



ExpressLabel Integration Guide

Version 1.10

Revision History

Date	Version	Description
17/09/2008	1.0	First revision of technical user guide for ExpressLabel
04/11/2008	1.1	Add code examples for connecting to ExpressLabel
06/11/2008	1.2	Add PHP code example for connecting to ExpressLabel
11/11/2008	1.3	Add information about label construction and return message
17/11/2008	1.4	Updates to rendering instructions
17/11/2008	1.5	Processing of feedback
01/09/2009	1.6	Incorporated new guides on barcode size.
23/03/2010	1.7	Incorporated latest Ops Piece Label Specification
02/06/2010	1.8	Added information about the exact match tag
10/09/2010	1.9	Changed Barcode type to Code128 length 28, changed ascii only characters to utf-8, added SLA statement.
08/12/2010	1.10	Documentation updated to include the new tags in support of the French domestic label data and a XML connection example.

Table of Contents

1. Introduction	5
1.1 Legend	6
2. Registration	7
3. Making a request to the TNT server	8
3.1 Service Level Agreement (SLA)	8
4. Example XML Label Request Document	9
5. Input XML format	11
5.1 Header	11
5.2 The LabelRequest element	11
5.3 The consignment element	12
5.4 The ConsignmentIdentity element	12
5.5 The collectionDateTime element	13
5.6 The sender and delivery elements	13
5.7 The contact element	14
5.8 The product element	14
5.9 The account element	14
5.10 The specialInstructions element	14
5.11 The cashAmount element	15
5.12 The totalNumberOfPieces element	15
5.13 The pieceLine element	15
6. Processing the XML Response	17
6.1 High level description of an XML Label Response Document	17
6.1.1 Routing Label Overview	18
6.1.2 Header and Root Element	19
6.1.3 consignment Sections	19
6.2 Detailed description of an XML Label Response Document	20
6.2.1 Logo	20
6.2.2 Market Indicator	20
6.2.3 Transport Mode	21
6.2.4 Free Circulation Indicator	22
6.2.5 Sort Split Indicator	22
6.2.6 Hazardous	23
6.2.7 X-Ray	23
6.2.8 Consignment Number	23
6.2.9 Product	24
6.2.10 Piece	24
6.2.11 Weight	25
6.2.12 Option	26
6.2.13 Customer Reference	26
6.2.14 Account	27
6.2.15 Collection Date	27
6.2.16 Origin Depot	28
6.2.17 Sender & Delivery Address	28
6.2.18 Contact	29

6.2.19 Routing	30
6.2.20 Airport Sort Code	31
6.2.21 Destination Depot	31
6.2.22 Postcode / Cluster Code	32
6.2.23 Legal Comments	33
6.2.24 Cash Amount	33
6.2.25 Special instructions	33
6.2.26 Barcode	33
6.2.27 Barcode For Customer	34
7. Errors	35
7.1 Application generated errors	35
7.2 Table of application generated error codes, messages and resolutions	36
8. Connecting to ExpressLabel	45
8.1 Choosing a request protocol	45
8.1.1 XML	45
8.1.2 Hessian	45
8.1.3 Burlap	45
8.2 Java XML Connection Example	46
8.3 Java Hessian Connection Example	46
8.3.1 Create a Marshaller	47
8.3.2 Populate the request data objects	47
8.3.3 Create the client integrator	48
8.4 PHP Example	50
9. Appendix A: XML elements definition (input)	53
10. Appendix B : XSD Data Types	63
10.1 Custom Data Types	64
11. Appendix C: ISO 3166-1 Alpha-2 Country Codes	65

1. Introduction

ExpressLabel is part of the ExpressConnect family, providing B2B interfaces into TNT's operational systems. The ExpressLabel interface is used to generate routing label data for TNT Consignments. This function, which is traditionally done by the depot, expedites consignment processing.

An example of a TNT routing label is given below. It contains information that is critical for timely delivery of the consignment and which maximises efficiency of the network through pre-validation of the information.

		INT / AIR X-RAY		C 2	
Con No. 123456782		Service Express			
Piece 1 of 3		Weight 1.11kg		Option Priority	
Customer Reference piece1		Origin CVT		Pickup Date 03 Nov 2010	
S/R Account No 100445		Routing EMA LGG			
Sender Address John Smith First Floor, Room 3 TNT House ATHERSTONE CV9 1TT GB		Sort			
Delivery Address TNT Corporate Head Office Neptunusstraat 41-63 2132 JA Hoofddorp AMSTERDAM 1011 AA NL		Dest Depot SP8-4			
Postcode / Cluster Code 01					
					
1100123456782011641024000000					

This manual provides a technical guide to the ExpressLabel interface. It is designed to help developers understand the interface sufficiently to program an application to request label data to be used in rendering a routing label. Using XML (eXtensible Markup Language), ExpressLabel provides routing label functionality for batches of consignments. The majority of examples in the document will be XML based as it is easier to understand the data involved.

The resultant response contains the data required to create labels for the consignments submitted. There is a 1-to-many relationship between consignments and labels as each consignment may have several pieces, each of which requires a label. A number of schemes have been designed for rendering the output as a label generally involving XSL Transformation, whether to create HTML or PDF documents.

An ExpressConnect Login Id will be arranged by your TNT representative. The customer must supply a list of valid TNT account numbers to be used with the system. A secure connection is thus set up, using both authentication and secure protocols, to submit requests and to receive processed consignment labels based on the published URL:

<https://express.tnt.com/expresslabel/documentation/getlabel>

Sample scripts are provided to show how a connection can be achieved together with example requests and responses to illustrate the data required; see [Connecting to ExpressLabel](#). Consideration is given below to the networking and security requirements to ensure that this is successful.

The data that is returned can then be stored for later use, or rendered for sending to a printer and attaching to the consignment.

This document is structured as follows:

- Registration
- Making a request to the TNT server
- Example XML Label Request Document
- Input XML Format - the structure and content of the request
- Processing the response - information about the data that will be returned
- Errors - possible error messages and the steps you can take to resolve them
- Connecting to ExpressLabel
- Appendices

1.1 Legend

The following conventions have been used throughout this document.

Normal	The majority of text in this document is in this style. Section in this style are part of the narrative of the document
Code	Sections or words in this text indicate a section of XML, XML element, or section of code.
[0..1]	Digits within square brackets indicate the number of times an element may occur in an XML document. Examples include: [1] The element must appear once in the document [0..*]. The element may appear once, many times (unlimited) or not at all. [0..1] The element is optional. If it appears, it must appear only once. [1..5] The element may appear any number of times between 1 and 5 times
xsd:string	This indicates one of the schema types, in this case a string. More information on defined schema data types can be found at http://www.w3.org/TR/xmlschema-2/#built-in-datatypes
...	Means that the section has been omitted for the sake of clarity. This usually means that the omitted elements are described elsewhere or that the section where they appear is a repetition of a previous stanza. For example: <pre> <house> <room> <width unitOfMeasure="m">12</width> </room> <!-- the next room contains the same dimensions sections as the one above. --> <room>...</room> </house> </pre>

2. Registration

Each customer is set up with a username and password, required for all communications with the system. The *userid* takes the form of an email address. You should choose an email address that is monitored: we may use this email to communicate membership information and service status reports. In addition, it may be used to communicate changes and improvements to the service

In return, you will be given the following information:

- A consignment number range allocation
- Default collection time for your depot
- Value for the following attributes of your consignments:
 1. Line of Business
 2. Group Id
 3. Sub Group Id
 4. Available Services - this is the list of product codes from which you can select
 5. Valid option codes are provided for each service or product by your TNT representative

3. Making a request to the TNT server

To make a label request, you must construct an XML file which conforms to the standard set out in this document. The submission will be validated to check for any problems with the structure of the XML. This facility is provided to allow you to self-diagnose problems with the XML.

Before sending an XML document over the internet to TNT, you should verify that you understand the XML format by successfully using the ExpressLabel Website provided by TNT, contact your TNT representative for further information. The audience for the website is intended to be developers who can use it as a tool to test their XML and to analyse results, prior to and during the development of client applications. It includes a Test Harness that allows the developer to submit sample XML to the service.

Please note that the site requires you to enter your user id and password before displaying the test harness.

- ExpressLabel now supports UTF-8 characters.
- The demonstration site links to a production like environment so that it replicates exactly the results that the customer will achieve with their completed application.

Having made a successful submission of an XML document via the test page, you are ready to set up a socket connection and make a programmatic submission using HTTP POST to the following URL:

`https://express.tnt.com/expresslabel/documentation/getlabel`

Please be aware that all submissions to the aforementioned URL will require you to supply your user id and password as part of the POST request.

More detailed information on connecting to the ExpressConnect servers can be found in [Connecting to ExpressLabel](#).

3.1 Service Level Agreement (SLA)

The SLA is specified as follows:

There is a 1.3 second SLA which is the time between the first byte of the request being received on the TNT server and the first byte of the response leaving the server. There is no way to control the network between the client and TNT so this element can not be included.

4. Example XML Label Request Document

An example is provided below of a typical label request. The request specifies a single consignment, with consignment number 123456782 and a customer reference of *Robert's computer*. When the consignment is in the TNT network the consignment number and customer reference can be used to track its progress.

A key uniquely identifies the consignment *for the given request*; where the request batches multiple consignment label requests, each consignment element must be allocated a unique key. The collection date and time are specified. The collection date can be up to 5 days ahead. The collection time will be provided by your TNT representative. Finally, the collection address is specified in the sender stanza.

```
<?xml version="1.0" encoding="UTF-8"?>
<labelRequest>
  <consignment key="CON1">
    <consignmentIdentity>
      <consignmentNumber>123456782</consignmentNumber>
      <customerReference>Robert's computer</customerReference>
    </consignmentIdentity>
    <collectionDateTime>2008-06-12T13:00:00</collectionDateTime>
    <sender>
      <name>Karen Bradley</name>
      <addressLine1>TNT Express</addressLine1>
      <addressLine2>TNT House</addressLine2>
      <addressLine3>Holly Lane</addressLine3>
      <town>Atherstone</town>
      <exactMatch>Y</exactMatch>
      <province>Warks</province>
      <postcode>CV9 1TT</postcode>
      <country>GB</country>
    </sender>
  </consignment>
</labelRequest>
```

The delivery address specified where the consignment will be delivered. The next two sections product and account, contain information about the product or service for the consignment, and the TNT account to be used for the consignment.

```

    <delivery>
      <name>TNT Corporate Head Office</name>
      <addressLine1>Neptunusstraat 41-63</addressLine1>
      <addressLine2>2132 JA Hoofddorp</addressLine2>
      <town>Amsterdam</town>
      <exactMatch>Y</exactMatch>
      <province/>
      <postcode>1011 AA</postcode>
      <country>NL</country>
    </delivery>
    <product>
      <lineOfBusiness>2</lineOfBusiness>
      <groupId>0</groupId>
      <subGroupId>0</subGroupId>
      <id>EX</id>
      <type>N</type>
      <option>PR</option>
    </product>
    <account>
      <accountNumber>100445</accountNumber>
      <accountCountry>GB</accountCountry>
    </account>
    <totalNumberOfPieces>3</totalNumberOfPieces>
    <pieceLine>
      <identifier>1</identifier>
      <goodsDescription>piecelinegoods desc</goodsDescription>
      <pieceMeasurements>
        <length>1.11</length>
        <width>1.11</width>
        <height>1.11</height>
        <weight>1.11</weight>
      </pieceMeasurements>
      <pieces>
        <sequenceNumbers>1,2</sequenceNumbers>
        <pieceReference>keyboard and mouse</pieceReference>
      </pieces>
      <pieces>
        <sequenceNumbers>3</sequenceNumbers>
        <pieceReference>computer tower</pieceReference>
      </pieces>
    </pieceLine>
  </consignment>
</labelRequest>

```

Following the product, and account, the next sections describe the contents of the consignment. A totalNumberOfPieces element is required to identify the total number of pieces or packages in the consignment. The last section, pieceLine, describes the packages, giving their measurements, description, and reference numbers.

5. Input XML format

The input XML format for ExpressLabel is a list of consignments for which routing labels are required.

The structure for a label request is described below in detail. Alternatively, refer to Appendix A which contains a summary of the points below.

Note

It should be noted that XML defines a number of characters which are reserved. These include the greater-than (>), less-than (<), ampersand (&), and percent (%) characters. Where these appear in the data which is being submitted to ExpressLabel, the characters must be escaped or the content surrounded with a CDATA section.

A common requirement is to submit an address which includes a company name such as: "Andrews & Plummer". The ampersand must therefore be escaped as per the XML rules (&) or alternatively the whole or part of the text must be wrapped in a CDATA section as follows:

```
...  
<name><![CDATA[Andrews & Plummer]]></name>
```

The request has the following structure:

- Header – always required, this defines the XML document
- Label Request– A list of consignment for which a routing label is required

5.1 Header

The header section will begin every ExpressLabel request XML document submitted to TNT.

