OpenVPMS ESCI

Integration Guide 1.0-beta-3

Contents

1 Overview

OpenVPMS ESCI (e-Supply Chain Interface) is a standards-based API to enable OpenVPMS to place orders and receive order responses and invoices to and from suppliers electronically. It is based on the exchange of Universal Business Language (UBL) 2.0 documents via web services.

1.1 Contents

This initial version of ESCI provides the following:

- This Integration document.
- Registry API
- Order placement API
- Inbox API
- Web Service Description Language (WSDL) and Extensible Markup Language (XML) Schema Definitions (XSD) files, defining the web services
- JAX-WS interfaces, defining the web services in Java
- Java Architecture for XML Binding (JAXB) wrappers for the UBL documents
- Example implementations of the ESCI web services
- Example clients, to submit UBL documents to the ESCI web services
- Example Order, OrderResponseSimple and Invoice UBL documents
- Java classes to submit UBL documents to ESCI web services

1.2 How it Works

ESCI defines a number of web services which are implemented by the supplier, namely:

- Order – enables clients to submit orders to the supplier.
- Inbox – enables clients to receive documents posted by the supplier
- Registry - provides a simple lookup mechanism to resolve the other supplier web services. For this release, these are the Order and Inbox services.

OpenVPMS uses the specific supplier's Registry web service to find the URLs of the supplier's Order and Inbox web services

OpenVPMS creates UBL Order documents and submits these to the supplier via the supplier's Order web service.

Orders are processed asynchronously by the supplier.

A UBL OrderResponseSimple document is placed in practice's Inbox, indicating success or failure.

When an order has been shipped, the supplier invoices the practice by placing an UBL Invoice document in its Inbox.

OpenVPMS checks the Inbox web service regularly to process any incoming Order Response and/or Invoice documents.

OpenVPMS notifies the users of incoming documents using messages.

OpenVPMS also processes these documents by either updating order status or by generating new deliveries.

The Practice actions the messages and processes the delivery information as per their internal procedures.

1.3 Development Environment

ESCI is based around open standards and the API's can be implemented on any platform or development environment that supports development of Web services as per WS-I Basic Profile 1.1 (http://www.ws-i.org/Profiles/BasicProfile-1.1.html)

The ESCI example implementation and clients are written in Java and will operate on any platform that supports:

- Java 1.6.x or higher

  See http://www.oracle.com/technetwork/java/javase/downloads/index.html

- Apache Tomcat

  At the time of writing, Apache Tomcat 7 was available, although Apache Tomcat 5.5 and higher should be compatible.

  See http://tomcat.apache.org/download-70.cgi

To install a complete test environment suitable for integration testing with OpenVPMS you will need:

- OpenVPMS 1.5

  As of writing, only an interim version of OpenVPMS-1.5 was available. This must be used with ESCI 1.0-beta-3 – earlier versions are not compatible. This is available from http://repository.openvpms.org/releases/org/openvpms/openvpms-release/1.5-beta-3/openvpms-release-1.5-beta-3.zip

  With the exception of the Java version above, the installation requirements are the same as those documented at http://www.openvpms.org/openvpms-installation-windows

  Installation for other platforms is documented at: http://www.openvpms.org/implementors-handbook

- Test Database

  The OpenVPMS 1.5 release includes a demo database that can be used for this purpose. To load this database first make sure you have run the **dataload setup** command as outlined in the **OpenVPMS Database Setup** section of the Installation notes referenced above**.**

  Then run the following command to load the demo data:

```
dataload.bat -d ../import/data/demo -s
```

1.4 Supplier Environment

In order to do full integration testing and production setup with OpenVPMS the supplier needs to have developed and deployed the web services as described in this document.

The supplier will also need to setup suitable authentication and account details for the veterinary practice.

Suppliers may choose to implement ESCI as secure web services accessible over the internet. Alternatively, they may choose to deploy an application in the practice which provides these services to OpenVPMS – this application would then utilise their pre-existing communication protocol.

1.5 Veterinary Practice Environment

In order for the Veterinary Practice to utilise ESCI services they will need:

- OpenVPMS 1.5
- Internet access (dependent on the supplier mode of deployment)
- Details of the supplier ESCI Registry service URL, authentication and Account details.
- Optional: Supplier installed and configured middleware application.
- OpenVPMS Supplier configuration as per the notes included in this document.

2 Installing the ESCI Package

The ESCI package is contained in a zip named:

    openvpms-esci-package-<version>.zip

In the above, ***<version>*** indicates the release version.

Extract this using your favourite zip program, or the Java jar tool:

```
jar xvf openvpms-esci-package-<version>.zip
```

This will extract to a directory named openvpms-esci-package-***<version>.***

For the rest of this document:

• ***<ESCI_HOME>***	Refers to the base directory where the ESCI package was installed. This must be replaced with the actual directory path on the command line, and in configuration files.
• ***<TOMCAT_HOME>***	Refers to Tomcat's base directory. This must be replaced with the actual directory path on the command line, and in configuration files.
• **JAVA_HOME**	Refers to the environment variable pointing to the base directory of the java installation. This is configured as part of the OpenVPMS installation

In shells and configuration files, they need to be replaced with their actual values.

Directory Structure

The package has the following sub-directories:

doc/	ESCI documentation
bin/	contains scripts for running the client examples
conf/	contains configuration files for the examples
data/	contains sample UBL documents
lib/	contains jars used by the examples in bin/
src/	contains the source code for the examples
webapp/	contains the example web application
wsdl/	contains the WSDL and XSD files defining the ESCI web services

2.2 Configuring Tomcat

SSL Configuration

Tomcat needs to be configured to use SSL (Secure Socket Layer). This is a technology which allows web browsers and web servers to communicate over a secured connection. To simplify things, the examples come with a self-signed certificate "tomcat", contained in the keystore: ***<ESCI_HOME>***`/conf/tomcat.jks`.

This certificate only works on "localhost".

Edit `<CATALINA_HOME>/conf/server.xml` and uncomment and change the SSL connector:

```
<-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
           maxThreads="150" scheme="https" secure="true"
           keystoreFile="<ESCI_HOME>/conf/tomcat.jks"
           keystorePass="changeit"
           clientAuth="false" sslProtocol="TLS" />
```

The certificate also needs to be imported into the Java runtime's **`cacerts`** file. Take a backup of the file first, so it can be restored after testing is complete.

Windows[1]:

```
keytool -exportcert -alias tomcat -keystore tomcat.jks -storepass changeit -file tomcat.cert
keytool -importcert -alias tomcat -keystore %JAVA_HOME%\jre\lib\security\cacerts   \
    -storepass changeit -file tomcat.cert
```

Enter "yes" when prompted, i.e.:

```
Trust this certificate? [no]:  yes
```

A longer discussion on SSL and key stores can be found at http://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html

User Configuration

For these examples, a user needs to be added to Tomcat's user database.

The default tomcat installation stores user information in:

<TOMCAT_HOME>`/conf/tomcat-users.xml`

Add the following entries:

```
    <role rolename="ESCI"/>
    <user username="esci" password="example" roles="ESCI"/>
```

See http://tomcat.apache.org/tomcat-7.0-doc/realm-howto.html#UserDatabaseRealm for more details.

2.3 Example web application installation

To install the example web application, copy:

<ESCI_HOME>/`webapp/esci-example.war`

To the directory:

[1] Ignore the trailing \ character. This is to indicate that the command needs to be entered on a single line

```
<TOMCAT_HOME>/webapps
```

2.4 Testing the installation

To test the installation, start Tomcat so that it logs to the current console window:

Windows:

```
cd <TOMCAT_HOME>\bin
catalina.bat run
```

Unix:

```
cd <TOMCAT_HOME>/bin
catalina.sh run
```

Now open up a web browser and enter the URL[2]:

```
https://localhost:8443/esci-example/RegistryService?wsdl
```

A dialog should pop up, requesting a user name and password. Enter `'esci'` and `'example'` respectively (without quotes).

The WSDL for the RegistryService should be displayed.

[2] NOTE: Your browser will warn you that the connection is untrusted as the security certificate wasn't issued by a trusted certificate authority. For the purposes of testing, this can be ignored. In a production environment, the certificate needs to be issued by a trusted authority, to simplify deployment

3 Using ESCI Examples

3.1 Submitting orders

To submit an order:

```
cd <ESCI_HOME>/bin
order.bat -f ../data/order1.xml -u esci -p example
```

The example OrderService will:

- log the order to the console
- create a response
- create an invoice for the order

This connects to the default registry URL. To override this, specify **–r** on the command line, e.g.:

```
order.bat -f ../data/order1.xml -u esci -p example \
    -r https://localhost:8443/esci-example/RegistryService
```

3.2 Checking for responses

Both order responses and invoices are placed in the inbox of the practice placing the order.

