Internet Open Trading Protocol

This Document

This document is being submitted to the IETF Trade working group by the Open Trading Protocol Consortium for information only.

The document and the information it contains is provided on an "as is" basis and the Open Trading Protocol Consortium disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

Version: 0.9.9, 17 August 1998

Printed: 17/08/98 19:18

Table of Contents

Figures

0. Preface

The Internet Open Trading Protocol (OTP) provides an interoperable framework for Internet commerce. It is payment system independent and encapsulates payment systems such as SET, Mondex, CyberCash, DigiCash, GeldKarte, etc. OTP is able to handle cases where such merchant roles as the shopping site, the payment handler, the Delivery Handler of goods or services, and the provider of customer support are performed by different parties or by one party.

The developers of OTP seek to provide a virtual capability that safely replicates the real world, the paper based, traditional, understood, accepted methods of trading, buying, selling, value exchanging that has existed for many hundreds of years. The negotiation of who will be the parties to the trade, how it will be conducted, the presentment of an offer, the method of payment, the provision of a payment receipt, the delivery of goods and the receipt of goods. These are events that are taken for granted in the course of real world trade. OTP has been produced to provide the same for the virtual world, and to prepare and provide for the introduction of new models of trading made possible by the expanding presence of the virtual world.

The other fundamental ideal of the OTP effort is to produce a definition of these trading events in such a way that no matter where produced, two unfamiliar parties using electronic commerce capabilities to buy and sell that conform to the OTP specifications will be able to complete the business safely and successfully.

In summary, OTP supports:

- Familiar trading models
- New trading models
- Global interoperability

The remainder of this section provides background to why OTP was developed. The specification itself starts in the next chapter.

0.1 Commerce on the Internet – a Different Model

The growth of the Internet and the advent of electronic commerce are bringing about enormous changes around the world in society, politics and government, and in business. The ways in which trading partners communicate, conduct commerce, are governed have been enriched and changed forever.

One of the very fundamental changes about which OTP is concerned is taking place in the way consumers and merchants trade. Characteristics of trading that have changed markedly include:

- **Presence**: Face-to-face transactions become the exception, not the rule. Already with the rise of mail order and telephone order placement this change has been felt in western commerce. Electronic commerce over the Internet will further expand the scope and volume of transactions conducted without ever seeing the people who are a part of the enterprise with whom one does business.
- **Authentication**: An important part of personal presence is the ability of the parties to use familiar objects and dialogue to confirm they are who they claim to be. The seller displays

one or several well known financial logos that declaim his ability to accept widely used credit and debit instruments in the payment part of a purchase. The buyer brings government or financial institution identification that assures the seller she will be paid. People use intangibles such as personal appearance and conduct, location of the store, apparent quality and familiarity with brands of merchandise, and a good clear look in the eye to reinforce formal means of authentication.

- **Payment Instruments**: Despite the enormous size of bank card financial payments associations and their members, most of the world's trade still takes place using the coin of the realm or barter. The present infrastructure of the payments business cannot economically support low value transactions and could not survive under the consequent volumes of transactions if it did accept low value transactions.
- **Transaction Values**: New meaning for low value transactions arises in the Internet where sellers may wish to offer for example, pages of information for fractions of currency that do not exist in the real world.
- **Delivery**: New modes of delivery must be accommodated such as direct electronic delivery. The means by which receipt is confirmed and the execution of payment change dramatically where the goods or services have extremely low delivery cost but may in fact have very high value. Or, maybe the value is not high, but once delivery occurs the value is irretrievably delivered so payment must be final and non-refundable but delivery nonetheless must still be confirmed before payment. Incremental delivery such as listening or viewing time or playing time are other models that operate somewhat differently in the virtual world.

0.2 Benefits of OTP

Electronic Commerce Software Vendors

Electronic Commerce Software Vendors will be able to develop e-commerce products which are more attractive as they will inter-operate with any other vendors' software. However since OTP focuses on how these solutions communicate, there is still plenty of opportunity for product differentiation.

Payment Brands

OTP provides a standard framework for encapsulating payment protocols. This means that it is easier for payment products to be incorporated into OTP solutions. As a result the payment brands will be more widely distributed and available on a wider variety of platforms.

Merchants

There are several benefits for Merchants:

- they will be able to offer a wider variety of payment brands,
- they can be more certain that the customer will have the software needed to complete the purchase
- through receiving payment and delivery receipts from their customers, they will be able to provide customer care knowing that they are dealing with the individual or organisation with which they originally traded
- new merchants will be able to enter this new (Internet) market-place with new products and services, using the new trading opportunities which OTP presents

Banks and Financial Institutions

There are also several benefits for Banks and Financial Institutions:

- they will be able to provide OTP support for merchants
- they will find new opportunities for OTP related services:
  - providing customer care for merchants
  - fees from processing new payments and deposits
- they have an opportunity to build relationships with new types of merchants

Customers

For Customers there are several benefits:

- they will have a larger selection of merchants with whom they can trade
- there is a more consistent interface when making the purchase
- there are ways in which they can get their problems fixed through the merchant (rather than the bank!)
- there is a record of their transaction which can be used, for example, to feed into accounting systems or, potentially, to present to the tax authorities

0.3 Baseline OTP

This specification is Baseline OTP. It is a Baseline in that it contains ways of doing trades on the Internet which are the most common. The team working on the OTP see an extended versions of this specification being developed as needs demand but at this stage feel a need to develop a limited function but usable specification in order that technology providers can develop pathway-pilot products that will be placed in the market in order to understand the real “market place” demands and requirements for electronic trading or electronic commerce. To proceed otherwise would be presumptuous, time consuming, expensive and foolish.

Accordingly the OTP Baseline specification has been produced for pathway-pilot product development, expecting to transact live trades to prove the interoperability of solutions based on this specification by end '98.

During this period it is anticipated that there will be no changes to the scope of this specification with the only changes made being limited to corrections where problems are found. Software solutions have been developed based on earlier versions of this specification which prove that the basic concepts work.

0.4 Objectives of Document

The objectives of this document are to provide a functional specification of version 0.9.9 of the Open Trading Protocols which can be used to design and implement systems which support electronic trading on the Internet using the Open Trading Protocols.

An overview of OTP is provided the OTP Business Description which explains the Business Requirements for OTP.

0.5 Purpose

The purpose of the document is:

- to allow potential developers of products based on the protocol to start development of software/hardware solutions which use the protocol
- to allow the financial services industry to understand a developing electronic commerce trading protocol that encapsulates (without modification) any of the current or developing payment schemes now being used or considered by their merchant customer base

0.6 Scope of Document

The protocol describes the content, format and sequences of messages that pass among the participants in an electronic trade - consumers, merchants and banks or other financial institutions, and customer care providers. These are required to support the electronic commerce transactions outlined in the objectives above.

The protocol is designed to be applicable to any electronic payment scheme since it targets the complete purchase process where the movement of electronic value from the payer to the payee is only one, but important, step of many that may be involved to complete the trade.

Payment Scheme which OTP could support include MasterCard Credit, Visa Credit, Mondex Cash, Visa Cash, GeldKarte, DigiCash, CyberCoin, Millicent, Proton etc.

Each payment scheme contains some message flows which are specific to that scheme. These scheme-specific parts of the protocol are contained in a set of payment scheme supplements to this specification.

The document does not prescribe the software and processes that will need to be implemented by each participant. It does describe the framework necessary for trading to take place.

This document also does not address any legal or regulatory issues surrounding the implementation of the protocol or the information systems which use them.

0.7 Intended Readership

Software and hardware developers; development analysts; business and technical planners; industry analysts; merchants; bank and other payment handlers; owners, custodians, and users of payment protocols.

0.8 History

Version 0.1	20 February 1997	Initial draft for comment
Version 0.2	14 April 1997	Revised draft including changes arising from comments
Version 0.2a	24 April 1997	Same as version 0-2 with typographic corrections
Version 0.3	9 October 1997	Revised draft for comment including revised encoding approach using [XML]
Version 0.4	31 October 1997	Published draft for limited public review by groups working within OTP dev
Version 0.9	12 January 1998	Revisions following limited public review – draft for public comment only.
Version 0.9.1	20 May 1998	Revisions following public review - internal OTP Consortium review.
Version 0.9.9	17 August 1998	Draft published for submission to IETF for information.

1. Introduction

The Open Trading Protocols (OTP) define a number of different types of OTP Transactions:

- **Purchase**. This supports a purchase involving an offer, a payment and optionally a delivery
- **Refund**. This supports the refund of a payment as a result of, typically, an earlier purchase
- **Value Exchange**. This involves two payments which result in the exchange of value from one combination of currency and payment method to another
- **Authentication**. This supports the remote authentication of a Consumer by another Trading Role using a variety of authentication methods, and the provision of an Organisation Component about a Consumer to another Trading Role for use in, for example the creation of an offer
- **Withdrawal**. This supports the withdrawal of electronic cash from a financial institution
- **Deposit**. This supports the deposit of electronic cash at a financial institution
- **Payment Instrument Customer Care**. This supports the provision of Payment Brand or Payment Method specific customer care of a Payment Instrument
- **Inquiry** This supports inquiries on the status of an OTP transaction which is either in progress or is complete
- **Ping** This supports a simple query which enables one OTP aware application to determine whether another OTP application running elsewhere is working or not.

These **OTP Transactions** are "Baseline" transactions since they have been identified as a minimum useful set of transactions. Later versions of OTP may include additional types of transactions.

Each of the OTP Transactions above involve:

- a number organisations playing a **Trading Role**, and
- a set of **Trading Exchanges**. Each Trading Exchange involves the exchange of data, between Trading Roles, in the form of a set of **Trading Components**.

Trading Roles, Trading Exchanges and Trading Components are described below.

1.1 Trading Roles

The **Trading Roles** identify the different parts which organisations can take in a trade. The five Trading Roles used within OTP are illustrated in the diagram below.

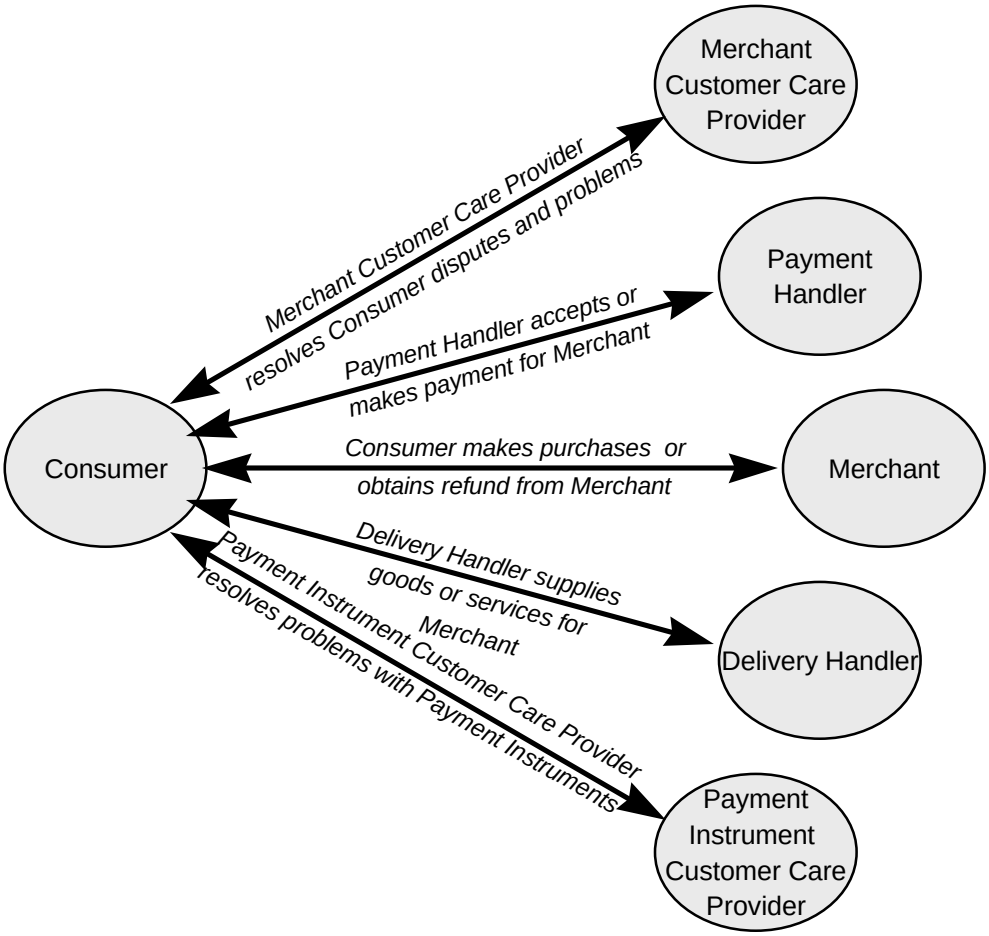


Figure 1 OTP Trading Roles

The roles are:

- **Consumer**. The person or organisation which is to receive and pay for the goods or services
- **Merchant**. The person or organisation from whom the purchase is being made and who is legally responsible for providing the goods or services and receives the benefit of the payment made
- **Payment Handler**. The entity that physically receives the payment from the Consumer on behalf of the Merchant
- **Delivery Handler**. The entity that physically delivers the goods or services to the Consumer on behalf of the Merchant.
- **Merchant Customer Care Provider**. The entity that is involved with customer dispute negotiation and resolution on behalf of the Merchant

- **Payment Instrument Customer Care Provider**. The entity that resolves problems with a particular Payment Instrument

Roles may be carried out by the same organisation or different organisations. For example:

- in the simplest case one physical organisation (e.g. a merchant) could handle the purchase, accept the payment, deliver the goods and provide merchant customer care
- at the other extreme, a merchant could handle the purchase but instruct the consumer to pay a bank or financial institution, request that delivery be made by an overnight courier firm and to contact an organisation which provides 24x7 service if problems arise.

Note that in this specification, unless stated to the contrary, when the words Consumer, Merchant, Payment Handler, **Delivery Handler** or Customer Care Provider are used, they refer to the Trading Role rather than an actual organisation.

An individual organisation may take multiple roles. For example a company which is selling goods and services on the Internet could take the role of Merchant when selling goods or services and the role of Consumer when the company is buying goods or services itself.

As roles occur in different places there is a need for the organisations involved in the trade to exchange data, i.e. to carry out **Trading Exchanges**, so that the trade can be completed.

1.2 Trading Exchanges

The Open Trading Protocols identify four Trading Exchanges which involve the exchange of data between the Trading Roles. The Trading Exchanges are:

- **Offer**. The Offer Exchange results in the Merchant providing the Consumer with the reason why the trade is taking place. It is called an Offer since the Consumer must accept the Offer if a trade is to continue
- **Payment**. The Payment Exchange results in a payment of some kind between the Consumer and the Payment Handler. This may occur in either direction
- **Delivery**. The Delivery Exchange transmits either the on-line goods, or delivery information about physical goods from the **Delivery Handler** to the Consumer, and
- **Authentication**. The Authentication Exchange can be used by any Trading Role to authenticate another Trading Role to check that they are who they appear to be.