This contains the XML declaration, which contains the character encoding used for the document and the standalone attribute, which should be set to “no”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

5.2 The LabelRequest element

A Label request at a high level is structured as shown in Diagram 1:

```
<labelRequest>  
  <consignment key="consignment1">...</consignment>  
  <consignment key="consignment2">...</consignment>  
  <consignment key="consignment3">...</consignment>  
  ...  
</labelRequest>
```

A labelRequest contains between 1 and 5 consignment elements. This allows batching of consignment routing label requests. Each consignment element contains the set of information needed to generate routing labels for the consignment referenced.

A consignment element contains a key attribute that identifies it uniquely *within the request*. The response will associate label data and validation errors which their consignment label requests through this key. The value of this key must be of type string and is only valid in express label for a single transaction - i.e. ExpressLabel does not retain any history of the keys used once a request has been processed and a response sent to the client system.

5.3 The consignment element

Each consignment element contains the following information:

```
<labelRequest>
  <consignment key=" xsd:string ">
    <consignmentIdentity> ... </consignmentIdentity> [1]
    <collectionDateTime> xsd:datetime </collectionDate> [1]
    <sender> ... </sender> [1]
    <delivery> ... </delivery> [1]
    <contact> ... </contact> [0..1]
    <product> ... </product> [1]
    <account> ... </account> [1]
    <specialInstructions> ... </specialInstructions> [0..1]
    <cashAmount> ... </cashAmount> [0..1]
    <totalNumberOfPieces> xsd:int </totalNumberOfPieces> [1]
    <pieceLine> ... </pieceLine> [1..99]
  </consignment> [1..5]
</labelRequest>
```

A consignment is made up of the following parts. Except where noted, every element is required once. A consignment contains `consignmentIdentity` information which uniquely identifies it within TNT. For more information, see [Section 5.4](#). The `collectionDateTime`, a standard schema `datetime` type is described in [Appendix A](#)

- The sender address from which the consignment will be collected. The structure of an address is described in further detail in [Section 5.6](#).
- The delivery address to which the consignment will be delivered. The structure of an address is described in [Section 5.6](#).
- The TNT product to be used for the consignment. See [Section 5.7](#)
- The TNT account that will be charged for the consignment. [Section 5.8](#)
- The total number of pieces in the consignment.
- One or many `pieceLine` elements. Piece line is described in more detail in [Section 5.10](#).

5.4 The ConsignmentIdentity element

The consignment identifies the consignment uniquely within TNT.

```
<consignmentIdentity>
  <consignmentNumber> xsd:string </consignmentNumber> [1]
  <customerReference> xsd:string </customerReference> [0..1]
</consignmentIdentity>
```

It contains a `consignmentNumber` element which should have a consignment number from a range allocated to you by your TNT representative. You should ensure that any given number is allocated to only one consignment. Failure to do so may result in the consignment being allocated a new number which will make it difficult to track.

You may also supply a customer reference. The customer reference can be used to track the consignment from collection to delivery.

5.5 The `collectionDateTime` element

The date and time the consignment will be collected. The time part of this value will be provided by your representative. The date is the date the consignment will be collected. The format of a date time is `yyyy-mm-ddThh:MM:ss`

Format	Date Component Description
yyyy	The year in four digits. e.g. 2008
mm	The month in digits, January is 01, December is 12
dd	Day of the month. Valid range is 1 to 31
hh	The collection hour in 24 hour notation.
MM	The minutes from 00 to 60. If the hour is 24, the minute value must be 00
ss	The seconds from 00 to 60. If the hour is 24, the second value must be 00

All other characters - the dashes, colons, and the capital T which separates the day from the hour - are literals which must appear as they are shown above.

5.6 The `sender` and `delivery` elements

The `sender` element represents your company address from which consignment will be collected by our driver. The `delivery` is the address the consignment will be delivered to. As the definition for these two addresses is identical, we've identified them below as *address_type_element*.

```
<address_type_element>
  <name> xsd:string </name> [1]
  <addressLine1> xsd:string </addressLine1> [1]
  <addressLine2> xsd:string </addressLine2> [0..1]
  <addressLine3> xsd:string </addressLine3> [0..1]
  <town> xsd:string </town> [1]
  <exactMatch> exactMatchEnum </exactMatch> [0..1]
  <province> xsd:string </province> [0..1]
  <postcode> xsd:string </postcode> [0..1]
  <country> xsd:string </country> [1]
</address_type_element>
```

These sections contain the following elements, all of type string:

- The name of company at the given premises
- `addressLine1` is the first line of the address. This line usually contains a building name or number
- Address lines 2 and 3 are optional. They may contain additional information to help locate the address
- The town for the address
- Whether the town name should be used as an exact or partial match

- The province is also known as the region or county.
- The postcode of the address
- The ISO 3166-1 Alpha-2 country code. More information can be found in [Appendix C](#)

5.7 The contact element

The contact element is specific to some labels but not all of them. The French Domestic label is an example of where this data is used. The contact data is returned in the output xml and then should be rendered to the label in this case.

```
<contact>
  <name> stringMaxLength30 </name> [0..1]
  <telephoneNumber> stringMaxLength30 </telephoneNumber> [0..1]
  <emailAddress> xsd:string </emailAddress> [0..1]
</contact>
```

5.8 The product element

The product element contains the following elements. The values in this section will be provided by your TNT representative.

```
<product>
  <lineOfBusiness> integerMin0Max9 </lineOfBusiness> [1]
  <groupId> integerMin0Max9 </groupId> [1]
  <subGroupId> integerMin0Max9 </subGroupId> [1]
  <id> stringMaxLength4 </id> [1]
  <type> productTypeEnum </type> [1]
  <option> stringMaxLength4 </option> [0..5]
</product>
```

You may supply up to five options for your consignment. Valid option codes are provided for each product by your TNT representative

5.9 The account element

Your TNT account is specified by giving the account number and the 2 digit ISO 3166-1 Alpha-2 country code in which the account is registered. See figure 1b.

```
<account>
  <accountNumber> xsd:long </accountNumber> [1]
  <accountCountry> xsd:string </accountCountry> [1]
</account>
```

5.10 The specialInstructions element

The special instructions data is used by some labels but not all. The French Domestic label, for example, has room on it for the special instructions to be displayed. As a result of not all labels requiring this data, it is an optional field.

```
<specialInstructions> xsd:string </specialInstructions> [0..1]
```

5.11 The cashAmount element

The cashAmount data is very similar to the specialInstructions element and is, therefore, an optional field.

```
<cashAmount> doubleTwoDecimalPlaces </cashAmount> [0..1]
```

5.12 The totalNumberOfPieces element

The totalNumberOfPieces element contains the integer number of pieces, or packages in your consignment. The maximum number of pieces supported is 99.

```
<totalNumberOfPieces> xsd:int </totalNumberOfPieces> [1]
```

5.13 The pieceLine element

You may specify any number of piece line elements. A piece line is defined as the group of packages which are identical in weight and dimensions. The stanza for a piece line element is shown below:

```
<pieceLine>
  <identifier> xsd:int </identifier> [1]
  <goodsDescription> xsd:string </goodsDescription> [1]
  <pieceMeasurements>
    <length> xsd:double </length> [1]
    <width> xsd:double </width> [1]
    <height> xsd:double </height> [1]
    <weight> xsd:double </weight> [1]
  </pieceMeasurements> [1]
  <pieces>
    <sequenceNumbers> xsd:string </sequenceNumber> [1..99]
    <pieceReference> xsd:string </pieceReference> [1]
  </pieces> [1..99]
</pieceLine> [1..99]
```

A piece line contains the following sub-elements:

- The pieceLines are assigned incrementally higher numbers from 1 i.e. the first piece line is assigned identifier value of 1, the second is assigned a value of 2, and so on.
- The goodsDescription is the full goods description for customs purposes.
- A pieceMeasurements contains the length, width, height, and weight of each piece in the pieceLine
- The pieces element may be repeated as many times as necessary but appears at least once. It allows flexibility in assigning piece references through which the parts of the consignment can be tracked. The sequenceNumbers element identifies the packages that should be assigned a specified pieceReference.

Perhaps the simplest way to explain the way that this works is through an example. Imagine that you are a supplier of computer parts. You have an order that is made up of computer mice, some of which are urgently needed. As a result, you would like to track the set of urgently required parts as a group. The order is made up of 6 mice, all packages identical, of which the first 4 are urgent. The section of XML that would allow you to do so might be:

```
<pieceLine>
  <identifier>1</identifier>
  <goodsDescription>Computer mice - laser</goodsDescription>
  <pieceMeasurements>
    <length>1.2</length>
    <width>1.2</width>
    <height>1.2</height>
    <weight>0.1</weight>
  </pieceMeasurements>
  <pieces>
    <sequenceNumbers>1,2,3,4</sequenceNumbers>
    <pieceReference>P000001-CL02</pieceReference>
  </pieces>
  <pieces>
    <sequenceNumbers>5,6</sequenceNumbers>
    <pieceReference>P000002-CL03</pieceReference>
  </pieces>
</pieceLine>
```


6. Processing the XML Response

It is envisaged that the majority of system using ExpressLabel will take one of the following approaches to render the label:

1. Use a component provided by TNT to be integrated into your application like Style Sheets.
2. Development of a custom rendering component by the client, in order the layout of TNT requirements.

The document that is returned by the system is a standard XML document. It contains a Header and a single root element containing the remaining elements. The document will contain a `<consignment>` element for each successful label request - one that passed validation and contained all of the information necessary to create a routing label. This document can be cached in a database or in a file in order to allow later printing and re-printing of labels.

In addition to `consignment` elements, there may be a number of `<brokenRules>` elements. These elements represent validation errors, and may be used to report on the availability of the system.

Before the document can be used to render routing labels for each consignment, there are a number of checks that must be made:

- The document is inspected to match `<consignment>` sections with the consignments they represent.
- These sections can be submitted to the rendering tool to produce routing labels.
- Alternatively, the document or portions of it may be cached in a database or file for re-printing or later printing.
- The document is inspected to check for `<brokenRules>` elements. Any invalid requests should be investigated and amendments made for a resubmission ([see Errors section](#)).

Before investigating each of these, the section below will briefly review an example response document.

6.1 High level description of an XML Label Response Document

The first section contains the usual XML header; see [Header and Root Element](#). The ROOT node for the response is a `routingLabels` element. This element typically contains a `consignment` element for each consignment label request.

A consignment stanza begins with a list of `pieceLabelData` sections, one per piece in the consignment. This represents the data that varies from piece to piece. Each `pieceLabelData` section contains:

- piece number
- piece reference
- The barcode to be rendered for the routing label.

The remaining data is common to all piece labels and is gathered together in the `consignmentLabelData` element. This section contains:

- The consignment number as it should be shown on the label
- The portions of the sender and delivery addresses that should be shown on the label.
- Account number
- Total number of pieces
- The product and option information
- Operational information to be displayed on the label e.g. date / market / transport/ free circulation / sort split text / x-ray etc.
- The routing or depot information pertaining to the consignment

6.1.1 Routing Label Overview

The main sections of the routing label are identified below:

<p>X-Ray required</p> <p>Hazardous production option included</p> <p>Market Display: International (INT) or Domestic (DOM)</p> <p>Transport mode: AIR, ROAD, or mixed mode</p> <p>Free Circulation Display: C may be present or not</p>	
<p>Logo</p> <p>Consignment Number</p> <p>Weight</p> <p>Piece Number</p> <p>Piece Reference or if not supplied then Customer Reference</p> <p>Account Number</p> <p>Sender / Origin Address</p> <p>Delivery / Destination Address</p> <p>Cluster Code or if not supplied then delivery address postcode</p> <p>Barcode</p>	<p> TNT INT / AIR HAZARDOUS X-RAY </p> <p> C 2 Sort Split Indicator </p> <p> Con No. 123456782 Piece 1 of 3 Weight 1.11kg </p> <p> Service Express Option PR HZ </p> <p> Customer Reference piece1 S/R Account No 100445 </p> <p> Origin CVT Pickup Date 03 Nov 2010 Collection Date </p> <p> Routing EMA LGG Transit Depots </p> <p> Sort Cell Indicator e.g. for LGG depot or Action Day of Week e.g. for QAR depot </p> <p> Sort Airport Sort Code </p> <p> Postcode / Cluster Code 01 Dest Depot SP8-4 Due Day of Month </p> <p> Destination Depot </p> <p> 1100123456782011641024000000 </p>

6.1.2 Header and Root Element

As explained above, the header section will begin every ExpressLabel response XML document sent back by TNT. This contains the XML declaration, which defines the encoding of the document. The root element, `labelResponse` may contain up to 5 `consignment` sections and a maximum of 10 `brokenRules` sections.