To check responses:

```
cd <ESCI_HOME>/bin
inbox.bat -l -u esci -p example
```

This will list available documents, e.g.:

```
Available documents: 2

<cac:DocumentReference
    xmlns="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2”
    xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2">
    <ID>21</ID>
    <DocumentType>OrderResponseSimple</DocumentType>
</cac:DocumentReference>

<cac:DocumentReference
    xmlns="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2”
    xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2">
    <ID>22</ID>
    <DocumentType>Invoice</DocumentType>
</cac:DocumentReference>
```

3.3 Retrieving documents from the Inbox

Documents can be retrieved by specifying their identifier and type, e.g.:

```
cd <ESCI_HOME>/bin
inbox.bat -g -i 21 -t OrderResponseSimple -u esci -p example
```

If available, the corresponding document will be displayed.

3.4 Acknowledging receipt of documents in the Inbox

Documents will continue to be returned by the InboxService until acknowledged. To acknowledge a document:

```
cd <ESCI_HOME>/bin
inbox.bat -a -i 21 -t OrderResponseSimple -u esci -p example
```

3.5 Posting documents to the Inbox

Documents can be posted to the Inbox as follows:

```
cd <ESCI_HOME>/bin
post.bat -f ../data/response1.xml -u esci -p example
```

Note that this service is not part of the ESCI API, and exists solely for testing purposes.

3.6 Validating documents

Documents can be validated using the **validate** tool. This validates documents against their UBL schema. Note that it does not validate that documents conform to the ESCI specification, nor does it check that the content is consistent.

E.g.:

```
cd <ESCI_HOME>/bin
validate.bat -f ../data/invoice1.xml
```

For a valid document this produces:

```
Document is valid.
```

For an invalid document, it displays the error, and any line information, if available. E.g.:

```
Document is invalid.
Line 62, column 34: cvc-complex-type.2.4.b: The content of element 'cac:OrderLineReference' is
not complete. One of '{"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-
2":LineID}' is expected.
```

4 OpenVPMS ESCI Configuration and Testing

The following describes how to setup OpenVPMS to use ESCI.

For testing purposes the sample web services will be used to receive orders and generate order responses and invoices.

The sample services provide limited validation of documents – they simply check that documents conform to their relevant UBL schema.

It will be necessary for suppliers to perform full integration testing between OpenVPMS and their ESCI service implementations in order to completely validate the operation of their services.

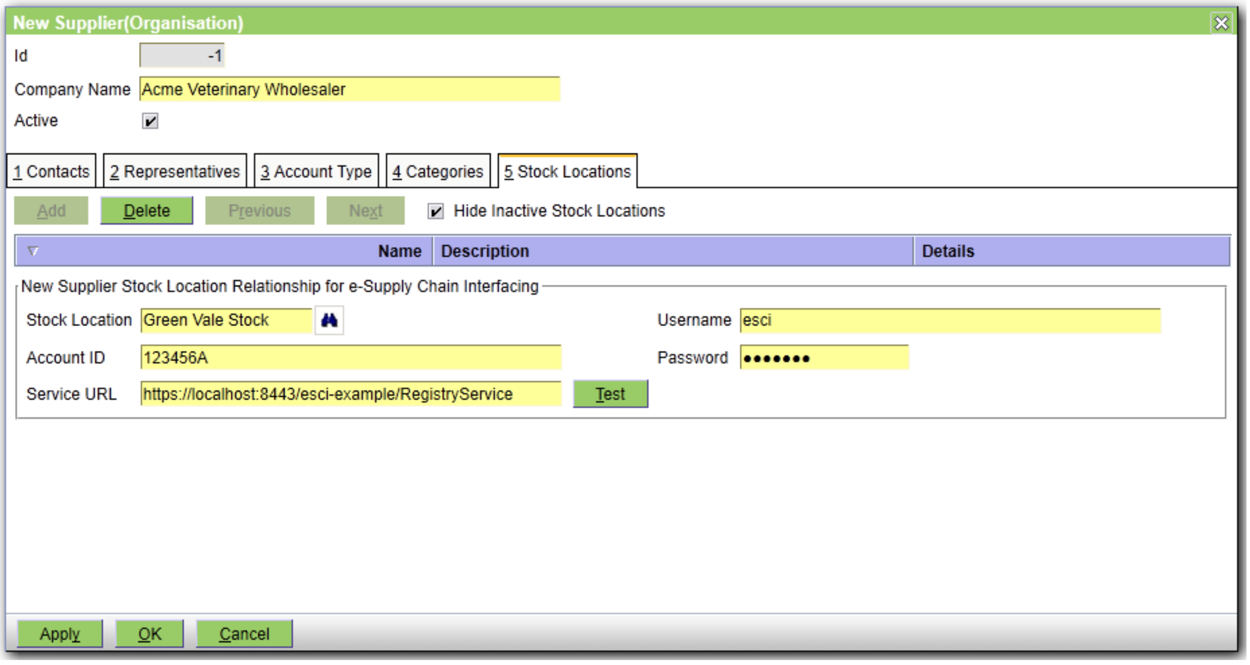
4.1 Configuring OpenVPMS

In order for OpenVPMS to send orders to the sample OrderService, it must first be configured. Login to the web application at http://localhost:8080/openvpms/login as user "admin", password "admin" and:

- Configure a supplier for ESCI

  A number of suppliers are included in the sample data. Go to the **Suppliers→Information** workspace, select one, and edit it. Alternatively, a new supplier can be created via the **New** button.

  - In the "Stock Locations" tab, click **Add** to add a new "Supplier Stock Location Relationship for e-Supply Chain Interfacing"
  - Select "Main Stock Clinic" for the Stock Location
  - Enter an Account ID.
  - Enter "https://localhost:8443/esci-example/RegistryService" for the Service URL
  - Enter "esci" for Username
  - Enter "example" for Password
  - Click **Test to** verify the details. A message "Successfully connected to supplier" should be displayed.
  - Click **OK** to finish editing the supplier


New Supplier(Organisation)

Id -1

Company Name Acme Veterinary Wholesaler

Active ✓

✓ Hide Inactive Stock Locations

New Supplier Stock Location Relationship for e-Supply Chain Interfacing

Stock Location Green Vale Stock

Username esci

Account ID 123456A

Password •••••••

Service URL https://localhost:8443/esci-example/RegistryService

Notes:

1. The Account ID is used to populate the SupplierAssignedAccountID in the Order document. Although not necessary for the example services, this may be required by the receiving supplier in a production environment.
2. OpenVPMS supports multiple stock locations. Each stock location may be associated with multiple suppliers. The supplier/stock location relationship determines which service URL and authentication details to use to connect to the supplier. These details must correspond to a unique Inbox; i.e. documents posted to the Inbox must be for the supplier/stock location only.
3. In the case where a supplier requires different account identifiers based on the products being ordered, e.g. Medication vs. Food, a separate supplier must be created.

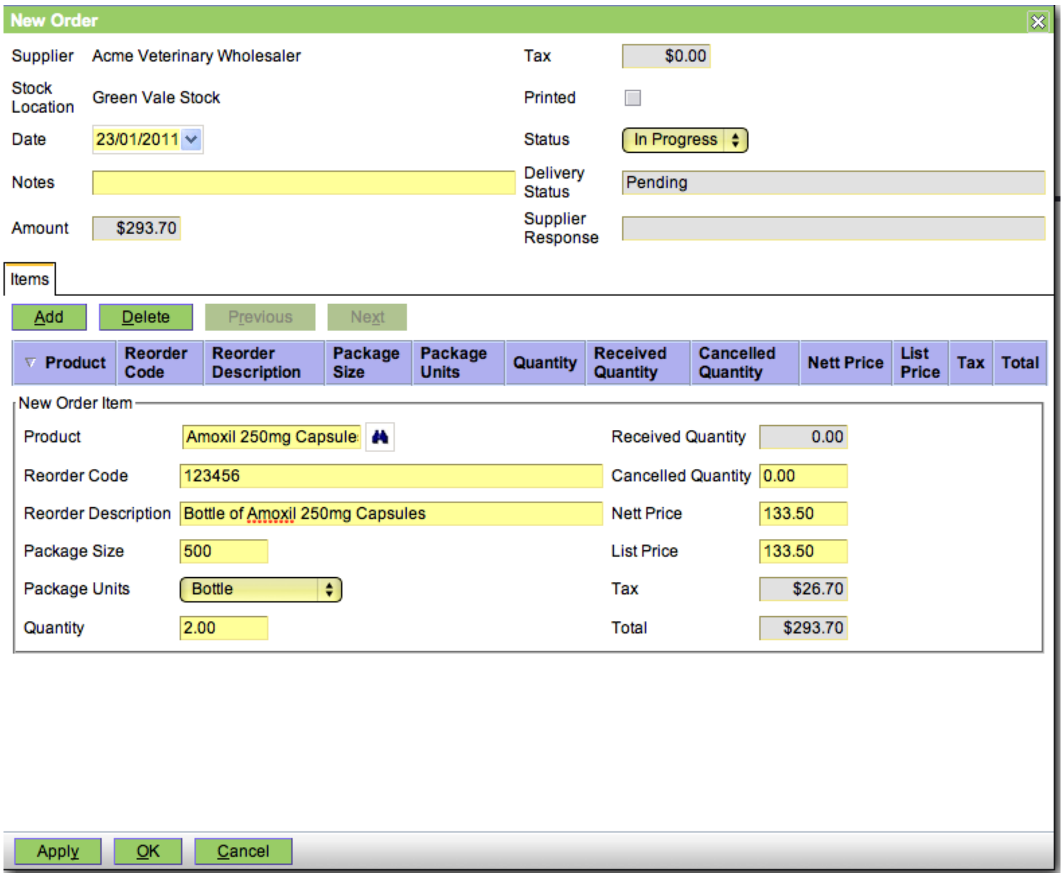
4.2 Sending an Order

- Create an order

  Ordering is done in the **Suppliers → Orders** workspace. Click **New** to create a new order**.**

