Whereto® CARD

Ticketing Order Interface specification

Order Interface version:	**1.1**
Document version:	**1.60**
Last modified:	**4 januari**
Reference:	**Ticketing order interface specification.doc**

Whereto®
CARD

Table of contents

1 Overview 1

1.1 Specific usages 3

2 Order Interface 4

2.1 Technology 4
2.2 Request XML definition 4
2.3 Order Response XML Definition 11
2.3.1 Comment with type URL 12
2.4 Checks on posted requests 13
2.4.1 Schema check 13
2.4.2 Client authentication 13
2.4.3 Order content check 13

3 Order Update Interface 14

3.1 Technology 14
3.2 Request XML definition 14
3.3 Response XML Definition 15

4 Appendix A: Order Interface 16

4.1 Order Request Schema 16
4.2 Example Request XML 19
4.2.1 Sending tickets only 19
4.2.2 Sending tickets and storing order 20
4.3 Order Response Schema 21
4.4 Order Interface XML Structure from a different perspective 1
4.4.1 Overview on Client in relation to orderRequest 1
4.4.2 Overview on Order in relation to orderRequest 2
4.4.3 Overview on Tickets in relation to orderRequest 3

5 Appendix B 4

1 Overview

The WheretoCard Ticketing platform offers an OrderInterface that enables clients to sell Mobile- and E-tickets through their own web shop. Clients using the OrderInterface can use the resend functionality contained in the Ticketing platform itself, to request redelivery of tickets.
Figure 1.0 gives an overview of the ordering process using the OrderInterface.