If the system has encountered a fatal error, this will appear as a `fault`. This is discussed in more details in the section on [Errors](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<labelResponse>
  <consignment>...</consignment> [0..5]
  <brokenRules>...</brokenRules> [0..10]
</labelResponse>
```

6.1.3 consignment Sections

A consignment section contains information required to print routing labels for all pieces of a consignment. The format of the consignment element includes the `consignmentNumber` child element. This value can be used to match consignment sections with the consignment they apply to.

Each consignment can result in multiple routing labels, one per package in the consignment. Each consignment element contains two types of information: data specific to each piece and data which are common to all pieces of the consignment. The consignment section is therefore composed of two parts, `pieceLabelData` and `consignmentLabelData`. There may be between 1 and 99 pieces i.e. `pieceLabelData` in each consignment response and exactly one `consignmentLabelData`.

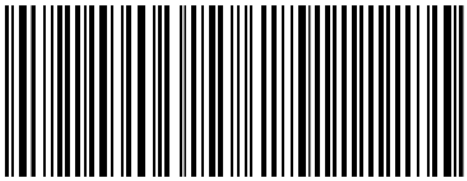
```
<pieceLabelData> [0..99]
  <pieceNumber> xsd:int </pieceNumber> [1]
  <pieceReference> xsd:string </pieceReference> [1]
  <barcode>...</barcode> [1]
</pieceLabelData> [1..99]
```

```
<consignmentLabelData> [1]
  <consignmentNumber> xsd:string </consignmentNumber> [1]
  ...
  <marketDisplay>...</marketDisplay> [1]
  <transportDisplay>...</transportDisplay> [0..1]
  <freeCirculationDisplay>...</freeCirculationDisplay> [0..1]
  <sortSplitText> xsd:string </sortSplitText> [0..1]
  <xrayDisplay>...</xrayDisplay> [0..1]
  <originDepot>...</originDepot> [1]
  <transitDepots>...</transitDepots> [0..1]
  <destinationDepot>...</destinationDepot> [1]
  <clusterCode> xsd:string </clusterCode> [0..1]
  ...
</consignmentLabelData> [1]
```

It is important to note that XML does not make any guarantees about the presence of whitespace (spaces/returns/tabs) in the document. You should use standard tools and methodologies for inspecting the contents of the document and not depend on finding elements in a specific character location.

The label below illustrates the parts of the label which vary by package:

```
<pieceLabelData> [0..99]
  <pieceNumber> xsd:int
  </pieceNumber> [1]
  <pieceReference> xsd:string
  </pieceReference> [1]
  <barcode>...</barcode> [1]
</pieceLabelData> [1..99]
```

TNT		INT / AIR X-RAY		C 2	
Con No. 123456782		Service Express		EMA LGG	
Piece of 3	Weight 1.11kg	Option Priority			
Customer Reference piece1		Origin CVT		Pickup Date 03 Nov 2010	
SR Account No 100445		Routing			
Sender Address John Smith First Floor, Room 3 TNT House ATHERSTONE CV9 1TT GB		Delivery Address TNT Corporate Head Office Neptunusstraat 41-63 2132 JA Hoofddorp AMSTERDAM 1011 AA NL		Sort	
Postcode / Cluster Code 01		Dest Depot SP8-4			
 1100123456782011641024000000					

6.2 Detailed description of an XML Label Response Document

The following sections contain a more detailed description of each part of the routing label.

6.2.1 Logo

The logo section of the label is used for branding purposes. A brand compliant TNT logo is available at <http://brandweb.tnt.com/index.asp> and included as part of the developer pack for ExpressLabel which will be provided by your TNT representative. The carrier name will be TNT in all cases.

6.2.2 Market Indicator

The market indicator can be set to either DOM (Domestic) or INT (International). The applicable response XML is:

```
<marketDisplay renderInstructions="yes"><![CDATA[ INT ]]></marketDisplay>
```

In the example XML code, the value that should be rendered in the label is "INT". The `renderInstructions` attribute is used to determine how the market should be displayed. Possible values include:

- yes - the element's value should be displayed black on white
- no - the element's value should not be displayed
- highlighted - the element's value should be displayed inverted (white on black)

The style for this element is as follows:

Font Size: 14pt
Font Family: Arial
Font Weight: Bold



6.2.3 Transport Mode

The transport indicator identifies the mode of transport that the consignment will be transported by, the possible values are:

- AIR
- ROAD
- ROAD – AIR (deprecated)
- AIR – ROAD (deprecated)

This appears in the response as follows:

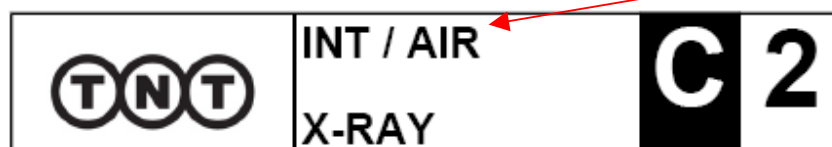
```
<transportDisplay renderInstructions="yes">AIR</transportDisplay>
```

The value in the example above is "AIR", the `renderInstructions` attribute is used to determine how the market should be displayed. Possible values include:

- yes - the element's value should be displayed black on white
- no - the element's value should not be displayed
- highlighted - the element's value should be displayed inverted (white on black)

The style for this element is as follows:

Font Size: 14pt
Font Family: Arial
Font Weight: Bold



6.2.4 Free Circulation Indicator

The free circulation indicator is based on the trade status of the consignment and is used for operational purposes by EU countries to determine if a consignment is customs controlled or not. Possible values are “C” or blank. The XML in the response that corresponds to this is as follows:

```
<freeCirculationDisplay renderInstructions="highlighted">
  <![CDATA[C]]>
</freeCirculationDisplay>
```

The value that should be displayed on the label is between the *freeCirculationDisplay* elements, for example, in this case it would be “C”. The *renderInstructions* attribute is used to determine how the market should be displayed. Possible values include:

- yes - the element's value should be displayed black on white
- no - the element's value should not be displayed
- highlighted - the element's value should be displayed inverted (white on black)

The style for this element is as follows:



6.2.5 Sort Split Indicator

The sort split indicator is used for operational purposes by the Liege Airhub to process the consignment. Possible values for this field are 1 or 2. The value is blank if there is a hazardous product option included in the request. If the consignment is hazardous then the sort split indicator will remain blank. The XML in the response that corresponds to this is as follows:

```
<sortSplitText><![CDATA[2]]></sortSplitText>
```

The value that should be displayed is the text content of the *sortSplitText* element, in this case "2". The style for this element is as follows:



6.2.6 Hazardous

The hazardous indicator on the label is not an xml tag in its own right. Instead it is put on the label when there is a production option HZ. If Hazardous is to be displayed, then the split-sort indicator (6.2.5) should be blank.

The value that should be displayed is the text "HAZARDOUS". The style for this element is as follows:



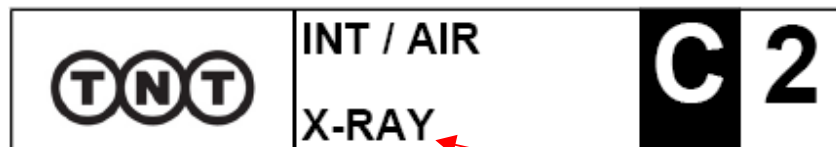
Font Size: 14pt
Font Family: Arial
Font Weight: Bold

6.2.7 X-Ray

The x-ray indicator is used for operational purposes. The XML in the response that corresponds to this is as follows:

```
<xrayDisplay code="" renderInstructions="yes">X-RAY</xrayDisplay>
```

The value that should be displayed is the text content of the `xrayDisplay` element, in this case "X-RAY". The style for this element is as follows:



Font Size: 14pt
Font Family: Arial
Font Weight: Bold

6.2.8 Consignment Number

The consignment number provides a unique identifier for the consignment. The international consignment number consists of 9 numbers; however, local domestic consignment numbers might differ from this. The XML in the response that corresponds to this is as follows:

```
<consignmentNumber>123456782</consignmentNumber>
```

The value that should be displayed on the label is between the `consignmentNumber` tags, in this case "123456782". The recommended styles for this element are as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Normal



Font Size: 18pt
Font Family: Arial
Font Weight: Bold

6.2.9 Product

The Product is used to identify the service level for the consignment, for example: Express, Economy. The maximum length of this field is 30 characters. The related XML is:

```
<product id="EX">Express</product>
```

The value that should be displayed on the label is between the *product* tags, in this example it is "Express". The recommended styles for this element are as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Bold



Font Size: 14pt or 11pt
depending on length of field
Font Family: Arial
Font Weight: Bold

6.2.10 Piece

The piece section of the label displays the current piece number and the total number of pieces for the consignment. If at the time of printing the label, the total number of pieces is unknown, the field will contain a special code (000 or XXX). The XML in the response that corresponds to this is as follows:

```
<pieceNumber>1</pieceNumber>
```


The value that should be displayed on the label is between the `pieceNumber` tags and `totalNumberOfPieces` element respectively, for example, in this case it would be “1 of 3”. The `pieceNumber` represents one of the `sequenceNumber` (s) specified in the request. The recommended styles for this element are as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Normal



Font Size: 14pt
Font Family: Arial
Font Weight: Bold

6.2.11 Weight

The piece section of the label also displays the consignment or piece weight. The preferred choice is for the actual weight to be displayed, if available, otherwise the contractual weight will be shown. For multi-piece shipments where no actual or contractual piece weights are available, no weight will be printed on the label. The contractual weight on consignment level will only be printed in case of single piece shipments (also for self-labeling customers). In case of a piece weight above 25kg, the weight will be displayed in inverse text. The applicable response XML is:

```
<weightDisplay renderInstructions="yes">
  <![CDATA[1.1 Kg]]>
</weightDisplay>
```

The value that should be displayed on the label is between the `weightDisplay` elements, for example, in this case it would be “1.1 Kg”. The `renderInstructions` attribute is used to determine how the weight should be displayed. Possible values include:

- `yes` - the element's value should be displayed black on white
- `no` - the element's value should not be displayed
- `highlighted` - the element's value should be displayed inverted (white on black)

The recommended style for this element is as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Normal



Font Size: 14pt
Font Family: Arial
Font Weight: Bold

For overweight shipments the styles will remain the same; however the weight will be displayed inverted (white on black), as shown below:

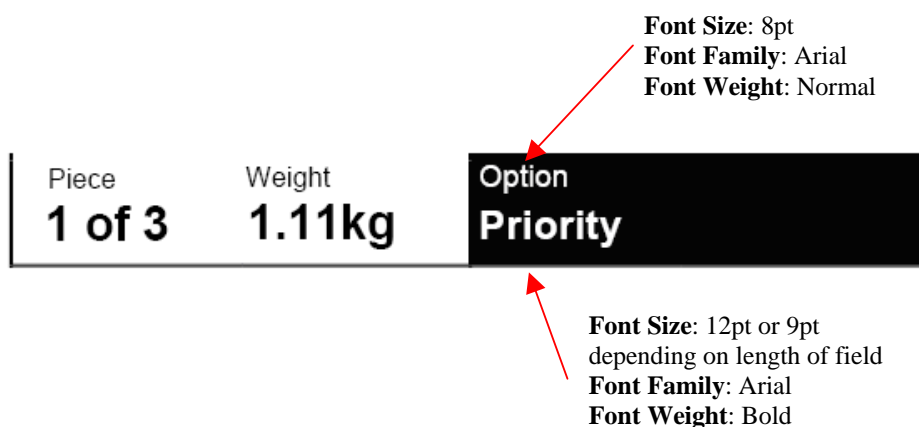


6.2.12 Option

The option section of the label displays any options that can be added as part of the product/service selected, for example priority (PR), insurance (IN). It is possible to have up to 5 options per consignment; the maximum length of an option description is 30 characters, for example, "Pre-delivery Notification (SMS)". The applicable response XML is:

```
<option id="PR"><![CDATA[Priority]]></option>
```

If the consignment only has one option, the value is displayed on the label; however, if the consignment has more than one option, contents of the id attributes are concatenated space delimited. E.g. "PR IN PNS". The recommended styles for this element are as follows:



6.2.13 Customer Reference

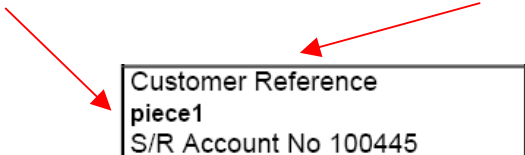
The customer reference provides a method for the customer to identify a particular consignment or piece as an alternative to the consignment number. If a piece reference is supplied, this will display as the *Customer Reference*. If no piece reference is supplied then the customer reference is displayed instead. If no customer reference is supplied then the field is left blank. The maximum length of any value is 24 characters. The XML in the response that corresponds to this is as follows:

```
<pieceReference><![CDATA[piece1]]></pieceReference>
```

The recommended styles for this element are as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Bold

Font Size: 8pt
Font Family: Arial
Font Weight: Normal



Customer Reference
piece1
S/R Account No 100445

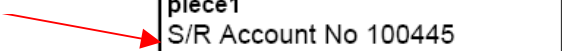
6.2.14 Account

The account maximum field length is 10 digits. Depending on the payment terms this is either the sender or the receiver account number. The XML in the response that corresponds to this is as follows:

```
<account>  
  <accountNumber>100445</accountNumber>  
  <accountCountry>GB</accountCountry>  
</account>
```

The value that should be displayed on the label is between the `accountNumber` tags, for example, in this case it would be “100445”. The recommended styles for this element are as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Normal



Customer Reference
piece1
S/R Account No 100445

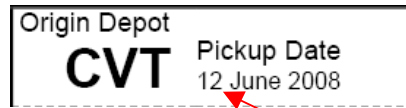
6.2.15 Collection Date

The collection date or pickup date indicates the date on which the consignment will be collected. The XML in the response that corresponds to this is as follows:

```
<collectionDate>2008-06-12</collectionDate>
```

The value that should be displayed on the label is between the `collectionDate` tags, this will need to be rendered into the format “dd MONTH yyyy”, in this case “12 June 2008”. The recommended styles for this element are as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Normal



Font Size: 8pt
Font Family: Arial
Font Weight: Normal

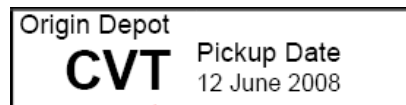
6.2.16 Origin Depot

The origin depot is the depot where the consignment will be taken to once it has been collected from the customer. The depot is displayed on the label as a unique 3 letter code e.g. “CVT”. The XML in the response that corresponds to this is as follows:

```
<originDepot>
  <depotCode>CVT</depotCode>
</originDepot>>
```

The value that should be displayed on the label is between the *depotCode* tags which sit within the *originDepot* Element. The recommended styles for this element are as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Normal



Font Size: 16pt
Font Family: Arial
Font Weight: Bold

6.2.17 Sender & Delivery Address

The sender and delivery address provide information about the location the consignment is going from and to. The field lengths are as follows:

- name – 40 characters
- address line length – 30 characters
- town – 40 characters
- province – 30 characters
- postcode – 9 characters
- country – 2 characters (Must be a valid ISO 3166-1 ALPHA-2 code (two-letter country codes in the ISO 3166-1 standard used to represent countries and dependent territories). See list of valid codes in [appendix C.](#))

The XML in the response that corresponds to this is as follows:

```

<sender>
  <name><![CDATA[Steve Matthews]]></name>
  <addressLine1><![CDATA[TNT Express]]></addressLine1>
  <addressLine2><![CDATA[TNT House]]></addressLine2>
  <town><![CDATA[ATHERSTONE]]></town>
  <province><![CDATA[Warks]]></province>
  <postcode><![CDATA[CV9 1TT]]></postcode>
  <country><![CDATA[GB]]></country>
</sender>
<delivery>
  <name><![CDATA[TNT Corporate Head Office]]></name>
  <addressLine1><![CDATA[Neptunusstraat 41-63]]></addressLine1>
  <addressLine2><![CDATA[2132 JA Hoofddorp]]></addressLine2>
  <town><![CDATA[AMSTERDAM]]></town>
  <province><![CDATA[]]></province>
  <postcode><![CDATA[1011 AA]]></postcode>
  <country><![CDATA[NL]]></country>
</delivery>

```

The value that should be displayed on the label is between the *sender* tags for the sender address and the *delivery* tags for the delivery address. The recommended style for these elements is as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Normal



Sender Address Express Manager TNT Express TNT House Atherstone CV9 1TT GB
Delivery Address TNT Corporate Head Office Neptunusstraat 41-63 2132 JA Hoofddorp Amsterdam 1011 AA NL

6.2.18 Contact

The contact details are part of the response xml only when the contact element is in the initial request. This tag is not required for many of the labels.

```

<contact>
  <name><![CDATA[Fred Blogs]]></name>
  <telephoneNumber>012345 456789</telephoneNumber>
</contact>

```

6.2.19 Routing

The route provides information about any transit depots that the consignment will pass through on route from origin to destination. There can be up to 4 transit depots that are displayed on the label from top to bottom in the order the consignment will pass through them. Each depot is represented by a unique three digit depot code, for example. "EMA (East Midlands Airport)". There are two special transit depot types that contain extra information that needs to be displayed on the label:

- **Sort depot** – The sort depot status applies to all *Document* consignments that pass through LGG (Liege). An extra piece of information known as the sort cell indicator is displayed alongside the depot code. This is used for operational purposes within the depot. Example e.g. 'LGG-8'.
- **Action Depot** – The action depot status applies to all consignments that pass through Duiven (QAR). An extra piece of information known as the "action day of week" is displayed alongside the depot code. This is used for operational purposes within the depot. Example e.g. 'QAR-1'.

The XML in the response that corresponds to this is as follows:

Standard Transit Depot

```
<transitDepot>
  <depotCode>EMA</depotCode>
</transitDepot>
```

The value that should be displayed on the label is between the depotCode tags, for example, in this case it would be "EMA". It is important to make sure that the depotCode tags are enclosed within a *transitDepot* element.

Sort Depot

```
<sortDepot>
  <depotCode>LGG</depotCode>
  <sortCellIndicator renderInstructions="yes">8</sortCellIndicator>
  <sortLocationCode>LGG</sortLocationCode>
</sortDepot>
```

The value that should be displayed on the label is between the *depotCode* elements, for example, in this case it would be "LGG". In addition to this the value between the *sortCellIndicator* elements will need to be added along with the depot code, in this case that would result in a value to be displayed of "LGG - 8". It is important to make sure that the elements used are those enclosed within a *sortDepot* element.

Action Depot

```
<actionDepot>
  <depotCode>QAR</depotCode>
  <actionDayOfWeek>1</actionDayOfWeek>
  <actionDate>2008-06-16</actionDate>
</actionDepot>
```

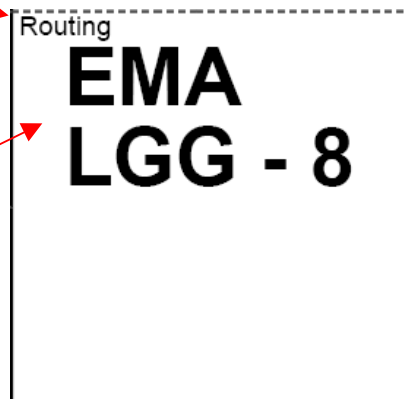
The value that should be displayed on the label is between the depotCode tags, for example, in this case it would be "QAR". In addition to this the value between the *actionDate* tags will need to be added alongside the

depot code, in this case that would result in a value to be displayed of "QAR - 1". It is important to make sure that the elements used are enclosed within an `actionDepot` element.

The recommended styles for this element are as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Normal

Font Size: 24pt
Font Family: Arial
Font Weight: Bold



6.2.20 Airport Sort Code

The airport sort code is used by the Liege Air hub (LGG) only and will be printed for non-docs traveling through LGG. The code helps Liege to make fine sorts on behalf of the destination hub. The XML in the response that corresponds to this is as follows:

```
<sortLocationCode>LGG</sortLocationCode>
```

The recommended styles for this element are as follows:

Font Size: 8pt
Font Family: Arial
Font Weight: Normal



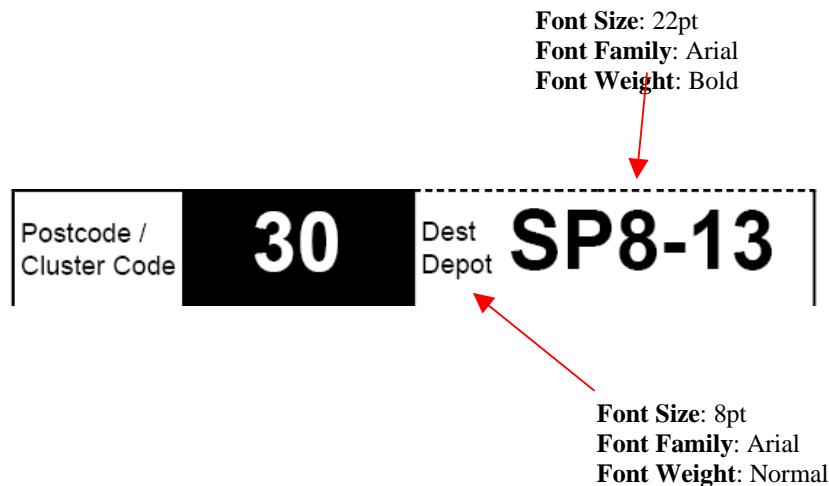
Font Size: 20pt
Font Family: Arial
Font Weight: Bold

6.2.21 Destination Depot

The destination depot is the final depot where the consignment will be taken to before it is delivered to the customer. The depot is displayed on the label as a unique 3 letter code e.g. "SP8". In addition to the depot code the due day of month is also displayed alongside the depot code and this indicates the estimated date the consignment should reach the destination depot. The XML in the response that corresponds to this is as follows:

```
<destinationDepot>
  <depotCode>SP8</depotCode>
  <dueDayOfMonth>16</dueDayOfMonth>
  <dueDate>2008-06-16</dueDate>
</destinationDepot>>
```

The value that should be displayed on the label is between the `depotCode` elements, for example, in this case it would be “SP8”. In addition to this the value between the `dueDayOfMonth` elements should be displayed alongside the depot code, for example, in this case it would be “SP8 - 13”. It is important to make sure that the `depotCode` elements are enclosed with a `destinationDepot` element. The recommended styles for this element are as follows:



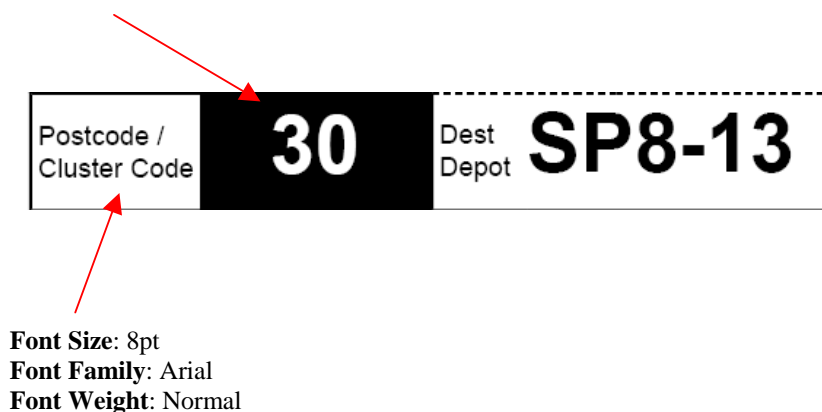
6.2.22 Postcode / Cluster Code

The cluster code is used by the destination depot for operational purposes. If the cluster code is not present then the delivery address postcode is used instead. The XML in the response that corresponds to this is as follows:

```
<clusterCode>30</clusterCode>
```

The value that should be displayed on the label is between the `clusterCode` elements, for example, in this case it would be "30". Even if the cluster code is not found the delivery postcode will still be included within the `clusterCode` elements. The recommended styles for this element are as follows:

Font Size: 26pt for cluster code and 16pt for postcode
Font Family: Arial
Font Weight: Normal



6.2.23 Legal Comments

Legal comments are provided for some labels but not all. Domestic country labels, such as the French Domestic label, have specific legal comments included.

```
<legalComments>SOU MIS AUX CONDITIONS GENERALES DU TRANSPORT</legalComments>
```

6.2.24 Cash Amount

The Cash Amount appears on some labels when the 'Cash On Delivery' production option is selected, as is the case of the French Domestic label. This is an optional tag in the request and if not present as part of the request xml will not be present in the output xml.

```
<cashAmount code="12.34" renderInstructions="yes">EUR 12,34</cashAmount>
```

6.2.25 Special instructions

This is another tag which only appears when provided on the request xml and only appears on some labels but not all.

```
<specialInstructions>Please be careful as there are currently no road markings</specialInstructions>
```

6.2.26 Barcode

The barcode is used operationally to quickly identify and process the consignment; the barcode standard is Code128 with a length of 28. Individual domestic country labels may have multiple barcodes. In this case the length of the additional barcodes may vary. The XML in the response that corresponds to this is as follows:

```
<barcode symbology="128C">1100123456783011641011000000</barcode>
```

The barcode value that should be displayed on the label is between the *barcode* elements. This value will need to be passed to a barcode generator such as [Barbecue](#) to create the actual barcode.

The recommended styles for this element are as follows:



Font Size: 9pt
Font Family: Arial
Font Weight: Normal

The barcode should be printed horizontally and use the following dimensions:

- **Height:** minimum 30mm
- **Width:** minimum 70mm
- **X-Dimension:** 0.4mm
- **Quiet zones:** Quiet zones are the white space left, right, above and below the barcode. They will be 10 * X-dimension but with a minimum of 5mm

This can only really be tested with a barcode reader and support can usually be found at your nearest depot.

6.2.27 Barcode For Customer

This is a tag which is only present in the output of xml for some labels but not all. The French Domestic label allows for the creation of a barcode for the `pieceReference` tag which is provided as part of the `pieces` tag in the request. The `barcodeForCustomer` does not have the same size constraints as the main TNT consignment barcode and may vary from label to label. However, the tag does have similar structure to the main barcode tag as can be seen from this example.