  - In the “New Order” dialog select the supplier edited above.
  - Select “Main Stock Clinic” for Stock Location
  - Click **OK** to launch the order editor
- Edit the order
  - Click **Add** on the “Items” tab to add a new order item
  - Select a product for Product, e.g. “Amoxil 250mg Capsules”
  - Enter a Reorder Code e.g. “123456”
  - Enter a package size e.g. 500
  - Enter a package units e.g. Bottle
  - Enter a Quantity e.g. 2
  - Enter a Nett Price e.g. 133.50
  - Enter a List Price e.g. 133.50

- Click **OK** to save the order and finish editing



New Order

Supplier	Acme Veterinary Wholesaler	Tax	$0.00
Stock Location	Green Vale Stock	Printed	☐
Date	23/01/2011	Status	In Progress
Notes		Delivery Status	Pending
Amount	$293.70	Supplier Response	

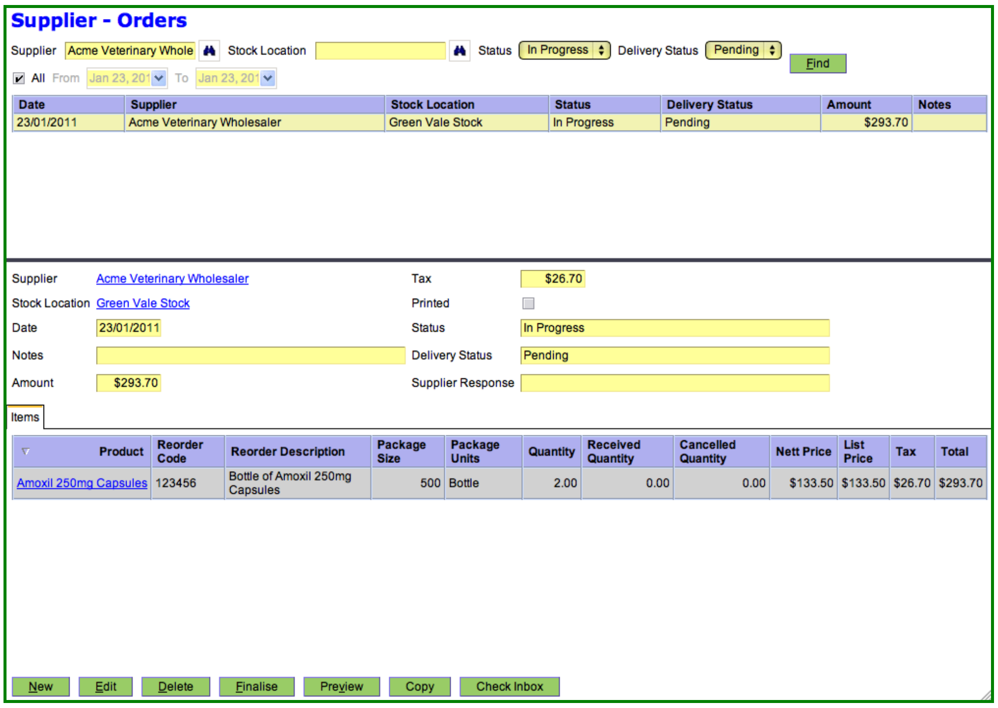
Items

Product	Reorder Code	Reorder Description	Package Size	Package Units	Quantity	Received Quantity	Cancelled Quantity	Nett Price	List Price	Tax	Total

New Order Item

Product	Amoxil 250mg Capsule	Received Quantity	0.00
Reorder Code	123456	Cancelled Quantity	0.00
Reorder Description	Bottle of Amoxil 250mg Capsules	Nett Price	133.50
Package Size	500	List Price	133.50
Package Units	Bottle	Tax	$26.70
Quantity	2.00	Total	$293.70

- Finalise the order
  - In the supplier orders table, select the order and click **Finalise**
  - Click **OK** on the **"**Are you sure you want to Finalise the Order?" prompt



Supplier - Orders

Supplier: Acme Veterinary Whole | Stock Location: | Status: In Progress | Delivery Status: Pending

☑ All From Jan 23, 201 To Jan 23, 201

Date	Supplier	Stock Location	Status	Delivery Status	Amount	Notes
23/01/2011	Acme Veterinary Wholesaler	Green Vale Stock	In Progress	Pending	$293.70	

Supplier	Acme Veterinary Wholesaler	Tax	$26.70
Stock Location	Green Vale Stock	Printed	☐
Date	23/01/2011	Status	In Progress
Notes		Delivery Status	Pending
Amount	$293.70	Supplier Response	

Items

Product	Reorder Code	Reorder Description	Package Size	Package Units	Quantity	Received Quantity	Cancelled Quantity	Nett Price	List Price	Tax	Total
Amoxil 250mg Capsules	123456	Bottle of Amoxil 250mg Capsules	500	Bottle	2.00	0.00	0.00	$133.50	$133.50	$26.70	$293.70

4.3 Checking the Inbox

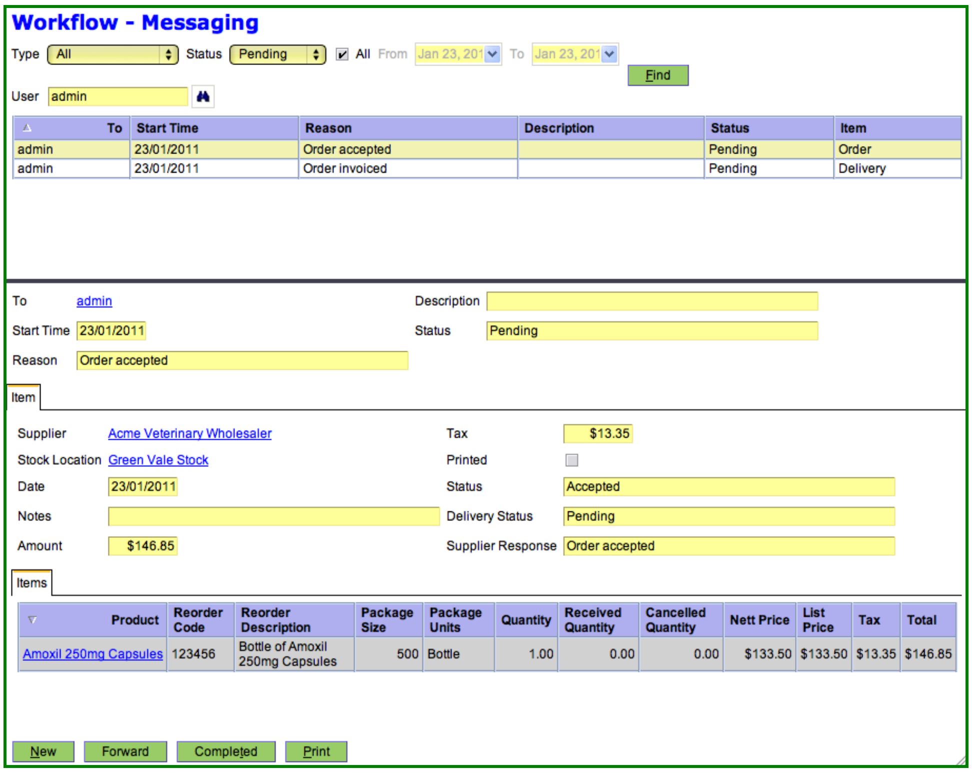
On receiving the Order the ESCI example services will do two things:

- Display the original order document on the Tomcat console.
- Queue an UBL **OrderResponseSimple** and an **Invoice** document in the Example ESCI Inbox service ready to be picked up by OpenVPMS when it next checks this service.

OpenVPMS periodically checks each configured Inbox. By default this is scheduled to occur once daily.

You can also manually check for documents in the **Suppliers → Orders** and **Supplier → Deliveries** Workspaces by clicking on the **Check Inbox** button.

