" minOccurs="0"/>
```

diagram

```
OrderManagerOption  co:ArrayOfString
  Item
  1..*  
```

namespace http://www.somewhere.org/ServiceActivation
type co:ArrayOfString
children Item
source <element name="OrderManagerOption" type="co:ArrayOfString" minOccurs="0"/>

element NewOrderValueException

```
<element name="NewOrderValueException">  
  <annotation>  
    <documentation>This is returned if an exception occurs.</documentation>  
  </annotation>  
  <complexType>  
    <choice>  
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>  
      <element name="RemoteException" type="co:RemoteException"/>  
    </choice>  
  </complexType>  
</element>
```

```
NewOrderValueException  IllegalArgumentException  RemoteException
  This is returned if an exception occurs.
```

namespace http://www.somewhere.org/ServiceActivation
children IllegalArgumentException RemoteException
annotation documentation
source <element name="NewOrderValueException">  
  <annotation>  
    <documentation>This is returned if an exception occurs.</documentation>  
  </annotation>  
  <complexType>  
    <choice>  
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>  
      <element name="RemoteException" type="co:RemoteException"/>  
    </choice>  
  </complexType>  
</element>

element NewOrderValueException/IllegalArgumentException
element NewOrderValueException/RemoteException

namespace http://www.somewhere.org/ServiceActivation

type co:IllegalArgumentException

children Message

source <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>

element NewOrderValueRequest
This request will return a new Order Value for usage in a client. The type of Order Value returned is based on the orderType attribute populated in this request. This request does not create a "real" order. It only supplies to the client an empty Order Value (i.e. not populated with data). The client would then make a createOrder.Request with that Order Value populated with appropriate values.
<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;OrderType&quot; type=&quot;string&quot;/&gt;</code></td>
</tr>
</tbody>
</table>
element *NewOrderValueResponse*

<table>
<thead>
<tr>
<th>diagram</th>
<th>NewOrderValueResponse ➔ OrderValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>children</td>
<td>OrderValue</td>
</tr>
<tr>
<td>annotation</td>
<td>This returns the empty OrderValue</td>
</tr>
</tbody>
</table>

source

```xml
<element name="NewOrderValueResponse">
    <annotation>
        <documentation>This returns the empty OrderValue</documentation>
    </annotation>
    <complexType>
        <sequence>
            <element name="OrderValue" type="sa:OrderValue"/>
        </sequence>
    </complexType>
</element>
```

element *NewOrderValueResponse/OrderValue*
**XML Schema**

```
<element name="OrderValue" type="sa:OrderValue"/>
```

**element NewQueryValueException**

```
<element name="NewQueryValueException">
  <illegalArgumentException/>
  <remoteException/>
</element>
```

**Source code**

```
namespace http://www.somewhere.org/ServiceActivation

type sa:OrderValue

children sa:BaseState OrderKey ServiceValues ClientId sa:Priority Description RequestedDeliveryDate ActualDeliveryDate OrderDate

source <element name="OrderValue" type="sa:OrderValue"/>
```

```
namespace http://www.somewhere.org/ServiceActivation

children IllegalArgumentException RemoteException

annotation This is returned if an exception occurs.
```
source  

```xml
<element name="NewQueryValueException">
  <annotation>
    <documentation>This is returned if an exception occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```
element NewQueryValueException/IllegalArgumentException

namespace http://www.somewhere.org/ServiceActivation

type co:IllegalArgumentException

children Message

source
<element name="IllegalArgumentException" type="co:IllegalArgumentException"/>

element NewQueryValueException/RemoteException

namespace http://www.somewhere.org/ServiceActivation

type co:RemoteException

children Message

source
<element name="RemoteException" type="co:RemoteException"/>

element NewQueryValueRequest
This request will return a new QueryValue for usage in a client. The type of QueryValue returned is based on the queryType element populated in this request. It supplies the client with an empty QueryValue (i.e. not populated with data). The client would then make a QueryOrderRequest with the QueryValue populated with data.

```
<element name="NewQueryValueRequest">
  <annotation>
    <documentation>
      This request will return a new QueryValue for usage in a client. The type of QueryValue returned is based on the queryType element populated in this request. It supplies the client with an empty QueryValue (i.e. not populated with data). The client would then make a QueryOrderRequest with the QueryValue populated with data.
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="QueryType" type="string"/>
    </sequence>
  </complexType>
</element>
```
element **NewQueryValueResponse**

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
</tr>
<tr>
<td>This returns the empty ServiceValue</td>
</tr>
</tbody>
</table>

**namespace** http://www.somewhere.org/ServiceActivation

**children**  
- **QueryValue**

**annotation**  
This returns the empty ServiceValue

**source**

```xml
<element name="NewQueryValueResponse">
  <annotation>
    <documentation>This returns the empty ServiceValue</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="QueryValue" type="sa:QueryValue"/>
    </sequence>
  </complexType>
</element>
```

element **NewQueryValueResponse/QueryValue**

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**namespace** http://www.somewhere.org/ServiceActivation

**type** `sa:QueryValue`

**source**

```xml
<element name="QueryValue" type="sa:QueryValue"/>
```

element **NewServiceValueException**
### XML Schema

**element** `NewServiceValueException` / `IllegalArgumentException`

- **namespace**: `http://www.somewhere.org/ServiceActivation`
- **children**: `IllegalArgumentException` `RemoteException`
- **annotation**
  ```xml
  <element name="NewServiceValueException">
    <annotation>
      <documentation>This is returned if an exception occurs.</documentation>
    </annotation>
    <complexType>
      <choice>
        <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
        <element name="RemoteException" type="co:RemoteException"/>
      </choice>
    </complexType>
  </element>
  ```

---

**element** `IllegalArgumentException`

- **namespace**: `http://www.somewhere.org/ServiceActivation`
- **type**: `co:IllegalArgumentException`
- **children**: `Message`
- **source**
  ```xml
  <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
  ```
**element** NewServiceValueException/RemoteException

<table>
<thead>
<tr>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**namespace** http://www.somewhere.org/ServiceActivation

**type** co:RemoteException

**children** Message

**source**

```xml
<element name="RemoteException" type="co:RemoteException"/>
```

**element** NewServiceValueRequest

<table>
<thead>
<tr>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**namespace** http://www.somewhere.org/ServiceActivation

**children** ServiceType

**annotation**

This request will return a new Service Value for usage in a client. The type of Service Value returned is based on the serviceType attribute populated in this request. This request does not create a "real" service. It only supplies to the client an empty ServiceValue (i.e., not populated with data). The client would then add this ServiceValue (populated with data) to an OrderValue and make a CreateOrderRequest.
<element name="NewServiceValueRequest">
  <annotation>
    <documentation>This request will return a new Service Value for usage in a client. The type of Service Value returned is based on the serviceType attribute populated in this request. This request does not create a "real" service. It only supplies to the client an empty ServiceValue (i.e. not populated with data). The client would then add this ServiceValue (populated with data) to an OrderValue and make a CreateOrderRequest.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="ServiceType" type="string"/>
    </sequence>
  </complexType>
</element>
**XML Schema**

**element** `NewServiceValueRequest/ServiceType`

```
<table>
<thead>
<tr>
<th>diagram</th>
<th><code>ServiceType</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;ServiceType&quot; type=&quot;string&quot;/&gt;</code></td>
</tr>
</tbody>
</table>
```

**element** `NewServiceValueResponse`

```
<table>
<thead>
<tr>
<th>diagram</th>
<th><code>NewServiceValueResponse</code> — <code>ServiceValue</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>children</td>
<td><code>ServiceValue</code></td>
</tr>
<tr>
<td>annotation</td>
<td>This returns the empty ServiceValue</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;NewServiceValueResponse&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;annotation&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;documentation&gt;</code>This returns the empty ServiceValue&lt;/documentation&gt;`</td>
</tr>
<tr>
<td></td>
<td><code>&lt;/annotation&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;complexType&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;sequence&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;element name=&quot;ServiceValue&quot; type=&quot;sa:ServiceValue&quot;/&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/sequence&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/complexType&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/element&gt;</code></td>
</tr>
</tbody>
</table>
```

**element** `NewServiceValueResponse/ServiceValue`
element OrderAttributeValueChangeEvent

An Event that is sent in the case in which some value of the order has changed. The event is thrown in the case of 1) set-Order(s).Request has been called and changed the value of an order 2) Some values of the order have been changed during the execution of an order by implementation. This event is not published if the state of the order has changed, that occurrence is handled by the OrderStateChangeEvent.
<element name="OrderAttributeValueChangeEvent">
  <annotation>
    <documentation>An Event that is sent in the case in which some value of the order has changed. The event is thrown in the case of 1) setOrder(s).Request has been called and changed the value of an order 2) Some values of the order have been changed during the execution of an order by implementation. This event is not published if the state of the order has changed, that occurrence is handled by the OrderStateChangeEvent.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="Event" type="sa:OrderAttributeValueChangeEventType" nul-lable="false"/>
    </sequence>
  </complexType>
</element>
element **OrderAttributeValueChangeEvent/Event**

```xml
<element name="Event" type="sa:OrderAttributeValueChangeEventType" nullable="false"/>
```

**namespace** http://www.somewhere.org/ServiceActivation

**type** `sa:OrderAttributeValueChangeEventType`

**children** `EventType EventTime Domain NotificationId SourceIndicator OrderValue`

**source**

An Event that is published in the case where a new order is created. The event is thrown when `createOrder(s).Request` has been called.