```
<barcodeForCustomer symbology="128C">135792468</barcodeForCustomer>
```

7. Errors

There are a number of different errors that may occur when using ExpressLabel. Many of these are likely to be encountered in the initial development phase and are concerned with the format of the XML message and the presence of data items.

The remaining messages are concerned with validation of the data items and the availability of the service. The error messages are shown below:

It would be sensible to ensure that your code is capable of handling all of the potential error messages returned by ExpressLabel.

7.1 Application generated errors

Application errors are included in your returned XML as they occur and take the following format:

- BrokenRules – These errors are returned as result of verification and validation of the request data.

```
<?xml version="1.0" encoding="UTF-8"?>
<labelResponse>
  <brokenRules key="CON1">
    <errorCode>1003</errorCode>
    <errorDescription>Consignment number is not the correct
length.</errorDescription>
  </brokenRules>
</labelResponse>
```

- Fault – These errors are returned as result of an unexpected exception that has occurred during a request.

```
<?xml version="1.0" encoding="UTF-8"?>
<labelResponse>
  <fault key="CON1"/>
</labelResponse>
```

- XML processing error – These errors are returned if the XML supplied with the request does not comply with the ExpressLabel request schema and thus cannot be successfully parsed.

```
Error 406: Unable to process request message:Bad DateTime format: 2008--
12T13:00:00
DateTime is not long enough
```

For possible BrokenRules errors see Table of application generated error codes, messages and resolutions on the following pages.

7.2 Table of application generated error codes, messages and resolutions

The Default Message column contains the English language message that is returned with each code. This value is intended to be used by systems integrators. For content management you should use the error code.

An element is considered empty if it contains no value or only whitespace. Therefore all of the following elements are empty:

```
<emptyElement/>

<emptyElement></emptyElement>

<emptyElement>                </emptyElement>
```

Key:

* Error codes marked with this symbol are not currently used as part of the application.

If this field is empty the value will default to "0" which will either result in a valid label or a 9200 error.

Error Code	Error Description	Default Message	Resolution
1001	The <code>consignmentIdentity</code> element is missing.	Consignment Identity must be provided.	Ensure <code>consignmentIdentity</code> element is provided. You will get this error is the element is altogether missing from the request. If the element is present but its child elements are missing, you will get Codes 1002, or 1004
1002	The <code>consignmentNumber</code> element is missing, empty or not numeric.	Consignment number must be entered.	Ensure <code>consignmentNumber</code> element is provided and has numeric data. Example problems are shown below: Empty element: <consignmentNumber/> Alpha characters in the con number value: <consignmentNumber>GB123456782X</consignmentNumber>
1003	Consignment number provided is not the correct length.	Consignment number is not the correct length.	Ensure consignment number is 9 digits long.

Error Code	Error Description	Default Message	Resolution
1004 *	The customerReference element is missing or empty.	Customer reference must be entered.	Ensure customerReference element is provided and has data.
1005	Customer reference exceeds maximum length.	Customer Reference has exceeded its maximum length.	Ensure customer reference is no longer than 25 characters.
2001	The collectionDateTime element is missing or empty.	Collection date must be entered.	Ensure collectionDateTime element is provided and has data.
3001	Sender address (the sender element) is missing	A sender address must be provided.	Ensure sender element is provided.
3002	Sender address name (the name element within the sender element) is missing or empty.	Sender address: name must be entered	Ensure the name element within the sender element is provided and has data.
3003	Sender address name exceeds its maximum length.	Sender address: name has exceeded its maximum length	Ensure sender address name is no longer than 40 characters.
3004	Sender address line 1 (the addressLine1 element within the sender element) is missing or empty.	Sender address: address line 1 must be entered.	Ensure the addressLine1 element within the sender element is provided and has data.
3005	Sender address line 1 exceeds its maximum length.	Sender address: address line 1 has exceeded its maximum length.	Ensure sender address line 1 is no longer than 30 characters.
3006	Sender address line 2 (the addressLine2 element within the sender element) exceeds its maximum length.	Sender address: address line 2 has exceeded its maximum length.	Ensure sender address line 2 is no longer than 30 characters.
3007	Sender address line 3 (the addressLine3 element within the sender element) exceeds its maximum length.	Sender address: address line 3 has exceeded its maximum length.	Ensure sender address line 3 is no longer than 30 characters.
3008	Sender address town (the town element within the sender element) is missing or empty.	Sender address: town must be entered	Ensure the town element within the sender element is provided and has data.
3009 *	Sender address town (the town element within the sender element) and postcode (the postcode element within the sender element) are null or empty. This may occur for countries that require both a town and postcode to be entered.	Sender address: town & postcode must be entered	Ensure the town and postcode elements within the sender element are provided and have data.

Error Code	Error Description	Default Message	Resolution
3010	Sender address town could not be found. This may occur for a town that is entered that is not valid for the country and province chosen.	Sender address: town could not be found	Ensure the town element within the sender element is entered correctly.
3011	Sender address town exceeds it maximum length.	Sender address: town has exceeded its maximum length	Ensure sender town is no longer than 40 characters.
3012	Sender address town results in more than one match.	Sender address: town results in more than one match	Ensure sender town and/or postcode are entered correctly. Do not use partial town or postcodes.
3013	Sender address province (the province element within the sender element) exceeds it maximum length.	Sender address: province has exceeded its maximum length	Ensure sender province is no longer than 30 characters.
3014 *	Sender province has been entered for a country that does not accept province as part of the address.	Sender address: provinces not accepted in the address for this country	Remove the province element from within the sender element.
3015 *	Sender address province could not be found. This may occur for a province that is entered that is not valid for the country and postcode chosen.	Sender address: province could not be found	Ensure sender province is entered correctly or remove the province element from within the sender element.
3016	Sender address postcode (the postcode element within the sender element) is invalid.	Sender address: postcode is invalid	Ensure the postcode element within the sender element is entered correctly.
3017	Sender address postcode has an invalid format. For example CV9 XTT is incorrect for a UK postcode.	Sender address: postcode format is invalid	Ensure the postcode element within the sender element is entered correctly and is no longer than 9 characters.
3018	Sender address postcode matches more than one town. This may occur if a partial postcode is entered.	Sender address: postcode matches more than one town	Ensure sender postcode and/or town are entered correctly.
3019	Sender address has an invalid town / postcode combination. For example if you entered a London postcode for Birmingham.	Sender address: invalid town postcode combination	Ensure sender postcode and/or town are entered correctly.
3020	Sender address postcode has been entered for a country that does not accept postcode as part of the address.	Sender address: postcode not accepted in the address for this country.	Remove the postcode element from within the sender element.

Error Code	Error Description	Default Message	Resolution
3021 *	Sender address postcode could not be found. This may occur for a postcode that is entered that is not valid for the country and town chosen.	Sender address: postcode could not be found	Ensure sender postcode is entered correctly.
3022 *	Sender address postcode is not within the valid range of postcodes for the town.	Sender address: postcode not within the valid range for the town selected	Ensure sender postcode and/or town are entered correctly.
3023	Sender address country (the country element within the sender element) is missing or empty.	Sender address: country must be entered	Ensure the country element within the sender element is provided and has data.
3024 *	Sender address country is not the correct length.	Sender address: country is not the correct length.	Ensure the sender country is exactly 2 characters long.
3025	Sender address country is not a valid ISO 3166-1 ALPHA-2 code (two-letter country codes in the ISO 3166-1 standard used to represent countries and dependent territories). See list of valid codes in appendix D.	Sender address: country is not valid ISO 3166-1 ALPHA-2.	Ensure sender country is entered correctly.
3026	Sender address has more than one match. The underlying system cannot distinguish between two physical addresses given the supplied sender address information. For example where a postcoded country has two towns with the same name and the user has not supplied a postcode.	Sender address: address has more than one match	Ensure sender postcode, town and/or country are entered correctly.
4001	Delivery address (the delivery element) is missing	A delivery address must be provided.	Ensure delivery element is provided.
4002	Delivery address name (the name element within the delivery element) is missing or empty.	Delivery address: name must be entered	Ensure the name element within the delivery element is provided and has data.
4003	Delivery address name exceeds its maximum length.	Delivery address: name has exceeded its maximum length	Ensure delivery address name is no longer than 40 characters.
4004	Delivery address line 1 (the addressLine1 element within the delivery element) is missing or empty.	Delivery address: address line 1 must be entered.	Ensure the addressLine1 element within the delivery element is provided and has data.

Error Code	Error Description	Default Message	Resolution
4005	Delivery address line 1 is too long.	Delivery address: address line 1 has exceeded its maximum length.	Ensure delivery address line 1 is no longer than 30 characters.
4006	Delivery address line 2 (the addressLine2 element within the delivery element) exceeds its maximum length.	Delivery address: address line 2 has exceeded its maximum length.	Ensure delivery address line 2 is no longer than 30 characters.
4007	Delivery address line 3 (the addressLine3 element within the delivery element) exceeds its maximum length.	Delivery address: address line 3 has exceeded its maximum length.	Ensure delivery address line 3 is no longer than 30 characters.
4008	Delivery address town (the town element within the delivery element) is missing or empty.	Delivery address: town must be entered	Ensure the town element within the delivery element is provided and has data.
4009 *	Delivery address town and postcode are null or empty. This may occur for countries that require both a town and postcode to be entered.	Delivery address: town & postcode must be entered	Ensure the town and postcode elements within the delivery element are provided and have data.
4010	Delivery address town could not be found. This may occur for a town that is not valid for the country and province chosen.	Delivery address: town could not be found	Ensure the town element within the delivery element is entered correctly.
4011	Delivery address town exceeds its maximum length.	Delivery address: town has exceeded its maximum length	Ensure delivery town is no longer than 40 characters.
4012	Delivery address town results in more than one match.	Delivery address: town results in more than one match	Ensure delivery town and/or postcode are entered correctly.
4013	Delivery address province (the province element within the delivery element) exceeds its maximum length.	Delivery address: province has exceeded its maximum length	Ensure delivery province is no longer than 30 characters.
4014 *	Delivery address province has been entered for a country that does not accept province as part of the address.	Delivery address: provinces not accepted in the address for this country	Remove the province element from within the delivery element.
4015 *	Delivery address province could not be found. This may occur for a province that is entered that is not valid for the	Delivery address: province could not be found	Ensure delivery province is entered correctly or remove the province element from within the delivery element.

Error Code	Error Description	Default Message	Resolution
	country and postcode chosen.		
4016	Delivery address postcode (the postcode element within the delivery element) is invalid.	Delivery address: postcode is invalid	Ensure the postcode element within the delivery element is entered correctly.
4017	Delivery address postcode has an invalid format. For example CV9 XTT is an incorrect format for a UK postcode.	Delivery address: postcode format is invalid	Ensure the postcode element within the delivery element is entered correctly and is no longer than 9 characters.
4018	Delivery address postcode matches more than one town. This may occur if a partial postcode is entered.	Delivery address: postcode matches more than one town	Ensure delivery postcode and/or town are entered correctly.
4019	Delivery address has an invalid town / postcode combination. This may occur if the postcode entered is not linked to the town entered, for example if you entered a London postcode for Birmingham.	Delivery address: invalid town postcode combination	Ensure delivery postcode and/or town are entered correctly.
4020	Delivery address postcode has been entered for a country that does not accept postcode as part of the address.	Delivery address: postcode not accepted in the address for this country.	Remove the postcode element from within the delivery element.
4021 *	Delivery address postcode could not be found. This may occur for a postcode that is entered that is not valid for the country and town chosen.	Delivery address: postcode could not be found	Ensure delivery postcode is entered correctly.
4022 *	Delivery address postcode is not within the valid range of postcodes for the town.	Delivery address: postcode not within the valid range for the town selected	Ensure delivery postcode and/or town are entered correctly.
4023	Delivery address country (the country element within the delivery element) is missing or empty.	Delivery address: country must be entered	Ensure the country element within the delivery element is provided and has data.
4024 *	Delivery address country is not the correct length.	Delivery address: country is not the correct length.	Ensure the delivery country is exactly 2 characters long.
4025	Delivery address country is not a valid ISO 3166-1 ALPHA-2 code See list of valid codes in appendix D.	Delivery address: country is not valid ISO 3166-1 ALPHA-2.	Ensure delivery country is entered correctly.

Error Code	Error Description	Default Message	Resolution
4026	Delivery address has more than one match. For example where a post coded country has two towns with the same name and the user has not supplied a postcode.	Delivery address: address has more than one match	Ensure delivery postcode, town and/or country are entered correctly.
4031	The <code>postcode</code> element is missing	Delivery address: postcode must be entered	Ensure the delivery postcode is present. Postcode is normally optional but some domestic labels (eg FR) have the postcode as mandatory
5001	The <code>product</code> element is missing	Product must be provided.	Ensure product element is provided.
5002 #	Line of business code is missing.	Line of business must be entered.	Ensure the <code>lineOfBusiness</code> element within the product element is provided and has data of 1 numeric character.
5003 #	Product group id is missing.	Product group id must be entered.	Ensure the <code>groupId</code> element within the product element is provided and has data of 1 numeric character.
5004 #	Product sub group id is missing.	Product sub group id must be entered.	Ensure the <code>subGroupId</code> element within the product element is provided and has data of 1 numeric character.
5005	Product id (the <code>id</code> element within the product element) missing or empty.	Product id must be entered.	Ensure the <code>id</code> element within the product element is provided and has data of no more than 4 characters.
5006 *	No service is available for the product entered.	No feasible service available for the requested product.	Ensure the details for the elements within the product element are entered correctly.
5007 *	One of the options is not valid for the product entered.	Option is not valid for the product chosen.	Ensure the <code>option</code> element or elements within the product element are entered correctly and each has data of no more than 4 characters.
6001	The <code>account</code> element is missing.	Account must be provided.	Ensure account element is provided.
6002	Account number is missing or empty.	Account number must be entered.	Ensure the <code>accountNumber</code> element within the account element is provided and has data.
6003	Account number is not the correct length.	Account number is not the correct length.	Ensure account number is no longer than 10 numeric characters.
6004	Account country is missing or empty. Country code needs to be a valid ISO 3166-1 ALPHA-2 code (two-letter country codes in the ISO 3166-1 standard used to represent countries and dependent territories). See list of valid codes in appendix D.	Account country must be entered.	Ensure the <code>accountCountry</code> element within the account element is provided and has data.
6005	Account country is not the correct length.	Account country is not the correct length.	Ensure account number is exactly 2 characters.
7001	The <code>totalNumberOfPieces</code>	Total number of pieces must be	Ensure <code>totalNumberOfPieces</code> element is provided with numeric

Error Code	Error Description	Default Message	Resolution
	element is missing or empty.	entered.	data.
7002	The total number of pieces is less than the number of pieces supplied.	Total number of pieces declared is less than the number of pieces supplied.	Check total number of pieces entered and the number of piece lines provided (see below for piece lines) and correct so total is not less than the number of piece lines provided.
7003	The total number of pieces exceeds the maximum allowed.	Cannot have a total number of pieces greater than 999.	Ensure value entered in <code>totalNumberOfPieces</code> element is no greater than 999.
7004	The piece sequence number exceeds the total number of pieces declared.	Piece sequence higher than the total number of pieces declared.	Ensure value entered in <code>totalNumberOfPieces</code> element is greater than or equal to the highest <code>sequenceNumbers</code> .
8001	No <code>pieceLine</code> elements are provided.	Piece lines must be provided.	Ensure at least one <code>pieceLine</code> element is provided.
8002	Piece line identifier is missing or empty.	Piece line identifier must be entered.	Ensure the <code>identifier</code> element within the <code>pieceLine</code> element is provided and has data.
8003	Piece line goods description is missing or empty.	Piece line goods description must be entered.	Ensure the <code>goodsDescription</code> element within the <code>pieceLine</code> element is provided and has data.
8004	Piece line goods description exceeds its maximum length.	Piece line goods description has exceeded its maximum length	Ensure the goods description is no longer than 30 characters.
9001	No piece measurements are provided.	Piece measurements must be provided.	Ensure <code>pieceMeasurements</code> element is provided within the <code>pieceLine</code> element.
9002	Piece length is missing or empty.	Piece measurements: length must be entered.	Ensure the <code>length</code> element is provided within the <code>pieceMeasurements</code> element with numeric data (2 decimal places).
9003	Piece length not in the valid range.	Piece measurements: length value not within valid range.	Ensure piece length value is no greater than 100.
9004	Piece width is missing or empty.	Piece measurements: width must be entered.	Ensure the <code>length</code> element is provided within the <code>pieceMeasurements</code> element with numeric data (2 decimal places).
9005	Piece width is not within the valid range.	Piece measurements: width value not within valid range.	Ensure piece width value is less than 100.
9006	Piece height is missing or empty.	Piece measurements: height must be entered.	Ensure the <code>height</code> element is provided within the <code>pieceMeasurements</code> element with numeric data (2 decimal places).
9007	Piece height is not within the valid range.	Piece measurements: height value not within valid range.	Ensure piece height value is less than 100.
9008	Piece weight is missing or empty.	Piece measurements: weight must be entered.	Ensure the <code>weight</code> element is provided within the <code>pieceMeasurements</code> element with numeric data (3 decimal places).
9009	Piece weight is not in the valid range.	Piece measurements: weight value not within valid range.	Ensure piece weight value is less than 100000.
9101	No pieces are provided.	Pieces must be provided.	Ensure <code>pieces</code> element is provided within the <code>pieceLine</code> element.

Error Code	Error Description	Default Message	Resolution
9102 *	Piece reference is missing or empty.	Piece reference must be entered.	Ensure <code>pieceReference</code> element is provided within the <code>pieces</code> element.
9103	Piece reference exceeds its maximum length.	Piece Reference has exceeded its maximum length.	Ensure piece reference is no longer than 25 characters.
9104	Piece sequences are missing or empty.	Piece sequence numbers must be entered.	Ensure <code>sequenceNumbers</code> element is provided with data within the <code>pieces</code> element.
9105	Piece sequences provided are invalid e.g. negative or zero.	Invalid piece sequence number value.	Ensure the <code>sequenceNumbers</code> element within the <code>piece</code> element is entered correctly. Note: 1 error is returned PER invalid sequence number, so for '0, -1, -2', three errors are returned.
9106	Piece line has too many pieces exceed the maximum allowed.	More than the maximum 99 pieces provided on a piece line	Remove one or more of <code>pieces</code> elements within the <code>pieceLine</code> element or remove some of sequence numbers from one of the <code>sequenceNumbers</code> element within one of the <code>pieces</code> elements such that total pieces defined does not exceed 99.
9200	A system CDC_6004 common codes error (no data returned) has been encountered. There are a number of different causes that may lead to this error being returned.	Issue with validating TNT common systems data	Contact local CIT representative with a copy of the XML supplied.

8. Connecting to ExpressLabel

ExpressLabel is a HTTP Web Service. It defines and supports the XML, Hessian, and Burlap data protocols. In the sections below we have shown various examples, using Java and PHP. XML is the preferred data protocol as it provides more backwards compatibility between versions.

In order to connect to ExpressLabel, therefore, you must construct a request using the chosen data protocol.

8.1 Choosing a request protocol

How do we choose a protocol? In order to select an appropriate protocol, you should identify the technology you intend to use. Various technologies provided different levels of support for Hessian, Burlap, and XML. In addition, the quality of internet connection between your application (hereafter known as the Client System) and ExpressLabel will contribute towards your decision.

The ExpressLabel system performs comparably for any of the protocols listed below. The contribution of the network to the performance of the system is therefore the main consideration in choosing the right protocol.

In order of size of the messages, Hessian is most efficient, followed by XML, and then Burlap. While in many locations, internet performance can be expected to be sub-second, this can vary according to the exact region, the time of day and the number of users connected. As a general rule, therefore, the XML protocol is the recommended.

8.1.1 XML

XML is the default and recommended choice for connecting to ExpressLabel. The message size is larger than with Hessian but XML has the big advantage that it is backwards compatible as new versions are released. XML is usually much simpler to develop as the message are human readable and can be constructed using simple String manipulation.

If an xml transfer is to be used the http header Content-Type must be set to either 'Content-Type: text/xml' or 'Content-Type: application/x-www-form-urlencoded'.

8.1.2 Hessian

Hessian is a binary protocol aimed at web services. For these reasons it is efficient and very suitable for internet connections, where the quality of the connection can vary over time. Hessian messages are typically 1/5 the size of equivalent XML messages. More information about Hessian can be found at <http://hessian.caucho.com>, and an example program is available with this document which outlines the main steps, see [Java Hessian Connection Example](#).

With Hessian the Objects are serialized and then de-serialized at the receiving end. This means that the objects which are being serialized must be the same at both ends. The biggest disadvantage of both Hessian and Burlap is that when an upgrade to ExpressLabel affects the transferred Objects, then the client side Objects must also be updated. This is a significant drawback to using a binary protocol.

8.1.3 Burlap

Burlap is a simplified XML protocol which offers an automated way to produce requests. It is provided by the same organisation which defines the Hessian protocol and therefore software written for Hessian can be seamlessly swapped to use Burlap, see <http://hessian.caucho.com/doc/burlap.xtp>.

As Burlap is more verbose than our own XML request definition, it is not recommended for production use. You may wish to use it for a period during development as it results in human-readable messages which can be helpful for debugging.

The following section provides code examples that explain how to connect to ExpressLabel.

8.2 Java XML Connection Example

Even for a simple XML connection a number of jar files are recommended,

- a. commons-httpclient-3.1.jar (<http://hc.apache.org/httpclient-3.x/>)
- b. commons-logging-1.1.1.jar (<http://commons.apache.org/>)
- c. commons-codec-1.3.jar (<http://commons.apache.org/>)

There are a number of useful xml libraries as part of the standard Java SE version, which may also be useful in parsing the result xml.