OTP Transactions are composed of various combinations of these Trading Exchanges. For example, an OTP **Purchase** transaction includes **Offer**, **Payment**, and **Delivery** Trading Exchanges. As another example, an OTP **Value Exchange** transaction is composed of an **Offer** Trading Exchange and two **Payment** Trading Exchanges.

Trading Exchanges consist of Trading Components that are transmitted between the various Trading Roles. Where possible, the number of round-trip delays in an OTP Transaction is minimised by packing the Components from several Trading Exchanges into combination OTP Messages. For example, the OTP *Purchase* transaction combines a Delivery Organisation Component with an Offer Response Component in order to avoid an extra Consumer request and response.

Each of the OTP Trading Exchanges is described in more detail below. For clarity of description, these describe the Trading Exchanges as though they were standalone operations. For performance reasons, the Trading Exchanges are intermingled in the actual OTP Transaction definitions.

1.2.1 Offer Exchange

The goal of the Offer Exchange is for the Merchant to provide the Consumer with information about the trade so that the Consumer can decide whether to continue with the trade. This is illustrated in the figure below.

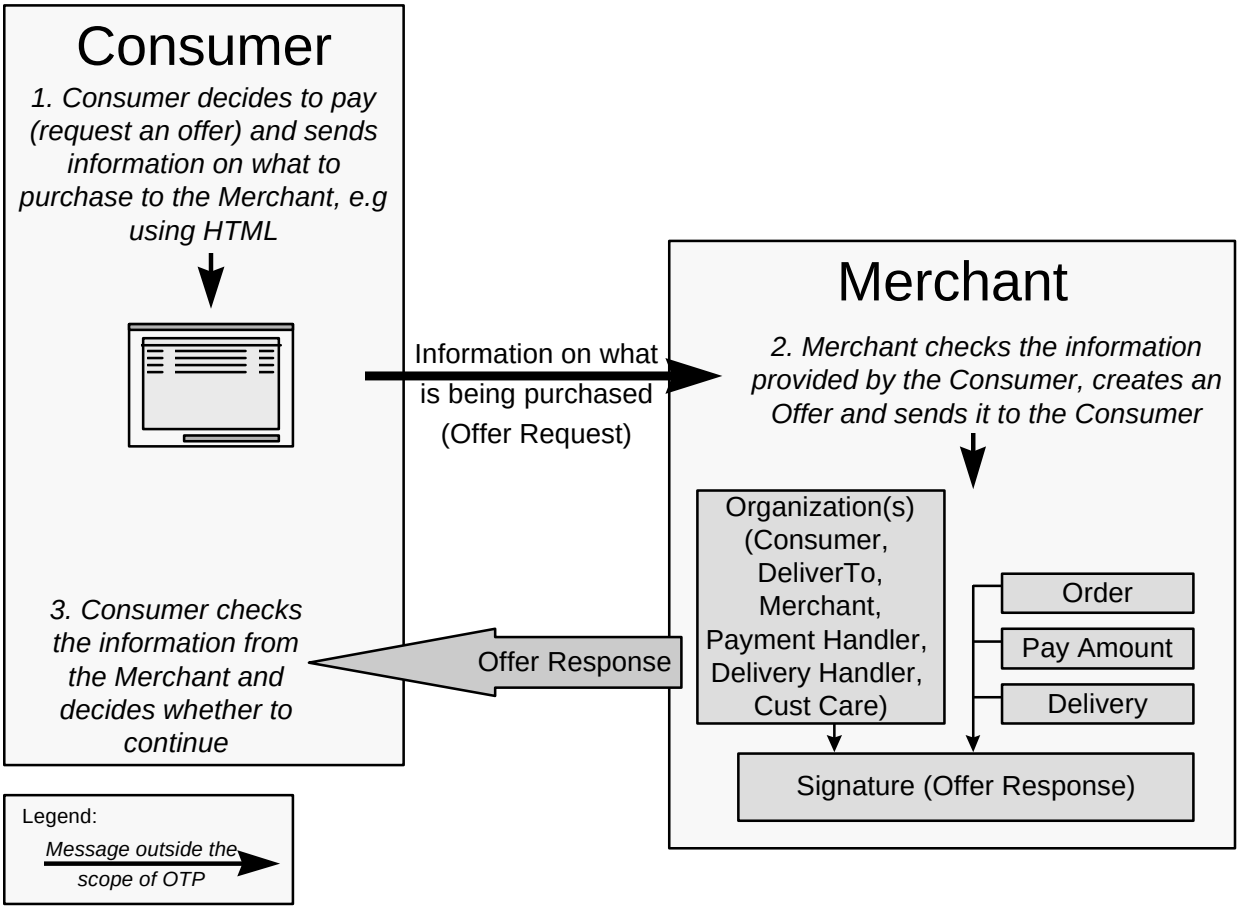


Figure 2 Offer Exchange

An Offer Exchange uses the following Trading Components that are passed between the Consumer and the Merchant:

- the Organisation Component contains information which describes the organisations which are taking a role in the trade:
  - the consumer provides information, about who the consumer is and, if goods or services are being delivered, where the goods or services are to be delivered to
  - the merchant augments this information by providing information about the merchant, the Payment Handler, the customer care provider and, if goods or services are being delivered, the Delivery Handler

- the Order Component contains descriptions of the goods or services which will result from the trade if the consumer agrees to the offer. This information is sent by the Merchant to the consumer who should verify it
- the Payment Component generated by the Merchant, contains details of how much to pay, the currency and the payment direction, for example the consumer could be asking for a refund. Note that there may be more than one payment in a trade
- the Delivery Component, also generated by the Merchant, is used if goods or services are being delivered. This contains information about how delivery will occur, for example by post or using e-mail
- the "Offer Response" Signature Component, if present, digitally signs all of the above components to ensure their integrity.

The exact content of the information provided by the Merchant to the Consumer will vary depending on the type of OTP Transaction. For example:

- low value purchases may not need a signature
- the amount to be paid may vary depending on the payment brand and payment protocol used
- some offers may not involve the delivery of any goods
- a value exchange will involve two payments
- a merchant may not offer customer care.

Information provided by the consumer to the merchant could be provided using a variety of methods, for example, it could be provided:

- using [HTML] pages as part of the "shopping experience" of the consumer.
- using the Open Profiling Standard [OPS] which has recently been proposed,
- in the form of Organisation and Order Components in a later version of OTP.

1.2.2 Payment Exchange

The goal of the Payment Exchange is for a payment to be made from the Consumer to a Payment Handler or vice versa using a payment brand and payment protocol selected by the Consumer. A secondary goal is to optionally provide the Consumer with a digitally signed Payment Receipt which can be used to link the payment to the reason for the payment as described in the Offer Exchange.

Payment Exchanges can work in a variety of ways. The most general case where the trade is dependent on the payment brand and protocol used is illustrated in the diagram below. Simpler payment exchanges are possible.

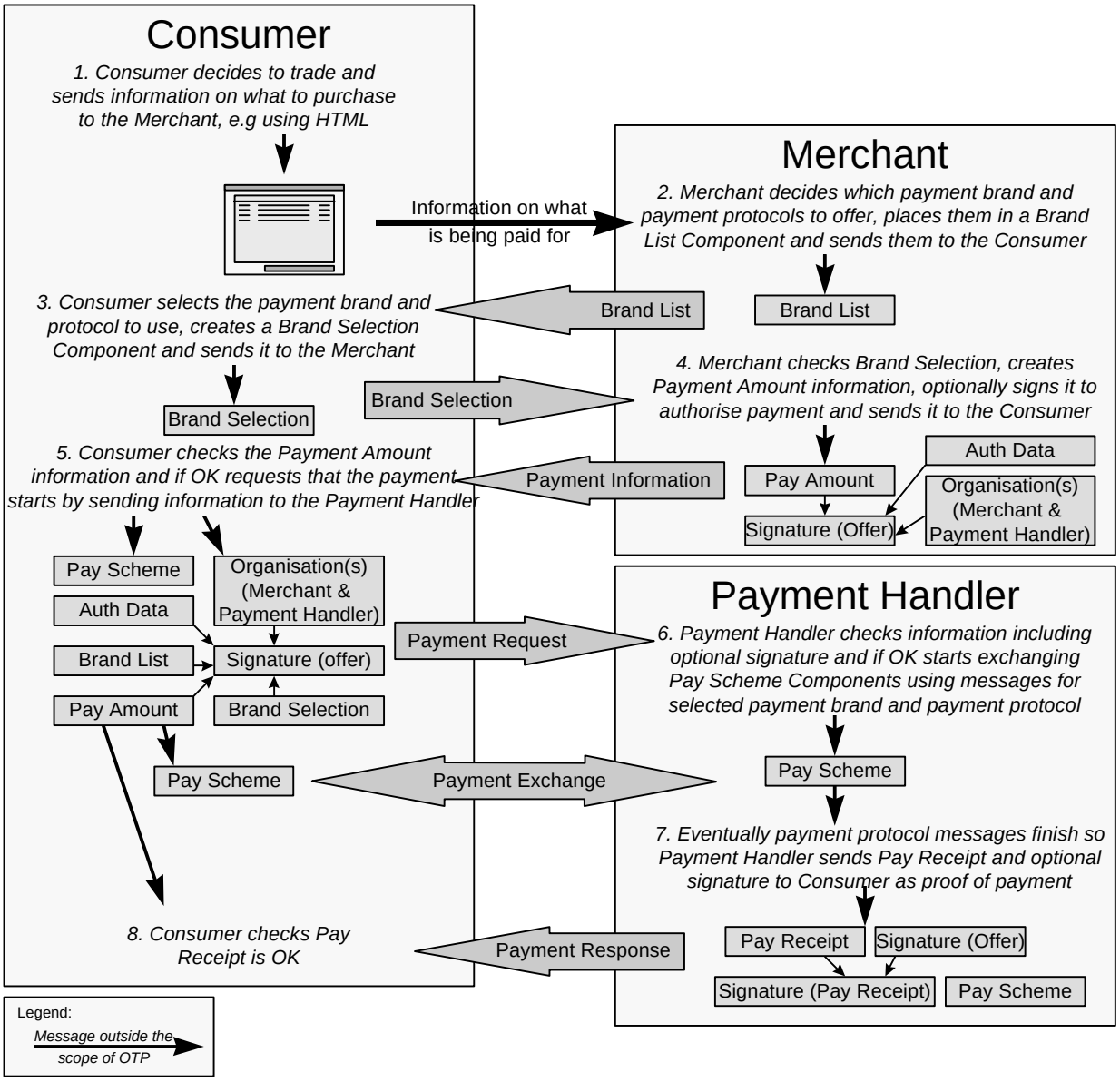


Figure 3 Payment Exchange

A Payment Exchange uses the following Trading Components that are passed between the Consumer, the Merchant and the Payment Handler:

- The Brand List Component contains a list of payment brands (for example, MasterCard, Visa, Mondex, GeldKarte) and payment protocols (for example SET Version 1.0, Secure Channel Credit Debit (SCCD - the name used for a credit or debit card payment where unauthorised access to account information is prevented through use of secure channel transport mechanisms such as SSL). The Merchant sends the Brand List to the Consumer. The consumer compares the payment brands and protocols on offer with those that the Consumer supports and makes a selection.
- The Brand Selection Component contains the Consumer's selection. Payment brand, protocol and possibly protocol-specific information is sent back to the Merchant. This information may be used to change information in the Offer Exchange. For example, a merchant could choose to offer a discount to encourage the use of a store card.
- The Organisation Components are generated by the Merchant. They contain details of the Merchant and Payment Handler Roles:
  - the Merchant role is required so that the Payment Handler can identify which Merchant initiated the payment. Typically, the result of the Payment Handler accepting (or making) a payment on behalf of the Merchant will be a credit or debit transaction to the Merchant's account held by the Payment Handler. These transactions are outside the scope of OTP
  - the Payment Handler role is required so that the Payment Handler can check that it is the correct Payment Handler to be used for the payment
- The optional Authentication Data Component contains challenge data which is used by the payment protocol to authenticate the consumer. Authentication may not always occur
- The Payment Component contains details of how much to pay, the currency and the payment direction, and identifies the Authentication Data Component to use.
- The "Offer Response" Signature Component, if present, digitally signs all of the above components to ensure their integrity. Note that the Brand List and Brand Selection Components are not signed until the payment information is created (step 3 in the diagram)
- The Payment Scheme Component contains messages from the payment protocol used in the Trade. For example they could be SET messages, Mondex messages, GeldKarte Messages or one of the other payment methods supported by OTP. The content of the Payment Scheme Component is defined in the supplements that describe how OTP works with various payment protocols.
- The Payment Receipt Component contains a record of the payment. The content depends upon the payment protocol used.
- The "Payment Receipt" Signature Component provides proof of payment by digitally signing both the Payment Receipt Component and the Offer Signature. The signature on the offer digitally signs the Order, Organisation and Delivery Components contained in the Offer. This signature effectively binds the payment to the offer.

The example of a Payment Exchange above is the most general case. Simpler cases are also possible. For example, if the amount paid is not dependent on the payment brand and protocol selected then the payment information generated by step 3 can be sent to the Consumer at the same time as the Brand List Component generated by step 1. These and other variations are described in the Baseline Purchase OTP Transaction (see section 7.3).

1.2.3 Delivery Exchange

The goal of the Delivery Exchange is to cause purchased goods to be delivered to the consumer either online or via physical delivery. A second goal is to provide a "delivery note" to the consumer, providing details about the delivery, such as shipping tracking number. A future goal is to have a signed delivery that can be used for customer care in the case of problems with physical delivery. This is illustrated in the diagram below.