---

**element** **OrderCreateEvent**

```xml
Event
```

**namespace** http://www.somewhere.org/ServiceActivation

**children** `Event`

**annotation**

An Event that is published in the case where a new order is created. The event is thrown when `createOrder(s).Request` has been called.
<element name="OrderCreateEvent">
  <annotation>
    <documentation>An Event that is published in the case where a new order is created. The event is thrown when createOrder(s).Request has been called.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="Event" type="sa:OrderCreateEventType" nullable="false"/>
    </sequence>
  </complexType>
</element>
**element** OrderCreateEvent/Event

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="XML Schema Diagram" /></td>
</tr>
</tbody>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| type | sa:OrderCreateEventType |
| children | EventType EventTime Domain NotificationId SourceIndicator OrderValue |
| source | <element name="Event" type="sa:OrderCreateEventType" nullable="false"/> |

**element** OrderRemoveEvent

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="XML Schema Diagram" /></td>
</tr>
</tbody>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| children | Event |

An Event that is published in the case where an order has been removed. The event is thrown when 1) removeOrder(s).Request has been called, 2) implementation may also throw this exception if it supports an automatic cleanup functionality to remove, for example, all completed orders.
<table>
<thead>
<tr>
<th>Annotation</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An Event that is published in the case where an order has been removed. The event is thrown when 1) removeOrder(s).Request has been called. 2) Implementation may also throw this exception if it supports an automatic cleanup functionality to remove, for example, all completed orders.</td>
</tr>
</tbody>
</table>

```xml
<element name="OrderRemoveEvent">
  <annotation>
    <documentation>An Event that is published in the case where an order has been removed. The event is thrown when 1) removeOrder(s).Request has been called. 2) Implementation may also throw this exception if it supports an automatic cleanup functionality to remove, for example, all completed orders.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="Event" type="sa:OrderRemoveEventType" nullable="false"/>
    </sequence>
  </complexType>
</element>
```
**element OrderRemoveEvent/Event**

| namespace | http://www.somewhere.org/ServiceActivation |
| type       | sa:OrderRemoveEventType                 |
| children   | EventType EventTime Domain NotificationId SourceIndicator OrderValue |
| source     | <element name="Event" type="sa:OrderRemoveEventType" nullable="false"/> |

**element OrderStateChangeEvent**

| namespace | http://www.somewhere.org/ServiceActivation |
| children   | Event                                      |
| annotation | An Event that is published in the case where the state of an order has changed. The event is thrown when 1) One of the Suspend/Resume/Start/Abort operations are called 2) While executing the order, the state of the order has been changed by the implementation |
An Event that is published in the case where the state of an order has changed. The event is thrown when 1) One of the Suspend/Resume/Start/Abort operations are called 2) While executing the order, the state of the order has been changed by the implementation.
element `OrderStateChangeEvent/Event`

```
<element name="Event" type="sa:OrderStateChangeEventType" nullable="false"/>
```

namespace `http://www.somewhere.org/ServiceActivation`
type `sa:OrderStateChangeEventType`
children `EventType EventTime Domain NotificationId SourceIndicator OrderValue`
source

---

element `Priority`

```
<element name="Priority" type="sa:OrderPriority"/>
```

namespace `http://www.somewhere.org/ServiceActivation`
type `sa:OrderPriority`
used by `OrderValue`
facets minInclusive 1 maxInclusive 5
source
**XML Schema**

**element QueryOrdersException**

```
<element name="QueryOrdersException">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```

**element QueryOrdersException/IllegalArgumentException**

```
<element name="IllegalArgumentException">
  <complexType>
    <annotation>
      <documentation>The Message element indicates the error message from the Exception. This is most likely the result from a Exception.getMessage() call.</documentation>
    </annotation>
    <attribute name="message" type="string"/>
  </complexType>
</element>
```
<element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
XML Schema

element QueryOrdersException/RemoteException

diagram

```
  co:RemoteException
    Message

RemoteException
```

- The Message element indicates the error message from the Exception. This is most likely the result from a Exception.getMessage() call.

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>co:RemoteException</td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;RemoteException&quot; type=&quot;co:RemoteException&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

element QueryOrdersRequest
The following request runs a (complex) query and returns the matching orders. The HowMany (from IteratorRequest) element is used to restrict the returning result set. If the result set exceeds the "HowMany" value than consequent response messages are returned.

**namespace** http://www.somewhere.org/ServiceActivation

**type** extension of co:IteratorRequest

**children** HowMany QueryValue AttributeNames

**annotation**

The following request runs a (complex) query and returns the matching orders. The HowMany (from IteratorRequest) element is used to restrict the returning result set. If the result set exceeds the "HowMany" value than consequent response messages are returned.
<element name="QueryOrdersRequest">
  <annotation>
    <documentation>The following request runs a (complex) query and returns the matching orders. The HowMany (from IteratorRequest) element is used to restrict the returning result set. If the result set exceeds the "HowMany" value than consequent response messages are returned.</documentation>
  </annotation>
  <complexType>
    <complexContent>
      <extension base="co:IteratorRequest">
        <sequence>
          <element name="QueryValue" type="sa:QueryValue"/>
          <element name="AttributeNames" type="co:ArrayOfString" nullable="true" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
### element `QueryOrdersRequest/QueryValue`

<table>
<thead>
<tr>
<th>Diagram</th>
<th><code>sa:QueryValue</code></th>
</tr>
</thead>
</table>

- **Namespace**: `http://www.somewhere.org/ServiceActivation`
- **Type**: `sa:QueryValue`
- **Source**:
  ```xml```
  <element name="QueryValue" type="sa:QueryValue"/>
  ```xml```

### element `QueryOrdersRequest/AttributeNames`

<table>
<thead>
<tr>
<th>Diagram</th>
<th><code>co:ArrayOfString</code></th>
</tr>
</thead>
</table>

- **Namespace**: `http://www.somewhere.org/ServiceActivation`
- **Type**: `co:ArrayOfString`
- **Children**: `Item`
- **Source**:  
  ```xml```
  <element name="AttributeNames" type="co:ArrayOfString" nullable="true" minOccurs="0"/>
  ```xml```

### element `QueryOrdersResponse`
The result set is a list (the amount in the list is determined by the HowMany element from the QueryOrdersRequest) of OrderValues. The EndOfReply (from IteratorResponse) indicates whether there are any more result sets being returned. The Sequence number (from IteratorResponse) indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.

The EndOfReply indicates whether there are any more result sets being returned.
<element name="QueryOrdersResponse">
  <annotation>
    <documentation>The result set is a list (the amount in the list is determined by the How-Many element from the QueryOrdersRequest) of OrderValues. The EndOfReply (from IteratorResponse) indicates whether there are any more result sets being returned. The Sequence number (from IteratorResponse) indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.</documentation>
  </annotation>
  <complexType>
    <complexContent>
      <extension base="co:IteratorResponse">
        <sequence>
          <element name="OrderValue" type="sa:ArrayOfOrderValue" nullable="true" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</element>
element QueryOrdersResponse/OrderValue

namespace: http://www.somewhere.org/ServiceActivation
type: sa:ArrayOfOrderValue
children: Item
source: <element name="OrderValue" type="sa:ArrayOfOrderValue" nullable="true" minOccurs="0"/>

The following exceptions are returned if an error occurs.

- UnsupportedOperationException
- IllegalArgumentException
- IllegalStateException
- RemoveException
- RemoteException

The following exceptions are returned if an error occurs.
<element name="RemoveOrderByKeyException">
   <annotation>
      <documentation>The following exceptions are returned if an error occurs.</documentation>
   </annotation>
   <complexType>
      <choice>
         <element name="UnsupportedOperationException" type="co:UnsupportedOperationException"/>
         <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
         <element name="IllegalStateException" type="co:IllegalStateException"/>
         <element name="RemoveException" type="co:RemoveException"/>
         <element name="RemoteException" type="co:RemoteException"/>
      </choice>
   </complexType>
</element>
### RemoveOrderByKeyException/UnsupportedOperationException

<table>
<thead>
<tr>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>UnsupportedOperationException</code></td>
</tr>
</tbody>
</table>

| Namespace | http://www.somewhere.org/ServiceActivation |
| Type      | `co:UnsupportedOperationException` |
| Children  | Message |

**Source**

```xml
<element name="UnsupportedOperationException" type="co:UnsupportedOperationException"/>
```

### RemoveOrderByKeyException/IllegalArgumentException

<table>
<thead>
<tr>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IllegalArgumentException</code></td>
</tr>
</tbody>
</table>

| Namespace | http://www.somewhere.org/ServiceActivation |
| Type      | `co:IllegalArgumentException` |
| Children  | Message |

**Source**

```xml
<element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
```

### RemoveOrderByKeyException/IllegalStateException

**Source**

```xml
<element name="IllegalStateException" type="co:IllegalStateException"/>
```
namespace http://www.somewhere.org/ServiceActivation

type co:IllegalStateException

children Message

source <element name="IllegalStateException" type="co:IllegalStateException"/>

element RemoveOrderByKeyException/RemoveException

c:co:RemoveException

children Message

source <element name="RemoveException" type="co:RemoveException"/>

element RemoveOrderByKeyException/RemoteException

source
element RemoveOrderByKeyRequest

The following request will terminate the workflow or an order. This request indicates that the client(s) does not need the referenced order anymore. The implementation can also remove orders automatically, e.g. batch cleanup of completed orders. If the order is removed successfully then a OrderRemoveEvent is published. Exceptions are raised if for example the Order does not exist or if the order is not in an appropriate state (i.e. RUNNING state). If the order is not in an appropriate state then the user would have to abort the order and then call RemoveOrder.

namespace http://www.somewhere.org/ServiceActivation

children OrderKey
The following request will terminate the lifetime or an order. This request indicates that the client(s) does not need the referenced order anymore. The implementation can also remove orders automatically, example batch cleanup of completed orders. If the order is removed successfully then a OrderRemoveEvent is published. Exceptions are returned if for example the Order does not exist or if the order is not in an appropriate state (i.e. RUNNING state). If the order is not in an appropriate state then the user would have to abort the order and then call removeOrder.

source

```xml
<element name="RemoveOrderByKeyRequest">
  <annotation>
    <documentation>The following request will terminate the lifetime or an order. This request indicates that the client(s) does not need the referenced order anymore. The implementation can also remove orders automatically, example batch cleanup of completed orders. If the order is removed successfully then a OrderRemoveEvent is published. Exceptions are returned if for example the Order does not exist or if the order is not in an appropriate state (i.e. RUNNING state). If the order is not in an appropriate state then the user would have to abort the order and then call removeOrder.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderKey" type="sa:OrderKey"/>
    </sequence>
  </complexType>
</element>
```
element **RemoveOrderByKeyRequest/OrderKey**

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>sa:OrderKey</td>
</tr>
<tr>
<td>children</td>
<td>Domain PrimaryKey Type</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;OrderKey&quot; type=&quot;sa:OrderKey&quot;/&gt;</td>
</tr>
</tbody>
</table>

**Diagram**

- **RemoveOrderByKeyRequest**
- **OrderKey**
- **Domain**
- **PrimaryKey**
- **Type**

**Source**

```
<element name="OrderKey" type="sa:OrderKey"/>
```

**element **RemoveOrderByKeyResponse**

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>annotation</td>
<td>Only the message header is returned. This is sufficient to indicate that the order was removed.</td>
</tr>
</tbody>
</table>
| source    | <element name="RemoveOrderByKeyResponse">
            <annotation>
            <documentation>Only the message header is returned. This is sufficient to indicate that the order was removed. </documentation>
            </annotation>
            <complexType>
            <sequence/>
            </complexType>
        </element> |

**Diagram**

- **RemoveOrderByKeyResponse**

**Source**

```
<element name="RemoveOrderByKeyResponse">
    <annotation>
        <documentation>Only the message header is returned. This is sufficient to indicate that the order was removed. </documentation>
    </annotation>
    <complexType>
        <sequence/>
    </complexType>
</element>
```

**element **RemoveOrdersByKeysException**
The following exceptions are returned if an error occurs:

```xml
<element name="RemoveOrdersByKeysException">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="UnsupportedOperationException" type="co:UnsupportedOperationException"/>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="IllegalStateException" type="co:IllegalStateException"/>
      <element name="RemoveException" type="co:RemoveException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```
element RemoveOrdersByKeysException/IllegalArgumentException

```
namespace http://www.somewhere.org/ServiceActivation

type co:IllegalArgumentException

children Message

source <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
```

element RemoveOrdersByKeysException/IllegalStateException

```
namespace http://www.somewhere.org/ServiceActivation

type co:UnsupportedOperationException

children Message

source <element name="UnsupportedOperationException" type="co:UnsupportedOperationException"/>
```
element `RemoveOrdersByKeysException/RemoveException`

```xml
<element name="RemoveException" type="co:RemoveException"/>
```

```
namespace http://www.somewhere.org/ServiceActivation

type co:RemoveException

children Message

source <element name="RemoveException" type="co:RemoveException"/>
```

```
namespace http://www.somewhere.org/ServiceActivation

type co:IllegalStateException

children Message

source <element name="IllegalStateException" type="co:IllegalStateException"/>
```

```xml
<element name="RemoveException" type="co:RemoveException"/>
```

```
namespace http://www.somewhere.org/ServiceActivation

type co:IllegalStateException

children Message

source <element name="IllegalStateException" type="co:IllegalStateException"/>
```

```
namespace http://www.somewhere.org/ServiceActivation

type co:IllegalStateException

children Message

source <element name="IllegalStateException" type="co:IllegalStateException"/>
```

```
namespace http://www.somewhere.org/ServiceActivation

type co:RemoveException

children Message

source <element name="RemoveException" type="co:RemoveException"/>
```
XML Schema

### RemoveOrdersByKeysRequest

**Diagram:**

```
RemoveOrdersByKeysRequest <-> OrderKey
```

**Documentation:**

This is the plural version of the removeOrder request. The only difference is that it takes a list of orders to be removed (indicated by a list of OrderKeys).