```
public class XMLTestApp {

    public static void main(String[] args) throws IOException {

        // Create the xml request
        StringBuilder xmlRequest = new StringBuilder();
        xmlRequest.append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
        xmlRequest.append("<labelRequest>");
        // complete the xml...

        // Post xml request to ExpressLabel
        String url =
            "https://express.tnt.com/expresslabel/documentation/getlabel";
        PostMethod post = new PostMethod(url);

        // Create the authentication element
        String userPassword = "user" + ":" + "password";
        String encoding = Base64.encodeBase64(userPassword.getBytes());
        post.setRequestHeader("Authorization", "Basic " + encoding);

        // Create the entity and put it in the post method
        RequestEntity entity =
            new StringRequestEntity(xmlRequest, "text/xml", null);
        post.setRequestEntity(entity);

        // Create a HttpClient to do the transfer
        HttpClient httpClient = new HttpClient();
        int result = httpClient.executeMethod(post);

        // Get the response as bytes or a stream for parsing the xml response
        byte[] xmlBytes = post.getResponseBody();
        InputStream is = post.getResponseBodyAsStream();
    }
}
```

8.3 Java Hessian Connection Example

In order to run this example, you will need a number of jars. These will be supplied by your CIT representative as part of the engagement for ExpressConnect adoption, once the client technology is identified.

The libraries required to run this code example are as follows and are included in the example code download, see your TNT representative for further information:

- a. cts-documentation-x.x.x.jar (Only the iface package is required if this is available separately)
- b. expresslabel-castor-x.x.x.jar
- c. cts-common-x.x.x.jar
- d. commons-httpclient-3.1.jar (<http://hc.apache.org/httpclient-3.x/>)
- e. hessian-3.1.3.jar (<http://hessian.caucho.com/#Java>)
- f. commons-logging-1.1.1.jar (<http://commons.apache.org/>)
- g. commons-codec-1.3.jar (<http://commons.apache.org/>)

where x.x.x will be the current version required for connection to production. Please note that the following code depends on a JDK5 implementation.

8.3.1 Create a Marshaller

The first step is to create a simple Hessian Marshaller class that will be used to convert Java objects to Hessian and back again, see code below:

```
public class Hessian2Marshaller {
    /**
     * Convert the Java objects into a byte array then send it to the service.
     */
    public byte[] marshal(Object obj) throws IOException {
        if (obj == null) {
            throw new IllegalArgumentException("No object supplied");
        }
        ByteArrayOutputStream out = new ByteArrayOutputStream();

        Hessian2Output s = new Hessian2Output(out);
        s.writeObject(obj);
        s.flush();
        out.flush();
        out.close();
        return out.toByteArray();
    }
}
```

```
/**
 * Take the input stream from the http request and convert it to Java
 */
public Object unmarshal(InputStream in) throws IOException {
    if (in == null)
        throw new IllegalArgumentException("No input stream supplied");
    }
    Hessian2Input hessianInput = new Hessian2Input(in);
    return hessianInput.readObject();
}
}
```

8.3.2 Populate the request data objects

The next step is to create an ELRoutingLabelRequestVO object to pass consignment data to ExpressLabel. The method below can be used to create a valid request. In the example code, the data is provided statically for the sake of clarity and brevity rather than populated from a form, client system, or file.

8.3.3 Create the client integrator

The client integrator pulls all the elements together to make a request and receive the response. In our example, we create a java class with a main method.

We initialise the request that will be posted to ExpressLabel. The call to the `setupRoutingRequestCon()` method mentioned previously will return an `ELRoutingLabelRequestVO` object containing all of the required request data for a single consignment. For a real application, this data would not be provided statically but injected from a file, website, or client system.

The objects are stored in an Array List as this enables multiple consignments to be processed in one call. There is currently a maximum limit of 5 consignments per batch.

```
public class HessianTestApp {

    public static void main(String[] args) throws IOException {

        // Create routing request object
        ELRoutingLabelRequestVO labelRequest = setupRoutingRequestCon();

        List<ELRoutingLabelRequestVO> labelRequests =
            new ArrayList<ELRoutingLabelRequestVO>();
        labelRequests.add(labelRequest);

        // Create routing response object
        List<ELRoutingLabelResponseVO> labelResponse =
            new ArrayList<ELRoutingLabelResponseVO>();
```

- We obtain an instance of the Hessian Marshaller.
- Call the marshal method, which returns a byte array containing the Hessian request data.