On clicking this button you should note that the Messages icon in the menu bar indicates new messages. Clicking this icon will take you to the **Workflow → Messages** workspace where you should see two (2) unread messages.



Workflow - Messaging

Type: All | Status: Pending | ☑ All | From: Jan 23, 201 | To: Jan 23, 201 | Find

User: admin

To	Start Time	Reason	Description	Status	Item
admin	23/01/2011	Order accepted		Pending	Order
admin	23/01/2011	Order invoiced		Pending	Delivery

To	admin	Description	
Start Time	23/01/2011	Status	Pending
Reason	Order accepted		

Item

Supplier	Acme Veterinary Wholesaler	Tax	$13.35
Stock Location	Green Vale Stock	Printed	☐
Date	23/01/2011	Status	Accepted
Notes		Delivery Status	Pending
Amount	$146.85	Supplier Response	Order accepted

Items

Product	Reorder Code	Reorder Description	Package Size	Package Units	Quantity	Received Quantity	Cancelled Quantity	Nett Price	List Price	Tax	Total
Amoxil 250mg Capsules	123456	Bottle of Amoxil 250mg Capsules	500	Bottle	1.00	0.00	0.00	$133.50	$133.50	$13.35	$146.85

One message relates to the received **OrderResponseSimple** document, and should have a reason of **Order Accepted**. The message displays the order that the response relates to, indicating that it now has a status of **Accepted**. The updated order can also be seen in the **Suppliers→Orders** workspace.

The second message relates to the received **Invoice** document, and should have a reason of **Order Invoiced**. OpenVPMS generates deliveries for Invoice documents - the message displays this delivery, with a status of **In Progress**. This can also be seem in the **Suppliers→Deliveries** workspace

OpenVPMS does not finalise the delivery automatically. This allows the delivery information to be reviewed and modified by the practice staff should there be any discrepancies with the physical delivery received by the practice.

On Finalising the Delivery, OpenVPMS will update stock quantities and purchase prices and perform any selling price updates required. The latter is determined by whether list prices have changed and if the specific Product Supplier information stipulates Auto price updates.

5 Implementing the API

5.1 Overview

In order to implement ESCI, the following web services must be developed:

- RegistryService
- OrderService
- InboxService

For Java, an interface for each service is provided. The java API documentation for these interfaces is located in:

```
<ESCI_HOME>/doc/api/index.html
```

For other platforms, code for the services can be generated from the WSDL files provided in the **wsdl/** directory of the examples package.

For .NET, this can be done using the svcutil tool. See http://msdn.microsoft.com/en-us/library/aa751905.aspx for more details.

In Visual Studio, this can be done by adding a reference to the web service as per: http://msdn.microsoft.com/en-us/library/bb628649.aspx.

5.2 RegistryService

The RegistryService provides a simple means to locate the other web services.

Method	Description	
getInboxService	Returns the InboxService URL.	
Parameters	None	
Returns	String	The InboxService URL
getOrderService	Returns the OrderService URL.	
Parameters	None	
Returns	String	The OrderService URL

5.3 OrderService

The OrderService enables the authenticated user to place orders with the supplier.

Method	Description
submitOrder	Submits an order to the supplier.

	If the order is accepted, an **OrderResponseSimple** document will be placed in the customer's corresponding Inbox, with **AcceptedIndicator=true**. If the order is rejected, the **OrderResponseSimple** document have **AcceptedIndicator=false** and the reason for rejection described in the **RejectionNote**. If the submit fails for any reason other than **DuplicateOrderException**, the client may resubmit the Order.	
Parameters	order:Order	The order to submit
Returns	None	
Throws	DuplicateOrderException	If the order has an identifier the same as another submitted previously

5.4 InboxService

The InboxService enables authenticated users to access documents placed in their Inbox by the supplier.

The supplier places OrderResponseSimple documents in a user's Inbox on receipt of an Order.

If an order is accepted, the supplier will place an Invoice document in the Inbox. This may happen immediately, or at some later stage, according to the supplier's business rules.

Method	**Description**	
getDocuments	Returns a list of references to documents in the Inbox. Each reference must refer to a unique document. The references must be in the same order that the documents were added, with the oldest document referenced first. For this release, only Invoice and OrderResponseSimple documents may be referenced.	
Parameters	None	
Returns	List of DocumentReference	
getDocument	Returns a document, given its reference.	
Parameters	reference:DocumentReference	The document reference
Returns	Document	The corresponding document, or null if none is found. For this release, only documents containing Invoice or OrderResponseSimple may be returned.
Acknowledge	Acknowledges a document. This removes the document from the Inbox. It will no longer be returned by **getDocument()** nor **getDocuments()**.	
Parameters	reference:DocumentReference	The document reference
Returns	None	
Throws	DocumentNotFoundException	If the reference doesn't refer to a valid document

5.5 Web Service Description Language (WSDL)

Each web service provides a description of itself, in the form of an XML document. This is typically accessed by performing an HTTP GET on a web service's URL with "?wsdl" appended.

E.g.:

https://localhost:8443/esci-example/RegistryService?wsdl

ESCI accesses the remote service's WSDL to ensure that it can communicate successfully with the service. In particular, it enables the appropriate SOAP version to be used (1.1 or 1.2). In future, it may be used to determine what services a supplier provides.

NOTE: It is important that this WSDL not be password protected. This can be achieved by allowing GET requests without authentication.

This requirement is to get around a limitation in JAX-WS which requires a JVM wide java.net.Authenticator to be registered to handle authentication of the WSDL. This limits the deployment options of ESCI, as multiple deployments in the one JVM aren't directly supported.

6 Document Exchange Scenarios

The following scenarios detail the document exchanges supported by ESCI.

For Order-Invoicing, they range from the simple One Order, One Invoice, to the more complex One Order, Multiple Invoices and One Invoice, Multiple Orders. There are a number of variants on these, namely One Order, One Invoice, unreferenced OrderLine(s) and One Invoice, Multiple Orders, unspecified Order Lines.

A supplier may reject an Order if it is incorrect or can't be fulfilled using Rejected Order.

Invoicing without a corresponding Order is supported by Invoice without Order.

These provide suppliers a great deal of flexibility to integrate ESCI with existing business processes.

6.1 One Order, One Invoice

In this scenario, OpenVPMS sends an Order to the supplier, who responds with an OrderResponseSimple with AcceptedIndicator=true, and later, an Invoice.

The Invoice has a reference to the Order (see Invoice.OrderReference), and each line of the Invoice references the corresponding Order line (see Invoice.InvoiceLine.OrderLineReference).

The OrderLineReferences need not specify an OrderReference – this is implied by the document-level OrderReference. If they do specify an OrderReference, it **must** correspond to that of the document-level OrderReference.

6.2 Rejected Order

In this scenario, OpenVPMS sends an Order to the supplier, who responds with an OrderResponseSimple, with AcceptedIndicator=false. The reason for the rejection is detailed in the RejectionNote.

OpenVPMS may change the order and resubmit it.

6.3 One Order, One Invoice, unreferenced OrderLine(s)

This is the same scenario as One Order, One Invoice, except that the Invoice has one or more InvoiceLines that have no OrderLineReference.

This can be used if products have been added by the supplier that weren't present in the original order.

6.4 One Order, Multiple Invoices

In this scenario, OpenVPMS sends an Order to the supplier, who responds with an OrderResponseSimple with AcceptedIndicator=true. Later, the supplier sends multiple Invoices, each referencing part of the original Order.

This can be used in the situation where an ordered item is on back order.

Alternatives to this approach include:

- Rejecting the original order if it cannot be fulfilled
- Delaying invoicing until the order can be fulfilled
- Generating a single Invoice with the expectation that it will be fulfilled later

6.5 Invoice without Order

In this scenario, the supplier sends an Invoice which has no associated Order. This might be used in the situation where an order is placed by phone, for example.

6.6 One Invoice, Multiple Orders

In this scenario, the supplier sends one Invoice that references multiple Orders. When referencing multiple Orders, the OrderReference **must not** be specified at the document level. It must be specified at the InvoiceLine level in the OrderLineReference instead. E.g.:

```
<cac:InvoiceLine>
    <cbc:ID>1</cbc:ID>
    <!-- snip -->
    <cac:OrderLineReference>
        <cbc:LineID>10</cbc:LineID>
        <cac:OrderReference>
            <cbc:ID>631</cbc:ID>
         </cac:OrderReference>
    </cac:OrderLineReference>
    <!-- snip -->
</cac:InvoiceLine>
<cac:InvoiceLine>
    <cbc:ID>2</cbc:ID>
    <!-- snip -->
    <cac:OrderLineReference>
        <cbc:LineID>61</cbc:LineID>
        <cac:OrderReference>
            <cbc:ID>647</cbc:ID>
         </cac:OrderReference>
    </cac:OrderLineReference>
    <!-- snip -->
</cac:InvoiceLine>
```

Note that this scenario may be used even if only one Order is referenced. In this case, it differs from One Order, One Invoice in that:

- the document-level order reference is not specified
- the line-level OrderLineReference specifies the OrderReference

6.7 One Invoice, Multiple Orders, unspecified Order Lines

This is the same scenario as One Invoice, Multiple Orders except that the original order lines are not specified.