**Namespace:**

http://www.somewhere.org/ServiceActivation

**Children:**

OrderKey

**Annotation:**

This is the plural version of the removeOrder request. The only difference is that it takes a list of orders to be removed (indicated by a list of OrderKeys).

---

**Namespace:**

http://www.somewhere.org/ServiceActivation

**Type:**

co:RemoteException

**Children:**

Message

**Source:**

```xml
<element name="RemoveOrdersByKeysRequest" type="co:RemoteException"/>
```

---

**Namespace:**

http://www.somewhere.org/ServiceActivation

**Type:**

co:RemoteException

**Children:**

Message

**Source:**

```xml
<element name="RemoteException" type="co:RemoteException"/>
```

---

**Namespace:**

http://www.somewhere.org/ServiceActivation

**Type:**

co:RemoteException

**Children:**

OrderKey

**Annotation:**

This is the plural version of the removeOrder request. The only difference is that it takes a list of orders to be removed (indicated by a list of OrderKeys).
<element name="RemoveOrdersByKeysRequest">
 <annotation>
  <documentation>This is the plural version of the removeOrder request. The only difference is that it takes a list of orders to be removed (indicated by a list of OrderKeys).</documentation>
 </annotation>
 <complexType>
  <sequence>
   <element name="OrderKey" type="sa:ArrayOfOrderKey"/>
  </sequence>
 </complexType>
</element>
element RemoveOrdersByKeysRequest/OrderKey

namespace: http://www.somewhere.org/ServiceActivation

type: sa:ArrayOfOrderKey

children: Item

source:
<element name="OrderKey" type="sa:ArrayOfOrderKey"/>

element RemoveOrdersByKeysResponse

annotation:
<documentation>Only the message header is returned. This is sufficient to indicate that the order was removed.</documentation>

source:
<element name="RemoveOrdersByKeysResponse">
  <annotation>
    <documentation>Only the message header is returned. This is sufficient to indicate that the order was removed.</documentation>
  </annotation>
  <complexType>
    <sequence/>
  </complexType>
</element>

element ResumeOrderException
The following exceptions are returned if an error occurs.

```xml
<element name="ResumeOrderException">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="IllegalStateException" type="co:IllegalStateException"/>
      <element name="RemoteException" type="co:RemoteException"/>
      <element name="UnsupportedOperationException" type="co:UnsupportedOperation-Exception"/>
    </choice>
  </complexType>
</element>
```

**element ResumeOrderException/IllegalArgumentException**
### XML Schema

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>co:IllegalArgumentException</td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;IllegalArgumentException&quot; type=&quot;co:IllegalArgumentException&quot;/&gt;</code></td>
</tr>
</tbody>
</table>
element ResumeOrderException/IllegalStateException

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>co:IllegalStateException</td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;IllegalStateException&quot; type=&quot;co:IllegalStateException&quot;/&gt;</td>
</tr>
</tbody>
</table>

element ResumeOrderException/RemoteException

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>co:RemoteException</td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;RemoteException&quot; type=&quot;co:RemoteException&quot;/&gt;</td>
</tr>
</tbody>
</table>

element ResumeOrderException/UnsupportedOperationException
**XML Schema**

### element ResumeOrderRequest

- **namespace**: http://www.somewhere.org/ServiceActivation
- **children**: OrderKey
- **annotation documentation**: This request puts the order from a state SUSPENDED back into a RUNNING state.

```xml
<element name="ResumeOrderRequest">
  <annotation>
    <documentation>This request puts the order from a state SUSPENDED back into a RUNNING state.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderKey" type="sa:OrderKey"/>
    </sequence>
  </complexType>
</element>
```
element `ResumeOrderRequest/OrderKey`

```xml
<element name="OrderKey" type="sa:OrderKey"/>
```

diagram

```xml
namespace http://www.somewhere.org/ServiceActivation

type sa:OrderKey

children Domain PrimaryKey Type

source
```

element `ResumeOrderResponse`

```xml
<element name="ResumeOrderResponse">
  <annotation>
    <documentation>Only the message header is returned. This is sufficient to indicate that the order has been resumed.</documentation>
  </annotation>
  <complexType>
    <sequence/>
  </complexType>
</element>
```

diagram

```xml
namespace http://www.somewhere.org/ServiceActivation

annotation documentation

source
```

element `ServiceState`
### XML Schema

<table>
<thead>
<tr>
<th>diagram</th>
<th>«ServiceState»</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>sa:ServiceStateType</td>
</tr>
<tr>
<td>used by</td>
<td>complexType ServiceValue</td>
</tr>
<tr>
<td>facets</td>
<td>enumeration active inactive</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;ServiceState&quot; type=&quot;sa:ServiceStateType&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element** **SetOrderByValueException**

<table>
<thead>
<tr>
<th>diagram</th>
<th>SetOrderByValueException — SetException IllegalArgumentException IllegalStateException RemoteException</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>children</td>
<td>SetException IllegalArgumentException IllegalStateException RemoteException</td>
</tr>
<tr>
<td>annotation</td>
<td>The following exceptions are returned if an error occurs.</td>
</tr>
</tbody>
</table>

The following exceptions are returned if an error occurs.
<element name="SetOrderByValueException">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="SetException" type="co:SetException"/>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="IllegalStateException" type="co:IllegalStateException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
element SetOrderByValueException/SetException

```
namespace http://www.somewhere.org/ServiceActivation
type co:SetException
children Message
source <element name="SetException" type="co:SetException"/>
```

```
namespace http://www.somewhere.org/ServiceActivation
type co:IllegalArgumentException
children Message
source <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
```

```
```
```
element SetOrderByValueRequest

```xml
<element name="SetOrderByValueRequest" type="co:RemoteException"/>
```
## XML Schema

**SetOrderByValueRequest**

This request is used to change the attributes of an Order. Only the attributes which are populated are changed. The state and key attributes are ignored.

**Source**

```xml
<element name="SetOrderByValueRequest">
  <annotation>
    <documentation>This request is used to change the attributes of an Order. Only the attributes which are populated are changed. The state and key attributes are ignored.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderValue" type="sa:OrderValue"/>
    </sequence>
  </complexType>
</element>
```

**Namespace**

http://www.somewhere.org/ServiceActivation

**Children**

- **OrderValue**

**Diagram**

[Diagram showing the relationship between SetOrderByValueRequest and OrderValue]
element SetOrderByValueResponse

- namespace: http://www.somewhere.org/ServiceActivation
- type: sa:OrderValue
- children: SetOrderByValueResponse
- source: 
  ```xml
  <element name="OrderValue" type="sa:OrderValue"/>
  ```

- Only the message header is returned. This is sufficient to indicate that the order was changed.
<element name="SetOrderByValueResponse">
  <annotation>
    <documentation>Only the message header is returned. This is sufficient to indicate that the order was changed. </documentation>
  </annotation>
  <complexType>
    <sequence/>
  </complexType>
</element>
element SetOrdersByKeysException

```
<element name="SetOrdersByKeysException">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="SetException" type="co:SetException"/>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="IllegalStateException" type="co:IllegalStateException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```

element SetOrdersByKeysException/SetException
element `SetOrdersByKeysException/IllegalArgumentException`

```xml
<element name="SetException" type="co:SetException"/>
```

**namespace** http://www.somewhere.org/ServiceActivation

**type** `co:SetException`

**children** `Message`

**source** <element name="SetException" type="co:SetException"/>

**diagramelement** `SetOrdersByKeysException/IllegalStateException`

```xml
<element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
```

**namespace** http://www.somewhere.org/ServiceActivation

**type** `co:IllegalArgumentException`

**children** `Message`

**source** <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>

**diagramelement** `SetOrdersByKeysException/IllegalStateException`
element `SetOrdersByKeysException/RemoteException`

```
<element name="IllegalStateException" type="co:IllegalStateException"/>
```

namespace `http://www.somewhere.org/ServiceActivation`

type `co:IllegalStateException`

children `Message`

source `<element name="IllegalStateException" type="co:IllegalStateException"/>`

diagram

```
co:IllegalStateException

---

IllegalStateException

---

Message

The Message element indicates the error message from the Exception. This is most likely the result from a Exception.getMessage() call.
```

element `SetOrdersByKeysRequest`

```
<element name="RemoteException" type="co:RemoteException"/>
```

namespace `http://www.somewhere.org/ServiceActivation`

type `co:RemoteException`

children `Message`

source `<element name="REMOTEEXCEPTION" type="co:RemoteException"/>`

diagram

```
co:RemoteException

---

RemoteException

---

Message

The Message element indicates the error message from the Exception. This is most likely the result from a Exception.getMessage() call.
```
**XML Schema**

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is a version of the setOrder request that allows the client to set a number of orders (passing in a list of orderKeys to identify those orders) with the same values (passing in a single orderValue).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>children</td>
<td>orderKey orderValue</td>
</tr>
<tr>
<td>annotation</td>
<td>This is a version of the setOrder request that allows the client to set a number of orders (passing in a list of orderKeys to identify those orders) with the same values (passing in a single orderValue).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;element name=&quot;SetOrdersByKeysRequest&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;annotation&gt;</code></td>
</tr>
<tr>
<td><code>&lt;documentation&gt;</code>This is a version of the setOrder request that allows the client to set a number of orders (passing in a list of orderKeys to identify those orders) with the same values (passing in a single orderValue).&lt;/documentation&gt;`</td>
</tr>
<tr>
<td><code>&lt;complexType&gt;</code></td>
</tr>
<tr>
<td><code>&lt;sequence&gt;</code></td>
</tr>
<tr>
<td><code>&lt;element name=&quot;orderKey&quot; type=&quot;sa:ArrayOfOrderKey&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;element name=&quot;orderValue&quot; type=&quot;sa:OrderValue&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/sequence&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/complexType&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/element&gt;</code></td>
</tr>
</tbody>
</table>