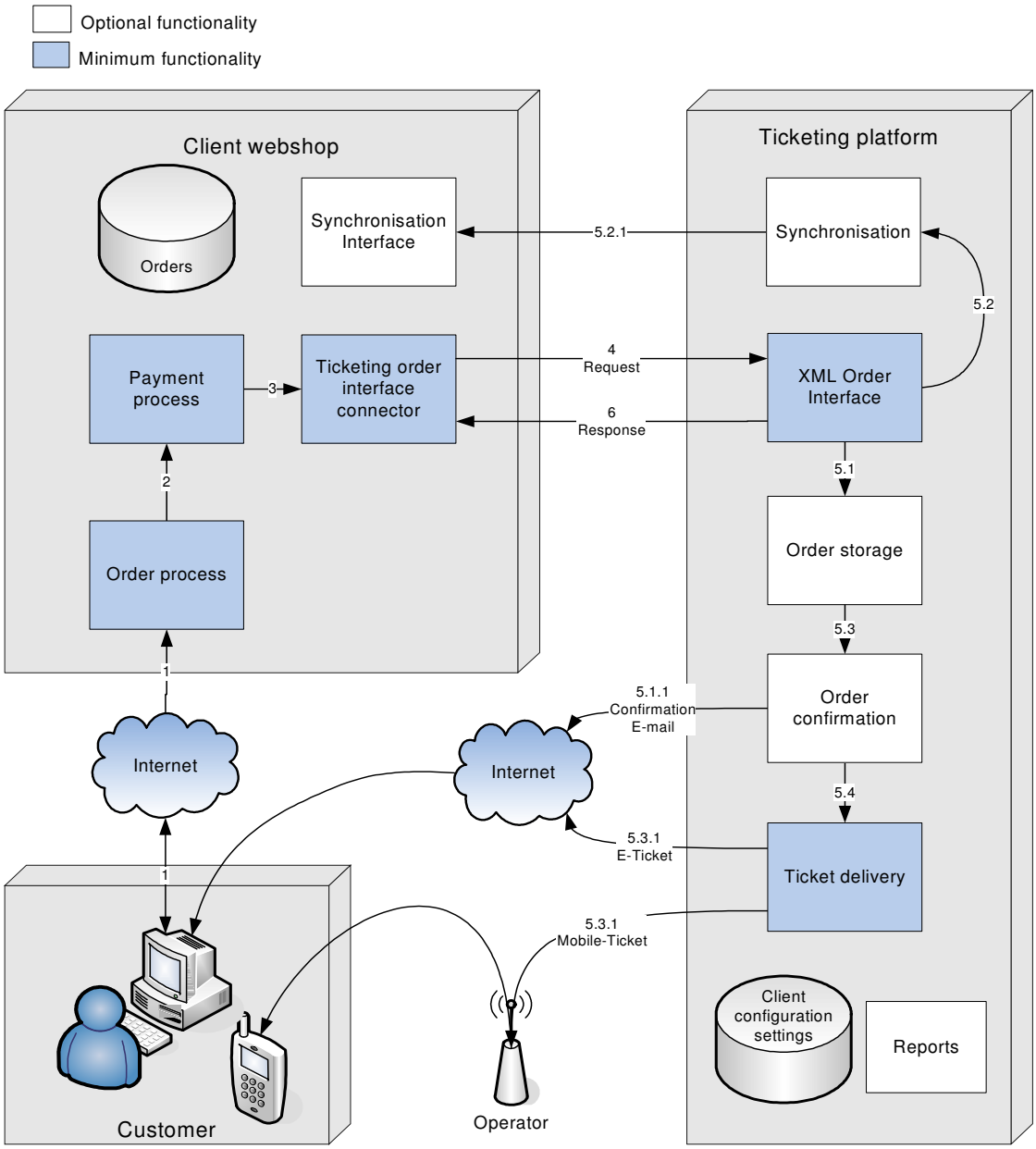

Figure 1.0: Overview Ticketing OrderInterface

The client handles the order intake, and payment, in his own web shop. Upon a successful payment, the client creates an order request based on the data provided by the customer on his web site. The client sends the order request over the Internet to the Ticketing OrderInterface. The OrderInterface will check if the message is correct, and will handle the delivery of the mobile ticket.
The success/failure response (6) upon a request (4) will be returned as a HTTP OK response message back to the client. The response contains XML information whether or not the request has been processed

Tickets are delivered as Mobile Tickets or as E-Tickets. Synchronizing the tickets back to the client is optional as well as sending an order confirmation E-mail. The Ticketing platform also can provide report functionality about sold tickets.

1.1 Specific usages

Depending on the functionality the web shop of the client offers, the OrderInterface can be used in different ways:

	Order Storage	Order Confirmation	Ticket Delivery	Ticket Synchronization
Delivery only			X	
Delivery and order storage	X		X	
Delivery, order storage and confirmation	X	X	X	
Delivery, order storage and synchronization	X		X	X
Delivery, order storage, confirmation and synchronization	X	X	X	X

Confirmation of the order is only possible when the order is being stored on the Ticketing side as well. The same applies to the synchronization. Reporting functionality can be used with any of the scenarios described above.

2 Order Interface

2.1 Technology

The Ticketing OrderInterface is based on open Internet standards as HTTPS and XML. This allows any platform (java, perl, Microsoft, etc) to communicate and integrate with the OrderInterface.

The XML document, containing an orderRequest, is posted as content of the HTTP POST request to the URL the OrderInterface is running on.

In the live environment, the OrderInterface is only accessible over HTTPS.