This may be used if the supplier:

- Does not support single invoices per order

- Does not preserve the original OrderLineReference

UBL requires that a LineID element be specified, so for this scenario, -1 is used for the LineID value. E.g.:

```
<cac:InvoiceLine>
    <cbc:ID>1</cbc:ID>
    <!-- snip -->
    <cac:OrderLineReference>
        <cbc:LineID>-1</cbc:LineID>
        <cac:OrderReference>
            <cbc:ID>631</cbc:ID>
         </cac:OrderReference>
    </cac:OrderLineReference>
    <!-- snip -->
</cac:InvoiceLine>
<cac:InvoiceLine>
    <cbc:ID>2</cbc:ID>
    <!-- snip -->
    <cac:OrderLineReference>
        <cbc:LineID>-1</cbc:LineID>
        <cac:OrderReference>
            <cbc:ID>647</cbc:ID>
         </cac:OrderReference>
    </cac:OrderLineReference>
    <!-- snip -->
</cac:InvoiceLine>
```

7 UBL Document Structure

OpenVPMS uses a subset of the UBL 2.0 specification. For this release, it uses the following documents:

- Order
- OrderResponseSimple
- Invoice

It also uses the UBL DocumentReference type, when referring to documents in an Inbox.

The usage of these will be specified in the following sections. Please read the notes associated with each as they provide further information on how OpenVPMS supplies and processes some of the key information in these documents.

7.1 Order

Element	Type	Use	Description
UBLVersionID	IdentifierType	1	UBL version identifier. Must be **2.0**
ID	IdentifierType	1	The buyer assigned order identifier
IssueDate	DateType	1	The date the buyer issued the order
IssueTime	TimeType	0..1	The time the buyer issued the order
BuyerCustomerParty	CustomerPartyType	1	Buyer details
SellerSupplierParty	SupplierPartyType	1	Seller details
AnticipatedMonetaryTotal	MonetaryTotalType	1	The anticipated total amounts
OrderLine	OrderLineType	1..*	The order items

Example

```
<Order xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
       xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
       xmlns="urn:oasis:names:specification:ubl:schema:xsd:Order-2">
   <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
   <cbc:ID>276</cbc:ID>
   <cbc:IssueDate>2010-03-21</cbc:IssueDate>
 + <cac:BuyerCustomerParty/>
 + <cac:SellerSupplierParty/>
 + <cac:AnticipatedMonetaryTotal/>
 + <cac:OrderLine/>
 + <cac:OrderLine/>
</Order>
```

Order.BuyerCustomerParty

Name	Type	Use	Description
CustomerAssignedAccountID	Identifier	1	The buyer assigned identifier for the customer
SupplierAssignedAccountID	Identifier	0..1	The seller assigned identifier for the customer
Party	PartyType	1	Buyer details

Example

```
<cac:BuyerCustomerParty>
    <cbc:CustomerAssignedAccountID>58</cbc:CustomerAssignedAccountID>
    <cbc:SupplierAssignedAccountID>123899123</cbc:SupplierAssignedAccountID>
  + <cac:Party/>
</cac:BuyerCustomerParty>
```

Notes

1. OpenVPMS sets the **SupplierAssignedAccountID** to the **Account ID** field setup in the Supplier Stock Location relationship as detailed in section 4.1, if present.
2. The **CustomerAssignedAccountID** is set to the internal unique ID for the Stock Location.

Order.BuyerCustomerParty.Party

Name	Type	Use	Description
PartyName	PartyNameType	1	Buyer name
PostalAddress	AddressType	1	Buyer postal address
Contact	ContactType	1	Buyer contact

Example

```
<cac:Party/>
   + <cac:PartyName/>
   + <cac:PostalAddress/>
   + <cac:Contact/>
</cac:Party>
```

Order.BuyerCustomerParty.Party.PartyName

Name	Type	Use	Description
Name	NameType	1	Buyer name

Example

```
<cac:PartyName>
    <cbc:Name>OpenVPMS Demonstration Practice</cbc:Name>
</cac:PartyName>
```

Order.BuyerCustomerParty.Party.PostalAddress

Name	Type	Use	Description
CityName	NameType	1	City or town name
PostalZone	TextType	1	Post code
CountrySubentitty	TextType	1	State
AddressLine	AddressLineType	1	Formatted address line

Example

```
<cac:PostalAddress>
    <CityName>Cape Woolamai</CityName>
    <cbc:PostalZone>3925</cbc:PostalZone>
    <cbc:CountrySubentity>Victoria</cbc:CountrySubentity>
  + <cac:AddressLine/>
</cac:PostalAddress>
```

Order.BuyerCustomerParty.Party.PostalAddress.AddressLine

Name	Type	Use	Description
Line	TextType	1	Formatted address line

Example

```
<cac:AddressLine>
    <cbc:Line>1 Broadwater Avenue</cbc:Line>
</cac:AddressLine>
```

Order.BuyerCustomerParty.Party.Contact

Name	Type	Use	Description
Name	NameType	0..1	Buyer contact name
Telephone	TextType	0..1	Buyer telephone
Telefax	TextType	0..1	Buyer fax
ElectronicMail	TextType	0..1	Buyer email

Example

```
<cac:Contact>
    <cbc:Name>Joe Bloggs</cbc:Name>
    <cbc:Telephone>59527054</cbc:Telephone>
    <cbc:Telefax>59527053</cbc:Telefax>
    <cbc:ElectronicMail>foo@bar.com</cbc:ElectronicMail>
</cac:Contact>
```

Order.SellerSupplierParty

Name	Type	Use	Description
CustomerAssignedAccountID	IdentifierType	1	The buyer assigned identifier for the supplier
Party	PartyType	1	Seller details

Example

```
<cac:SellerSupplierParty>
    <cbc:CustomerAssignedAccountID>948</cbc:CustomerAssignedAccountID>
  + <cac:Party/>
</cac:SellerSupplierParty>
```

Notes

1. OpenVPMS sets the **CustomerAssignedAccountID** to the internal unique ID for the Supplier.

Order.SellerSupplierParty.Party

Name	Type	Use	Description
PartyName	PartyNameType	1	Seller name
PostalAddress	AddressType	0..1	Seller postal address

Example

```
<cac:Party/>
   + <cac:PartyName/>
   + <cac:PostalAddress/>
</cac:Party>
```

Order.SellerSupplierParty.Party.PartyName

Name	Type	Use	Description
Name	NameType	1	Seller name

Example

```
<cac:PartyName>
    <cbc:Name>Vet Supplies</cbc:Name>
</cac:PartyName>
```

Order.SellerSupplierParty.Party.PostalAddress

Name	Type	Use	Description
CityName	NameType	1	City or town name
PostalZone	TextType	1	Post code
CountrySubentitty	TextType	1	State
AddressLine	AddressLineType	1	Formatted address line

Example

```
<cac:PostalAddress>
    <cbc:CityName>Tennant Creek</cbc:CityName>
     <cbc:PostalZone>0862</cbc:PostalZone>
     <cbc:CountrySubentity>Northern Territory</cbc:CountrySubentity>
   + <cac:AddressLine/>
</cac:PostalAddress>
```

Order.SellerSupplierParty.Party.PostalAddress.AddressLine

Name	Type	Use	Description
Line	TextType	1	Formatted address line

Example

```
<cac:AddressLine>
    <cbc:Line>2 Peko Rd</cbc:Line>
</cac:AddressLine>
```

Order.AnticipatedMonetaryTotal

Name	Type	Use	Description
PayableAmount	Amount	1	The anticipated payable amount
PayableAmount@currencyID	CurrencyCodeContentType	1	The currency code

Example

```
<cac:AnticipatedMonetaryTotal>
    <cbc:PayableAmount currencyID="AUD">55.00</cbc:PayableAmount>
</cac:AnticipatedMonetaryTotal>
```

Order.OrderLine

Name	Type	Use	Description
LineItem	LineItemType	1..*	Order line item

Example

```
<cac:OrderLine>
   + <cac:LineItem>
</cac:OrderLine>
```

Order.OrderLine.LineItem

Name	Type	Use	Description
ID	IdentifierType	1	The order line identifier, assigned by the buyer

Quantity	QuantityType	1	The quantity
Quantity@unitCode	String	1	The unit of measure code for the quantity.
LineExtensionAmount	AmountType	1	The line total, excluding taxes
LineExtensionAmount@currencyID	CurrencyCodeContentType	1	The currency code the LineExtensionAmount is expressed in
TotalTaxAmount	AmountType	1	The anticipated tax amount
TotalTaxAmount@currencyID	CurrencyCodeContentType	1	The currency code the TotalTaxAmount is expressed in
Price	PriceType	1	The item price
Item	ItemType	1	The item

Example

```
<cac:LineItem>
    <cbc:ID>277</cbc:ID>
    <cbc:Quantity unitCode="BX">1.000</cbc:Quantity>
    <cbc:LineExtensionAmount currencyID="AUD">55.00</cbc:LineExtensionAmount>
    <cbc:TotalTaxAmount currencyID="AUD">5.50</cbc:TotalTaxAmount>
  + <cac:Price/>
  + <cac:Item/>
</cac:LineItem>
```

Order.OrderLine.LineItem.Price

Name	Type	Use	Description
PriceAmount	AmountType	1	The price
PriceAmount@currencyID	CurrencyCodeContentType	1	The currency code that PriceAmount is expressed in
BaseQuantity	QuantityType	1	The base quantity for the price.
BaseQuantity@unitCode		1	The unit of measure code for the base quantity.