**element** SetOrdersByKeysRequest/orderKey

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>sa:ArrayOfOrderKey</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>sa:ArrayOfOrderKey</td>
</tr>
<tr>
<td>children</td>
<td>Item</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;orderKey&quot; type=&quot;sa:ArrayOfOrderKey&quot;/&gt;</code></td>
</tr>
</tbody>
</table>
element `SetOrdersByKeysRequest/orderValue`

```
<element name="orderValue" type="sa:OrderValue"/>
```

**namespace** http://www.somewhere.org/ServiceActivation

**type** `sa:OrderValue`

**children**
- `sa:BaseState`
- `OrderKey`
- `ServiceValues`
- `ClientId`
- `sa:Priority`
- `Description`
- `RequestedDeliveryDate`
- `ActualDeliveryDate`
- `OrderDate`

**source**

```
<element name="orderValue" type="sa:OrderValue"/>
```

---

**element `SetOrdersByKeysResponse`**

```
only the message header is returned. This is sufficient to indicate that the orders were changed.
```

**namespace** http://www.somewhere.org/ServiceActivation
Only the message header is returned. This is sufficient to indicate that the orders were changed.

```xml
<element name="SetOrdersByKeysResponse">
  <annotation>
    <documentation>Only the message header is returned. This is sufficient to indicate that the orders were changed. </documentation>
  </annotation>
  <complexType>
    <sequence />
  </complexType>
</element>
```
## element SetOrdersByValuesException

### diagram

```
SetOrdersByValuesException ———>
  
  SetException ———>
  
  IllegalArgumentException ———>
  IllegalStateException ———>
  RemoteException
```

The following exceptions are returned if an error occurs.

### namespace

http://www.somewhere.org/ServiceActivation

### children

SetException, IllegalArgumentException, IllegalStateException, RemoteException

### annotation
documentation

The following exceptions are returned if an error occurs.

### source

```xml
<element name="SetOrdersByValuesException">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="SetException" type="co:SetException"/>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="IllegalStateException" type="co:IllegalStateException"/>
      <element name="RemoteException" type="co:RemoteException"/>
    </choice>
  </complexType>
</element>
```

## element SetOrdersByValuesException/SetException
## XML Schema

### element SetOrdersByValuesException/IllegalArgumentException

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
</tr>
<tr>
<td>namespace</td>
</tr>
<tr>
<td>type</td>
</tr>
<tr>
<td>children</td>
</tr>
<tr>
<td>source</td>
</tr>
</tbody>
</table>

The `Message` element indicates the error message from the Exception. This is most likely the results from a `Exception.getMessage()` call.

### element SetOrdersByValuesException/IllegalStateException

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td>namespace</td>
</tr>
<tr>
<td>type</td>
</tr>
<tr>
<td>children</td>
</tr>
<tr>
<td>source</td>
</tr>
</tbody>
</table>

### element SetOrdersByValuesException/IllegalArgumentException

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
<tr>
<td>namespace</td>
</tr>
<tr>
<td>type</td>
</tr>
<tr>
<td>children</td>
</tr>
<tr>
<td>source</td>
</tr>
</tbody>
</table>

The `Message` element indicates the error message from the Exception. This is most likely the results from a `Exception.getMessage()` call.
element SetOrdersByValuesException/RemoteException

namespace http://www.somewhere.org/ServiceActivation
type co:RemoteException
children Message
source <element name="RemoteException" type="co:RemoteException"/>

element SetOrdersByValuesException/IllegalStateException

namespace http://www.somewhere.org/ServiceActivation
type co:IllegalStateException
children Message
source <element name="IllegalStateException" type="co:IllegalStateException"/>
### XML Schema

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>namespace</td>
</tr>
<tr>
<td>children</td>
</tr>
<tr>
<td>annotation</td>
</tr>
</tbody>
</table>
| source | `<element name="SetOrdersByValuesRequest">
  <annotation>
    <documentation>This is the plural version of the setOrder request. The only difference is that it takes a list of OrderValues.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderValue" type="sa:ArrayOfOrderValue"/>
    </sequence>
  </complexType>
</element>` |

**element** SetOrdersByValuesRequest/OrderValue

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>namespace</td>
</tr>
<tr>
<td>type</td>
</tr>
<tr>
<td>children</td>
</tr>
<tr>
<td>source</td>
</tr>
</tbody>
</table>

**element** SetOrdersByValuesResponse
element `SetOrdersByValuesResponse`

```xml
<element name="SetOrdersByValuesResponse">
  <annotation>
    <documentation>Only the message header is returned. This is sufficient to indicate that the orders were changed.</documentation>
  </annotation>
  <complexType>
    <sequence />
  </complexType>
</element>
```

**element `StartOrderException`**

```xml
<element name="StartOrderException">
  <complexType>
    <simpleContent>
      <restriction base="string">
        <enumeration value="IllegalArgumentException"/>
        <enumeration value="IllegalStateException"/>
        <enumeration value="RemoteException"/>
      </restriction>
    </simpleContent>
  </complexType>
</element>
```
<source>

```xml
<element name="StartOrderException">
   <annotation>
      <documentation>The following exceptions are returned if an error occurs.</documentation>
   </annotation>
   <complexType>
      <choice>
         <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
         <element name="IllegalStateException" type="co:IllegalStateException"/>
         <element name="RemoteException" type="co:RemoteException"/>
      </choice>
   </complexType>
</element>
```
</source>
```xml
<element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
```

```xml
<element name="IllegalStateException" type="co:IllegalStateException"/>
```

```xml
<element name="RemoteException" type="co:RemoteException"/>
```
**XML Schema**

**element StartOrderRequest**

- **annotation**
  - This request puts the order into a **RUNNING** state.

- **source**
  - `<element name="StartOrderRequest" />
    <annotation>
      <documentation>This request puts the order into a **RUNNING** state.</documentation>
    </annotation>
    <complexType>
      <sequence>
        <element name="OrderKey" type="sa:OrderKey"/>
      </sequence>
    </complexType>`
element **StartOrderRequest/OrderKey**

```
<element name="OrderKey" type="sa:OrderKey"/>
```

**namespace**: http://www.somewhere.org/ServiceActivation

**type**: `sa:OrderKey`

**children**: Domain PrimaryKey Type

**source**

```
<element name="StartOrderResponse">
  <annotation>
    <documentation>Only the message header is returned. This is sufficient to indicate that the order was started.</documentation>
  </annotation>
  <complexType>
    <sequence/>
  </complexType>
</element>
```

element **StartOrderResponse**

```
<element name="State">
  <complexType>
    <sequence/>
  </complexType>
</element>
```

element **State**
**XML Schema**

<table>
<thead>
<tr>
<th>diagram</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; State /&gt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>type</th>
<th>sa:OrderState</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>facets</th>
<th>enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td></td>
</tr>
<tr>
<td>open.not_running</td>
<td></td>
</tr>
<tr>
<td>open.not_running.not_started</td>
<td></td>
</tr>
<tr>
<td>open.not_running.suspened</td>
<td></td>
</tr>
<tr>
<td>open.running</td>
<td></td>
</tr>
<tr>
<td>closed</td>
<td></td>
</tr>
<tr>
<td>closed.completed</td>
<td></td>
</tr>
<tr>
<td>closed.aborted</td>
<td></td>
</tr>
<tr>
<td>closed.aborted.byclient</td>
<td></td>
</tr>
<tr>
<td>closed.aborted.bysserver</td>
<td></td>
</tr>
</tbody>
</table>

**source**

```xml
<element name="State" type="sa:OrderState" substitutionGroup="sa:BaseState"/>
```

element **SuspendOrderException**
The following exceptions are returned if an error occurs.

```xml
<element name="SuspendOrderException">
  <annotation>
    <documentation>The following exceptions are returned if an error occurs.</documentation>
  </annotation>
  <complexType>
    <choice>
      <element name="IllegalArgumentException" type="co:IllegalArgumentException"/>
      <element name="IllegalStateException" type="co:IllegalStateException"/>
      <element name="RemoteException" type="co:RemoteException"/>
      <element name="UnsupportedOperationException" type="co:UnsupportedOperationException"/>
    </choice>
  </complexType>
</element>
```

**element** `SuspendOrderException/IllegalArgumentException`
<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>co:IllegalArgumentException</td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;IllegalArgumentException&quot; type=&quot;co:IllegalArgumentException&quot;/&gt;</code></td>
</tr>
</tbody>
</table>
### element SuspendOrderException/IllegalStateException

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>co:IllegalStateException</td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;IllegalStateException&quot; type=&quot;co:IllegalStateException&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

The `Message` element indicates the error message from the Exception. This is most likely the result from a `Exception.getMessage()` call.

### element SuspendOrderException/RemoteException

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>co:RemoteException</td>
</tr>
<tr>
<td>children</td>
<td>Message</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;RemoteException&quot; type=&quot;co:RemoteException&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

The `Message` element indicates the error message from the Exception. This is most likely the result from a `Exception.getMessage()` call.

### element SuspendOrderException/UnsupportedOperationException
element SuspendOrderRequest

The following request put the order from a RUNNING state into a SUSPENDED state.

```xml
<element name="SuspendOrderRequest">
  <annotation>
    <documentation>The following request put the order from a RUNNING state into a SUSPENDED state.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="OrderKey" type="sa:OrderKey"/>
    </sequence>
  </complexType>
</element>
```
**element** SuspendOrderRequest/OrderKey

```xml
<element name="OrderKey" type="sa:OrderKey"/>
```

**element** SuspendOrderResponse

```xml
<source>
<element name="SuspendOrderResponse">
  <annotation>
    <documentation>Only the message header is returned. This is sufficient to indicate that the order has been suspended.</documentation>
  </annotation>
  <complexType>
    <sequence/>
  </complexType>
</element>
</source>
```

**complexType** ArrayOfOrderKey

```xml
<complexType name="ArrayOfOrderKey">
  <sequence/>
</complexType>
```
XML Schema

```xml
<complexType name="ArrayOfOrderKey">
  <annotation>
    <documentation>This is a representation of a OrderKey array.</documentation>
  </annotation>
  <sequence>
    <element name="Item" type="sa:OrderKey" nullable="true" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

element ArrayOfOrderKey/Item

```xml
<element name="Item" type="sa:OrderKey" nullable="true" maxOccurs="unbounded"/>
```

complexType ArrayOfOrderValue
element `ArrayOfOrderValue/Item`
**XML Schema**

```
<element name="Item" type="sa:OrderValue" nullable="true" maxOccurs="unbounded"/>
```

### ComplexType ArrayOfServiceValue

```
```

**Diagram:**

- **ArrayOfServiceValue**
  - **Item**

**Namespace:** http://www.somewhere.org/ServiceActivation

**Children:**
- Item

**Source:**