2.2 Request XML definition

Sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<orderRequest version="1.0">
	<client code="CLIENT_X">
		<username>defaultuser</username>
		<password>defaultpw</password>
	</client>
	<order referenceId="<unique number>">
		<properties>
			<property name="FIRST_NAME" value="first name"/>
			<property name="INFIX" value="infix"/>
			<property name="LAST_NAME" value="lastname"/>
			<property name="EMAIL" value="customerX@mail.nl"/>
		</properties>
		<cases>
			<case code="CASE_X_STD">
				<orderlines>
					<orderline code="ORD_LINE_TYPE_X_STD" ticket-ref="1234">
						<quantity>1</quantity>
						<price>1250</price>
						<action>Action description</action>
						<validPeriod from="2009-04-02T09:00:00" to="2009-05-02T18:30:00"/>
					</orderline>
					<orderline code="ORD_LINE_TYPE_Y_STD" ticket-ref="1234">
						<quantity>1</quantity>
						<price>1000</price>
						<action>Action description</action>
						<validPeriod from="2009-04-02T00:00:00" to="2008-01-01T00:00:00"/>
					</orderline>
				</orderlines>
			</case>
		</cases>
	</order>
	<tickets>
		<ticket ref="1234" delivery-channel="WEB" delivery-format="BARCODE">
			<properties>
				<property name="" value=""/>
			</properties>
		</ticket>
	</tickets>
</orderRequest>
```

In the next section each element is described in more detail. Appendix A contains a different perspective of the XML structure (see figures 4.4.1, 4.4.2 and 4.4.3).

/**orderRequest**
Description:
- Contains the order request structure built up with out of one client, one tickets and an optional orders element

/**orderRequest @ version**
Description:
- Contains the version number of the OrderInterface schema, if no version attribute is specified the default is 1.0

Type: String
Default: 1.0

//**client**
Description:
- Contains the credentials of a registered client that have been specifically configured for the client.

//**client @ code**
Description:
- Code that identifies the client. This code will be assigned to the client after it has been configured on the Ticketing platform

Type: String

///**username**
Description:
- Configured username, a client can have multiple usernames

///**password**
Description:
- Configured password that belongs to the supplied username

//**order**
Description:
- An order contains an owner, cases and properties element. The owner represents the customer that placed the order. The cases describe which articles (orderlines) were bought. The optional order properties can be used to store the customer credentials.
- This element is optional. Only use it if the OrderInterface has to store the order.

//**order @ referenceId**
Description:
- Optional Reference ID to identify individual orders. Optionally the OrderInterface can be configured to check on duplicate order reference IDs, blocking orders that already have been received.

Constraint: reference ID needs to be unique on client basis (max 32 characters).

//**orderProperties**
Description:
- Holds optional order properties for storing additional data of the customer like firstname and lastname

///**orderProperty**
Description:
- The additional order property holding a name value pair

///**orderProperty @ name**
Description:
- Contains the property name
- When using the option where the OrderInterface confirms the order to the customer by email, the EMAIL order property is necessary

Possible values: FIRST_NAME, INFIX, LAST_NAME, STREET, HOUSE_NUMBER,CITY, POSTAL_CODE, COUNTRY, EMAIL, PHONE_NUMBER

///**orderProperty @ value**
Description:
- Contains the property value

Constraint: The value may contain 100 characters at a maximum.

//**cases**
Description:
- A case represents the store. The store can sell products (orderlines). An order can be built up out of orderlines from different cases; these cases and their orderlines are listed here.

//**case**
Description:
- Contains the orderlines that were sold for the case

Note: The current implementation of the order interface cannot handle more than one case!

//**case @ code**
Description:
- Contains the code by which the case is known in the Ticketing system

Type: String

//**orderlines**
Description:
- Contains the orderlines that were sold for the parenting case

//**orderline**
Description:

- Contains the structure for a particular orderline. An orderline represents the product that was sold.

//**orderline @ code**
Description:
- The code by which the OrderInterface can determine which type of orderline is sold

//**orderline @ ticket-ref**
Description:
- Links the orderline to a ticket element (see ticket ref). Multiple orderlines can be linked to a single ticket, or each orderline can each have its own ticket. Each ticket will be delivered separately to the customer.

//**quantity**
Description:
- Contains the number of orderlines (items) were sold, like when a customer purchases two entrance cards this value will be 2.

//**timesValid**
Description:
- Contains the amount of times the orderline is valid to be used/scanned.

//**price**
Description:
The price in cents for the particular orderline.
Constraint:
Maximum value of 999.999.999 cents.

//**action**
Description:
Description of the action to be shown on the printed tickets.
Constraint:
Maximum number of characters is 64.

//**validPeriod @from and @to**
Description:
The dates and times between which the ticket is valid.
Constraint:
- Date/time format is 2009-12-31T23:59:59. If only a dates need to be supplied the hours, minutes and seconds can be set to zero.
- The to date/time should be greater then the from date/time.

Note: The valid period excludes the to date, so with a period of 2009-04-02T00:00:00 to 2009-04-03T00:00:00 will only be valid on the 2nd of April.

//**tickets**
Description:

- Contains the ticket(s) that need to be delivered to the customer. It is possible to specify multiple tickets

///**ticket**
Description:
- Ticket structure that holds the necessary information for a particular ticket

///**ticket @ ref**
Description:
- Optional reference value, necessary when linking orderlines to a ticket. From within the orderline a reference to a ticket can be made based on the ref value.

///**ticket @ delivery-channel**
Description:
- Specifies the channel over which the ticket will be sent to the customer
- If SMS, the deliveryAddress should contain the phone element
- If EMAIL, the deliveryAddress should contain the email element

Type: String
Possible values:

SMS	The ticket is sent in a SMS message
WEB	The ticket is sent to the user's browser (PDF)
EMAIL	The ticket is sent in an Email message

///**ticket @ delivery-format**
Description:
- Specifies the format in which the ticket will be sent to the customer
- BARCODE means the ticket contentCode will be converted to a barcode image
- TEXTCODE means the contentCode will be used as is

Type: String
Possible values: BARCODE, TEXTCODE

///**contentCode**
Description:
- A unique string that identifies the ticket. Scanning the barcode will result in this string. When this tag is omitted the Ticketing platform generates a unique contentCode.

Constraint:
- Maximum length depends on the type of barcode configured

///**ticketProperties**
Description:
- Contains additional property name-value pairs. Specifying these properties is optional.

Type: String

///**ticketProperty @ name**

Description:
- Name of the property

Type: String
Possible values:

TICKET_TEXT	The accompanying text for the ticket

///**ticketProperty @ value**
Description:
- Value of the property

Type: String
Constraint: The value may contain 100 characters at a maximum.

///**deliveryAddresses**
Description:
- Contains one or more deliveryAddress elements, allowing a ticket to be delivered to multiple addresses. The deliveryAddresses tag is mandatory when the requested delivery channel is set to SMS or EMAIL. In case of a WEB delivery the deliveryAddresses tag should be omitted.

///**deliveryAddress**
Description:
- DeliveryAddress structure that holds the necessary information for a particular deliveryAddress
- When the ticket delivery-channel is set to SMS all deliveryAddresses need to contain the phone element
- When the ticket delivery-channel is set to EMAIL all deliveryAddresses need to contain the email element

///**phone**
Description:
- Phone structure that holds the necessary information for a particular phone

///**phone @ number**
Description:
- The phone number the ticket is going to be delivered to

Type: String
Example: +31612345678

///**device**
Description:
- Describes the device details of the target phone. The information about manufacturer and type of phone is necessary to determine the type of message (NPM or EMS) to send. The device details of manufacturer and type will be supplied to the client.

///**manufacturer**
Description:

- Specifies the phone manufacturer

Example: NOKIA

///**type**

Description:

- Specifies the manufacturers phone type

Example: 7210

///**email**

Description:

- The email address the ticket is going to be delivered to

2.3 Order Response XML Definition

Basically there are two HTTP statuses to expect when posting orderRequests to the OrderInterface:

- “200 OK”
- “401 UNAUTHORISED”

When receiving other HTTP responses than listed above, these are due to external factors.

In case the request has been successfully received, the OrderInterface returns a HTTP OK response (200) message. The body of this response contains additional XML information about the final status of the request. See section 4.3 for the XML response Schema definition.
In case the request has been processed successfully, the following XML is returned:

```
<orderResponse status="OK">
    <comment type=”url”>
        <![CDATA[http://ticketing.wheretocard.nl/client/page/printOrder?ori=xxx&ui=yyy]]>
    </comment>
</orderResponse>
```

The <comment> tag is optional. Within this tag additional information could be returned to the sender. In example above an URL is contained within the <comment> tag. The type attribute tells something about the comment. In this case it tells that the text specified within the <comment> tag is an URL.

In case the request was not processed, the response contains an error element holding additional information why the request was not processed:

```
< orderResponse status="NOT_OK">
    <error>
        <code>number</code>
        <description><![CDATA[more detailed description]]></description>
    </error>
</ orderResponse >
```

The error codes in the error element are divided into logical groups. Basically there are three groups of errors defined, each having their own range of error codes.

Error code start range	**Cause**
OI_0**0**XX	Configuration
OI_0**1**XX	Client credentials are wrong
OI_0**2**XX	XML request contains errors

See Appendix B for a complete list of known error codes.

2.3.1 Comment with type URL

The URL contained in the response can be used to print the order. By contacting the URL a PDF document is created and printed in the user's browser. When contacting the URL it could happen that an error occurs. The possible errors are:

Code	Descripton
PO_0001	Internal server error
PO_0002	Unknown error
PO_0011	URL check failed, order not found
PO_0012	URL check failed, the url contains an invalid ui code
PO_0013	If the URL expiration flag is set, this error indicates that the time has been expired
PO_0014	The IP address of the computer the client connected from to order the tickets is different from the one the client used to print the tickets.

If an error was detected duering the print order process and you would like to inform the user about this error on a page that is in the look&feel of your company, you should append the pageError parameter to the URL (that was generated by the Ticketing platform). This URL points to the error page that is maintained by your company. The user is redirected to the URL that is included in the pageError parameter. The URL to this error page is extended with the errorCode parameter that can by used in the response to the user.

Example:

URL generated by the Ticketing Platform:
http://ticketing.wheretocard.nl/client/page/printOrder?ori=xxx&ui=yyy

URL with the pageError parameter appended by your company:
http://ticketing.wheretocard.nl/client/page/printOrder?ori=xxx&ui=yyy&pageError=http://www.companyx.nl/myErrorPage

URL the user is redirected to when an error was detected duering the print order process:
http://www.companyx.nl/myErrorPage?errorCode=PO_XXXX

2.4 Checks on posted requests

Upon sending data to the OrderInterface, the following checks are performed on the document that was sent.

- XML against Schema check
- Client authentication
- Order content check

Failing any of these checks results in discarding the complete order request. For instance when you have specified 4 tickets, and one of these contains a faulty phone number, none of the tickets will get sent, nor the order will be stored.

2.4.1 Schema check

The OrderInterface Schema (see 3) describes the minimum set of tags that must exist in the order request and their order. It also describes the mandatory tags and the possible values of certain tags. The Schema check takes place at the time the XML parser parses the order request. See appendix A for the Schema itself.

2.4.2 Client authentication

The client must provide his client code, username and a password in the XML document. The Ticketing OrderInterface uses this information to perform client authentication. The application first checks if the client is known and then if the given user is allowed to have access to it.

2.4.3 Order content check

This check verifies if the references in the XML document are valid.

Emailaddresses should match the following pattern: [\w-]+(\.[\w-]+)*@([\w-]+\.)+[a-zA-Z]{2,7}
Meaning the email address should contain a '@' and should have at least one character prior to the @ sign. The part after the '@' sign should contain a '.' and some characters.

Phonenumbers should match the following pattern: [+](\d{8,20})
Due to the differences in phone numbers on an international level, there are no strict rules on the phone numbers, except that they should start with a + sign and must have a length between 8 and 20 characters. This check is only performed in when a SMS ticket is requested.

Supplied data that needs to match the **configured data** in the ticketing system:

- Client code, username and password
- Device manufacturers and their corresponding types (only for SMS tickets)
- Case codes
- Orderline codes

3 Order Update Interface

3.1 Technology

Like with the order interface this interface is uses an XML request send over HTTP to execute updates on existing orders. The HTTP response contains the status of the process in XML format.

The update order interface is located on http://<host>/ticketService/order/updateOrder. The interface expects the XML to be located in the body of the HTTP request.

3.2 Request XML definition

In the request for an order update a client authentication and one or more orders are supplied.

Sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<orderRequest>
    <client code="CLIENT_X">
        <username>defaultuser</username>
        <password>defaultpw</password>
    </client>
    <update-orders>
        <order referenceId="<unique number>">
            <validFrom>2009-04-02T09:00:00</validFrom>
            <validTo>2009-05-02T18:30:00</validTo>
        </order>
    </update-orders>
</orderRequest>
```

The client authentication is the same as in the order interface request.

//**update-orders**
Description:

- Contains the orders that are going to be updated.

///**order**
Description:

- Represents an order that is to be updated.