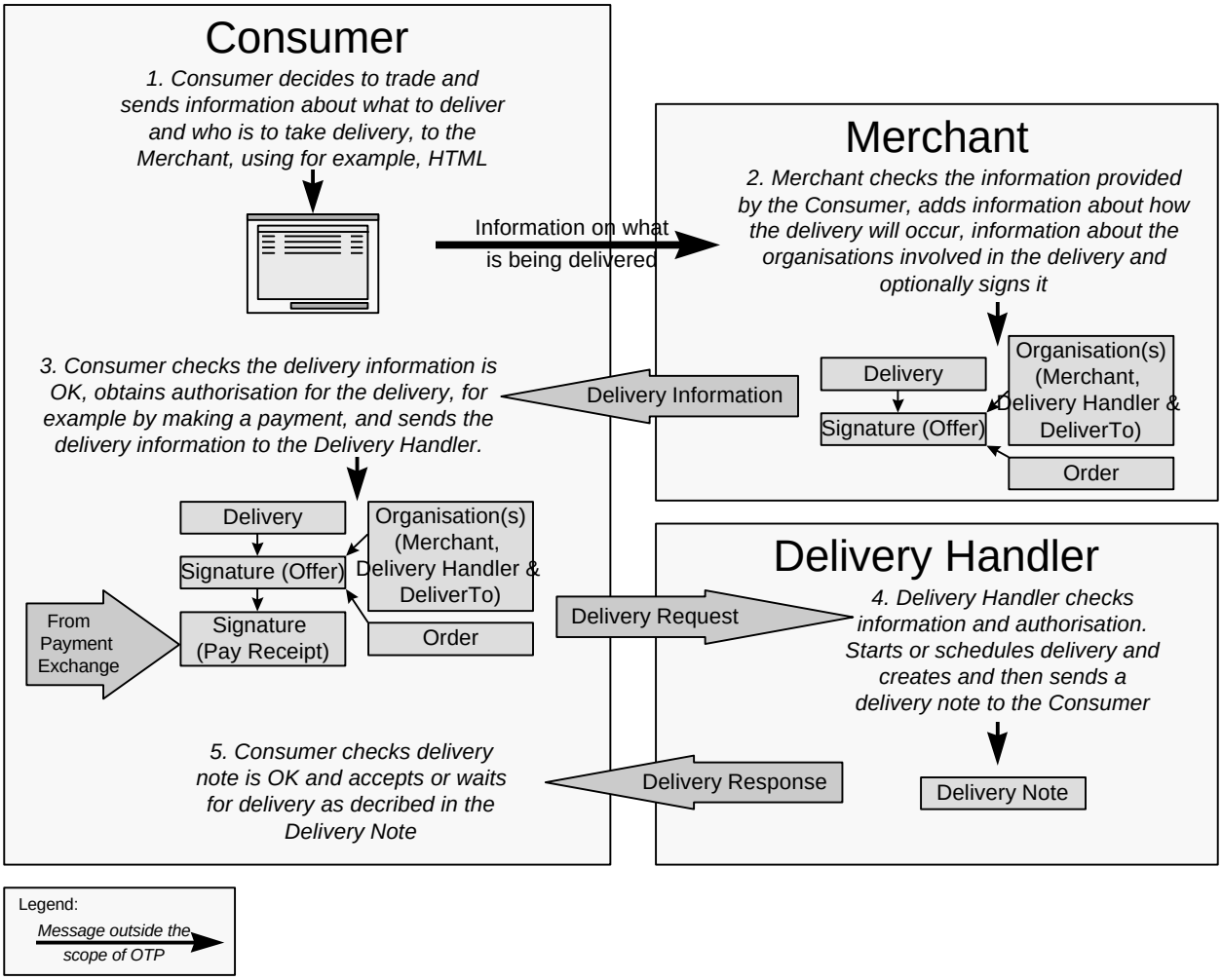


Figure 4 Delivery Exchange

A Delivery Exchange uses the following Trading Components that are passed between the Consumer, the Merchant and the Delivery Handler:

- The Organisation Component(s) contain details of the Deliver To, Delivery Handler and Merchant Roles:
  - the Deliver To role indicates where the goods or services are to be delivered to
  - the Delivery Handler role is required so that the Delivery Handler can check that she is the correct Delivery Handler to do the delivery
  - the Merchant role is required so that the Delivery Handler can identify which Merchant initiated the delivery

- The Order Component, contains information about the goods or services to be delivered
- The Delivery Component contains information about how delivery will occur, for example by post or using e-mail.
- The "Offer Response" Signature Component, if present, digitally signs all of the above components to ensure their integrity.
- The " Payment Receipt" Signature Component provides proof of payment by digitally signing the Payment Receipt Component and the Offer Signature. This is used by the Delivery Handler to check that delivery is authorised
- The Delivery Note Component contains customer care information related to a physical delivery, or alternatively the actual "electronic goods". The Consumer's software does not interpret information about a physical delivery but should have the ability to display the information, both at the time of the delivery and later if the Consumer selects the Trade to which this delivery relates from a transaction list.

1.2.4 Authentication Exchange

The goal of the Authentication Exchange is to allow one organisation, for example a financial institution, to be able to check that another organisation, for example a consumer, is who they appear to be. It uses a "challenge-response" mechanism. This is illustrated in the diagram below.

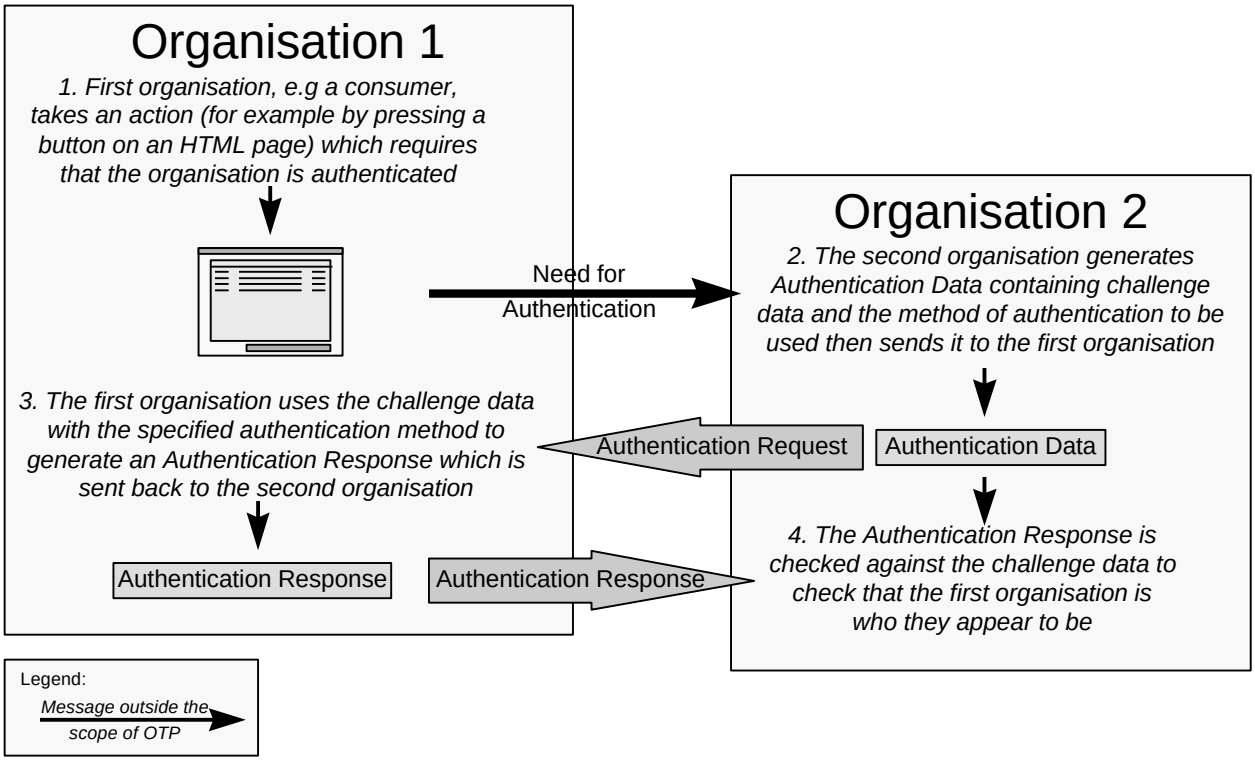


Figure 5 Authentication Exchange

An Authentication Exchange uses the following Trading Components that are passed between the two organisations:

- the Authentication Data Component which contains the challenge data to be used in the "challenge-response" mechanism and indicates the authentication method to be used. It is sent by one organisation to the other.
- the Authentication Response Component which contains the challenge response generated by the recipient of the Authentication Data Component. It is sent back to the first organisation for verification.

1.3 Scope of Baseline OTP

This specification describes the OTP Transactions which make up Baseline OTP. As described in the preface, OTP will evolve over time. This section defines the initial conformance criteria for implementations that claim to “support OTP.”

The main determinant on the scope of an OTP implementation is the roles which the solution is designed to support. The roles within OTP are described in more detail in section 1.1 Trading Roles. To summarise the roles are: Merchant, Consumer, Payment Handler, Delivery Handler and Customer Care Provider.

Payment Handlers who can be of three types:

- those who accept a payment as part of a purchase or make a payment as part of a refund,
- those who accept value as part of a deposit transaction, or
- those that issue value a withdrawal transaction

The following table defines, for each role, the OTP Transactions and Trading Blocks which must be supported for that role.

	Merchants						
	Store	**ECash Value Issuer**	**ECash Value Acquirer**	**Consumer**	**Payment Handler**	**Delivery Handler**	**Pay Inst. Cuts Care**
TRANSACTIONS							
Purchase	Must			Must			
Refund	Must			b) Depends			
Authentication	May	Must	May	b) Depends			
Value Exchange	May			Must			
Withdrawal		Must		b) Depends			
Deposit			Must	b) Depends			
Inquiry	Must	Must	Must	Must	Must	Must	Must
Ping	Must	Must	Must	Must	Must	Must	Must
Pay Inst. Cust. Care				b) Depends			Must

	Merchants						
	Store	**ECash Value Issuer**	**ECash Value Acquirer**	**Consumer**	**Payment Handler**	**Delivery Handler**	**Pay Inst. Cuts Care**
TRADING BLOCKS							
TPO	Must	Must	Must	Must			
TPO Selection	Must	Must	Must	Must			
Auth-Request	a) Depends		a) Depends	a) Depends			
Auth-Reply	a) Depends		a) Depends	a) Depends			
Offer Response	Must	Must	Must	Must			
Payment Request				Must	Must		
Payment Exchange				Must	Must		
Payment Response				Must	Must		
Delivery Request				Must		Must	
Delivery Response				Must		Must	
Pay Inst. Cust Care Req.				b) Depends			Must
Pay Inst. Cust Care Resp				b) Depends			Must
Inquiry Request	Must	Must	Must	Must	Must	Must	Must
Inquiry Response	Must	Must	Must	Must	Must	Must	Must
Ping Request	Must	Must	Must	Must	Must	Must	Must
Ping Response	Must	Must	Must	Must	Must	Must	Must
Signature	Must	Must	Must	Limited	Must	Must	b) Depends
Error	Must	Must	Must	Must	Must	Must	Must

In the above table:

- “Must” means that a Trading Role must support the Transaction or Trading Block.
- “May” means that an implementation may support the Transaction or Trading Block at the option of the developer.
- “Depends” means implementation of the Transaction or Trading Block depends on one of the following conditions:
  a) if Baseline Authentication OTP Transaction is supported;
  b) if required by a Payment Method as defined in its OTP Supplement document.
- "Limited" means the Trading Block must be understood and its content manipulated but not in every respect. Specifically, on the Signature Block, Consumers do not have to be able to validate digital signatures.

An OTP solution **must** support all the OTP Transactions and Trading Blocks required by **at least one role** (column) as described in the above table for that solution to be described as "supporting OTP".

2. Protocol Structure

The previous section provided an introduction which explained:

- **Trading Roles** which are the different roles which organisations can take in a trade: Consumer, Merchant, Payment Handler, Delivery Handler and Merchant and Payment Instrument Customer Care Provider, and
- **Trading Exchanges** where each Trading Exchange involves the exchange of data, between Trading Roles, in the form of a set of **Trading Components**.

This section describes:

- how Trading Components are constructed into **Trading Blocks** and the **OTP Messages** which are physically sent in the form of [XML] documents between the different Trading Roles,
- how OTP Messages are exchanged between Trading Roles to create an **OTP Transaction**
- the XML definitions of an **OTP Message** including a **Transaction Reference Block** - an XML element which identifies an OTP Transaction and the OTP Message within it
- the definitions of the XML **ID Attributes** which are used to identify OTP Messages, Trading Blocks and Trading Components and how these are referred to using **Element References** from other XML elements such as
- OTP **Signature Components** which use digital signature techniques to preserve the integrity of OTP Messages and provide the trust relationships required by OTP
- how extra XML Elements and new user defined values for existing OTP codes can be used when **Extending OTP**, and finally

2.1 Overview

2.1.1 OTP Message Structure

The structure of an OTP Message and its relationship with Trading Blocks and Trading Components is illustrated in the diagram below.

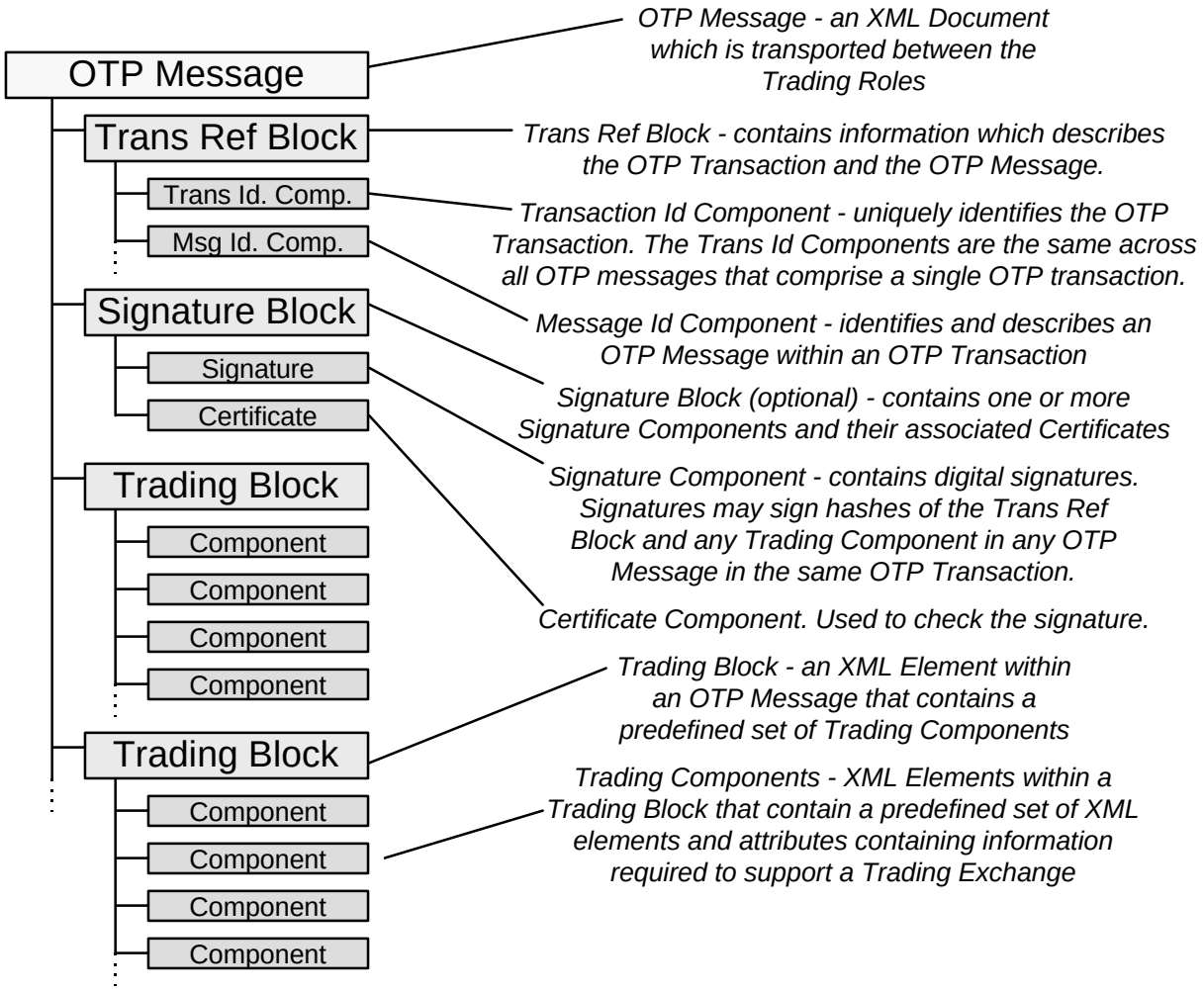


Figure 6 OTP Message Structure

The diagram also introduces the concept of a **Transaction Reference Block**. This block contains, amongst other things, a globally unique identifier for the OTP Transaction. Also each block and component is given an ID Attribute (see section 2.4) which is unique within an OTP Transaction. Therefore the combination of the ID attribute and the globally unique identifier in the Transaction Reference Block is sufficient to uniquely identify any Trading Block or Trading Component.