```
```

**Annotation:**

This is a representation of a ServiceValue array.
<complexType name="ArrayOfServiceValue">
  <annotation>
    <documentation>This is a representation of a ServiceValue array.</documentation>
  </annotation>
  <sequence>
    <element name="Item" type="sa:ServiceValue" nullable="true" maxOccurs="unbounded"/>
  </sequence>
</complexType>
### Element ArrayOfServiceValue/Item

**Diagram**

```
Item ➔ sa:ServiceValue ➔ sa:ServiceState ➔ ServiceKey
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>sa:ServiceValue</td>
</tr>
<tr>
<td>children</td>
<td>sa:ServiceState, ServiceKey</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;Item&quot; type=&quot;sa:ServiceValue&quot; nullable=&quot;true&quot; maxOccurs=&quot;unbounded&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

### ComplexType OrderAttributeValueChangeEventType

**Diagram**

```
OrderAttributeValueChangeEventType ➔ co:BaseEventType
```

- **EventType**
- **EventTime**
- **Domain**
- **NotificationId**
- **SourceIndicator**
- **OrderValue**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>extension of co:BaseEventType</td>
</tr>
<tr>
<td>children</td>
<td>EventType, EventTime, Domain, NotificationId, SourceIndicator, OrderValue</td>
</tr>
<tr>
<td>used by</td>
<td>element</td>
</tr>
<tr>
<td>annotation</td>
<td>Base Content of Attribute Value Change Event</td>
</tr>
</tbody>
</table>
<complexType name="OrderAttributeValueChangeEventType">
  <annotation>
    <documentation>Base Content of Attribute Value Change Event</documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseEventType">
      <sequence>
        <element name="OrderValue" type="sa:OrderValue" nullable="false"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

element OrderAttributeValueChangeEventType/OrderValue

```
<element name="OrderValue" type="sa:OrderValue" nullable="false"/>
```

complexType OrderCreateEventType

```
namespace http://www.somewhere.org/ServiceActivation
type sa:OrderValue
children sa:BaseState OrderKey ServiceValues ClientId sa:Priority Description RequestedDeliveryDate ActualDeliveryDate OrderDate
source
```
**XML Schema**

---

**namespace**
http://www.somewhere.org/ServiceActivation

**type**
extension of `co:BaseEventType`

**children**
EventTime Domain NotificationId SourceIndicator OrderValue

**used by**

```
OrderCreateEvent/Event
```

**annotation**
Base Content of Create Event Type

**source**

```xml
<complexType name="OrderCreateEventType">
  <annotation>
    <documentation>Base Content of Create Event Type</documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseEventType">
      <sequence>
        <element name="OrderValue" type="sa:OrderValue" nullable="false"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

**element**

```
OrderCreateEventType/OrderValue
```
**XML Schema**

```xml
<complexType name="OrderValue">
  <sequence>
    <element name="sa:BaseState"/>
    <element name="OrderKey"/>
    <element name="ServiceValues"/>
    <element name="ClientId"/>
    <element name="sa:Priority"/>
    <element name="Description"/>
    <element name="RequestedDeliveryDate"/>
    <element name="ActualDeliveryDate"/>
    <element name="OrderDate"/>
  </sequence>
</complexType>
```

**complexType** `OrderKey`

```xml
<complexType name="OrderKey">
  <simpleContent>
    <restriction base="http://www.somewhere.org/ServiceActivation">
      <attribute name="Domain" />
      <attribute name="PrimaryKey" />
      <attribute name="Type" />
    </restriction>
  </simpleContent>
</complexType>
```
The OrderKey uniquely identifies an order.

```xml
<complexType name="OrderKey">

  <annotation>
    <documentation>The OrderKey uniquely identifies an order.</documentation>
  </annotation>

  <sequence>
    <element name="Domain" type="string"/>
    <element name="PrimaryKey" type="string"/>
    <element name="Type" type="string"/>
  </sequence>

</complexType>
```
element **OrderKey/Domain**

<table>
<thead>
<tr>
<th>diagram</th>
<th></th>
<th></th>
</tr>
</thead>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| type      | string                                      |
| source    | `<element name="Domain" type="string"/>`  |

element **OrderKey/PrimaryKey**

<table>
<thead>
<tr>
<th>diagram</th>
<th></th>
<th></th>
</tr>
</thead>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| type      | string                                      |
| source    | `<element name="PrimaryKey" type="string"/>`  |

element **OrderKey/Type**

<table>
<thead>
<tr>
<th>diagram</th>
<th></th>
<th></th>
</tr>
</thead>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| type      | string                                      |
| source    | `<element name="Type" type="string"/>`  |

complexType **OrderRemoveEventType**
**element** OrderRemoveEventType/OrderValue
complexType OrderStateChangeEventType

namespace http://www.somewhere.org/ServiceActivation

type sa:OrderValue

children sa:BaseState OrderKey ServiceValues ClientId sa:Priority Description RequestedDeliveryDate ActualDeliveryDate OrderDate

source <element name="OrderValue" type="sa:OrderValue" nullable="false"/>

complexType OrderStateChangeEventType
element OrderStateChangeEventType/OrderValue
complexType OrderValue

namespace http://www.somewhere.org/ServiceActivation

type sa:OrderValue

children
    sa:BaseState
    OrderKey
    ServiceValues
    ClientId
    sa:Priority
    Description
    RequestedDeliveryDate
    ActualDeliveryDate
    OrderDate

source
    <element name="OrderValue" type="sa:OrderValue" nullable="false"/>

complexType OrderValue
The following is an XML representation of an order. An order is a request to perform an operation. Typical operations are Activate, Deactivate etc. OrderValue is the base type of an order. The intent of OrderValue is to be derived by a more specific type such as Activate. This is achieved by declaring OrderValue abstract, meaning OrderValue can not be used directly in instance documents, but only derived types of OrderValue can be used in instance documents. The base OrderValue contains the elements state (contains the current state of the order which must be one of OrderState), orderKey (uniquely identifies an order), serviceValue (indicates a business service to be sold to a customer), clientId (Client id of the customer), priority (Priority of the order, must be one of OrderPriority), and a miscellaneous description field.

namespace  http://www.somewhere.org/ServiceActivation

children  sa:BaseState OrderKey ServiceValues ClientId sa:Priority Description RequestedDeliveryDate ActualDeliveryDate OrderDate

used by  elements ArrayOfOrderValue/Item NewOrderValueResponse/OrderValue CreateOrderByValueRequest/OrderValue SetOrderByValueRequest/OrderValue SetOrdersByKeysRequest/OrderValue GetOrderByKeyResponse/OrderValue OrderAt-
tributeValueChangeEventType/OrderValue OrderCreateEventType/OrderValue OrderRemoveEventType/OrderValue OrderStateChangeEventType/OrderValue
<table>
<thead>
<tr>
<th>annotation</th>
<th>documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following is an XML representation of an order. An order is a request to perform an operation. Typical operations are Activate, Deactivate etc. OrderValue is the base type of an order. The intent of OrderValue is to be derived by a more specific type such as Activate. This is achieved by declaring OrderValue abstract, meaning OrderValue cannot be used directly in instance documents, but only derived types of OrderValue can be used in instance documents. The base OrderValue contains the elements state (contains the current state of the order which must be one of OrderState), orderKey (uniquely identifies an order), serviceValue (indicates a business service to be sold to a customer), clientId (Client id of the customer), priority (Priority of the order, must be one of OrderPriority), and a miscellaneous description field.</td>
<td></td>
</tr>
</tbody>
</table>

```xml
<complexType name="OrderValue" abstract="true">
  <annotation>
    <documentation>The following is an XML representation of an order. An order is a request to perform an operation. Typical operations are Activate, Deactivate etc. OrderValue is the base type of an order. The intent of OrderValue is to be derived by a more specific type such as Activate. This is achieved by declaring OrderValue abstract, meaning OrderValue cannot be used directly in instance documents, but only derived types of OrderValue can be used in instance documents. The base OrderValue contains the elements state (contains the current state of the order which must be one of OrderState), orderKey (uniquely identifies an order), serviceValue (indicates a business service to be sold to a customer), clientId (Client id of the customer), priority (Priority of the order, must be one of OrderPriority), and a miscellaneous description field.</documentation>
  </annotation>
  <sequence>
    <element ref="sa:BaseState" nullable="true" minOccurs="0"/>
    <element name="OrderKey" type="sa:OrderKey" nullable="true" minOccurs="0"/>
    <element name="ServiceValues" type="sa:ArrayOfServiceValue" minOccurs="0"/>
    <element name="ClientId" type="string" nullable="true" minOccurs="0"/>
    <element ref="sa:Priority" nullable="true" minOccurs="0"/>
    <element name="Description" type="string" nullable="true" minOccurs="0"/>
    <element name="RequestedDeliveryDate" type="timeInstant" nullable="true" minOccurs="0"/>
    <element name="ActualDeliveryDate" type="timeInstant" nullable="true" minOccurs="0"/>
    <element name="OrderDate" type="timeInstant" nullable="true" minOccurs="0"/>
  </sequence>
</complexType>
```
### element OrderValue/OrderKey

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| type | sa:OrderKey |
| children | Domain, PrimaryKey, Type |
| source | `<element name="OrderKey" type="sa:OrderKey" nullable="true" minOccurs="0"/>` |

### element OrderValue/ServiceValues

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| type | sa:ArrayOfServiceValue |
| children | Item |
| source | `<element name="ServiceValues" type="sa:ArrayOfServiceValue" minOccurs="0"/>` |

### element OrderValue/ClientId

<table>
<thead>
<tr>
<th>diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

| namespace | http://www.somewhere.org/ServiceActivation |
| type | string |
| source | `<element name="ClientId" type="string" nullable="true" minOccurs="0"/>` |
**XML Schema**

**element OrderValue/Description**

<table>
<thead>
<tr>
<th>diagram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;Description&quot; type=&quot;string&quot; nullable=&quot;true&quot; minOccurs=&quot;0&quot;/&gt;</td>
</tr>
</tbody>
</table>

**element OrderValue/RequestedDeliveryDate**

<table>
<thead>
<tr>
<th>diagram</th>
<th>RequestedDeliveryDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>timeInstant</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;RequestedDeliveryDate&quot; type=&quot;timeInstant&quot; nullable=&quot;true&quot; minOccurs=&quot;0&quot;/&gt;</td>
</tr>
</tbody>
</table>

**element OrderValue/ActualDeliveryDate**

<table>
<thead>
<tr>
<th>diagram</th>
<th>ActualDeliveryDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>timeInstant</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;ActualDeliveryDate&quot; type=&quot;timeInstant&quot; nullable=&quot;true&quot; minOccurs=&quot;0&quot;/&gt;</td>
</tr>
</tbody>
</table>

**element OrderValue/OrderDate**

<table>
<thead>
<tr>
<th>diagram</th>
<th>OrderDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>timeInstant</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;OrderDate&quot; type=&quot;timeInstant&quot; nullable=&quot;true&quot; minOccurs=&quot;0&quot;/&gt;</td>
</tr>
</tbody>
</table>
complexType QueryValue

| diagram | <complexType name="QueryValue" abstract="true">  
| namespace | http://www.somewhere.org/ServiceActivation  
| used by | NewQueryValueResponse/QueryValue QueryOrdersRequest/QueryValue  
| source | <sequence/>  

complexType ServiceKey

| diagram | <complexType name="ServiceKey">  
| namespace | http://www.somewhere.org/ServiceActivation  
| children | Domain PrimaryKey Type  
| used by | ServiceValue/ServiceKey  
| annotation | The ServiceKey uniquely identifies a service.  
| source | <element name="Domain" type="string"/>  

The ServiceKey uniquely identifies a service.
### XML Schema

**element** `ServiceKey/Domain`

<table>
<thead>
<tr>
<th>diagram</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;Domain&quot; type=&quot;string&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element** `ServiceKey/PrimaryKey`

<table>
<thead>
<tr>
<th>diagram</th>
<th>PrimaryKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;PrimaryKey&quot; type=&quot;string&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element** `ServiceKey/Type`

<table>
<thead>
<tr>
<th>diagram</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;Type&quot; type=&quot;string&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**complexType** `ServiceValue`
The following is an XML representation of a service. An service is something of business value that can be associated to a subscriber. Typical are DSL, 3G etc. ServiceValue is the base type of a service. The intent of ServiceValue is to be derived by a more specific type such DSL. This is achieved by declaring ServiceValue abstract, meaning ServiceValue can not be used directly in instance documents, but only derived types of ServiceValue can be used in instance documents. The base ServiceValue contains the elements ServiceValue contains the current state of the order which must be one of ServiceStateType, ServiceKey (uniquely identifies an service)

| namespace | http://www.somewhere.org/ServiceActivation |
| children  | sa:ServiceState, ServiceKey |
| used by   | ArrayOfServiceValue/Item, NewServiceValueResponse/ServiceValue |
| annotation| The following is an XML representation of a service. An service is something of business value that can be associated to a subscriber. Typical are DSL, 3G etc. ServiceValue is the base type of a service. The intent of ServiceValue is to be derived by a more specific type such DSL. This is achieved by declaring ServiceValue abstract, meaning ServiceValue can not be used directly in instance documents, but only derived types of ServiceValue can be used in instance documents. The base ServiceValue contains the elements ServiceState (contains the current state of the order which must be one of ServiceStateType), ServiceKey (uniquely identifies an service) |
<complexType name="ServiceValue" abstract="true">
  <annotation>
    <documentation>The following is an XML representation of a service. An service is something of business value that can be associated to a subscriber. Typical are DSL, 3G etc. ServiceValue is the base type of a service. The intent of ServiceValue is to be derived by a more specific type such DSL. This is achieved by declaring ServiceValue abstract, meaning ServiceValue can not be used directly in instance documents, but only derived types of ServiceValue can be used in instance documents. The base ServiceValue contains the elements ServiceState (contains the current state of the order which must be one of ServiceStateType), serviceKey (uniquely identifies an service)</documentation>
  </annotation>
  <sequence>
    <element ref="sa:ServiceState" nullable="true" minOccurs="0"/>
    <element name="ServiceKey" type="sa:ServiceKey" nullable="true" minOccurs="0"/>
  </sequence>
</complexType>
element **ServiceValue/ServiceKey**