///**order @ referenceId**
Description:

- Mandatory Reference ID to identify individual orders.

Constraint: reference ID needs to be the same as the referenceId used to create the order via the order interface.
Constraint:

- reference ID needs to be the same as the referenceId used to create the order via the order interface.

////**validFrom**
Description:

- Optional element that specifies the new start of the order valid periond.
Constraint:
- Date/time format is 2009-12-31T23:59:59. If only a dates need to be supplied the hours, minutes and seconds can be set to zero.

////**validTo**
Description:
- Optional element that specifies the new end of the order valid periond.
  Constraint:
- Date/time format is 2009-12-31T23:59:59. If only a dates need to be supplied the hours, minutes and seconds can be set to zero.
- The validTo date/time should be greater then the validFrom date/time.

Note: The valid period excludes the to date, so with a period of 2009-04-02T00:00:00 to 2009-04-03T00:00:00 will only be valid on the 2nd of April.

3.3 Response XML Definition

When processing the request and updating the orders is completed successfully a response XML is returned with a status attribute indicating success. The XML response looks like this:

```
<orderResponse status="OK"/>
```

When something fails in the order update process or while reading the XML request the response XML has the status NOT_OK. In this case also an error subelement is supplied, containing an error code and a description why the processing has failed. The same codes as in the order interface are used.

The error XML response looks like this:

```
<orderResponse status="NOT_OK">
    <error>
        <code>number</code>
        <description><![CDATA[more detailed description]]></description>
    </error>
</orderResponse
```

4 Appendix A: Order Interface

4.1 Order Request Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by E (L) -->
<!--W3C Schema generated by XMLSPY v2004 rel. 3 U (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:jxb="http://java.sun.com/xml/ns/jaxb" jxb:version="1.0">
	<xsd:element name="orderRequest" type="orderRequestType"/>
	<xsd:complexType name="orderRequestType">
		<xsd:sequence>
			<xsd:element name="client" type="clientType"/>
			<xsd:element name="order" type="orderType" minOccurs="0"/>
			<xsd:element name="tickets" type="ticketsType"/>
		</xsd:sequence>
		<xsd:attribute name="version" type="xsd:string" use="optional" default="1.0"/>
	</xsd:complexType>
	<xsd:complexType name="caseType">
		<xsd:sequence>
			<xsd:element name="orderlines" type="orderlinesType"/>
		</xsd:sequence>
		<xsd:attribute name="code" type="xsd:string" use="required"/>
	</xsd:complexType>
	<xsd:complexType name="casesType">
		<xsd:sequence>
			<xsd:element name="case" type="caseType" maxOccurs="unbounded"/>
		</xsd:sequence>
	</xsd:complexType>
	<xsd:complexType name="clientType">
		<xsd:sequence>
			<xsd:element name="username" type="xsd:string"/>
			<xsd:element name="password" type="xsd:string"/>
		</xsd:sequence>
		<xsd:attribute name="code" type="xsd:string" use="required"/>
	</xsd:complexType>
	<xsd:complexType name="deliveryAddressType">
		<xsd:choice>
			<xsd:element name="phone" type="phoneType"/>
			<xsd:element name="email">
				<xsd:simpleType>
					<xsd:restriction base="xsd:string">
						<xsd:pattern value="[\w\-]+(\.[\w\-]+)*@([\w\-]+\.)+[a-zA-Z]{2,7}"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:element>
		</xsd:choice>
	</xsd:complexType>
	<xsd:complexType name="deliveryAddressesType">
		<xsd:sequence>
			<xsd:element name="deliveryAddress" type="deliveryAddressType" maxOccurs="unbounded"/>
		</xsd:sequence>
	</xsd:complexType>
	<xsd:complexType name="deviceType">
		<xsd:sequence>
			<xsd:element name="manufacturer" type="xsd:string"/>
			<xsd:element name="type" type="xsd:string"/>
		</xsd:sequence>
	</xsd:complexType>
	<xsd:complexType name="orderType">
		<xsd:sequence>
			<xsd:element name="properties" type="orderPropertiesType" minOccurs="0"/>
			<xsd:element name="cases" type="casesType"/>
		</xsd:sequence>
		<xsd:attribute name="referenceId" type="xsd:string" use="optional"/>
	</xsd:complexType>
	<xsd:complexType name="orderlineType">
		<xsd:sequence>
```