```
// Get instance of ELRoutingLabelResponseVO Marshaller and BasicRequestHandler
Hessian2Marshaller hessian2Marshaller = new Hessian2Marshaller();

// Convert Request object to Hessian
byte[] hessianRequest = hessian2Marshaller.marshal(labelRequests);
```

- Setup a POST request to call ExpressLabel using the Apache HttpClient classes
- Define the URL where ExpressLabel is hosted and initialize the POST request object. You may wish to externalize the configuration of this information in case the details change in the future.
- Encode username and password required by ExpressLabel security and add to request headers.
- Create a request entity which acts as a container for the Hessian request & add to POST request object.


```
// Post Hessian request to ExpressLabel
String url = "https://express.tnt.com/expresslabel/documentation/getlabel";
String userPassword = "user" + ":" + "password";
String encoding = Base64.encodeBase64(userPassword.getBytes());

PostMethod post = new PostMethod(url);

post.addRequestHeader("Authorization", "Basic " + encoding);

RequestEntity entity = new InputStreamRequestEntity(
    new ByteArrayInputStream(hessianRequest), "hessian2");

post.setRequestEntity(entity);
```

- Send POST request to ExpressLabel.
- Retrieve Hessian response in the form of a byte array.
- Call the unmarshal method on the Hessian Marshaller, which converts the Hessian response into an Array List of ELRoutingLabelResponseVO objects.

```
HttpClient httpclient = new HttpClient();
try {
    httpclient.executeMethod(post);
    byte[] hessianResponse = post.getResponseBody();

    // Convert Hessian response to ELRoutingLabelResponseVO objects
    labelResponse = (List<ELRoutingLabelResponseVO>)
        hessian2Marshaller.unmarshal(new ByteArrayInputStream(hessianResponse));
} catch (HttpException e) {
    e.printStackTrace();
} finally {
    // Release current connection to the connection pool once you are done
    post.releaseConnection();
}
```

- Display the response. In this example, we have output the response to the console. In production code, the response would be processed to produce a label to be sent to the printer or saved to a database (Not part of ExpressLabel) for later processing. An excerpt of the extraction is provided below:

```

RoutingLabelDataVO routingLabelData = labelResponse.get(0).getRoutingLabelData();

System.out.println("Broken Rules");
System.out.println(routingLabelData.getBrokenRules() + "\n");

if (labelResponse.get(0).getRoutingLabelData().getBrokenRuleCount() == 0) {

    System.out.println("Consignment Identity");

    System.out.println(routingLabelData.getConsignmentData().getConsignmentIdentity
());
    System.out.println("collectionDate=" +
        routingLabelData.getConsignmentData().getCollectionDate());
    System.out.println("market=" +
        routingLabelData.getConsignmentData().getMarketDisplay().getOutputText());
}

```

8.4 PHP Example

This example, written in PHP requires that the curl and OpenSSL extensions are running within the PHP installation. To confirm that these extensions are enabled, run `phpinfo()` and look for the following sections in the results.

Curl

cURL support	enabled
cURL Information	libcurl/7.16.0 OpenSSL/0.9.8 zlib/1.2.3

OpenSSL

OpenSSL support	enabled
OpenSSL Version	OpenSSL 0.9.8g 19 Oct 2007

The ExpressLabelConnection class demonstrates how to connect to a remote server and send a request via an http post method. This is all encapsulated in a single method call `httpPost`, which takes the following parameters:

- `$url` : The full url that the request should be submitted to `https://server:port/application-path`
- `$strRequest`: The string data that is to be posted as part of the request
- `$userId`: If authentication is required. The user id to log on with
- `$password`: If authentication is required. The password to log on with

We begin by declaring the class and the `httpPost` method.

```

<?PHP
class ExpressLabelConnection {
    private $errorCode = 0;
    private $errorMessage = "";
    private $socketResponse = "";

    function httpPost($url, $strRequest,$userId, $password) {

```

We open a connection using the properties specified earlier. Our `strRequest` contains the XML which has been constructed in the format specified elsewhere in this document. We submit the request and read the response, looking for any connection errors in the response. Finally we close the connection. A few methods are defined for returning error messages and/or the response.

```

        $ch=curl_init();
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

        $userPass = "";
        if ((trim($userId)!="") && (trim($password)!="")) {
            $userPass = $userId.".".$password;
            curl_setopt($ch, CURLOPT_USERPWD, $userPass);
        }

        curl_setopt($ch, CURLOPT_POST, 1) ;
        curl_setopt($ch, CURLOPT_POSTFIELDS, $strRequest);

        $isSecure = strpos($url,"https://");

        if ($isSecure===0) {
            curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
            curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
        }

        $result = curl_exec($ch);
        $this->errorCode = curl_errno($ch);
        $this->errorMessage = curl_error($ch);
        $this->socketResponse = $result;

        curl_close($ch);
    }

    ...
}
?>

```

An example using the above class is shown below:

```
<?PHP

// Include the PHP file containing the ExpressLabelConnection class
include_once("ExpressLabelConnection.PHP");

//Set a variable containing the url that we will be posting to
$url = 'https://express.tnt.com/expresslabel/documentation/getlabel';

// set a variable containing the data to send in the post
$strRequest = '### some xml data ###';

// set variables containing the authentication data
$userId = "myuserid";
$password = "mypassword";

// get a new instance of the ExpressLabelConnection
$expressLabelConnection = new ExpressLabelConnection();

// Call the httpPost function to send the data to the expresslabel server
$expressLabelConnection->httpPost ($url, $strRequest, $userId, $password);

// Check the error code and process accordingly
if ($expressLabelConnection->getErrorCode() == 0) {
    // All has worked correctly.
    //Now get the response data and do something with it.
    $responseData = $expressLabelConnection->getSocketResponse();
} else {
    // None Zero return code. Something has gone wrong.
    //Handle the error.