2.1.2 OTP Transactions

A predefined set of OTP Messages exchanged between the Trading Roles constitute an **OTP Transaction**. This is illustrated in the diagram below.

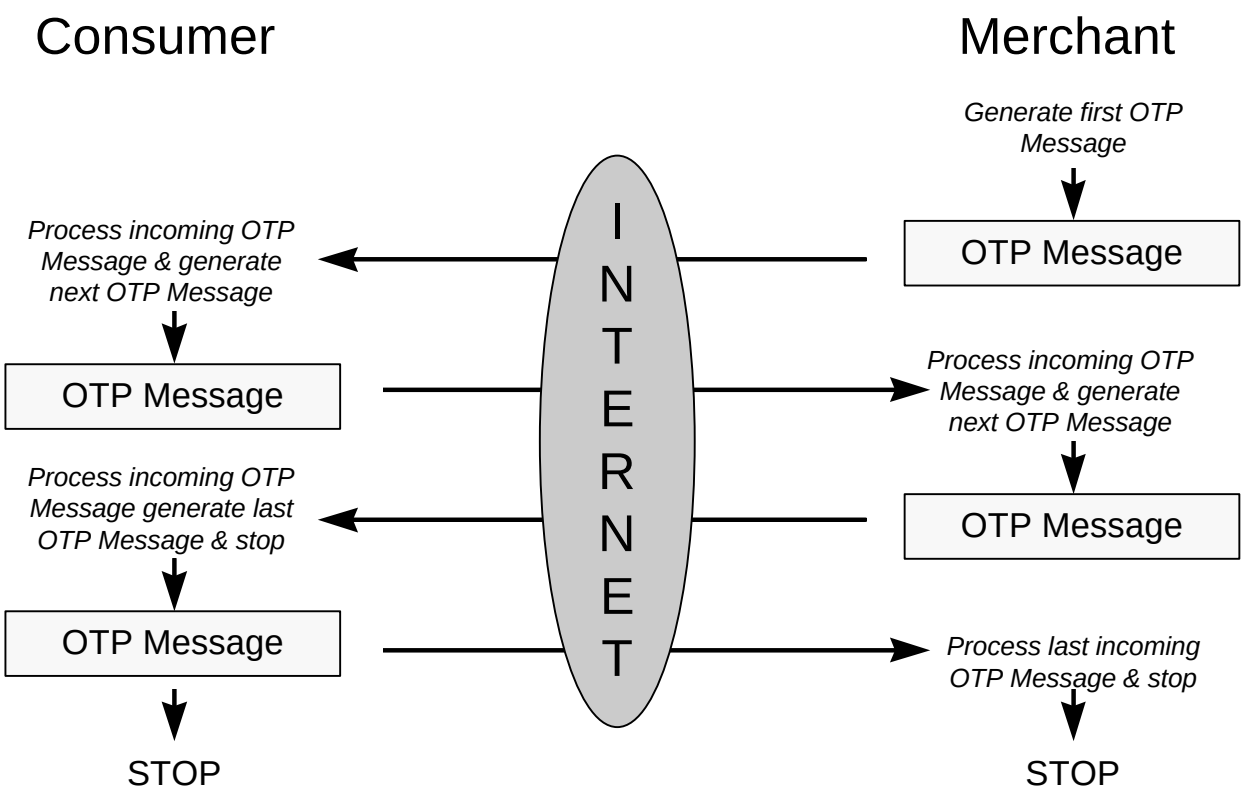


Figure 7 An OTP Transaction

In the above diagram the Internet is shown as the transport mechanism. This is not necessarily the case. OTP Messages can be transported using a variety of transport mechanisms.

The OTP Transactions (see section 7) in this version of OTP are specifically:

- **Purchase**. This supports a purchase involving an offer, a payment and optionally a delivery
- **Refund**. This supports the refund of a payment as a result of, typically, an earlier purchase
- **Value Exchange**. This involves two payments which result in the exchange of value from one combination of currency and payment method to another
- **Authentication**. This supports the remote authentication of a Consumer by another Trading Role using a variety of authentication methods, and the provision of an Organisation Component about a Consumer to another Trading Role for use in, for example the creation of an offer
- **Withdrawal**. This supports the withdrawal of electronic cash from a financial institution
- **Deposit**. This supports the deposit of electronic cash at a financial institution

- **Payment Instrument Customer Care**. This supports the provision of Payment Brand or Payment Method specific customer care of a Payment Instrument
- **Inquiry** This supports inquiries on the status of an OTP transaction which is either in progress or is complete
- **Ping** This supports a simple query which enables one OTP aware application to determine whether another OTP application running elsewhere is working or not.

2.2 OTP Message

As described earlier, OTP Messages are [XML] documents which are physically sent between the different organisations that are taking part in a trade.

The XML definition of an OTP Message is as follows.

```
<!ELEMENT OtpMessage (TransRefBlk, SigBlk?, ErrorBlk?,
(        AuthReqBlk |
         AuthRespBlk |
         DeliveryReqBlk |
         DeliveryRespBlk |
         InquiryReqBlk |
         InquiryRespBlk |
         OfferRespBlk |
         PayExchBlk |
         PayReqBlk |
         PayInstCCExchBlk |
         PayInstCCReqBlk |
         PayInstCCRespBlk
         PayRespBlk |
         PingReqBlk |
         PingRespBlk |
         TpoBlk |
         TpoSelectionBlk |
         )*
) >
```

Content:

`TransRefBlk`	This contains information which describes an OTP Message within an OTP Transaction (see section 2.3 immediately below)

`AuthReqBlk,`
`AuthRespBlk,`
`DeliveryReqBlk,`
`DeliveryRespBlk`
`ErrorBlk`
`InquiryReqBlk,`
`InquiryRespBlk,`
`OfferRespBlk,`
`PayExchBlk,`
`PayReqBlk,`
`PayInstCCExchBlk,`
`PayInstCCReqBlk,`
`PayInstCCRespBlk`
`PayRespBlk,`
`PingReqBlk,`
	These are the Trading Blocks.
	The Trading Blocks present within an OTP Message, and the content of a Trading Block itself is dependent on the type of OTP Transaction being carried out - see the definition of each transaction in section 7 Open Trading Protocol Transactions.
	Full definitions of each Trading Block are described in section 6.

```
 PingRespBlk,
 SigBlk,
 TpoBlk,
 TpoSelectionBlk
```

2.2.1 XML Document Prolog

The OTP Message is the root element of the XML document. It therefore needs to be preceded by an appropriate XML Document Prolog. For example:

```
<?XML Version='1.0'?>
<!DOCTYPE OtpMessage >
<OtpMessage>
   ...
</OtpMessage>
```

2.3 Transaction Reference Block

A Transaction Reference Block contains information which identifies the OTP Transaction and OTP Message. The Transaction Reference Block contains:

- a Transaction Id Component which globally uniquely identifies the OTP Transaction. The Transaction Id Components are the same across all OTP messages that comprise a single OTP transaction,
- a Message Id Component which provides control information about the OTP Message as well as uniquely identifying the OTP Message within an OTP Transaction, and
- zero or more Related To Components which link this OTP Transaction to either other OTP Transactions or other events using the identifiers of those events.

The definition of a Transaction Reference Block is as follows:

```
<!ELEMENT TransRefBlk (TransId, MsgId, RelatedTo*) >
<!ATTLIST TransRefBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Transaction Reference Block within the OTP Transaction (see section 2.4 ID Attributes).

Content:

`TransId`	See 2.3.1 Transaction Id Component immediately below.
`MsgId`	See 2.3.2 Message Id Component immediately below.
`RelatedTo`	See 2.3.3 Related To Component immediately below.

2.3.1 Transaction Id Component

This contains information which globally uniquely identifies the OTP Transaction. Its definition is as follows:

```
<!ELEMENT TransId EMPTY>
<!ATTLIST TransId
  ID  ID              #REQUIRED
  Version             NMTOKEN  #FIXED '1.0'
  OtpTransId          NMTOKEN  #REQUIRED
  OtpTransType        CDATA    #REQUIRED >
  TransTimeStamp      CDATA    #REQUIRED >
```

Attributes:

`ID`

An identifier which uniquely identifies the Transaction Id Component within the OTP Transaction.

`Version`

This identifies the version of OTP, and therefore the structure of the OTP Messages, which the OTP Transaction is using.

`OtpTransId`

Contains data which uniquely identifies the OTP Transaction. It must conform to the rules for Message Ids in [RFC 822].

`OtpTransType`

This is the type of OTP Transaction being carried out. For Baseline OTP it identifies a "standard" OTP Transaction and implies the sequence and content of the OTP Messages exchanged between the Trading Roles. The valid values for Baseline OTP are:

- `BaselineAuthentication`
- `BaselineDeposit`
- `BaselinePurchase`
- `BaselineRefund`
- `BaselineWithdrawal`
- `BaselineValueExchange`
- `BaselineInquiry`
- `BaselinePing`
- `BaselinePayInstrumentCustomerCare`
- `x-ddd:nnn`

A value for `OtpTransType`of `x-ddd:nnn`indicates a user defined transaction type. See section 2.7.3 User Defined Codes.

In later versions of OTP, this list will be extended to support different types of standard OTP Transaction based on market demand. It is also likely to support the type `Dynamic`which indicates that the sequence of steps within the transaction are non-standard.

`TransTimeStamp`

Where the system initiating the OTP Transaction has an internal clock, it is set to the time at which the OTP Transaction started in [UTC] format.

The main purpose of this attribute is to provide an alternative way of identifying a transaction by specifying the time at which it started.

Some systems, for example, hand held devices may not be able to generate a time stamp. In this case this attribute should contain the value "NA" for Not Available.

2.3.2 Message Id Component

The Message Id Component provides control information about the OTP Message as well as uniquely identifying the OTP Message within an OTP Transaction. Its definition is as follows.

```
<!ELEMENT MsgId EMPTY >
<!ATTLIST MsgId
  ID  ID              #REQUIRED
  RespOtpMsg          NMTOKEN  #IMPLIED
  xml:lang            NMTOKEN  #REQUIRED
  SoftwareId          CDATA    #REQUIRED
  TimeStamp           CDATA    #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the OTP Message within the OTP Transaction (see section 2.4 ID Attributes). Note that if an OTP Message is resent then the value of this attribute remains the same.
`RespOtpMsg`	This contains the `ID` attribute of the Message Id Component of the OTP Message to which this OTP Message is a response. In this way all the OTP Messages in an OTP Transaction are unambiguously linked together. This field is required on every OTP Message except the first OTP Message in an OTP Transaction.
`xml:lang`	Defines the language used by attributes or child elements within this component, unless overridden by an `xml:lang` attribute on a child element. See section 2.9 Identifying Languages.
`SoftwareId`	This contains information which identifies the software which generated the OTP Message. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:lang`. It must contain, as a minimum: • the name of the software manufacturer • the name of the software • the version of the software, and • the build of the software
`TimeStamp`	Where the device sending the message has an internal clock, it is set to the time at which the OTP Message was created in [UTC] format.

2.3.3 Related To Component

The Related To Component links OTP Transactions to either other OTP Transactions or other events using the identifiers of those events. Its definition is as follows.

```
<!ELEMENT RelatedTo (PackagedContent) >
<!ATTLIST RelatedTo
  ID  ID              #REQUIRED
  xml:lang            NMTOKEN  #REQUIRED
  RelationshipType    NMTOKEN  #REQUIRED
  Relation            CDATA    #REQUIRED
  RelnKeyWords        NMTOKENS #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Related To Component within the OTP Transaction.
`xml:lang`	Defines the language used by attributes or child elements within this component, unless overridden by an `xml:lang` attribute on a child element. See section 2.9 Identifying Languages.
`RelationshipType`	Defines the type of the relationship. Valid values are: • `OtpTransaction` in which case the Packaged Content Element contains an `OtpTransId` of another OTP Transaction • `Reference` in which case the Packaged Content Element contains the reference of some other, non-OTP document. • `x-ddd:nnn` a user defined code (see section 2.7.3)
`Relation`	The `Relation` attribute contains a phrase in the language defined by `xml:lang` which describes the nature of the relationship between the OTP transaction that contains this component and another OTP Transaction or other event. The exact words to be used are left to the implementer of the OTP software. The purpose of the attribute is to provide the Trading Roles involved in an OTP Transaction with an explanation of the nature of the relationship between the transactions. Care should be taken that the words used to in the `Relation` attribute indicate the "direction" of the relationship correctly. For example: one transaction might be a refund for another earlier transaction. In this case the transaction which is a refund should contain in the `Relation` attribute words such as "refund for" rather than "refund to" or just "refund".
`RelnKeyWords`	This attribute contains keywords which could be used to help identify similar relationships, for example all refunds. It is anticipated that recommended keywords will be developed through examination of actual usage. In this version of the specification there are no specific recommendations and the keywords used are at the discretion of the implementer.

Content:

`PackagedContent`	The Packaged Content (see section 2.8) contains data which identifies the related transaction. Its format varies depending on the value of the `RelationshipType`

2.4 ID Attributes

OTP Messages, Blocks (i.e. Transaction Reference Blocks and Trading Blocks) and Trading Components (including the Transaction Id Component and the Signature Component) are each given an XML "ID" attribute which is used to identify an instance of these XML elements. These identifiers are used so that one element can be referenced by another. All these attributes are given the attribute name `ID`.

The values of each `ID` attribute are unique within an OTP transaction i.e. the set of OTP Messages which have the same globally unique Transaction ID Component. This means that it is possible to use these IDs to refer to and locate the content of any OTP Message, Block or Component from any other OTP Message, Block or Component in the same OTP Transaction using **Element References** (see section 2.5).

This section defines the rules for setting the values for the ID attributes of OTP Messages Blocks and Components.

2.4.1 OTP Message ID Attribute Definition

The `ID` attribute of the Message Id Component of an OTP Message must be unique within an OTP Transaction. It's definition is as follows:

```
OtpMsgId_value  ::= OtpMsgIdPrefix OtpMsgIdSuffix
OtpMsgIdPrefix  ::= NameChar (NameChar)*
OtpMsgIdSuffix  ::= Digit (Digit)*
```

`OtpMsgIdPrefix`

Apart from messages which contain an Inquiry Request Trading Block (see section 6.14), the same prefix is used for all messages sent by the Merchant or Consumer role as follows:

- `"M"` - Merchant
- `"C"` - Consumer

For messages which contain an Inquiry Request Trading Block, the prefix is set to `"I"` for Inquiry.

The prefix for the other roles in a trade is contained within the Organisation Component for the role and are typically set by the Merchant. The following is recommended as a guideline and must not be relied upon:

- `"P"` - First (only) Payment Handler
- `"R"` - Second Payment Handler
- `"D"` - Delivery Handler

As a guideline, prefixes should be limited to one character.

`NameChar` has the same definition as the [XML] definition of `NameChar.`

`OtpMsgIdSuffix`

The suffix consists of one or more digits. The suffix must be unique within a Trading Role within an OTP Transaction. The following is recommended as a guideline and must not be relied upon:

- the first OTP Message sent by a trading role is given the suffix `"1"`
- the second and subsequent OTP Messages sent by the same trading role are incremented by one for each message
- no leading zeroes are included in the suffix

Put more simply the Message Id Component of the first OTP Message sent by a Consumer would have an `ID` attribute of, `"C1"`, the second `"C2"`, the third `"C3"` etc.

`Digit` has the same definition as the [XML] definition of `Digit.`

2.4.2 Block and Component ID Attribute Definitions

The ID Attribute of Blocks and Components must also be unique within an OTP Transaction. Their definition is as follows:

```
BlkOrCompId_value ::= OtpMsgId "." IdSuffix
IdSuffix ::= Digit (Digit)*
```

`OtpMsgId`

The ID attribute of the Message ID Component of the OTP Message where the Block or Component is first used.

In OTP, Trading Components and Trading Blocks are copied from one OTP Message to another. The `ID` attribute does not change when an existing Trading Block or Component is copied to another OTP Message.

`IdSuffix`

The suffix consists of one or more digits. The suffix must be unique within the `ID` attribute of the Message ID Component used to generate the `ID` attribute. The following is recommended as a guideline and must not be relied upon:

- the first Block or Component sent by a trading role is given the suffix `"1"`
- the ID attributes of the second and subsequent Blocks or Components are incremented by one for each new Block or Component added to an OTP Message
- no leading zeroes are included in the suffix

Put more simply, the first new Block or Component added to the second OTP Message sent, for example, by a consumer would have a an `ID` attribute of `"C2.1"`, the second `"C2.2"`, the third `"C2.3"` etc.

`Digit` has the same definition as the [XML] definition of `Digit`.

2.4.3 Example of use of ID Attributes

The diagram below illustrates how `ID` attribute values are used.

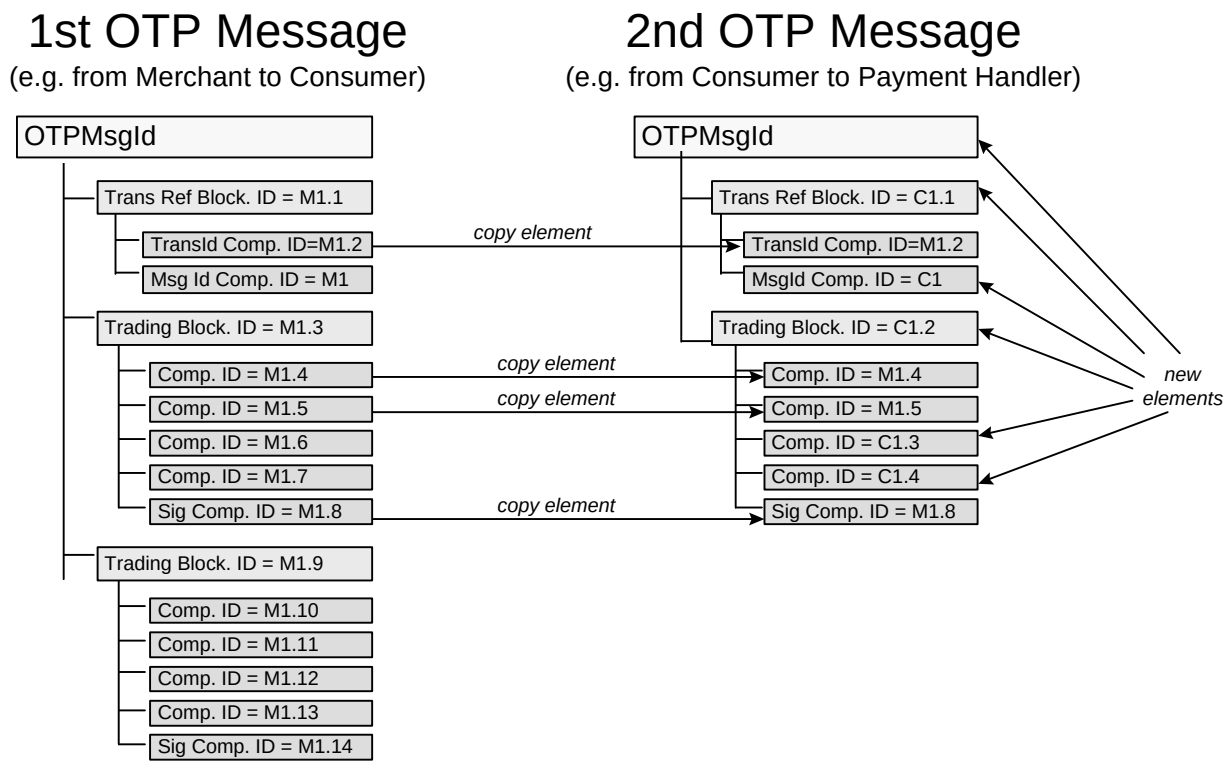


Figure 8 Example use of `ID` attributes

2.5 Element References

A Trading Component or one of its child XML elements, may contain an XML attribute that refers to another Block (i.e. a Transaction Reference Block or a Trading Block) or Trading Component (including a Transaction Id and Signature Component). These **Element References** are used for many purposes, a few examples include:

- identifying an XML element whose hash value is included in a Signature Component,
- referring to the Payment Handler Organisation Component which is used when making a Payment

An Element Reference always contains the value of an `ID` attribute of a Block or Component.

Identifying the OTP Message, Trading Block or Trading Component which is referred to by an Element Reference, involves finding the XML element which:

- belongs to the same OTP Transaction (i.e. the Transaction Id Components of the OTP Messages match), and
- where the value of the ID attribute of the element matches the value of the Element Reference.

[Note] *The term "match" in this specification has the same definition as the [XML] definition of match.*
[Note End]

An example of "matching" an Element Reference is illustrated in the example below.

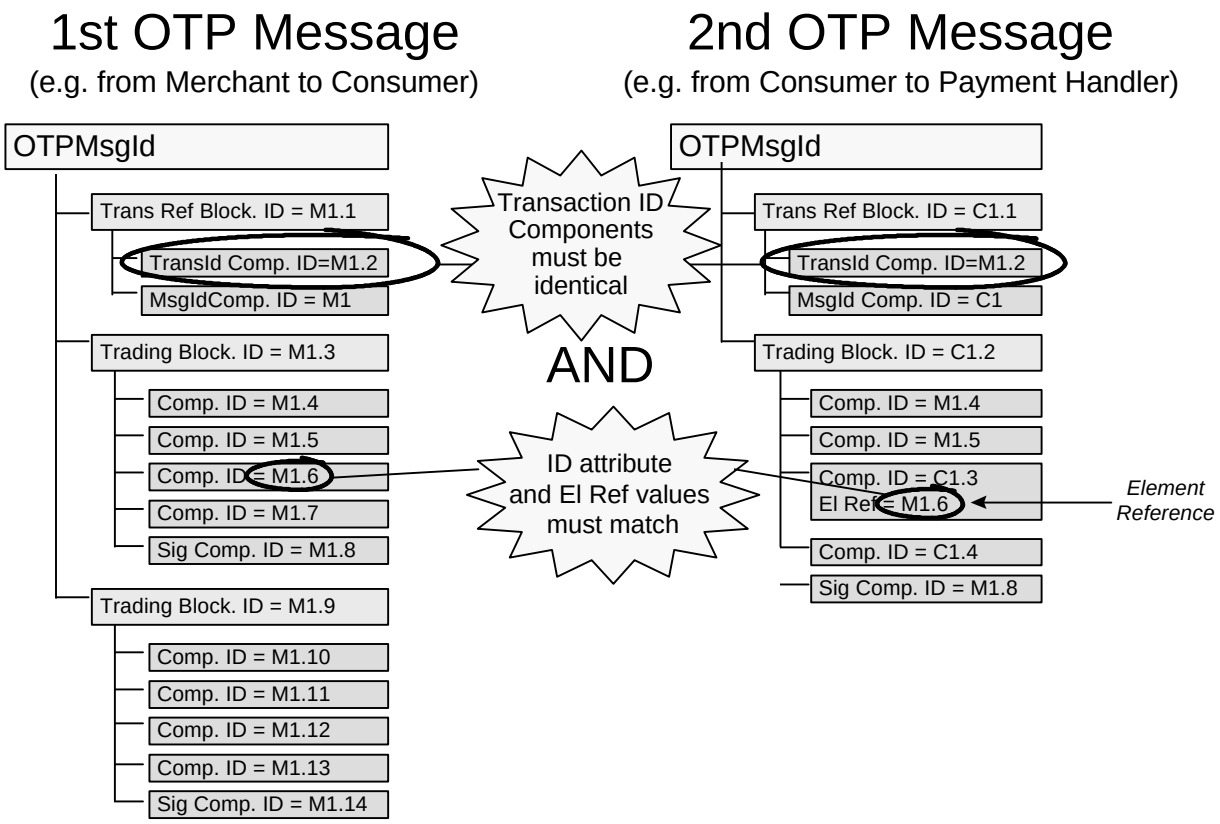


Figure 9 Element References

[Note] *Element Reference attributes are defined as `"NMTOKEN"` rather than `"IDREF"` (see [XML]). This is because an `IDREF` requires that the XML element referred to is in the same XML Document. With OTP this is not necessarily the case.*
[Note End]

2.6 Brands and Brand Selection

One of the key features of OTP is the ability for a merchant to offer a list of Brands from which a consumer may make a selection. This section provides an overview of what is involved and provides guidance on how selection of a brand and associated payment instrument can be carried out by a Consumer. It covers:

- definitions of Payment Instruments and Brands - what are Payment Instruments and Brands in an OTP context. Further categorises Brands as optionally a "Dual Brand" or a "Promotional Brand",
- identification and selection of Promotional Brands - Promotional Brands offer a Consumer some additional benefit, for example loyalty points or a discount. This means that both Consumers and Merchant must be able to correctly identify that a valid Promotional Brand is being used.

Also see the following sections:

- Brand List Component (section 5.6) which contains definitions of the XML elements which contain the list of Brands offered by a Merchant to a Consumer, and
- Brand Selection Component (section 5.7) for details of how a Consumer records the Brand that was selected.

2.6.1 Definition of Payment Instrument

A Payment Instrument is the means by which Consumer pays for goods or services offered by a Merchant. It can be, for example:

- a credit card such as MasterCard or Visa;
- a debit card such as MasterCard's Maestro;
- a smart card based electronic cash payment instrument such as a Mondex Card, a GeldKarte card or a Visa Cash card
- a software based electronic payment account such as a CyberCash or DigiCash account.

All Payment Instruments have a number, typically an account number, by which the Payment Instrument can be identified.

2.6.2 Definition of Brand

A Brand is the mark which identifies a particular type of Payment Instrument. A list of Brands are the payment options which are presented by the Merchant to the Consumer and from which the Consumer makes a selection. Each Brand may have a different Payment Handler. Examples of Brands include:

- payment association and proprietary Brands, for example MasterCard, Visa, American Express, Diners Club, American Express, Mondex, GeldKarte, CyberCash, etc.
- promotional brands (see below). These include:
  - store brands, where the Payment Instrument is issued to a Consumer by a particular Merchant, for example Walmart, Sears, or Marks and Spencer (UK)
  - cobrands, for example American Advantage Visa, where an organisation uses their own brand in conjunction with, typically, a payment association Brand.

2.6.3 Definition of Dual Brand

A Dual Brand means that a single payment instrument may be used as if it were two separate Brands. For example there could be a single Japanese "UC" MasterCard which can be used as either a UC card or a regular MasterCard. The UC card Brand and the MasterCard Brand could each have their own separate Payment Handlers. This means that:

- the merchant treats, for example "UC" and "MasterCard" as two separate Brands when offering a list of Brands to the Consumer,
- the consumer chooses a Brand, for example either "UC" or "MasterCard,
- the consumer OTP aware application determines which Payment Instrument(s) match the chosen Brand, and selects, perhaps with user assistance, the correct Payment Instrument to use.

2.6.4 Definition of Promotional Brand

A Promotional Brand means that, if the Consumer pays with that Brand, then the Consumer will receive some additional benefit which can be received in two ways:

- at the time of purchase. For example if a Consumer pays with a "Walmart MasterCard" at a Walmart web site, then a 5% discount might apply, which means the consumer actually pays less,
- from their Payment Instrument (card) issuer when the payment appears on their statement. For example loyalty points in a frequent flyer scheme could be awarded based on the total payments made with the Payment Instrument since the last statement was issued.

Note that:

- the first example (obtaining the benefit at the time of purchase), requires that:
  - the Consumer is informed of the benefits which arise if that Brand is selected
  - if the Brand is selected, the Merchant changes the relevant OTP Components in the Offer Response to reflect the correct amount to be paid
- the second (obtaining a benefit through the Payment Instrument issuer) does not require that the Offer Response is changed
- each Promotional Brand should be identified as a separate Brand in the list of Brands offered by the Merchant. For example: "Walmart", "Sears", "Marks and Spencer" and "American Advantage Visa", would each be a separate Brand.

2.6.5 Identifying Promotional Brands

There are two problems which need to handled in identifying Promotional Brands:

- how does the Merchant or their Payment Handler positively identify the promotional brand being used at the time of purchase
- how does the Consumer reliably identify the correct promotional brand from the Brand List presented by the Merchant

The following is a description of how this could be achieved.

[Note] *Please note that the approach described here is a model approach that solves the problem. Other equivalent methods may be used.*

[Note End]

2.6.5.1 Merchant/Payment Handler Identification of Promotional Brands

Correct identification that the Consumer is paying using a Promotional Brand is important since a Consumer might fraudulently claim to have a Promotional Brand that offers a reduced payment amount when in reality they do not.

Two approaches seem possible:

- use some feature of the Payment Instrument or the payment method to positively identify the Brand being used. For example, the SET certificate for the Brand could be used, if one is available, or
- use the Payment Instrument (card) number to look up information about the Payment Instrument on a Payment Instrument issuer database to determine if the Payment Instrument is a promotional brand

Note that:

- the first assumes that SET is available.
- the second is only possible if the Merchant, or alternatively the Payment Handler, has access to card issuer information.

OTP does not provide the Merchant with Payment Instrument information (e.g. a card or account number). This is only sent as part of the encapsulated payment protocol to a Payment Handler. This means that:

- the Merchant would have to assume that the Payment Instrument selected was a valid Promotional Brand, or
- the Payment Handler would have to check that the Payment Instrument was for the valid Promotional Brand and fail the payment if it was not.

A Payment Handler checking that a brand is a valid Promotional Brand is most likely if the Payment Handler is also the Card Issuer.

2.6.5.2 Consumer Selection of Promotional Brands

Two ways by which a Consumer can correctly select a Promotional Brand are:

- the Consumer visually matching a logo for the Promotional Brand which has been provided to the Consumer by the Merchant,
- the Consumer's OTP aware application matching a code for the Promotional Brand which the application has registered against a similar code contained in the list of Brands offered by the Merchant.

In the latter case, the code contained in the Consumer wallet must match exactly the code in the list offered by the Merchant otherwise no match will be found. Ways in which the Consumer's OTP Aware Application could obtain such a code include:

- the Consumer types the code in directly. This is error prone and not user friendly, also the consumer needs to be provided with the code. This approach is not recommended,

- using some information contained in the software or other data associated with the Payment Instrument. This could be:
  - a SET certificate for Brands which use this payment method
  - a code provided by the payment software which handles the particular payment method, this could apply to, for example, GeldKarte, Mondex, CyberCash and DigiCash
- the consumer making a initial "manual" link between a Promotional Brand in the list of Brands offered by the Merchant and an individual Payment Instrument, the first time the promotional brand is used. The OTP Aware application would then "remember" the code for the Promotional Brand for use in future purchases

[Note] *It is not the intention of the developers of this specification to develop a prescriptive list of payment brands. It is anticipated that owners of brands will develop distinctive names for Brands which should mean that name clashes are unlikely.*

[Note End]

2.7 Extending OTP

Baseline OTP defines a minimum protocol which systems supporting OTP must be able to accept. As new versions of OTP are developed, additional types of OTP Transactions will be defined. In addition to this, Baseline and future versions of OTP will support user extensions to OTP through two mechanisms:

- extra XML elements, and
- new user-defined values for existing OTP codes.

2.7.1 Extra XML Elements

The XML element and attribute names used within OTP constitute an [XML Namespace]. This allows OTP to support the inclusion of additional XML elements within OTP messages through the use of [XML Namespaces].

[Note] *In drafts of the [XML] specification, the concept of "Namespaces" have been discussed. However they are not present in the XML documentation submitted for approval (see XML draft dated 8 December 1997) although it appears as if they may be included in version 1.1 of XML. It is considered by the authors of this document that OTP would be an ideal example of a Namespace so that other XML elements with potentially the same name can be included unambiguously in XML documents which conform to this specification. If Namespaces, or an equivalent, is not developed for XML as a whole then OTP is likely to propose its own equivalent. The Views of other organisations on this topic are sought.*

[Note End]

Extra XML elements may be included at any level within an OTP message including:

- new Trading Blocks
- new Trading Components
- new XML elements within a Trading Component.

The following rules apply:

- any new XML element must be declared according to the rules for [XML Namespaces]. This means that:
  - the namespace must be declared to the XML parser
  - each element must have a start and end tags which conform to the rules for XML Namespaces
- new XML elements which are either Trading Blocks or Trading Components must contain an ID attributes with an attribute name of `ID`.

In order to make sure that extra XML elements can be processed properly, OTP reserves the use of a special attribute, `Otp:Critical`, which takes the values `True` or `False` and may appear in extra elements added to an OTP message.

The purpose of this attribute is to allow an OTP aware application to determine if the OTP transaction can safely continue. Specifically:

- if an extra XML element has an `"Otp:Critical"`attribute with a value of `"True"` and an OTP aware application does not know how to process the element and its child elements, then the OTP transaction must fail. See section 5.17 Error Component.
- if an extra XML element has an `"Otp:Critical"`attribute with a value of `"False"` then the OTP transaction may continue if the OTP aware application does not know how to process it. In this case:
  - any extra XML elements contained within an XML element defined within the OTP namespace, must be included with that element whenever the OTP XML element is used or copied by OTP
  - the content of the extra element must be ignored except that it must be included when it is hashed as part of the generation of a signature
- if an extra XML element has no `"Otp:Critical"`attribute then it must be treated as if it had an `"Otp:Critical"`attribute with a value of `"True"`
- if an XML element contains an `"Otp:Critical"`attribute, then the value of that attribute is assumed to apply to all the child elements within that element

In order to ensure that documents containing `"Otp:Critical"`are valid, it is declared as part of the DTD for the extra element as:

```
Otp:Critical  (True | False ) #IMPLIED
```

2.7.2 Opaque Embedded Data

If OTP is to be extended using Opaque Embedded Data then a Packaged Content Element (see section 2.8) should be used to encapsulate the data.

2.7.3 User Defined Codes

User defined codes provide a simple way to identify additional values for the codes contained within this specification.

The definition of a user defined code is as follows:

```
user_defined_code ::= ( "x-" | "X-" ) domain_name ":" name
```

`domain_name` A name which identifies the organisation which is creating the user defined

code (see [DNS]). The purpose of this field is to reduce the probability of two organisations creating the same user-defined name

`name` — A name specified by the organisation which owns the `domain_name` which identifies the user defined code within the `domain_name`

User defined codes are identified in this specification as `"x-ddd:nnn"`. The values of User Defined Codes must conform to the rules for the specific code (see explanations of the individual codes).

2.8 Packaged Content Element

The Packaged Content element supports the concept of an embedded data stream, transformed to both protect it against misinterpretation by transporting systems and to ensure XML compatibility. Examples of its use in OTP include:

- to encapsulate payment scheme messages, such as SET messages,
- to encapsulate a description of an order.

In general it is used to encapsulate any data stream.

This data stream has two standardised attributes that allow for decoding and interpretation of the contents. Its definition is as follows.