```xml
<element name="ServiceKey" type="sa:ServiceKey" nullable="true" minOccurs="0"/>
```

**namespace**  http://www.somewhere.org/ServiceActivation

**type**  sa:ServiceKey

**children**  Domain PrimaryKey Type

**source**

```xml
<element name="ServiceKey" type="sa:ServiceKey" nullable="true" minOccurs="0"/>
```

**simpleType** **OrderManagerOption**

**namespace**  http://www.somewhere.org/ServiceActivation

**type**  restriction of string

**facets**

- OrderAutoRemove
- suspendedOrder
- resumeOrder
- setPriority
- getPriority

**enumeration**

- OrderAutoRemove
- suspendedOrder
- resumeOrder
- setPriority
- getPriority
<table>
<thead>
<tr>
<th>Source</th>
<th>XML Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;simpleType name=&quot;OrderManagerOption&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;restriction base=&quot;string&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;enumeration value=&quot;OrderAutoRemove&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;enumeration value=&quot;suspendedOrder&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;enumeration value=&quot;resumeOrder&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;enumeration value=&quot;setPriority&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;enumeration value=&quot;getPriority&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/restriction&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/simpleType&gt;</td>
</tr>
</tbody>
</table>
simpleType **OrderPriority**

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>restriction of unsignedInt</td>
</tr>
<tr>
<td>used by</td>
<td>element</td>
</tr>
<tr>
<td><strong>Priority</strong></td>
<td></td>
</tr>
<tr>
<td>facets</td>
<td>minInclusive</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>maxInclusive</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>source</td>
<td>&lt;simpleType name=&quot;OrderPriority&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;restriction base=&quot;unsignedInt&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;minInclusive value=&quot;1&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;maxInclusive value=&quot;5&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/restriction&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/simpleType&gt;</td>
</tr>
</tbody>
</table>

simpleType **OrderState**

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>restriction of string</td>
</tr>
<tr>
<td>used by</td>
<td>element</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td></td>
</tr>
</tbody>
</table>
### XML Schema

<table>
<thead>
<tr>
<th>facets</th>
<th>enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>enumeration</td>
</tr>
<tr>
<td>open.not_running</td>
<td>enumeration</td>
</tr>
<tr>
<td>open.not_running.not_started</td>
<td>enumeration</td>
</tr>
<tr>
<td>open.not_running.suspened</td>
<td>enumeration</td>
</tr>
<tr>
<td>open.running</td>
<td>enumeration</td>
</tr>
<tr>
<td>closed</td>
<td>enumeration</td>
</tr>
<tr>
<td>closed.completed</td>
<td>enumeration</td>
</tr>
<tr>
<td>closed.aborted</td>
<td>enumeration</td>
</tr>
<tr>
<td>closed.aborted.byclient</td>
<td>enumeration</td>
</tr>
<tr>
<td>closed.aborted.byserver</td>
<td>enumeration</td>
</tr>
</tbody>
</table>

```xml
<simpleType name="OrderState">
  <restriction base="string">
    <enumeration value="open"/>
    <enumeration value="open.not_running"/>
    <enumeration value="open.not_running.not_started"/>
    <enumeration value="open.not_running.suspened"/>
    <enumeration value="open.running"/>
    <enumeration value="closed"/>
    <enumeration value="closed.completed"/>
    <enumeration value="closed.aborted"/>
    <enumeration value="closed.aborted.byclient"/>
    <enumeration value="closed.aborted.byserver"/>
  </restriction>
</simpleType>
```
simpleType **OrderType**

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>restriction of string</td>
</tr>
<tr>
<td>facets</td>
<td></td>
</tr>
<tr>
<td>activate</td>
<td></td>
</tr>
<tr>
<td>deactivate</td>
<td></td>
</tr>
<tr>
<td>modify</td>
<td></td>
</tr>
<tr>
<td>remove</td>
<td></td>
</tr>
</tbody>
</table>

```xml
<source>
<simpleType name="OrderType">
  <restriction base="string">
    <enumeration value="activate"/>
    <enumeration value="deactivate"/>
    <enumeration value="modify"/>
    <enumeration value="remove"/>
  </restriction>
</simpleType>
</source>
```

simpleType **ServiceStateType**

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/ServiceActivation">http://www.somewhere.org/ServiceActivation</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>restriction of string</td>
</tr>
<tr>
<td>used by</td>
<td><strong>ServiceState</strong></td>
</tr>
<tr>
<td>facets</td>
<td></td>
</tr>
<tr>
<td>active</td>
<td></td>
</tr>
<tr>
<td>inactive</td>
<td></td>
</tr>
</tbody>
</table>
<simpleType name="ServiceStateType">
<restriction base="string">
<enumeration value="active"/>
<enumeration value="inactive"/>
</restriction>
</simpleType>
complexType **co:ArrayOfString**

This is a representation of a String array.

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>children</td>
<td>Item</td>
</tr>
<tr>
<td>used by</td>
<td>GetOrderByKeyRequest/AttributeName GetOrdersByKeysRequest/AttributeNames GetOrdersByTemplatesRequest/AttributeNames QueryOrdersRequest/AttributeNames GetSupportedOperationsResponse/OrderManagerOption GetOrderTypesResponse/OrderType GetQueryTypesResponse/QueryType GetServiceTypes.Response/ServiceType</td>
</tr>
<tr>
<td>annotation</td>
<td>This is a representation of a String array.</td>
</tr>
</tbody>
</table>
| source          | <complexType name="ArrayOfString">  
|                 |   <annotation>  
|                 |     <documentation>This is a representation of a String array.</documentation>  
|                 |   </annotation>  
|                 |   <sequence>  
|                 |     <element name="Item" type="string" nullable="true" maxOccurs="unbounded"/>  
|                 |   </sequence>  
|                 | </complexType> |

**Diagram**

```
co:ArrayOfString  ----> Item
```

**Element co:ArrayOfString/Item**

<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td>&lt;element name=&quot;Item&quot; type=&quot;string&quot; nullable=&quot;true&quot; maxOccurs=&quot;unbounded&quot;/&gt;</td>
</tr>
</tbody>
</table>

**complexType co:BaseEventType**
XML Schema

```xml
<complexType name="BaseEventType" abstract="true">
  <annotation>
    <documentation>Base Event</documentation>
  </annotation>
  <sequence>
    <element name="EventType" type="string" nullable="false"/>
    <element name="EventTime" type="timeInstant" nullable="false"/>
    <element name="Domain" type="string" nullable="false"/>
    <element name="NotificationId" type="string" nullable="false"/>
    <element name="SourceIndicator" type="string" nullable="false"/>
  </sequence>
</complexType>
```

element co:BaseEventType/EventType

```xml
<complexType name="BaseEventType" abstract="true">
  <annotation>
    <documentation>Base Event</documentation>
  </annotation>
  <sequence>
    <element name="EventType" type="string" nullable="false"/>
  </sequence>
</complexType>
```
**element co:BaseEventType/EventTime**

<table>
<thead>
<tr>
<th>diagram</th>
<th>EventTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></td>
</tr>
<tr>
<td>type</td>
<td>timeInstant</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;EventTime&quot; type=&quot;timeInstant&quot; nullable=&quot;false&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element co:BaseEventType/Domain**

<table>
<thead>
<tr>
<th>diagram</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;Domain&quot; type=&quot;string&quot; nullable=&quot;false&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element co:BaseEventType/NotificationId**

<table>
<thead>
<tr>
<th>diagram</th>
<th>NotificationId</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;NotificationId&quot; type=&quot;string&quot; nullable=&quot;false&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**element co:BaseEventType/SourceIndicator**

<table>
<thead>
<tr>
<th>diagram</th>
<th>SourceIndicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>source</td>
<td><code>&lt;element name=&quot;SourceIndicator&quot; type=&quot;string&quot; nullable=&quot;false&quot;/&gt;</code></td>
</tr>
</tbody>
</table>

**complexType co:BaseException**
**XML Schema**