```
                <xsd:element name="quantity" type="xsd:int"/>
                <xsd:element name="timesValid" type="xsd:int" minOccurs="0"/>
                <xsd:element name="price" type="xsd:integer" minOccurs="0"/>
                <xsd:element name="action" type="xsd:string" minOccurs="0"/>
                <xsd:element name="validPeriod" type="periodType" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="code" type="xsd:string" use="required"/>
        <xsd:attribute name="ticket-ref" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="periodType">
        <xsd:attribute name="from" type="orderDateTime" use="required"/>
        <xsd:attribute name="to" type="orderDateTime" use="required"/>
</xsd:complexType>
<xsd:complexType name="orderlinesType">
        <xsd:sequence>
                <xsd:element name="orderline" type="orderlineType" maxOccurs="unbounded"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="phoneType">
        <xsd:sequence>
                <xsd:element name="device" type="deviceType"/>
        </xsd:sequence>
        <xsd:attribute name="number" use="required">
                <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                                <xsd:pattern value="[+](\d{8,20})"/>
                        </xsd:restriction>
                </xsd:simpleType>
        </xsd:attribute>
</xsd:complexType>
<xsd:simpleType name="orderDateTime">
        <xsd:annotation>
                <xsd:appinfo>
                        <jxb:javaType name="java.util.Date"
                                parseMethod="com.lcmg.common.xml.MyDatatypeConverter.parseDateTime"
                                printMethod="com.lcmg.common.xml.MyDatatypeConverter.printDateTime" />
                </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>
<xsd:complexType name="orderPropertiesType">
        <xsd:sequence>
                <xsd:element name="property" type="orderPropertyType" maxOccurs="unbounded"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="orderPropertyType">
        <xsd:attribute name="name" use="required">
                <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="FIRST_NAME"/>
                                <xsd:enumeration value="INFIX"/>
                                <xsd:enumeration value="LAST_NAME"/>
                                <xsd:enumeration value="STREET"/>
                                <xsd:enumeration value="HOUSE_NUMBER"/>
                                <xsd:enumeration value="CITY"/>
                                <xsd:enumeration value="POSTAL_CODE"/>
                                <xsd:enumeration value="COUNTRY"/>
                                <xsd:enumeration value="EMAIL"/>
                                <xsd:enumeration value="PHONE_NUMBER"/>
                        </xsd:restriction>
                </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="ticketPropertiesType">
        <xsd:sequence>
                <xsd:element name="property" type="ticketPropertyType" maxOccurs="unbounded"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ticketPropertyType">
```

```
            <xsd:attribute name="name" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="TICKET_TEXT"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="value" type="xsd:string" use="required"/>
        </xsd:complexType>
        <xsd:complexType name="ticketType">
            <xsd:sequence>
                <xsd:element name="contentCode" type="xsd:string" minOccurs="0"/>
                <xsd:element name="properties" type="ticketPropertiesType" minOccurs="0"/>
                <xsd:element name="deliveryAddresses" type="deliveryAddressesType" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="ref" type="xsd:string" use="required"/>
            <xsd:attribute name="delivery-channel" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="SMS"/>
                        <xsd:enumeration value="EMAIL"/>
                        <xsd:enumeration value="WEB"/>
                        <xsd:enumeration value="NONE"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="delivery-format" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="BARCODE"/>
                        <xsd:enumeration value="TEXTCODE"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
        </xsd:complexType>
        <xsd:complexType name="ticketsType">
            <xsd:sequence>
                <xsd:element name="ticket" type="ticketType" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
</xsd:schema>
```

4.2 Example Request XML

4.2.1 Sending tickets only

```
<?xml version="1.0" encoding="UTF-8"?>
<orderRequest version="1.0">
    <client code="CLIENT_X">
        <username>defaultuser</username>
        <password>defaultpw</password>
    </client>
    <tickets>
        <ticket ref="" delivery-channel="SMS" delivery-format="BARCODE">
            <contentCode>1234</contentCode>
            <properties>
                <property name="TICKET_TEXT" value="This is your ticket for..."/>
            </properties>
            <deliveryAddresses>
                <deliveryAddress>
                    <phone number="+31612345678">
                        <device>
                            <manufacturer>NOKIA</manufacturer>
                            <type>7210</type>
                        </device>
                    </phone>
                </deliveryAddress>
            </deliveryAddresses>
        </ticket>
        <ticket ref="" delivery-channel="SMS" delivery-format="BARCODE">
            <contentCode>5678</contentCode>
            <properties>
                <property name="TICKET_TEXT" value="This is your ticket for..."/>
            </properties>
            <deliveryAddresses>
                <deliveryAddress>
                    <phone number="+31687654321">
                        <device>
                            <manufacturer>NOKIA</manufacturer>
                            <type>3310</type>
                        </device>
                    </phone>
                </deliveryAddress>
            </deliveryAddresses>
        </ticket>
    </tickets>
</orderRequest>
```

4.2.2 Sending tickets and storing order