```
<!ELEMENT PackagedContent (#PCDATA)>
<!ATTLIST PackagedContent
  Content            NMTOKEN  "PCDATA"
  Transform (NONE|BASE64)     "NONE" >
```

Attributes:

`Content` — This identifies what type of data is contained within the Content of the Packaged Content Element. The valid values for the `Content` attribute are as follows:

- `PCDATA` The content of the Packaged Content Element can be treated as PCDATA with no further processing.
- `MIME`. The content of the Packaged Content Element is a complete MIME item. Processing should include looking for MIME headers inside the Packaged Content Element.
- `MIME:mimetype` The content of the Packaged Content Element is MIME content, with the following header `"Content-Type: mimetype"`. Although it is possible to have `MIME:mimetype` with the `Transform` attribute set to `NONE`, it is far more likely to have `Transform` attribute set to `BASE64`. Note that if `Transform` is `NONE` is used, then the entire content must still conform to PCDATA. Some characters will need to be encoded either as the XML default entities, or as numeric character entities.
- `XML`. The content of the Packaged Content Element can be treated as an XML document. Entities and CDATA sections, or `Transform` set to `BASE64`, must be used to ensure that the Packaged Content Element contents are legitimate PCDATA.
- `x-ddd:usercode` The content is private, where `ddd` represents a domain name of a user, and `usercode` represents a particular

content format defined by that user. The guidelines around a x-ddd are very loose. Given company FFGGHH Inc., all of `x-www.ffgghh.com`, `x-ffgghh.com` and and `x-ffgghh` are legitimate examples. However, only one should be the correct format, as defined by FFGGHH Inc.

`Transform` — This identifies the transformation that has been done to the data before it was placed in the content. Valid values are:

- `NONE`. The PCDATA content of the Packaged Content Element is the correct representation of the data. Note that entity expansion must occur first (i.e. replacement of & and 	) before the data is examined. CDATA sections may legitimately occur in a Packaged Content Element where the `Transform` attribute is set to `NONE`.
- `BASE64`. The PCDATA content of the Packaged Content Element represents a BASE64 encoding of the actual content.

Content:

`PCDATA` — This is the actual data which has been embedded. The format of the data and rules on how to decode it are contained in the `Content` and the `Transform` attributes

Note that any special details, especially custom attributes, must be represented at a higher level.

2.9 Identifying Languages

OTP uses [XML] Language Identification to specify which languages are used within the content and attributes of OTP Messages.

The following principles have been used in order to determine which XML elements contain an `xml:lang` Attributes:

- a mandatory `xml:lang` attribute is contained on every Trading Component which contains attributes or content which may need to be displayed or printed in a particular language
- an optional `xml:lang` attribute is included on child elements of these Trading Components. In this case the value of `xml:lang`, if present, overrides the value for the Trading Component.

`xml:lang` attributes which follow these principles are included in the Trading Components and their child XML elements defined in section 5.

2.10 Secure and Insecure Net Locations

OTP contains several "Net Locations" which identify places where, typically, OTP Messages may be sent. Net Locations come in two types:

- "Secure" Net Locations which are net locations where privacy of data is secured using, for example, encryption methods such as [SSL], and
- "Insecure" Net Locations where privacy of data is not assured.

Where both types of net location are present, the following rules apply:

- either a Secure Net Location or an Insecure Net Location or both must be present
- if only one of the two Net Locations is present, then the one present must be used
- if both are present, then the either may be used depending on preference the preference of the sender of the message.

3. OTP Error Handling

OTP is designed as a request/response protocol where each message is composed of a number of Trading Blocks which contain a number of Trading Components. There are a several interrelated considerations in handling errors, re-transmissions, duplicates, and the like. These factors mean OTP aware applications must manage message flows more complex than the simple request/response model. Also a wide variety of errors can occur in messages as well as at the transport level or in Trading Blocks or Components.

This section describes at a high level how OTP handles errors, retries and idempotency. It covers:

- the different types of errors which can occur. This is divided into:
  - "technical errors" which are independent of the meaning of the OTP Message,
  - "business errors" which indicate that there is a problem specific to the process (payment or delivery) which is being carried out, and
- the depth of the error which indicates whether the error is at the transport, message or block/component level
- how the different trading roles should handle the different types of messages which they may receive.

3.1 Technical Errors

Technical errors are those which are independent of the meaning of the message. This means, they can affect any attempt at OTP communication. Typically they are handled in a standard fashion with a limited number of standard options for the user. Specifically these are:

- retrying the transmission, or
- cancelling the transaction.

When communications are operating sufficiently well, a technical error is indicated by an Error Component (see section 0) in an Error Block (see section 6.19) sent by the party which detected the error in an OTP message to the party which sent the erroneous message.

If communications too poor, a message which was sent may not reach its destination. In this case a time-out might occur.

The Error codes associated with Technical Errors are recorded in Error Components (see section 0) which lists all the different technical errors which can be set.

3.2 Business Errors

Business errors may occur when the OTP messages are "technically" correct. They are connected with a particular process, for example, an offer, payment, or delivery, where each process has a different set of possible business errors.

For example, "Insufficient funds" is a reasonable payment error but makes no sense for a delivery while "Back ordered" is a reasonable delivery error but not meaningful for a payment. Business errors are indicated in the Status Component (see section 5.14) of a "response block" of the normal type, for example a Payment Response Block or a Delivery Response Block. This allows whatever additional response related information is needed to accompany the error indication.

Business errors must usually be presented to the user so that they can decide what to do next. For example, if the error is insufficient funds in a Brand Independent Purchase (see section 7.3.1), the user might wish to choose a different payment instrument/account of the same brand or a different brand or payment system. Alternatively, if the OTP based implementation allows it and it makes sense for that instrument, the user might want to put more funds into the instrument/account and try again.

3.3 Error Depth

The three levels at which OTP errors can occur are the transport level, the message level, and the block level. Each is described below.

3.3.1 Transport Level

This level of error indicates a fundamental problem in the transport mechanism over which the OTP communication is taking place.

All transport level errors are technical errors and are indicated by either an explicit transport level error indication, such as a "No route to destination" error from TCP/IP, or by a time out where no response has been received to a request.

The only reasonable automatic action when faced with transport level errors is to retry and, after some number of automatic retries, to inform the user.

The explicit error indications that can be received are transport dependent and the documentation for appropriate OTP Transport supplement should be consulted for errors and appropriate actions.

Appropriate time outs to use are a function of both the transport being used and of the payment system if the request encapsulates payment information. The transport and payment system specific documentation should be consulted for time out and automatic retry parameters. Frequently there is no way to directly inform the other party of transport level errors but they should generally be logged and if automatic recovery is unsuccessful and there is a human user, the user should be informed.

3.3.2 Message Level

This level of error indicates a fundamental technical problem with an entire OTP message. For example, the XML is not Well formed, or the message is too large for the receiver to handle or there are errors in the Transaction Reference Block (see section 2.3) so it is not possible to figure out what transaction the message relates to.

All message level errors are technical errors and are indicated by an Error Component (see section 0) sent to the other party. The Error Component includes a `Severity` attribute which indicates whether the error is a `Warning` and may be ignored, a `TransientError` which indicates that a retry may resolve the problem or a `HardError` in which case the transaction must fail.

The Technical Errors (see section 5.17.2 Error Codes) that are Message Level errors are:

- XML not well formed. The document is not well formed XML (see [XML])
- XML not valid. The document is not valid XML (see [XML])
- block level technical errors (see section 3.3.3) on the Transaction Reference Block (see section 2.3) and the Signature Block only. This should only be carried out if the XML is valid

Note that checks on the Signature Block includes checking, where possible, that each Signature Component is correctly calculated. If the Digital Signature Element is incorrectly calculated then the data that should have been covered by the signature can not be trusted and must be treated as erroneous. A description of how to check a signature is correctly calculate is contained in section 4.2 Checking a Signature is Correctly Calculated.

3.3.3 Block Level

A Block level error indicates a problem with a block or one of its components in an OTP message (apart from Transaction Reference or Signature Blocks). The message has been transported properly, the overall message structure and the block/component(s) including the Transaction Reference and Signature Blocks are meaningful but there is some error related to one of the other blocks.

Block level errors can be either:

- technical errors, or
- business errors

Technical Errors are further divided into:

- Block Level Attribute and Element Checks, and
- Block and Component Consistency Checks

If a technical error occurs related to a block or component, then an Error Component is returned and, unless it is merely a warning, the usual response block is suppressed.

3.3.3.1 Block Level Attribute and Element Checks

Block Level Attribute and Element Checks occur only within the same block. Checks which involve cross-checking against other blocks are covered by Block and Component Consistency Checks.

The Block Level Attribute & Element checks are:

- checking that each attribute value within each element in a block conforms to any rules contained within this OTP specification
- checking that the content of each element conforms to any rules contained within this OTP specification
- if the previous checks are OK, then cross-checking attribute values and element content against other attribute values or element content within any other components in the same block.

3.3.3.2 Block and Component Consistency Checks

Block and Component Consistency Checks consist of:

- checking that the combination of blocks and/or components present in the OTP Message are consistent with the rules contained within this OTP specification
- checking for consistency between attributes and element content within the blocks within the same OTP message.
- checking for consistency between attributes and elements in blocks in this OTP message and blocks received in earlier OTP messages for the same OTP transaction

3.3.3.3 Block Business Errors

If a business error occurs in a process such as a Payment or a Delivery, then the usual type of response block is returned. The Status Component (see section 5.14) within that response block indicates the error and its severity. No Error Component or Error Block is generated for business errors.

3.4 Idempotency, Processing Sequence, and Message Flow

OTP messages are actually a combination of blocks and components as described in 2.1.1 OTP Message Structure. Especially in future extensions of OTP, a rich variety of combinations of such blocks and components can occur. It is important that the multiple transmission/receipt of the "same" request for state changing action not result in that action occurring more than once. This is called idempotency. For example, a customer paying for an order would want to pay the full amount only once. Most network transport mechanisms have some probability of delivering a message more than once or not at all, perhaps requiring retransmission. On the other hand, a request for status can reasonably be repeated and should be processed fresh each time it is received.

Correct implementation of OTP can be modelled by a particular processing order as detailed below. Any other method that is indistinguishable in the messages sent between the parties is equally acceptable.

3.4.1 Server Role Processing Sequence

"Server roles" are any Trading Role which is not the Consumer role. They are "Server roles" since they typically receive a request which they must service and then produce a response.

The model processing sequence for a Server role is indicated in the diagram below.

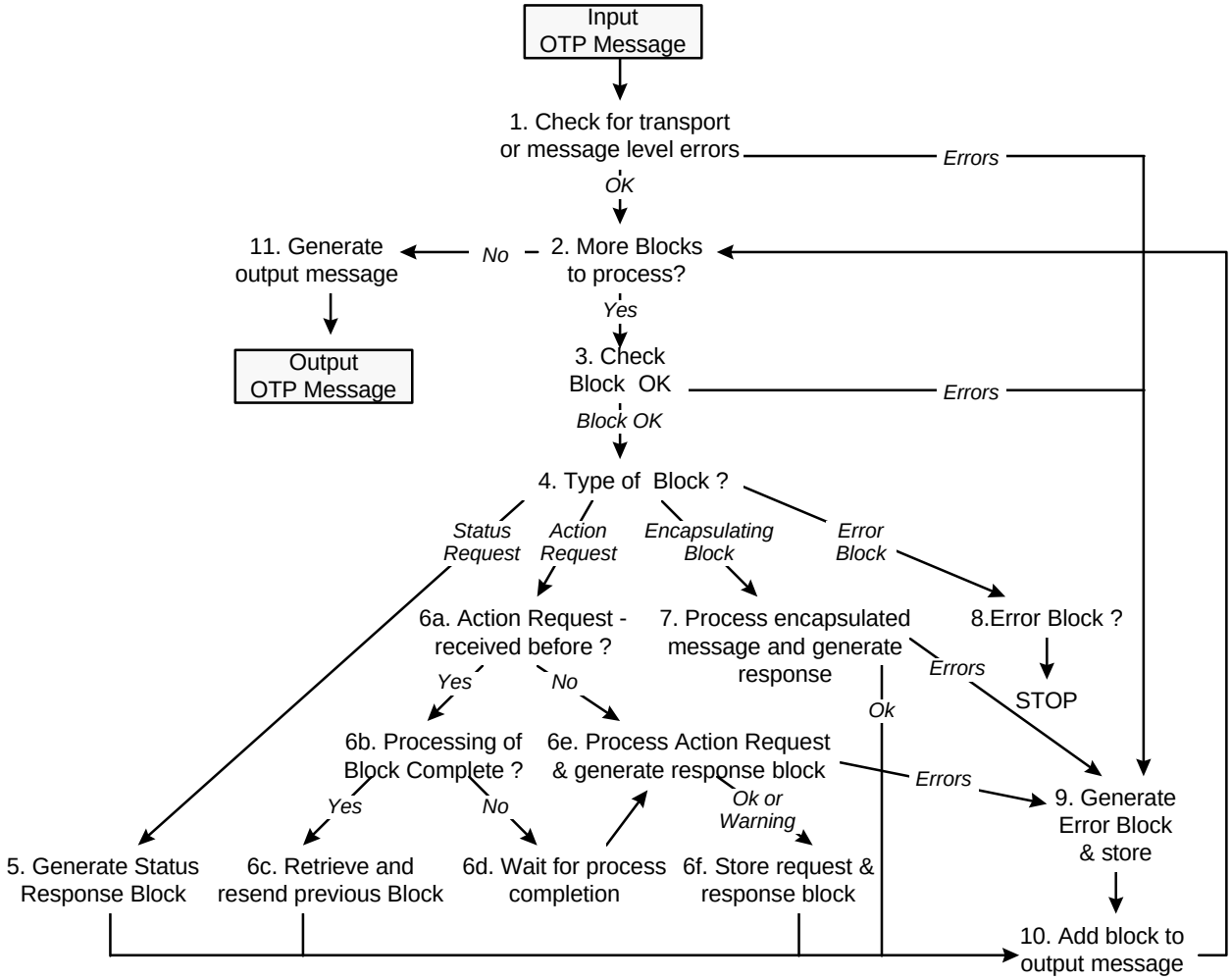


Figure 10 Server Role Processing Sequence

Each of the processes in the sequence is described in more detail below.

3.4.1.1 Check for Transport or Message Level Error

On receipt of an OTP request message (step 1), first check for transport or message level errors (see sections 3.3.1 and 3.3.2). These are errors which indicate that the entire message is corrupt and can not reliably be associated with any particular transaction or, if it can be associated with a transaction, the interior information in the message can not be reliably accessed.

If the `OtpTransId`attribute in the Transaction Id Component (see section 2.3.1) can be determined, set up a response message with an appropriate Error Component. Perform local actions such as making log entries.

If the value of the `OtpTransId`attribute is not recognised as belonging to an OTP transaction when other Blocks in the OTP Message indicate that it should be recognised, then report the error using an Error Component with a `Severity`of `HardError`, an `ErrorCode`set to `AttValNotRecog`(attribute value not recognised), and an Error Location element (see section 5.17.3) that points to the `OtpTransId`attribute.

No idempotency related actions are necessary.

3.4.1.2 Process all the blocks

If there are no message level errors, process each of the blocks within the message which has not been processed (step 2).

Once all the blocks have been processed, generate a response message (step 11) and send it to the requester unless there are fatal transport level problems. As recommended for the particular transport used, a limited number of automatic response retransmission attempts may be appropriate.