    $errorMessage = ($expressLabelConnection->getErrorMessage());
}

?>
```

9. Appendix A: XML elements definition (input)

These tables contain all of the possible Input XML nodes and some information about their use. All elements are mandatory and must contain just one instance unless stated in the table below.

XML elements which do not contain a value but are merely a container for other elements are noted as **Container element**, the elements contained therein are described in the rows immediately below. Please remember that field values should be escaped using the `![CDATA[]]` notation. At a minimum, all address fields should be escaped. Failure to escape these areas could result in unexpected problems if the value in an element contains an Ampersand (&).

For all data types described with a type prefix of 'xsd:' see [XSD Data Types](#).

For all customer data types see Section 10.1 Custom Data Types.

Developer Notes

XML ELEMENT	DESCRIPTION	COMMENTS
labelRequest	Top level element for a message to ExpressLabel. Container element	The root XML element for a Label Request message. This element may contain up to 5 label requests, each one represented by a <code>consignment</code> element.
consignment	The consignment element represents a single request for a routing label(s). Container element	<p>Up to 5 label requests, each one in a <code>consignment</code> element, may be submitted. The information within this element will be used to validate the addresses, determine the route, and produce the label information. Each parcel or piece in the consignment will require its own routing label. For this reason, routing labels are also known as piece labels.</p> <p>Each <code>consignment</code> element must contain a <code>key</code> attribute that uniquely identifies the label request. The value of the key may be any alphanumeric string - typically this value will be an integer value i.e. the first request will have <code>key="1"</code>, second value of <code>key="2"</code> and so on. Any validation errors will be tagged with this key so that you can identify the request in error.</p>
consignmentIdentity	Element to hold the Consignment Number and Customer Reference. Container element	This element identifies the consignment for the operational systems. TNT requires a TNT valid consignment number and you may tag the consignment with your own String customer reference. The customer reference can be used to track a consignment throughout its journey.

Developer Notes

consignmentNumber	Type = consignmentStringLength	The TNT consignment number.
customerReference	Type = xsd:string	<p>Contains the optional customer reference for the consignment. A customer reference is a way for a customer to designate a name for the consignment.</p> <p>This value can be used to track the consignment at a later date.</p>
collectionDateTime	Type = xsd:dateTime	<p>The date and time that the consignment will be collected, to be supplied in the format CCYY-MM-DD'T'hh:mm:ss</p> <p>For example 5:30 p.m. on 30th December 2008 will be supplied as 2008-12-30T17:30:00.</p>
sender	Type = nameAndAddressRequestType / nameAndAddressResponseType Container element	<p>The sender element holds the address from which the consignment is physically collected. This will be used to obtain a route for the consignment and will also appear on the label.</p> <p>Information relating to name and address for a participant in the consignment. Examples of a participant are:</p> <ul style="list-style-type: none"> The Sender - the company sending the consignment The Receiver - the company receiving the consignment The Collection Address - the address from which the consignment is picked up The Delivery Address - the address to which the consignment should be delivered

Developer Notes

		Address information comprises of: name, address line 1, address line 2 (optional), address line 3 (optional), town, postcode (optional depending on the country), province (optional) and country (see below).
delivery	Type = nameAndAddressRequestType / nameAndAddressResponseType Container element	The delivery element holds the address that the consignment is physically delivered to. This will be used to obtain a route for the consignment and will also appear on the label. The delivery element contains all the same children as sender. See below.
name	Type = stringMaxLength40	Either the name of the company as recognised by TNT, or the contact name at the address
addressLine1	Type = stringMaxLength30	The first address line usually comprising house number and street. This address line is the most commonly used of the three address lines and is therefore mandatory.
addressLine2	Type = stringMaxLength30	This address line may not be used by the supporting system and therefore should not contain information essential to the address.
addressLine3	Type = stringMaxLength30	This address line is sometimes used to identify an address more exactly to obtain a route.

Developer Notes

		NOTE - this address line will not appear on any routing label produced.
town	Type = stringMaxLength40	The town name as recognised by TNT.
exactMatch	Type = booleanEnum (see comment)	Optional field to specify if the town name should be used as an exact match or a partial match. Some town names are subsets of other names even with the postcode included. This can lead to error code 3026 or 4026 (multiple addresses found). This flag can help remove these errors. The default value for this option is Y. booleanEnum: Data type for flagging if an exact town name match is required. Must be Y or N
province	Type = stringMaxLength30	Optional field to contain the province, county, state, or area for the given address.
postcode	Type = stringMaxLength9	Postcode or zip code is considered a mandatory field where it is required for a given country. If the postcode is not provided, it may not be possible to deliver the consignment as indicated by your chosen service.
country	Type = stringMinLength2MaxLength2	The ISO 3166-1 Alpha-2 country code for the country of the given address.
contact	Type = contactType	This tag holds the information about the contact person for this consignment

Developer Notes

	Container element	
name	Type = stringMaxLength30	This is the name of the contact and appears on French Domestic labels
telephoneNumber	Type = stringMaxLength30	This is the telephoneNumber of the contact and appears on French Domestic labels
emailAddress	Type = xsd:string	This is the email address of the contact
product	Type = productType Container element	Information relating to the TNT product chosen for this consignment (see elements below). This element contains a lineOfBusiness, groupId, subGroupId, type, id, and a list of options. The product ids you should use will be allocated to you by your TNT representative.
lineOfBusiness	Type = integerMin0Max9	Line of business for product chosen. This value is allocated to you by your TNT representative.
groupId	Type = integerMin0Max9	Group id for product chosen. This value is allocated to you by your TNT representative.
subGroupId	Type = integerMin0Max9	Sub group id for product chosen. This value is allocated to you by your TNT representative.

Developer Notes

id	Type = stringMaxLength4	TNT identifier for product chosen. This value is allocated to you by your TNT representative.
type	Type = ProductTypeEnum (see Comment)	Type of service for product chosen. Values (as defined for ProductTypeEnum) are "D" for a document or "N" for non-documents.
option	Type = stringMaxLength4	Identifier for the type of option chosen for this consignment. Examples include "IN" for Insurance, "PR" for Priority. This element is optional and may have up to 5 occurrences. The list of options allowed will be supplied to you by your TNT representative.
account	Type = accountType Container element	The TNT account paying for the transport of this consignment. Includes information about a TNT account which includes the account number and country code (see below).
accountNumber	Type = longMaxLength10	TNT account number, which is the 9 or 10 digit number assigned by the TNT sales person.
accountCountry	Type = stringMinLength2MaxLength2	ISO 3166-1 Alpha-2 country code for the country in which the TNT account is registered.
specialInstructions	Type = xsd:string	These are the special instructions which should appear on a French Domestic label

Developer Notes

cashAmount	Type = doubleTwoDecimalPlaces	This is the amount which will appear on the French Domestic label when the 'Cash on Delivery' option is selected as part of the product. The code for this option is either 'CO' or 'RP'
totalNumberOfPieces	Type = xsd:int	<p>The total number of pieces this consignment contains. In cases where only some of the pieces are being submitted, this value should contain the total number of pieces in the consignment, not the total number of pieces in the request.</p> <p>This is used to print the sequence numbers on the labels, e.g. 1 of x, where x is the value provided here.</p>
pieceLine	Type = pieceLineType Container element	<p>A piece line is a group of pieces (or parcels) that have the same weight and dimensions. This is a convenience to reduce the amount of data to be transmitted. In plain English this equates to specifying "<i>5 pieces of 0.2m x 0.3m x 0.4m each weighing 1kg</i>" rather than specifying each piece separately.</p> <p>Piece line information comprises a unique identifier, goodsDescription, pieceMeasurements (length, width, height and weight). See elements below.</p>
identifier	Type = xsd:int	Identifier for the piece line so that it can be referenced during processing. Each piece line type should have a unique number.

Developer Notes

goodsDescription	Type = stringMaxLength30	Full description of goods being shipped (catalogue numbers or part numbers will not suffice. The Customs Authorities want to know what each item actually is so please carefully describe the goods).
pieceMeasurements	Type = measurementsType Container element	Dimension (height, width, length) and weight measurements relating to the pieces defined by this piece line type. Data must be provided in metres for dimensions, kilograms for weight. See below.
length	Type = doubleMaxExclusive100MinInclusive0.01	The length in metres. The length is the longest dimension of the piece.
width	Type = doubleMaxExclusive100MinInclusive0.01	The width in metres.
height	Type = doubleMaxExclusive100MinInclusive0.01	The height in metres.
weight	Type = doubleMaxExclusive100000MinInclusive0.01	The weight in kilograms.
pieces	Type = pieceType Container element	At least one of these sections should be provided per consignment up to a maximum of one per piece, and up to 99 per consignment.

Developer Notes

		<p>This element is used to identify all the pieces that should be grouped together by the given reference. The list of sequence numbers is included.</p> <p>One <code>sequenceNumber</code> element per piece with a single <code>pieceReference</code> element.</p>
sequenceNumbers	Type = xsd:string	List of the piece sequence numbers expressed as a comma delimited list, e.g. <i>1,2,5,n</i> out of a total of <i>n</i> pieces. The pieces are grouped by the piece reference.
pieceReference	Type = xsd:string	Customer reference for this piece or pieces. This value can be used to track the piece later date.

<div>Developer Notes</div>

10. Appendix B : XSD Data Types

XSD DATA TYPE	DESCRIPTION
dateTime	<p>Data expected in the format CCYY-MM-DD'T'hh:mm:ss where CC indicates century, YY year, MM month in 2 digit format, DD day of the month, hh hour of the day, mm minutes and ss seconds.</p> <p>For example 5:30 p.m. on 30th December 2008 would be 2008-12-30T17:30:00. This data type describes instances identified by the combination of a date and a time. It is described in Chapter 5.4 of ISO 8601 and the W3C XML Schema Recommendation. Its lexical space is the extended format: [-]CCYY-MM-DDThh:mm:ss[Z (+ -)hh:mm]</p>
double	Numeric data with decimal places in the range -9007199254740991..9007199254740991.
int	Numeric data without decimal places in the range -2147483648..2147483647
long	Numeric data without decimal places in the range -9223372036854775808..9223372036854775807
string	XML compatible alphanumeric data.

Developer Notes

10.1 Custom Data Types

CUSTOM DATA TYPE	DESCRIPTION
stringMinLengthXMaxLengthY	Data type of type xsd:string with a minimum length of <i>X</i> and a maximum length of <i>Y</i> . If 'MinLength' is not specified in the data type name (e.g. stringMaxLength4) there is no minimum length.
stringMaxLength4	As above but without a minimum length
integerMin0Max9	Data type of type xsd:int with a minimum value of 0 and a maximum value of 9.
doubleMaxExclusiveXMinInclusiveY	Data type of type xsd:double with a value that must be less than <i>X</i> . E.g. For 'doubleMaxExclusive100MinInclusive0.01' the value cannot exceed 99.99 or be 0 or less
doubleTwoDecimalPlaces	This defines a double value with two decimal places only.
longMaxLengthX	Data type of type xsd:long with a maximum length of <i>X</i> . E.g. For 'longMaxLength10' the maximum value is 9999999999 (ten 9's).
booleanEnum	This data type defines an enumeration of Y or N only
productTypeEnum	This data type defines 'D' or 'N', standing for Document or Non-document
consignmentStringLength	This only allows for strings of 9 or 16 characters in length.

Developer Notes

11. Appendix C: ISO 3166-1 Alpha-2 Country Codes

Country names	ISO 3166-1-alpha-2 code
A	
AFGHANISTAN	AF
ÅLAND ISLANDS	AX
ALBANIA	AL
ALGERIA	DZ
AMERICAN SAMOA	AS
ANDORRA	AD
ANGOLA	AO
ANGUILLA	AI
ANTARCTICA	AQ
ANTIGUA AND BARBUDA	AG
ARGENTINA	AR
ARMENIA	AM
ARUBA	AW
AUSTRALIA	AU

Developer Notes

AUSTRIA	AT
AZERBAIJAN	AZ
B	
BAHAMAS	BS
BAHRAIN	BH
BANGLADESH	BD
BARBADOS	BB
BELARUS	BY
BELGIUM	BE
BELIZE	BZ
BENIN	BJ
BERMUDA	BM
BHUTAN	BT
BOLIVIA	BO
BOSNIA AND HERZEGOVINA	BA
BOTSWANA	BW
BOUVET ISLAND	BV
BRAZIL	BR

Developer Notes

BRITISH INDIAN OCEAN TERRITORY	IO
BRUNEI DARUSSALAM	BN
BULGARIA	BG
BURKINA FASO	BF
BURUNDI	BI
C	
CAMBODIA	KH
CAMEROON	CM
CANADA	CA
CAPE VERDE	CV
CAYMAN ISLANDS	KY
CENTRAL AFRICAN REPUBLIC	CF
CHAD	TD
CHILE	CL
CHINA	CN
CHRISTMAS ISLAND	CX
COCOS (KEELING) ISLANDS	CC

Developer Notes

COLOMBIA	CO
COMOROS	KM
CONGO	CG
CONGO, THE DEMOCRATIC REPUBLIC OF THE	CD
COOK ISLANDS	CK
COSTA RICA	CR
CÔTE D'IVOIRE	CI
CROATIA	HR
CUBA	CU
CYPRUS	CY
CZECH REPUBLIC	CZ
D	
DENMARK	DK
DJIBOUTI	DJ
DOMINICA	DM
DOMINICAN REPUBLIC	DO
E	

Developer Notes

ECUADOR	EC
EGYPT	EG
EL SALVADOR	SV
EQUATORIAL GUINEA	GQ
ERITREA	ER
ESTONIA	EE
ETHIOPIA	ET
F	
FALKLAND ISLANDS (MALVINAS)	FK
FAROE ISLANDS	FO
FIJI	FJ
FINLAND	FI
FRANCE	FR
FRENCH GUIANA	GF
FRENCH POLYNESIA	PF
FRENCH SOUTHERN TERRITORIES	TF
G	
GABON	GA

Developer Notes

GAMBIA	GM
GEORGIA	GE
GERMANY	DE
GHANA	GH
GIBRALTAR	GI
GREECE	GR
GREENLAND	GL
GRENADA	GD
GUADELOUPE	GP
GUAM	GU
GUATEMALA	GT
GUERNSEY	GG
GUINEA	GN
GUINEA-BISSAU	GW
GUYANA	GY
H	
HAITI	HT
HEARD ISLAND AND MCDONALD	HM

Developer Notes

ISLANDS

HOLY SEE (VATICAN CITY STATE) VA

HONDURAS HN

HONG KONG HK

HUNGARY HU

I

ICELAND IS

INDIA IN

INDONESIA ID

IRAN, ISLAMIC REPUBLIC OF IR

IRAQ IQ

IRELAND IE

ISLE OF MAN IM

ISRAEL IL

ITALY IT

J

JAMAICA JM

JAPAN JP

Developer Notes

JERSEY	JE
JORDAN	JO
K	
KAZAKHSTAN	KZ
KENYA	KE
KIRIBATI	KI
KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF	KP
KOREA, REPUBLIC OF	KR
KUWAIT	KW
KYRGYZSTAN	KG
L	
LAO PEOPLE'S DEMOCRATIC REPUBLIC	LA
LATVIA	LV
LEBANON	LB
LESOTHO	LS
LIBERIA	LR
LIBYAN ARAB JAMAHIRIYA	LY

Developer Notes

LIECHTENSTEIN	LI
LITHUANIA	LT
LUXEMBOURG	LU
M	
MACAO	MO
MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	MK
MADAGASCAR	MG
MALAWI	MW
MALAYSIA	MY
MALDIVES	MV
MALI	ML
MALTA	MT
MARSHALL ISLANDS	MH
MARTINIQUE	MQ
MAURITANIA	MR
MAURITIUS	MU
MAYOTTE	YT

Developer Notes

MEXICO	MX
MICRONESIA, FEDERATED STATES OF	FM
MOLDOVA	MD
MONACO	MC
MONGOLIA	MN
MONTENEGRO	ME
MONTSERRAT	MS
MOROCCO	MA
MOZAMBIQUE	MZ
MYANMAR	MM
N	
NAMIBIA	NA
NAURU	NR
NEPAL	NP
NETHERLANDS	NL
NETHERLANDS ANTILLES	AN
NEW CALEDONIA	NC

Developer Notes

NEW ZEALAND	NZ
NICARAGUA	NI
NIGER	NE
NIGERIA	NG
NIUE	NU
NORFOLK ISLAND	NF
NORTHERN MARIANA ISLANDS	MP
NORWAY	NO
O	
OMAN	OM
P	
PAKISTAN	PK
PALAU	PW
PALESTINIAN TERRITORY, OCCUPIED	PS
PANAMA	PA
PAPUA NEW GUINEA	PG
PARAGUAY	PY

Developer Notes

PERU	PE
PHILIPPINES	PH
PITCAIRN	PN
POLAND	PL
PORTUGAL	PT
PUERTO RICO	PR
Q	
QATAR	QA
R	
RÉUNION	RE
ROMANIA	RO
RUSSIAN FEDERATION	RU
RWANDA	RW
S	
SAINT BARTHÉLEMY	BL
SAINT HELENA	SH
SAINT KITTS AND NEVIS	KN
SAINT LUCIA	LC

Developer Notes

SAINT MARTIN	MF
SAINT PIERRE AND MIQUELON	PM
SAINT VINCENT AND THE GRENADINES	VC
SAMOA	WS
SAN MARINO	SM
SAO TOME AND PRINCIPE	ST
SAUDI ARABIA	SA
SENEGAL	SN
SERBIA	RS
SEYCHELLES	SC
SIERRA LEONE	SL
SINGAPORE	SG
SLOVAKIA	SK
SLOVENIA	SI
SOLOMON ISLANDS	SB
SOMALIA	SO
SOUTH AFRICA	ZA

Developer Notes

SOUTH GEORGIA AND THE SOUTH GS
SANDWICH ISLANDS

SPAIN ES

SRI LANKA LK

SUDAN SD

SURINAME SR

SVALBARD AND JAN MAYEN SJ

SWAZILAND SZ

SWEDEN SE

SWITZERLAND CH

SYRIAN ARAB REPUBLIC SY

T

TAIWAN, PROVINCE OF CHINA TW

TAJIKISTAN TJ

TANZANIA, UNITED REPUBLIC OF TZ

THAILAND TH

TIMOR-LESTE TL

TOGO TG

Developer Notes

TOKELAU	TK
TONGA	TO
TRINIDAD AND TOBAGO	TT
TUNISIA	TN
TURKEY	TR
TURKMENISTAN	TM
TURKS AND CAICOS ISLANDS	TC
TUVALU	TV
U	
UGANDA	UG
UKRAINE	UA
UNITED ARAB EMIRATES	AE
UNITED KINGDOM	GB
UNITED STATES	US
UNITED STATES MINOR OUTLYING ISLANDS	UM
URUGUAY	UY
UZBEKISTAN	UZ

Developer Notes

V

VANUATU

VU

VATICAN CITY STATE

see HOLY SEE

VENEZUELA

VE

VIET NAM

VN

VIRGIN ISLANDS, BRITISH

VG

VIRGIN ISLANDS, U.S.

VI

W

WALLIS AND FUTUNA

WF

WESTERN SAHARA

EH

Y

YEMEN

YE

Z

ZAMBIA

ZM

ZIMBABWE

ZW

a.- All documents provided by TNT for the purpose of technical integration or deployed applications; is TNT PROPRIETARY INFORMATION shared with the customer to support the commercial relationship, not be shared beyond this purpose.

b.- When information needed to be shared with a third party to support operations (for example 3PL), or any other technical third party such as but not limited to software developer; it is the customer sole responsibility to do so, and manage the relationship with their supplier in terms of confidentiality, extending TNT's requirements. Ensuring that their supplier does not retain any information after cease of relationship, nor use TNT information for purpose different than enabling the transactions stated in the commercial agreement.

Developer Notes