```
<?xml version="1.0" encoding="UTF-8"?>
<orderRequest version="1.0">
	<client code="CLIENT_X">
		<username>defaultuser</username>
		<password>defaultpw</password>
	</client>
	<order referenceId="">
		<properties>
			<property name="FIRST_NAME" value=""/>
			<property name="EMAIL" value="customer@mail.nl"/>
		</properties>
		<cases>
			<case code="CASE_X_STD">
				<orderlines>
					<orderline code="ORD_LINE_TYPE_X_STD" ticket-ref="23">
						<quantity>2</quantity>
						<price>1250</price>
						<action>Action description</action>
						<validPeriod from="2009-04-02T09:00:00" to="2009-05-02T18:30:00"/>
					</orderline>
					<orderline code="ORD_LINE_TYPE_Y_STD" ticket-ref="23">
						<quantity>1</quantity>
						<price>1000</price>
						<action>Action description</action>
						<validPeriod from="2009-04-02T09:00:00" to="2009-05-02T18:30:00"/>
					</orderline>
				</orderlines>
			</case>
		</cases>
	</order>
	<tickets>
		<ticket ref="23" delivery-channel="WEB" delivery-format="BARCODE">
		</ticket>
	</tickets>
</orderRequest>
```

4.3 Order Response Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="code">
        <xsd:complexType/>
    </xsd:element>
    <xsd:element name="description">
        <xsd:complexType/>
    </xsd:element>
    <xsd:complexType name="errorType">
        <xsd:sequence>
            <xsd:element ref="code"/>
            <xsd:element ref="description"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="orderResponse">
        <xsd:complexType>
            <xsd:sequence minOccurs="0">
                <xsd:choice>
                    <xsd:element name="error" type="errorType"/>
                    <xsd:element name="comment" minOccurs="0">
                        <xsd:complexType>
                            <xsd:simpleContent>
                                <xsd:extension base="xsd:string">
                                    <xsd:attribute name="type" type="xsd:string" use="required"/>