```
<complexType name="BaseException">
  <annotation>
    <documentation>The BaseException is the parent complexType of the Exceptions.</documentation>
  </annotation>
  <sequence>
    <element name="Message" type="string">
      <annotation>
        <documentation>The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.</documentation>
      </annotation>
    </element>
  </sequence>
</complexType>
```

**element** `co:BaseException/Message`

```
<Message>
  The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.
</Message>
```
<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
</tr>
<tr>
<td>annotation</td>
<td>The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.</td>
</tr>
</tbody>
</table>
| source             | <element name="Message" type="string">  
|                    | <annotation>  
|                    | <documentation>The Message element indicates the error message from the Exception. This is most likely the results from a Exception.getMessage() call.</documentation>  
|                    | </annotation>  
|                    | </element> |
complexType **co:CreateException**

<table>
<thead>
<tr>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **Message**
  - The CreateException exception can be returned by all create(...) requests defined in the OSS through Java XML/JMS interface. The exception is used as a standard application-level exception to report a failure to create a managed entity or a collection of managed entities. This exception is thrown when a particular managed entity or group of managed entities cannot be created.

<table>
<thead>
<tr>
<th>Namespace</th>
<th><a href="http://www.somewhere.org/Common">http://www.somewhere.org/Common</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>extension of <strong>co:BaseException</strong></td>
</tr>
<tr>
<td>Children</td>
<td><strong>Message</strong></td>
</tr>
<tr>
<td>Used by</td>
<td>CreateOrderByValueException/CreateException/Co:CreateException</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The CreateException exception can be returned by all create(...) requests defined in the OSS through Java XML/JMS interface. The exception is used as a standard application-level exception to report a failure to create a managed entity or a collection of managed entities. This exception is thrown when a particular managed entity or group of managed entities cannot be created.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;complexType name=&quot;CreateException&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;annotation&gt;</code></td>
</tr>
<tr>
<td><code>&lt;documentation&gt;</code> The CreateException exception can be returned by all create(...) requests defined in the OSS through Java XML/JMS interface. The exception is used as a standard application-level exception to report a failure to create a managed entity or a collection of managed entities. This exception is thrown when a particular managed entity or group of managed entities cannot be created. &lt;/documentation&gt;`</td>
</tr>
<tr>
<td><code>&lt;/annotation&gt;</code></td>
</tr>
<tr>
<td><code>&lt;complexType&gt;</code></td>
</tr>
<tr>
<td><code>&lt;extension base=&quot;co:BaseException&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;sequence/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/extension&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/complexType&gt;</code></td>
</tr>
</tbody>
</table>
complexType `co:DuplicateKeyException`

The `co:DuplicateKeyException` exception is returned if a managed entity cannot be created because an object with the same key already exists. This exception is only used when a managed entity key is provided in a `create(...) request` and when client controlled naming is used. This exception is returned by the `create requests defined in the OSS through Java XML/JMS interface.`

```xml
<complexType name="DuplicateKeyException">
  <annotation>
    <documentation> The DuplicateKeyException exception is returned if a managed entity cannot be created because an object with the same key already exists. This exception is only used when a managed entity key is provided in a `create(...) request` and when client controlled naming is used. This exception is returned by the `create requests defined in the OSS through Java XML/JMS interface.` </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```

complexType `co:FinderException`
The exception is used as a standard application-level exception to report a failure to find the requested managed entities. This exception is returned when a collection of one or more entity cannot be found. This exception should not be returned by requests that return a collection of managed entities using an associative lookup approach (they should return a null list instead).

```
<complexType name="FinderException">
    <annotation>
        <documentation> The exception is used as a standard application-level exception to report a failure to find the requested managed entities. This exception is returned when a collection of one or more entity cannot be found. This exception should not be returned by requests that return a collection of managed entities using an associative lookup approach (they should return a null list instead). </documentation>
    </annotation>
    <complexContent>
        <extension base="co:BaseException">
            <sequence/>
        </extension>
    </complexContent>
</complexType>
```
The `IllegalArgumentException` exception is returned by the OSS through Java XML/JMS interface to report that the request could not be completed because one of the arguments passed in is invalid.

```xml
<complexType name="IllegalArgumentException">
  <annotation>
    <documentation>The IllegalArgumentException exception is returned by the OSS through Java XML/JMS interface to report that the request could not be completed because one of the arguments passed in is invalid.</documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```
The IllegalArgumentException exception is returned by the OSS through Java XML/JMS interface to report that the request could not be completed because one of the arguments passed in is invalid.

```xml
<complexType name="IllegalStateException">
  <annotation>
    <documentation>The IllegalArgumentException exception is returned by the OSS through Java XML/JMS interface to report that the request could not be completed because one of the arguments passed in is invalid.</documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```
namespace: http://www.somewhere.org/Common

children: HowMany

used by: GetOrdersByTemplatesRequest QueryOrdersRequest

annotation: This is a representation of an Iterator Request.
<complexType name="IteratorRequest">
  <annotation>
    <documentation>This is a representation of an Iterator Request.</documentation>
  </annotation>
  <sequence>
    <element name="HowMany" type="positiveInteger" nullable="true" minOccurs="0">
      <annotation>
        <documentation>The HowMany element indicates how many result instances should be returned in the Response message. If the value supplied is greater than the complete result set then just the result set is returned in the response message, and End-Of-Reply element will be set to 'true'. If the HowMany element exceeds a preset maximum (for performance reasons) then the number of instances returned will equal the preset application maximum. Not specifying this element will return the entire result set.</documentation>
      </annotation>
    </element>
  </sequence>
</complexType>
The HowMany element indicates how many result instances should be returned in the Response message. If the value supplied is greater than the complete result set then just the result set is returned in the response message, and EndOfReply element will be set to ‘true’. If the HowMany element exceeds a preset application maximum (for performance reasons) then the number of instances returned will equal the preset application maximum. Not specifying this element will return the entire result set.
| source | <element name="HowMany" type="positiveInteger" nullable="true" minOccurs="0">  
|        | <annotation> 
|        |     <documentation>The HowMany element indicates how many result instances should be returned in the Response message. If the value supplied is greater than the complete result set then just the result set is returned in the response message, and End-Of-Reply element will be set to 'true'. If the HowMany element exceeds a preset maximum (for performance reasons) then the number of instances returned will equal the preset application maximum. Not specifying this element will return the entire result set.</documentation> 
|        | </annotation> 
|        | </element> |
### complexType `co:IteratorResponse`

```xml
<complexType name="co:IteratorResponse">
  <sequence>
    <element name="co:IteratorResponse"/>
  </sequence>
  <annotation>
    This is a representation of an Iterator Response.
  </annotation>
</complexType>
```

#### Diagram

- **Sequence**
  - The sequence number indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.

- **EndOfReply**
  - The EndOfReply indicates whether there are any more result sets being returned.

#### namespace
- `http://www.somewhere.org/Common`

#### children
- `Sequence`  
- `EndOfReply`

#### used by
- `GetOrdersByTemplatesResponse`  
- `QueryOrdersResponse`

#### annotation
- This is a representation of an Iterator Response.
<complexType name="IteratorResponse">
  <annotation>
    <documentation>This is a representation of an Iterator Response.</documentation>
  </annotation>
  <sequence>
    <element name="Sequence" type="positiveInteger">
      <annotation>
        <documentation>The Sequence number indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.</documentation>
      </annotation>
    </element>
    <element name="EndOfReply" type="boolean">
      <annotation>
        <documentation>The EndOfReply indicates whether there are any more result sets being returned.</documentation>
      </annotation>
    </element>
  </sequence>
</complexType>
element **co:IteratorResponse/Sequence**

```
<element name="Sequence" type="positiveInteger">
  <annotation>
    <documentation>
      The Sequence number indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.
    </documentation>
  </annotation>
</element>
```

**namespace** http://www.somewhere.org/Common
**type** positiveInteger
**annotation** documentation
The Sequence number indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.

**source**

```
<element name="Sequence" type="positiveInteger">
  <annotation>
    <documentation>
      The Sequence number indicates the result set order i.e. since there can be a number of response messages generated and there is no mechanism to insure the responses are sent sequentially, there needs to be a method of ordering the response messages that are returned.
    </documentation>
  </annotation>
</element>
```

**element **co:IteratorResponse/EndOfReply**

```
<element name="EndOfReply" type="boolean">
  <annotation>
    <documentation>
      The EndOfReply indicates whether there are any more result sets being returned.
    </documentation>
  </annotation>
</element>
```

**namespace** http://www.somewhere.org/Common
**type** boolean
**annotation** documentation
The EndOfReply indicates whether there are any more result sets being returned.
<element name="EndOfReply" type="boolean">
  <annotation>
    <documentation>The EndOfReply indicates whether there are any more result sets being returned.</documentation>
  </annotation>
</element>
complexType `co:ObjectNotFoundException`

The `ObjectNotFoundException` exception is returned by an OSS through Java request to indicate that the specified managed entity does not exist. Only the request that are declared to return a single managed entity use this exception. This exception should not be returned by methods that return a collection of managed entities. This exception is returned when a singular managed entity cannot be found.

```xml
<complexType name="ObjectNotFoundException">
  <annotation>
    <documentation> The ObjectNotFoundException exception is returned by an OSS through Java request to indicate that the specified managed entity does not exist. Only the request that are declared to return a single managed entity use this exception. This exception should not be returned by methods that return a collection of managed entities. This exception is returned when a singular managed entity cannot be found. </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence />
    </extension>
  </complexContent>
</complexType>
```

complexType `co:RemoteException`

```xml
<complexType name="RemoteException">
  <annotation>
    <documentation> The RemoteException indicates the error message from the Exception. This is likely the result from a Exception.getMessage() call. </documentation>
  </annotation>
  <complexContent>
    <sequence />
  </complexContent>
</complexType>
```
The RemoteException is returned when an error occurs during any remote object operation. This is most likely the result from a Exception.getMessage() call.

The RemoteException is returned when an errors occurs during any remote object operation.

The RemoteException is returned when an errors occurs during any remote object operation.

<complexType name="RemoteException">
  <annotation>
    <documentation>The RemoteException is returned when an errors occurs during any remote object operation.</documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
**complexType** `co:RemoveException`  

```xml
<complexType name="RemoveException">
  <annotation>
    <documentation>
      The RemoveException exception is returned at an attempt to remove a collection of one or more managed entity when the XML/JMS interface does not allow the managed entity to be removed. This exception is returned when a collection of one or more managed entity cannot be removed.
    </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence />
    </extension>
  </complexContent>
</complexType>
```

- **namespace**: http://www.somewhere.org/Common
- **type**: extension of `co:BaseException`
- **children**: `Message`
- **used by**: `RemoveOrderByKeyException:RemoveException RemoveOrdersByKeysException:RemoveException`

---

**complexType** `co:SetException`
The `SetException` exception is returned at an attempt to update a collection of one or more managed entity when the XML/JMS interface does not allow the managed entity to be updated. This exception is returned when a collection of one or more managed entity cannot be updated.

```xml
<complexType name="SetException">
  <annotation>
    <documentation>
      The SetException exception is returned at an attempt to update a a collection of one or more managed entity when the XML/JMS interface does not allow the managed entity to be updated. This exception is returned when a collection of one or more managed entity cannot be updated.
    </documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```

**complexType** `co:UnsupportedOperationException`
The UnsupportedOperationException exception is returned by the OSS through Java XML/JMS interface to report that the invoked request could not be completed because it is not implemented.