Example

```
<cac:Price>
    <cbc:PriceAmount currencyID="AUD">55.000</cbc:PriceAmount>
    <cbc:BaseQuantity unitCode="BX">1</cbc:BaseQuantity>
</cac:Price>
```

Notes

1. BaseQuantity is limited to 1 for this release of ESCI.

Order.OrderLine.LineItem.Item

Name	Type	Use	Description
Description	TextType	0..1	The item description
Name	TextType	1	The item name
BuyersItemIdentification	ItemIdentificationType	1	The buyer assigned item identifier
SellersItemIdentification	ItemIdentificationType	1	The seller assigned item identifier

Example

```
<cac:Item>
    <cbc:Description>Advantage Dog Large 10-25Kg Maroon</cbc:Description>
    <cbc:Name>Advantage Dog Large</cbc:Name>
  + <cac:BuyersItemIdentification/>
  + <cac:SellersItemIdentification>
</cac:Item>
```

Order.OrderLine.LineItem.Item.BuyersItemIdentification

Name	Type	Use	Description
ID	IdentifierType	1	The buyer assigned item identifier

Example

```
<cac:BuyersItemIdentification>
    <cbc:ID>945</cbc:ID>
</cac:BuyersItemIdentification>
```

Notes

1. OpenVPMS sets the **BuyersItemIdentification** to the internal unique ID for the Product being ordered.

Order.OrderLine.LineItem.Item.SellersItemIdentification

Name	Type	Use	Description
ID	IdentifierType	1	The seller assigned item identifier

Example

```
<cac:SellersItemIdentification>
    <cbc:ID>ADVAND623</cbc:ID>
</cac:SellersItemIdentification>
```

Notes

1. OpenVPMS sets the **SellersItemIdentification** to the **Reorder Code** entered in the order line.
2. OpenVPMS stores the Reorder Code for each supplier in Product Supplier relationship information which can be found in the **Product → Information** workspace on the **Supplier** tab.

7.2 OrderResponseSimple

Element	Type	Use	Description
UBLVersionID	IdentifierType	1	UBL version identifier. Must be **2.0**
ID	IdentifierType	1	The seller assigned response identifier
IssueDate	DateType	1	The date when the response was issued
Note	TextType	0..*	Free form text
AcceptedIndicator	IndicatorType	1	Indicates if the order was accepted (true) or not (false)
RejectionNote	TextType	0..1	The reason for rejection if the order was not accepted
OrderReference	OrderReferenceType	1	The order that the response refers to
SellerSupplierParty	SupplierPartyType	1	The seller details
BuyerCustomerParty	CustomerPartyType	1	The buyer details

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<OrderResponseSimple
    xmlns="urn:oasis:names:specification:ubl:schema:xsd:OrderResponseSimple-2"
    xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
    xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2">
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:ID>66890-9-09</cbc:ID>
    <cbc:IssueDate>2005-06-20</cbc:IssueDate>
    <cbc:Note>sample</cbc:Note>
    <cbc:AcceptedIndicator>true</cbc:AcceptedIndicator>
  + <cac:OrderReference/>
  + <cac:SellerSupplierParty/>
  + <cac:BuyerCustomerParty/>
</OrderResponseSimple>
```

Notes

1. The ID must be unique for all OrderResponseSimple documents for a given buyer. The Order identifier cannot be used for this purpose as ESCI will support order amendment in a future vesion of ESCI – a new OrderResponseSimple must be generated with another identifier in this case.

OrderResponseSimple.OrderReference

Element	Type	Use	Description
ID	IdentifierType	1	The order identifier

Example

```
<cac:OrderReference>
    <cbc:ID>726</cbc:ID>
</cac:OrderReference>
```

OrderResponseSimple.SellerSupplierParty

Element	Type	Use	Description
CustomerAssignedAccountID	IdentifierType	0..1	The buyer assigned identifier for the seller
AdditionalAccountID	IdentifierType	0..1	The seller assigned identifier for the seller

Example

```
<cac:SellerSupplierParty>
    <cbc:CustomerAssignedAccountID>1234</cbc:CustomerAssignedAccountID>
</cac:SellerSupplierParty>
```

Notes

1. OpenVPMS requires that one or both of **CustomerAssignedAccountID** or **AdditionalAccountID** be provided.
2. If **CustomerAssignedAccountID** is used, it must correspond to that sent in the original order. This is the preferred identifier.
3. If **AdditionalAccountID** is used, it must match the **Account ID** field setup in the Supplier Stock Location relationship, as detailed in section 4.1.

OrderResponseSimple.BuyerCustomerParty

Element	Type	Use	Description
CustomerAssignedAccountID	IdentifierType	0..1	The buyer assigned identifier for the buyer
SupplierAssignedAccountID	IdentifierType	0..1	The seller assigned identifier for the buyer

Example

```
<cac:BuyerCustomerParty>
    <cbc:CustomerAssignedAccountID>1234</cbc:CustomerAssignedAccountID>
</cac:BuyerCustomerParty>
```

Notes

1. OpenVPMS requires that one or both of **CustomerAssignedAccountID** or **SupplierAssignedAccountID** be provided.
2. If **CustomerAssignedAccountID** is used, it must correspond to that sent in the original order. This is the preferred identifier.
3. If **SupplierAssignedAccountID** is used, it must match the **Account ID** field setup in the Supplier Stock Location relationship, as detailed in section 4.1.

7.3 Invoice

Element	Type	Use	Description
UBLVersionID	IdentifierType	1	UBL version identifier. Must be **2.0**
ID	IdentifierType	1	The invoice identifier, assigned by the supplier
IssueDate	DateType	1	The date the invoice was issued
IssueTime	TimeType	0..1	The time the invoice was issued
Note	TextType	0..*	Supplier provided note
OrderReference	OrderReferenceType	0..1	Reference to the original order
AccountSupplierParty	SupplierPartyType	1	The supplier information
AccountingCustomerParty	CustomerPartyType	1	The customer information
AllowanceCharge	AllowanceChargeType	0..*	Any allowances or charges associated with the invoice. For this release, only charges (e.g. freight) are supported.
TaxTotal	TaxTotalType	0..1	The total tax on the invoice
LegalMonetaryTotal	MonetaryTotalType	1	Total amounts
InvoiceLine	InvoiceLineType	1..*	Invoice items

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Invoice xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
         xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
         xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2">
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:ID>12345</cbc:ID>
    <cbc:IssueDate>2010-11-01</cbc:IssueDate>
    <cbc:IssueTime>17:12:08.265</cbc:IssueTime>
    <cbc:Note>a note</cbc:Note>
    <cbc:Note>another note</cbc:Note>
  + <cac:OrderReference/>
  + <cac:AccountingSupplierParty/>
  + <cac:AccountingCustomerParty/>
  + <cac:AllowanceCharge/>
  + <cac:TaxTotal/>
  + <cac:LegalMonetaryTotal/>
  + <cac:InvoiceLine/>
  + <cac:InvoiceLine/>
</Invoice>
```

Invoice.OrderReference

Element	Type	Use	Description

ID	IdentifierType	1	Order identifier, from the original order

Example

```
<cac:OrderReference>
    <cbc:ID>20122</cbc:ID>
</cac:OrderReference>
```

Notes

1. When specified, all InvoiceLine elements that reference an order line using the OrderLineReference element must refer to this order.

2. When an Invoice refers to multiple orders, this element should not be specified. Each InvoiceLine must fully specify the associated order.

Invoice.AccountingSupplierParty

Element	Type	Use	Description
CustomerAssignedAccountID	IdentifierType	0..1	The customer assigned identifier for the supplier
AdditionalAccountID	IdentifierType	0..1	The supplier assigned identifier for the supplier

Example

```
<cac:AccountingSupplierParty>
    <cbc:CustomerAssignedAccountID>12089</cbc:CustomerAssignedAccountID>
</cac:AccountingSupplierParty>
```

Notes

1. OpenVPMS requires that one or both of **CustomerAssignedAccountID** or **AdditionalAccountID** be provided.
2. If **CustomerAssignedAccountID** is used, it must correspond to that sent in the original order. This is the preferred identifier.
3. If **AdditionalAccountID** is used it must match the **Account ID** field setup in the Supplier Stock Location relationship, as detailed in section 4.1.