                                </xsd:extension>
                            </xsd:simpleContent>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:choice>
            </xsd:sequence>
            <xsd:attribute name="status" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="OK"/>
                        <xsd:enumeration value="NOT_OK"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

4.4 Order Interface XML Structure from a different perspective

4.4.1 Overview on Client in relation to orderRequest

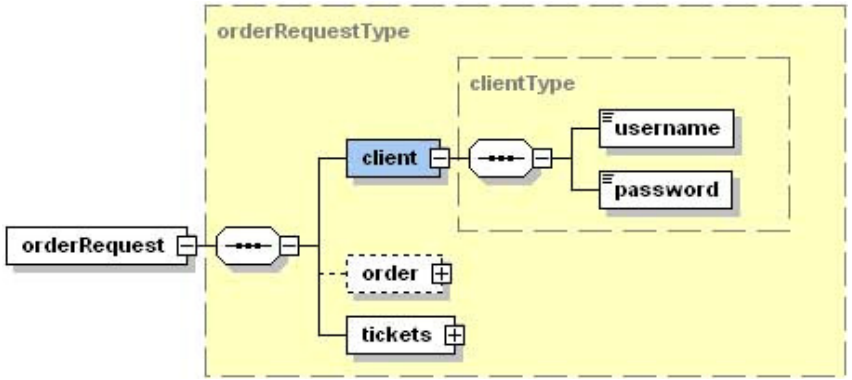

Figure 3.4.1 orderRequest - client

4.4.2 Overview on Order in relation to orderRequest

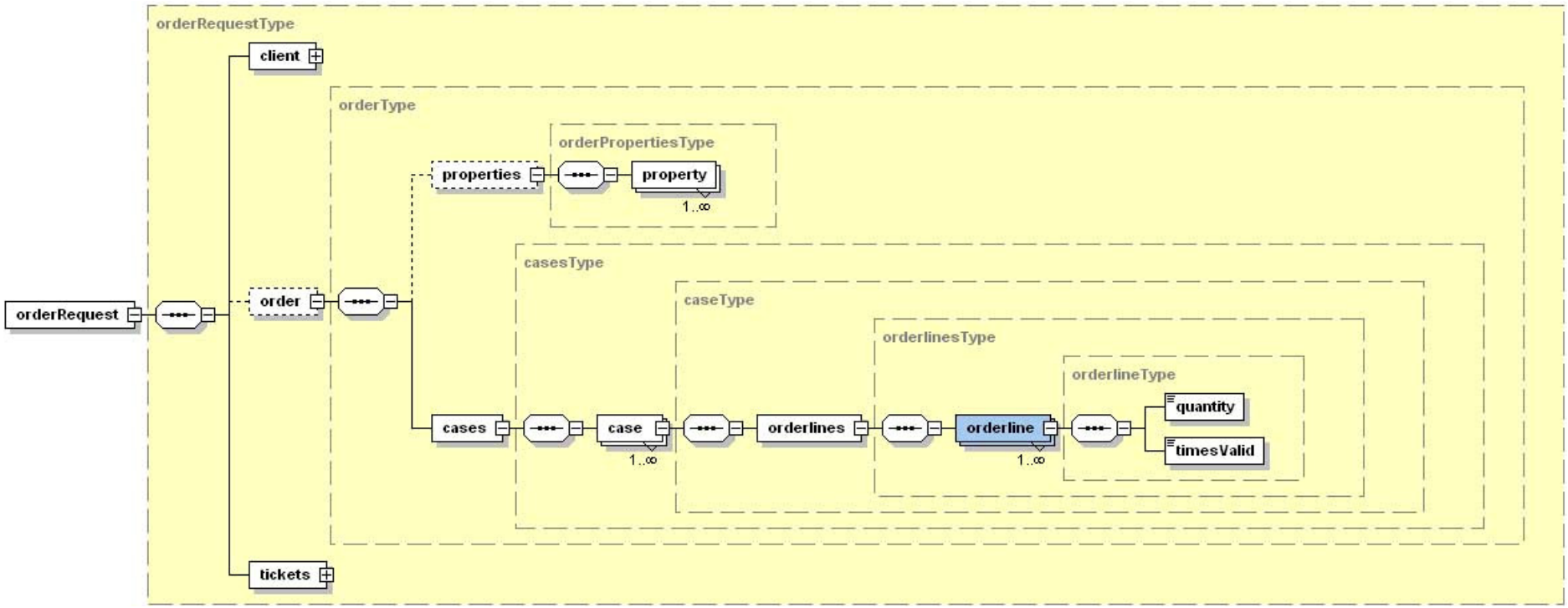

Figure 3.4.2 orderRequest - order

4.4.3 Overview on Tickets in relation to orderRequest

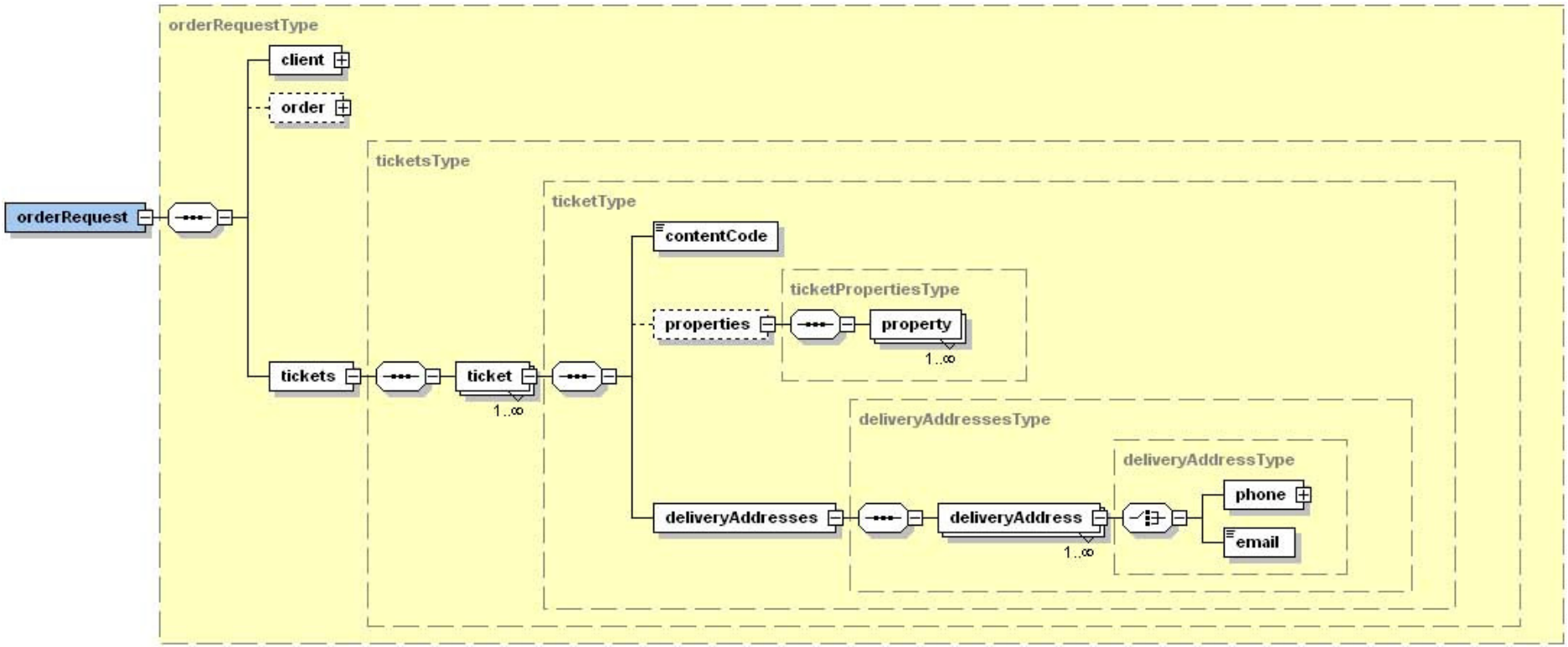

Figure 3.4.3 orderRequest - tickets

5 Appendix B

	Configuration
OI_0001	Internal server error
OI_0099	Unknown error
	Client credentials are wrong
OI_0101	Supplied client not found
OI_0102	Interface not enabled for client
	XML request contains errors
OI_0201	Submitted client not found
OI_0202	Interface not enabled for client
OI_0203	Missing (or empty) order reference id attribute.
OI_0204	Request contains invalid orderline.ticket-ref <-> ticket.ref link reference(s)
OI_0205	Unknown delivery channel
OI_0206	Unknown delivery format
OI_0207	Device manufacturerer/type is not known
OI_0208	The delivery address is not specified
OI_0209	The submitted data for order reference id (max 32 chars), or one of the properties (max 100 chars) is too long.
OI_0210	Submitted number of persons is negative or 0
OI_0211	Submitted times valid is negative or 0
OI_0212	Supplied email address is missing or the format is not correct
OI_0213	The submitted orderline type does not belong to the client
OI_0214	Missing required orderlinetype
OI_0215	Orderline type is not owned by the submitted case
OI_0216	Could not generate ticket code for request, all ticket codes are in use (batch full).