```xml
<complexType name="UnsupportedOperationException">
  <annotation>
    <documentation>
The UnsupportedOperationException exception is returned by the OSS through Java XML/JMS interface to report that the invoked request could not be completed because it is not implemented.</documentation>
  </annotation>
  <complexContent>
    <extension base="co:BaseException">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
```
FAQ

To better understand the design of the interface, here are some questions and answers on specific issues of the API:

Q: Why does the base interface of all service values, interface ManagedEntityValue, introduce get-AttributeValue() / setAttributeValue? Why not use Java’s reflection capabilities to access attributes in a generic way?
A: Because reflection is at least 10 times slower.

Q: Why is the service-related data placed in a separate class? Why haven’t they been included in OrderValue?
A: We assume that the service-related data would be placed in a service inventory. Also, the service data has to be stored by the service provider as long as the customer has subscribed to the service.

In contrast, the order data can be archived some time after the order has been fulfilled.

From object-oriented perspective, an order can be seen as a command object on the service. The main difference is that this order command may take quite a long time to execute, from 250 ms up to weeks.

Q: It seems that I can access order and service attributes in two ways, either by calling order.getAttribute("attribute") or by calling order.getAttribute(). Why is that?
A: We are expecting two kinds of clients: service-specific clients and generic clients. Service-specific clients are coded for a specific set of service, e.g. a DSL Order Manager System. Service-specific clients will be aware of the special order and service types for these services and will be hard-coded to the strongly typed interface. They will use the order.get<Attribute>() methods.

But there will also be generic clients, e.g. workflow management systems. These clients are configured at the customer’s site to fulfill certain orders using the business process of the customer. In most cases, these clients use a weakly typed programming model and thus would use order.getAttributeValue("attribute").

Q: If an order can only be executed partially, how is that handled by the OrderManager? (The service would be active/operational already after the partial implementation, but e.g. there would be not as much capacity or high-availability yet as there is planned to be finally ...)
A: One way to handle this situation is: The original order is executed until it is COMPLETED. However, some service attributes contain values that are different than what was requested (e.g. order.getService().get-Bandwidth() returns only 1024 instead of 2048.). Based on that, a new ModifyOrder is created (manually or automatically) that initiated the changes.
Appendix

Glossary

The following terms are used in this document:

client
implementation
customer
user
north-bound interface

Open issues

Service Activation Specific

(major) How do we model the Service Management Information Base? How are complex product offerings represented as ServiceValues? Currently an OrderValue can have an an array of ServiceValues. That is insufficient because a) the client can not inspect what is allowed and b) some services require more complex modelling. Are there any restrictions on the attribute types to support generic clients?

Requirements:

• Clients can introspect about the allowed service "tree" and attributes.
• Implementation effort is reasonable, service bundeling from existing attributes should not require one class per service bundle.

(minor) Network Order concepts missing. But do we need to model it? What is the difference?

(minor) Should we introduce more array/template methods, for e.g. startOrder and removeOrder? (San Jose)

General

Should we declare IllegalArgumentException + UnsupportedOperation as RuntimeException? Consequence:

+ clients can handle Illegal Argument from server and client similar
+ signals a system-error clearly and causes log-file entry and transaction abort
- Problems with bulk-operations (but we have these anyway)

ManagedEntityValue: Should we stay only with the generic set/getAttributeValue methods and remove all other methods, e.g. getAttributeType(), getSettableAttributes()? Instead the Introspector should be used. Consequences:

+ simpler API
+ more features: short description, methods, property editor, icons, ...
+ tool support for producing ServiceValue Beans.
- can the Introspector be used inside EJBs?

The implicit relationship between connection factories and topics in the Lookup service is not very nice, e.g. does the code above really work? It relies on the fact that listXy returns entries in the same sequence. Should we have that more explicit in the lookup service by introducing a new abstraction, e.g. BusinessComponentInterface etc. This instance would then include: one EJB home, one factory and one topic and an Attribute list. We would then have only one list method.

Is there any need to restrict the possible values for the transaction attributes in the deployment descriptor? Otherwise we might see problem with Plug&Work. (A)
Ideas for Future Development of the API

New exception for createOrder/setOrder if the order/service data is wrong. (E)

Add more bulk operations, and perhaps introduce a special semantics for sub-orders? (E,N)

It shall be possible to set relations between orders (e.g. dependency and sub-orders). (E)

Notifications for Service object creation/modification/deletion from the ServiceManager (if it comes from the node?). (E)

Notification of abnormal or faulty situation in the node related to subscription management. (E)

There should be an attribute of services called Service Category (Primary Service, Secondary Service, CPE, Hunting, Feature with SID relationship, Feature Voice/Fax-mail) (E)

validate() method on ServiceValue and OrderValue? (N)

Support of substring matching in the template methods?

Follow XML related JCPs, esp. JAXB.

Ideas for Additional Items in the Documentation

Contents

Put values for the Enums into the generated spec.

Explain how to implement the API in a generic way. (m)

Explain the idea of second level attributes, e.g. order.getAttributeValue ("service.bandwidth") (m)

Explain how to carry out an order with multiple services: we should have an order that affects multiple service objects. This also includes filtering. (E, N)

If yes, how to get more detailed information on which service was provided and which failed? Do we require certain attributes on the order for that? (E)

First explain XVT, then XML/JMS.

Format

Lines are too wide and/or font size is too small.

Generated Content includes class names, but not ChapterSubHead from chapter 1-5.

Quality of some images is poor.

ClassHead: font size too large, no space and no ruler before the paragraph.

Font size of for <pre> and <code> is different.
Index

A
ABORTED 81
ABORTED_BYCLIENT 82
ABORTED_BYSERVER 82
abortOrder 44, 56
ACTIVATE 83
ACTIVE 87
ACTUAL_DELIVERY_DATE 69
API_TYPE 91

C
CLIENT_ID 69, 75
CLOSED 82
COMPLETED 82
create 54, 66, 98
createOrder 45
createOrderByValue 57
createOrders 45
createOrdersByValues 57

D
DEACTIVATE 83
DESCRIPTION 69

E
Event 105
EXPEDITE 80

F
fromXml 110

G
GET_ORDERS 79
GET_PRIORITY 79
getActualDeliveryDate 70
getAttributeNames 102
getAttribute 98
getAttributeType 70, 86
getAttributeValue 102
getClientId 70
getCurrentState 77
description 70
getDomain 99, 105
getEventTime 106
getEventType 106
getFactory 91
getHome 92
getJeopardyDate 70
getJndiName 98
getManagedEntityKey 102
getNamingAttributes 92
getNext 55, 67
getNotificationId 106
getObject 99
getOrder 46
getOrderByKey 58
getOrderDate 70
getOrderKey 70, 77
getOrders 46, 47
getOrdersByKeys 58
getOrdersByTemplates 59
getOrderTypes 47, 59
getOrderValue 76, 78
getPopulatedAttributeNames 103
getPrimaryKey 100
getPriority 71
getPurchaseOrder 71
getQueryTypes 48, 59
getQueue 92
getReason 77
getRequestedDeliveryDate 71
getRootName 110
getServiceKey 86
getServices 71
getServiceTypes 48, 60
getSettableAttributeNames 103
getSourceIndicator 106
getState 71, 86
getSupportedOperations 48, 60
getTopic 92
getType 100
getXmlHeader 110

H
HIGH 80

I
IllegalArgumentException 107, 108
IllegalStateException 84
INACTIVE 87
isFullyPopulated 103
Index

isPopulated 103

J
JEOPARDY_DATE 69

K
KEY 102

L
listAttributeValues 93
listDomains 93
listFactories 93, 94
listHomes 94, 95
listQueues 95, 96
listTopics 96
Lookup 90
LookupHome 97
LookupResult 98
LOW 80

M
ManagedEntityKey 99
ManagedEntityValue 101
MEDIUM 81
MODIFY 83

N
NAME 91
newOrderValue 48, 60
newQueryValue 49, 61
newServiceValue 49, 61
NORMAL 81
NOT_RUNNING 82
NOT_STARTED 82

O
OPEN 82
ORDER_DATE 69
ORDER_PRIMARY_KEY 75
ORDER_TYPE 75
OrderAttributeValueChangeEvent 77
OrderCreateEvent 75
OrderKey 74
OrderManager 42
OrderManagerHome 54
OrderManagerOption 78
OrderMessageProperty 74
OrderPriority 80
OrderRemoveEvent 78
OrderState 81
OrderStateChangeEvent 76
OrderType 83
OrderValue 67
OrderValueIterator 54
OSSJ_DOMAIN_PROP_NAME 75
OSSJ_EVENT_TYPE_PROP_NAME 75

P
PRIORITY 69
PRODUCT_NAME 91
PRODUCT_VERSION 91
PURCHASE_ORDER 69

Q
queryOrders 49, 61
QueryValue 107

R
REMOVE 83
REMOVE_ORDER 79
REMOVE_ORDERS 79
removeOrder 50
removeOrderByKey 62
removeOrders 50
removeOrdersByKey 62
REQUESTED_DELIVERY_DATE 69
reset 107
RESUME_ORDER 79
resumeOrder 51, 63
RUNNING 82

S
ServiceKey 86
SERVICES 69
ServiceState 86
ServiceValue 85
SET_PRIORITY 80
setActualDeliveryDate 71
setAttributeValue 103
setClientId 72
setDescription 72
setDomain 100, 106
setEventType 106
setEventType 106
SetException 108, 109
setFullyPopulated 104
setJeopardyDate 72
setManagedEntityKey 104
setNotificationId 106
setOrder 51
setOrderByIdValue 63
setOrderDate 72
setOrderKey 72
setOrders 52
setOrdersByKeys 64
setOrdersByValues 64
setPrimaryIdKey 100
setPriority 73
setPurchaseOrder 73
setRequestedDeliveryDate 73
setService 73
setServiceKey 86
setServices 73
setSourceIndicator 107
setState 74, 86
setType 100
startOrder 52, 65
STATE 69, 85
SUSPEND_ORDER 80
SUSPENDED 82
suspendOrder 53, 65

toXml 110

unpopulateAllAttributes 104
unpopulateAttribute 104
UnsupportedOperationException 109

VENDOR_NAME 91

XMLOrderManager 55
XMLOrderManagerHome 66
XMLOrderValueIterator 66
XmlSerializable 109