Invoice.AccountingCustomerParty

Element	Type	Use	Description
CustomerAssignedAccountID	IdentifierType	0..1	The customer assigned identifier for the customer
SupplierAssignedAccountID	IdentifierType	0..1	The supplier assigned identifier for the customer

Example

```
<cac:AccountingCustomerParty>
    <cbc:CustomerAssignedAccountID>12087</cbc:CustomerAssignedAccountID>
</cac:AccountingCustomerParty>
```

Notes

1. OpenVPMS requires that one or both of **CustomerAssignedAccountID** or **SupplierAssignedAccountID** be provided.
2. If **CustomerAssignedAccountID** is used, it must correspond to that sent in the original order. This is the preferred identifier.
3. If **SupplierAssignedAccountID** is used it must match the **Account ID** field setup in the Supplier Stock Location relationship, as detailed in section 4.1.

Invoice.AllowanceCharge

Element	Type	Use	Description
ChargeIndicator	IndicatorType	1	Set **true** if the AllowanceCharge is a charge, **false** if it is an allowance. For this release, only **true** (i.e. charges) is supported
AllowanceChargeReason	AllowanceChargeReasonType	1	The allowance/charge reason. Free-form text.
Amount	AmountType	1	The allowance/charge amount.
TaxCategory	TaxCategoryType	0..1	The tax category. Must be provided if TaxTotal is specified.
TaxTotal	TaxTotalType	0..1	The total tax for the allowance/charge. Defaults to **0.00** if not specified

Example

```
<cac:AllowanceCharge>
    <cbc:ChargeIndicator>true</cbc:ChargeIndicator>
    <cbc:AllowanceChargeReason>Freight</cbc:AllowanceChargeReason>
    <cbc:Amount currencyID="AUD">10</cbc:Amount>
  + <cac:TaxCategory/>
  + <cac:TaxTotal/>
</cac:AllowanceCharge>
```

Invoice.AllowanceCharge.TaxCategory

Element	Type	Use	Description
ID	IdentifierType	1	The tax category identifier
Percent	PercentType	1	The tax rate, expressed as a percentage
TaxScheme	TaxSchemeType	1	The tax scheme

Example

```
<cac:TaxCategory>
    <cbc:ID>S</cbc:ID>
    <cbc:Percent>10.00</cbc:Percent>
    + <cac:TaxScheme/>
<cac:TaxCategory>
```

Invoice.AllowanceCharge.TaxCategory.TaxScheme

Element	Type	Use	Description
TaxTypeCode	CodeType	1	The tax type code

Example

```
<cac:TaxScheme>
    <cbc:TaxTypeCode>GST</cbc:TaxTypeCode>
</cac:TaxScheme>
```

Invoice.AllowanceCharge.TaxTotal

Element	Type	Use	Description
TaxAmount	AmountType	1	The tax total
TaxAmount@currencyID	CurrencyCodeContentType	1	The currency code that TaxAmount is expressed in

Example

```
<cac:TaxTotal>
    <cbc:TaxAmount currencyID="AUD">1.00</cbc:TaxAmount>
</cac:TaxTotal>
```

Invoice. TaxTotal

Element	Type	Use	Description
TaxAmount	AmountType	1	The tax total
TaxAmount@currencyID	CurrencyCodeContentType	1	The currency code that TaxAmount is expressed in

Example

```
<cac:TaxTotal>
    <cbc:TaxAmount currencyID="AUD">15.00</cbc:TaxAmount>
</cac:TaxTotal>
```

Invoice. LegalMonetaryTotal

Element	Type	Use	Description
LineExtensionAmount	AmountType	1	The total of the line level LineExtensionAmounts
LineExtensionAmount@currencyID	CurrencyCodeContentType	1	The currency code that LineExtensionAmount is expressed in
TaxExclusiveAmount	AmountType	1	The total amount, exclusive of taxes
TaxExclusiveAmount@currencyID	CurrencyCodeContentType	1	The currency code that TaxExclusiveAmount is expressed in
ChargeTotalAmount	AmountType	0..1	The total charge amount
ChargeTotalAmount@currencyID	CurrencyCodeContentType	1	The currency code that ChargeTotalAmount is expressed in
PayableAmount	AmountType	1	The payable amount
PayableAmount@currencyID	CurrencyCodeContentType	1	The currency code that PayableAmount is expressed in

Example

```
<cac:LegalMonetaryTotal>
    <cbc:LineExtensionAmount currencyID="AUD">140</cbc:LineExtensionAmount>
    <cbc:TaxExclusiveAmount currencyID="AUD">150</cbc:TaxExclusiveAmount>
    <cbc:ChargeTotalAmount currencyID="AUD">10</cbc:ChargeTotalAmount>
    <cbc:PayableAmount currencyID="AUD">165.00</cbc:PayableAmount>
</cac:LegalMonetaryTotal>
```

Invoice. LegalMonetaryTotal.LineExtensionAmount

This is the total of the line level LineExtensionAmount amounts i.e.:

```
LineExtensionAmount = Σ LineExtensionAmount (at line level)
```

Invoice. LegalMonetaryTotal.TaxExclusiveAmount

This is the total amount, exclusive of taxes i.e.

```
TaxExclusiveAmount =
Σ LineExtensionAmount (at line level) +
Σ Amount (from AllowanceCharge at document level where ChargeIndicator = “true”) -
Σ Amount (from AllowanceCharge at document level where ChargeIndicator = “false”)
```

Invoice. LegalMonetaryTotal.ChargeTotalAmount

This is the total charge amount i.e.:

```
ChargeTotalAmount =
Σ Amount (from AllowanceCharge at document level where ChargeIndicator = “true”)
```

Invoice. LegalMonetaryTotal.PayableAmount

This is the total payable amount for the invoice i.e.:

```
PayableAmount = TaxExclusiveAmount (at document level) +
ΣTaxAmount (from TaxTotal at document level)
```

Invoice. InvoiceLine

Element	Type	Use	Description
ID	IdentifierType	1	The seller assigned identifier for the invoice line
InvoicedQuantity	QuantityType	1	The quantity of items on the invoice line
InvoicedQuantity@unitCode	String	1	The unit of measure code for the quantity
LineExtensionAmount	AmountType	1	The line total, excluding taxes
LineExtensionAmount@currencyID	CurrencyCodeContentType	1	The currency code that LineExtensionAmount is expressed in
OrderLineReference	OrderLineReferenceType	0..1	A reference to the original order line
PricingReference	PricingReferenceType	0..1	Used to express the wholesale price
TaxTotal	TaxTotalType	0..1	The total tax for the invoice line
Item	ItemType	1	The item being invoiced
Price	PriceType	1	The item price

Example

```
<cac:InvoiceLine>
    <cbc:ID>1</cbc:ID>
    <cbc:InvoicedQuantity unitCode="BX">1</cbc:InvoicedQuantity>
    <cbc:LineExtensionAmount currencyID="AUD">100</cbc:LineExtensionAmount>
  + <cac:OrderLineReference/>
  + <cac:PricingReference/>
  + <cac:TaxTotal/>
  + <cac:Item/>
  + <cac:Price/>
</cac:InvoiceLine>
```

Invoice.InvoiceLine.OrderLineReference

Element	Type	Use	Description
LineID	IdentifierType	1	The order line identifier
OrderReference	OrderReferenceType	0..1	The order reference

Example

```
<cac:OrderLineReference>
    <cbc:LineID>12384</cbc:LineID>
</cac:OrderLineReference>
```

Invoice.InvoiceLine.OrderLineReference.OrderReference

Element	Type	Use	Description
ID	IdentifierType	1	The order identifier

Example

```
<cac:OrderReference>
    <cbc:ID>726</cbc:ID>
</cac:OrderReference>
```

Notes

1. **OrderReference** only needs to be provided if an invoice is associated with multiple orders.

Invoice.InvoiceLine.PricingReference

Element	Type	Use	Description
AlternativeConditionPrice	PriceType	1	The price expressed in terms other than the actual price, e.g., the list price vs. the contracted price, or the price in bags vs. the price in kilos, or the list price in bags vs. the contracted price in kilos

Example

```
<cac:PricingReference>
  + <cac:AlternativeConditionPrice/>
</cac:PricingReference>
```

Invoice.InvoiceLine.PricingReference.AlternativeConditionPrice

Element	Type	Use	Description
PriceAmount	AmountType	1	The wholesale price
PriceAmount@currencyID	CurrencyCodeContentType	1	The currency code that PriceAmount is expressed in
PriceTypeCode	CodeType	1	The price type code. Must be **WH**

Example

```
<cac:AlternativeConditionPrice>
    <cbc:PriceAmount currencyID="AUD">105</cbc:PriceAmount>
    <cbc:PriceTypeCode>WH</cbc:PriceTypeCode>
</cac:AlternativeConditionPrice>
```

Invoice.InvoiceLine.TaxTotal

Element	Type	Use	Description
TaxAmount	AmountType	1	The invoice line tax amount
TaxAmount@currencyID	CurrencyCodeContentType	1	The currency code that TaxAmount is expressed in
TaxSubtotal	TaxSubtotalType	1	The tax subtotal

Example

```
<cac:TaxTotal>
    <cbc:TaxAmount currencyID="AUD">10.00</cbc:TaxAmount>
  + <cac:TaxSubtotal/>
</cac:TaxTotal>
```

Invoice.InvoiceLine.TaxTotal.Taxsubtotal

Element	Type	Use	Description
TaxAmount	AmountType	1	The tax amount
TaxAmount@currencyID	CurrencyCodeContentType	1	The currency code that TaxAmount is expressed in
TaxCategory	TaxCategoryType	1	The tax category

Example

```
<cac:TaxSubtotal>
    <cbc:TaxAmount currencyID="AUD">10.00</cbc:TaxAmount>
  + <cac:TaxCategory/>
</cac:TaxSubtotal>
```

Invoice.InvoiceLine.TaxTotal.Taxsubtotal.TaxCategory

Element	Type	Use	Description
ID	IdentifierType	1	The tax category identifier
Percent	PercentType	1	The tax rate, as a percentage
TaxScheme	TaxSchemeType	1	The tax scheme

Example

```
<cac:TaxCategory>
    <cbc:ID>S</cbc:ID>
    <cbc:Percent>10.00</cbc:Percent>
  + <cac:TaxScheme/>
</cac:TaxCategory>
```

Invoice.InvoiceLine.TaxTotal.Taxsubtotal.TaxCategory.TaxScheme

Element	Type	Use	Description
TaxTypeCode	CodeType	1	The tax type code

Example

```
<cac:TaxScheme>
    <cbc:TaxTypeCode>GST</cbc:TaxTypeCode>
</cac:TaxScheme>
```

Invoice.InvoiceLine.Item

Element	Type	Use	Description
Name	TextType	1	The item name
BuyersItemIdentification	ItemIdentificationType	0..1	The buyer assigned item identifier
SellersItemIdentification	ItemIdentificationType	0..1	The seller assigned item identifier

Example

```
<cac:Item>
    <cbc:Name>product1 name</cbc:Name>
  + <cac:BuyersItemIdentification/>
  + <cac:SellersItemIdentification/>
</cac:Item>
```

Notes

1. OpenVPMS requires that one or both of **BuyersItemIdentification** and **SellersItemIdentification** be provided.
2. If **BuyersItemIdentification** is used, it must correspond to the **BuyersItemIdentification** from the original **OrderLine**.
3. If **SellersItemIdentification** is used, it must correspond to the **Reorder Code** found in OpenVPMS product's supplier information for the supplier generating the invoice. **SellersItemIdentification** may be used if there is no **OrderLine** associated with an **InvoiceLine**.

Invoice.InvoiceLine.Item.BuyersItemIdentification

Name	Type	Use	Description
ID	IdentifierType	1	The buyer assigned item identifier

Example

```
<cac:BuyersItemIdentification>
    <cbc:ID>12091</cbc:ID>
</cac:BuyersItemIdentification>
```

Invoice.InvoiceLine.Item.SellersItemIdentification

Name	Type	Use	Description
ID	IdentifierType	1	The seller assigned item identifier

Example

```
<cac:SellersItemIdentification>
    <cbc:ID>product1</cbc:ID>
</cac:SellersItemIdentification>
```

Invoice.InvoiceLine.Price

Name	Type	Use	Description
PriceAmount	AmountType	1	The price for the base quantity of the item, excluding any taxes.
PriceAmount@currencyID	CurrencyCodeContentType	1	The currency code that PriceAmount is expressed in
BaseQuantity	QuantityType	0..1	The base quantity that the price applies to.
BaseQuantity@unitCode	String	1	The unit of measure code for the quantity.

Example

```
cac:InvoiceLine>
    <cbc:InvoicedQuantity unitCode="BO">10</cbc:InvoicedQuantity>
    <cbc:LineExtensionTotalAmount currencyID="AUD">240.00</cbc:LineExtensionTotalAmount>
    <cac:Item>
        <cbc:Name>Alamycin 10</cbc:Name>
        <cac:SellersItemIdentification>
            <cbc:ID>1024</cbc:ID>
        </cac:SellersItemIdentification>
    </cac:Item>
    <cac:Price>
        <cbc:PriceAmount currencyID="AUD">24.00</cbc:PriceAmount>
        <cbc:BaseQuantity unitCode="BO">1</cbc:BaseQuantity>
    </cac:Price>
</cac:InvoiceLine>
```

The example above shows that 10 bottles of Alamycin 10 have been invoiced at a total price of $240, and that each bottle has a price of $24.

The *PriceAmount* and the *BaseQuantity* describe the supplier's base unit for the product. Note that *PriceAmount* is always specified tax exclusive. For this release, only a BaseQuantity of 1 is supported.

7.4 DocumentReference

Element	Type	Use	Description
ID	IdentifierType	1	Identifies the document being referred to
IssueDate	IssueDateType	0..1	The date the document was issued
DocumentType	TextType	1	The document type e.g. **OrderResponseSimple, Invoice**

7.5 Document

Element	Type	Use	Description
Any	Any	1	The UBL document

Example

```
<esci:Document>
  + <invoice:Invoice/>
</esci:Document>
```

8 Code Lists

8.1 Currency codes

Currency codes used in amounts must conform to the ISO 4217 standard. This specifies currencies as three letter codes e.g. **AUD** for “Australian Dollar”, **GBP** for **“**Pound Sterling”.

While UBL allows different currency codes to be expressed for different amounts in a document, this release of ESCI only supports a single currency per document.

UBL provides a code list to validate currency codes against, at http://docs.oasis-open.org/ubl/cs-UBL-2.0/cl/gc/cefact/CurrencyCode-2.0.gc

8.2 Unit of Measure codes

Unit of measure codes are used in quantities to specify the type of the quantity, e.g. **BX** for “box”, and **EA** for “each”. These must conform to the UN/ECE rec 20 standard.

UBL provides a code list to validate these against, at http://docs.oasis-open.org/ubl/cs-UBL-2.0/cl/gc/cefact/UnitOfMeasureCode-2.0.gc

8.3 Tax Category Identifiers

Tax category identifiers are used to specify the category of tax applied to an item, e.g. **S** for “Standard-Rated”, **Z** for **“**Zero-Rated”, **E** for “Exempt from tax”.

These must be specified according to UN/CEFACT Code List 5305 “Duty or Tax or Fee Category Code” version D09B. See http://www.unece.org/uncefact/codelist/standard/UNECE_DutyorTaxorFeeCategoryCode_D09B.xsd for more details.

8.4 Tax Type Codes

Tax schemes specify the type of tax payable via a TaxTypeCode, e.g. **GST**, or **VAT**. These must be specified according to UN/CEFACT Code List 5153 “Duty Tax Fee Type Code” version D09B. See http://www.unece.org/uncefact/codelist/standard/UNECE_DutyTaxFeeTypeCode_D09B.xsd for more details.

8.5 Price Type Codes

Price type code identifiers are used to identify the type of a price. For ESCI, these are currently only relevant to the **InvoiceLine.PricingReference.AlternativeConditionPrice** element, to specify that the price is a wholesale price.

These must be specified according to UN/CEFACT Code List 5375 “Price Type Code” version D09B. See http://www.unece.org/uncefact/codelist/standard/UNECE_PriceTypeCode_D09B.xsd for more details.

8.6 Country codes

There is currently no requirement to include country codes in addresses. If specified however, any country code must conform to the ISO 3166-1 “Country codes” standard. This expresses country codes using two characters e.g. **AU** for “Australia”, **NZ** for “New Zealand”.

UBL provides a code list to validate these against, at http://docs.oasis-open.org/ubl/cs-UBL-2.0/cl/gc/default/CountryIdentificationCode-2.0.gc

8.7 Country Sub-entity codes

There is currently no requirement to include country sub-entity codes (e.g. states, territories or provinces) in addresses. If specified however, any code must conform to the ISO 3166-2 “Country subdivision code” standard.

See http://www.iso.org/iso/country_codes/background_on_iso_3166/iso_3166-2.htm for the source of the standard. ISO charges a fee for the code list – see http://en.wikipedia.org/wiki/ISO_3166-2 for a synopsis and references to older versions.

9 Future Development

Areas for future development include:

- Order cancellation
- Order amendment
- Catalogues
- Tools to validate that ESCI documents conform to both the XML schema and the ESCI specification
- Different calculation models