It may be desirable to log the complete response message at the server. Failure of the requester to receive a response may lead to a time-out and a retransmission of the request. Following the procedures above, a duplicate request message should produce a duplicate of the previous response except for changes in status and transient error conditions that have changed.

3.4.1.3 Check the Block is OK

Check the block is OK (see section 3.3.3). For each block level error found, an appropriate Error Component should be created to be included in the OTP Message sent back to the Consumer. Note that some checking of the Transaction Reference Block and the Signature Block has occurred as part of Message Level error checking.

If one or more of the Error Components contain a `Severity` attribute with a value of `TransientError` or `HardError`, then no response block need be generated and no further processing of the block, including idempotency related actions are necessary.

3.4.1.4 Determine the Type of the Block

Trading Blocks that survive the above steps and thus have no errors, or at worst have added a warning error component to the response, can receive further processing. The nature of the processing depends (step 4) on whether the block is a Status Request, Action Request, an Error Block or contains an Encapsulated Message.

3.4.1.5 Status Request Blocks

Status Request Blocks (step 5) are either:

- Inquiry Request Trading Block (see section 6.14), or
- Ping Request Block (see section 6.16).

These status requests do not change state and are processed fresh to get the current status. The appropriate response block should be added to the OTP message being composed.

No idempotency actions are required.

3.4.1.6 Action Request Blocks

Blocks which request an action and change state need to be subject to idempotency duplicate filtering by checking to see if the same block for the same transaction has been previously stored (step 6a) at the server as described later.

If the Block has been received previously then:

- if processing of the previously stored block is complete (step 6b) then the same OTP Block as previously produced must be included for resending to the Consumer (step 6c).
- if processing is not complete, wait until the processing is complete (step 6d) before sending the response.

If the block has not been received before, the action request is processed normally (step 6e) producing a response block that is added to the response message. This might or might not indicate a business error.

If there is a transient error indicated by an Error Component that contains a `Severity` attribute set to `TransientError`, then apart from sending the Error Block, no further actions should be taken so the action can be retried.

If there is no Transient Error, then the transaction id, the request block, and the response block must be stored (step 6f) so they can be found as described above (step 6a) should a duplicate OTP action request block be received for this transaction in the future.

[Note] *Most business errors should be labeled as a `TransientError` as there is usually some possibility they will be corrected over time or some user action exists that can fix them. Requesters are expected to understand business errors and the appropriate time scale for user actions for retrying.*

[Note End]

3.4.1.7 Encapsulating Blocks

Blocks which encapsulate a payment protocol (step 7) pass along the enclosed information to the payment system involved.

OTP does not know the meaning of the enclosed information. It is thus up to the payment system involved to handle error detection and idempotency. Payment systems adapted for the Internet include idempotency handling because duplicates are always possible. Should a payment system have no idempotency handling, a layer between OTP and the payment system must be added to take care of this.

No OTP level idempotency actions are required for encapsulating blocks. The payment system must return material to be encapsulated in the OTP response message along with indications as to whether the exchange will continue or this is the final response and an indication whether an error occurred. If a payment protocol error has occurred, an Error Component is added to the response block.

3.4.1.8 Error Block Received

An error block (step 8) should not occur in a request and should be treated as an unexpected element with a Severity of `HardError`. No response to the block should be made in order to avoid the risk of loops.

[Note] *Consumers should send Error Blocks to a server specified in the `ErrorNetLoc` attribute of the appropriate Trading Role element as a response to the detection of an error in an OTP Message that has been received (see section 3.4.1.9 Generate Error Block). This may be the same server as is used to accept OTP Messages which contain no error. In this case, the error block must not considered as a fatal error.*

[Note End]

3.4.1.9 Generate Error Block

If any of the previous steps resulted in an error being detected and an Error Component being created then generate an Error Block (step 9) containing the Error Components that describe the error(s).

Unless the error is a "Transient Error", the Error Component(s) and the request block which caused the Error Components to be generated should be stored so that it can be reused if the action request is received again (step 6a).

"Transient Errors" are not stored so that if the original Response Block is received again, then it can be processed as if it had never been received before.

3.4.1.10 Add Block to Output Message

Any Blocks which have been created as a result of processing the block received are added to the output message.

3.4.2 Client Role Processing Sequence

The "Client role" in OTP is the Consumer Trading Role.

[Note] *A company or organisation that is a Merchant, for example, may take on the Trading Role of a Consumer when making a purchase or downloading or withdrawing electronic cash.*
[Note End]

The model processing sequence for a Client role is indicted in the diagram below.

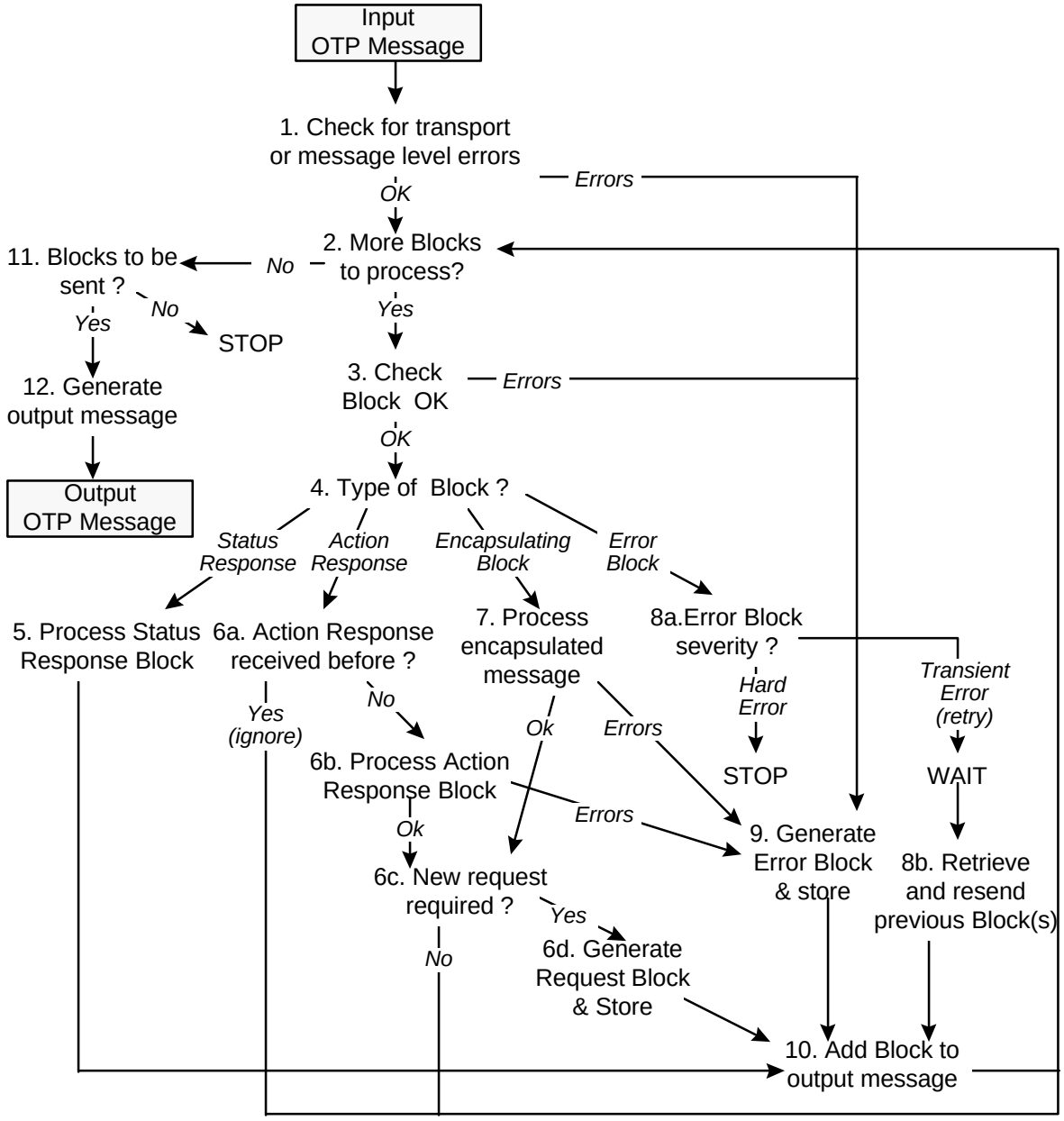


Figure 11 Client Role Processing Sequence

Each of the processes in the sequence is described in more detail below.

3.4.2.1 Check for Transport or Message Level Error

On receipt of an OTP response message (step 1), first check for transport or message level errors (see sections 3.3.1 and 3.3.2). These are errors which indicate that the entire message is corrupt and can not reliably be associated with any particular transaction or, if it can be associated with a transaction, the interior information in the message can not be reliably accessed. Set up an error indication message with an Error Component indicating the error.

If the value of the `OtpTransId`attribute is not recognised as belonging to an OTP transaction when other Blocks in the OTP Message indicate that it should be recognised, then report the error using an Error Component with a `Severity`of `HardError`, an `ErrorCode`set to `AttValNotRecog`(attribute value not recognised), and an Error Location element (see section 5.17.3) that points to the `OtpTransId`attribute.

On failure to receive an expected response message, the time out strategy indicated in the documentation for the transport method being used should be followed. This may include some number of automatic retransmissions of the request. If a user is present, they may be offered options of continuing to retransmit the request or of cancelling the transaction.

3.4.2.2 Process all the blocks

If there are no message level errors, process each of the blocks within the message which has not been processed (step 2).

Once all the blocks have been processed, check to see if there are any blocks to be sent (step 11). There may be no blocks to send if the last response message received was the last message of the transaction.

If blocks are to be sent then generate a request message (step 12) and send it to the server. It may be desirable to log the complete request message at the client. Failure of the server to receive a response may lead to a time-out and a retransmission of the request.

3.4.2.3 Check the Block is OK

If there are no message level errors process each of the blocks within the message (step 2).

Check the block is OK (see section 3.3.3). For each block level error found, an appropriate Error Component should be created to be included in an Error Component sent back to the Server.

If one or more of the Error Components contain a `Severity`attribute with a value of `TransientError`or `HardError`, no further processing of the block should occur and it is likely that this will result in termination of the transaction.

3.4.2.4 Determine the Type of the Block

Trading Blocks that survive the above steps and thus have no errors, or at worst have added a warning error component to the error indication message, can receive further processing. The nature of the processing depends (step 4) on whether the block is a Status Response, Action Response, an Error Block or contains an Encapsulated Message.

3.4.2.5 Status Response Blocks

Status Response Blocks (step 4) are either:

- Inquiry Response Trading Blocks (see section 6.15), or
- Ping Response Blocks (see section 6.17)

In general, such blocks should be considered a status update. The best action to take at the requester may depend on whether this is in response to a user originated or automatic status request, whether a status display that could be updated is being presented to the user, and whether the status response block shows a change in status from a previous response block for the same type of status. Thus client detection of duplication in successive status response blocks may be useful.

3.4.2.6 Action Response Blocks

Check to determine if the Block has been received previously (step 6a). If it has then it should be ignored.

These indicate an action taken at the server in response to an action request block or a business error. If the response indicates success the block should be processed (step 6b) and, if required by the transaction (step 6c) , another Action Request Block generated and stored (step 6d).

The Response Block should always be stored with the transaction id and until the transaction is terminated. If the Response Block indicates a transient business error, appropriate manually chosen or automatic steps to fix the problem or cancel the transaction should be provided.

3.4.2.7 Encapsulating Blocks

Blocks which encapsulate a payment protocol (step 7) pass along the enclosed information to the payment system involved.

OTP does not know the meaning of the enclosed information. It is up to the payment system involved to handle error detection and idempotency. Payment systems adapted for the Internet include idempotency handling because duplicates are always possible. Should a payment system have no idempotency handling, a layer between OTP and the payment system must be added to take care of this.

No OTP level idempotency actions are required for encapsulating blocks. The payment system must return an indication of whether an error occurred. In addition, for a continuing exchange, it must return material to be encapsulated in the next OTP request/exchange (step 6d). If the response was a final response for that payment exchange and there was an error, the payment system may optionally return material to be encapsulated in the error indication.

3.4.2.8 Error Block

An error block in a response (step 8a) indicates some problem was detected by the server. If all of the error components are warnings, they may be optionally logged and/or presented to the user.

Transient errors may be used to provide a manual or automatic resending (step 8b) of a block previously stored or alternatively may result in transaction cancellation. Hard errors will always terminate the transaction, unless they are in optional blocks, with appropriate indication to he user.

3.4.2.9 Generate Error Block

If an error indication message was created above, try to send it to the server unless all of the error components are of the warning severity in which case attempted transmission to the server is optional.

[Note] *To avoid error message loops, such an error indication from a requester must be sent to the Error Net Location specified in the Trading Role Element (see section 5.5.2) for the Organisation that is the server. Any errors encountered in sending such an error indication should be, at most, logged and must not result in any further attempts to transmit any error indication.*

[Note End]

3.4.2.10 Add Block to Output Message

Any Blocks which have been created as a result of processing the block received are added to the output message.

4. Security Considerations

This section considers the security associated with OTP. It covers:

- an overview of how OTP uses digital signatures
- how to check a signature is correctly calculated
- how Payment Handlers and Delivery Handlers check they can carry out payments or deliveries on behalf of a Merchant.
- how OTP handles data integrity and privacy

4.1 Digital Signatures and OTP

In general, signatures when used with OTP:

- are always treated as a OTP Components (see section 5)
- hash one or more OTP Components or Trading Blocks, possibly including other Signature Components, in any OTP message within the same OTP Transaction
- identify:
  - which Organisation signed (generated) the signature, and
  - which Organisation(s) should verify the signature in order to check that the Action the Organisation should take can occur.

The way in which Signatures Components hash one or more elements is illustrated in the figure below.

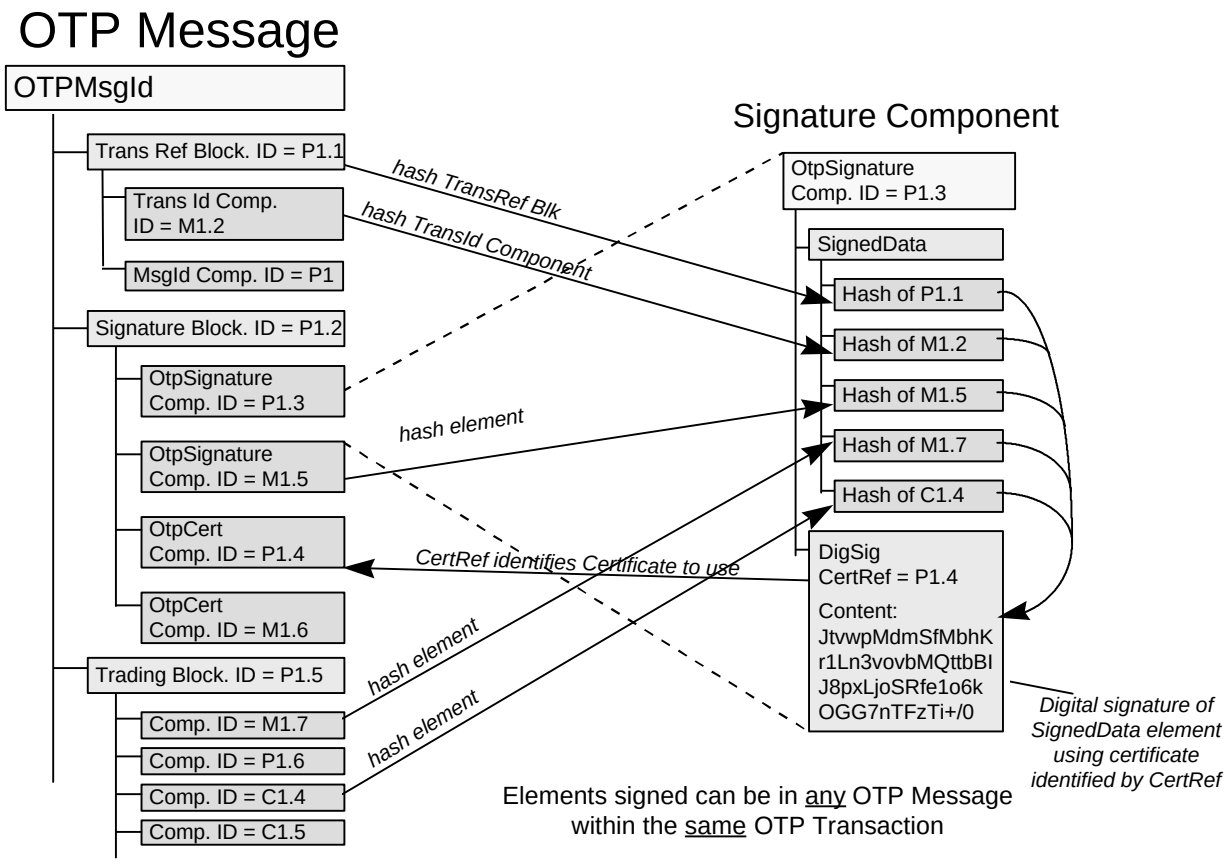


Figure 12 Signature Hashing

[Note] *The classic example of one signature signing another in OTP, is when an Offer is first signed by a Merchant creating an "Offer Response" signature, which is then later signed by a Payment Handler together with a record of the payment creating a "Payment Receipt" signature. In this way, the payment in an OTP Transaction is bound to the Merchant's offer.*
[Note End]

The detailed definitions of how signatures are created is contained in the paper "Digital Signature for XML - Proposal", see [XMLSIG]. That document should be read in conjunction with this section.

The remainder of this section contains:

- an example of how OTP uses signatures
- how the `SignerOrgRef`and `VerifierOrgRef`attributes within a Signature Component are used to identify the organisations associated with the signature
- how signatures may use either Symmetric or Asymmetric Cryptography
- Mandatory and Optional use of Signatures by OTP, and
- how OTP uses signatures to prove actions complete successfully

4.1.1 OTP Signature Example

An example of how signatures are used is illustrated in the figure below which shows how the various components and elements in a Baseline Purchase relate to one another. Refer to this example in the later description of how signatures are used to check a payment or delivery can occur (see section 4.3).

[Note] *A Baseline Purchase transaction has been used for illustration purposes. The usage of the elements and attributes is the same for all types of OTP Transactions.*

[Note End]

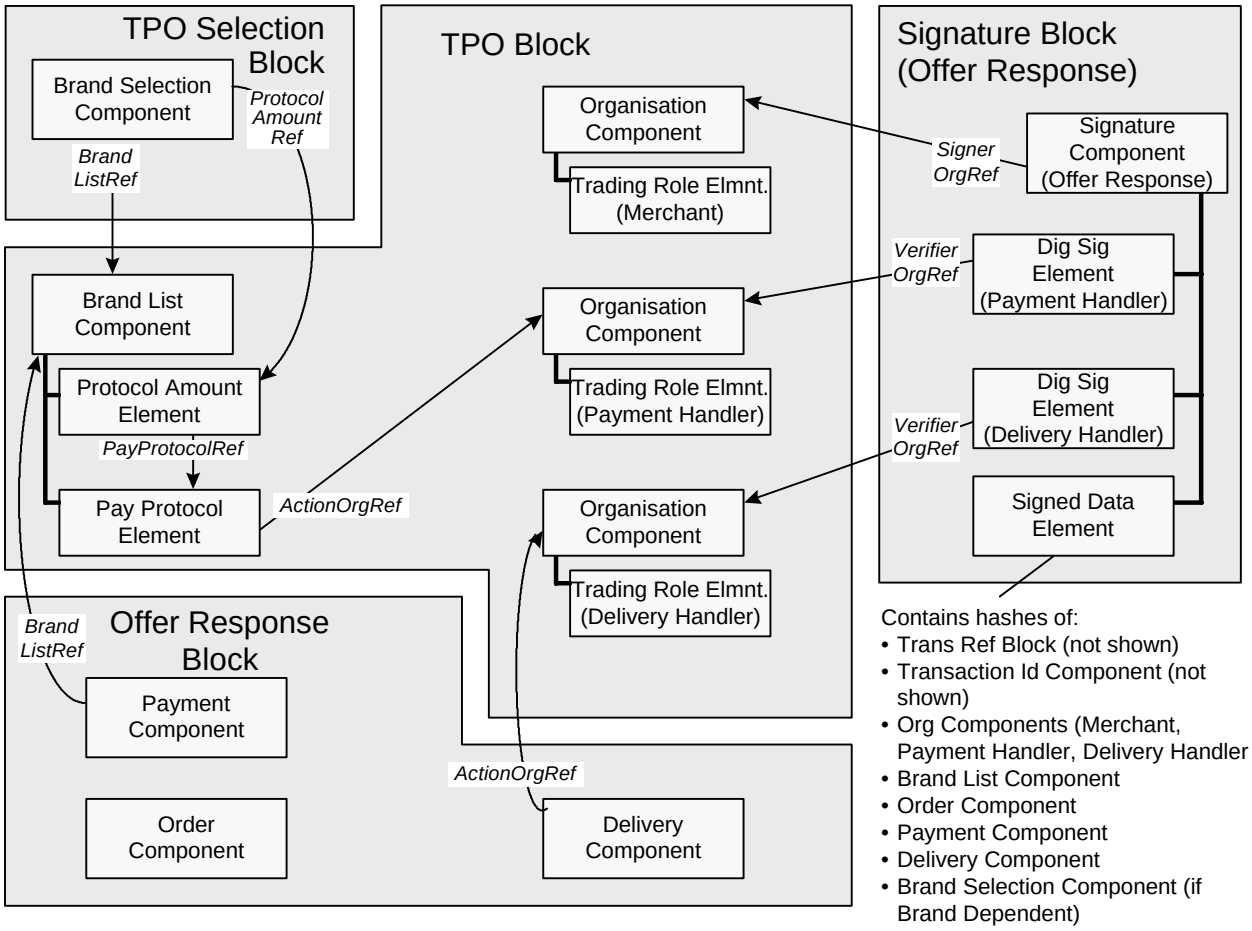


Figure 13 Example use of Signatures for Baseline Purchase

4.1.2 SignerOrgRef and VerifierOrgRef Attributes

The `SignerOrgRef`attribute on the Signature Component contains an Element Reference (see section 2.5) that points to the Organisation Component of the Organisation which generated the Signature. In this example its the Merchant.

Note that the type of the Signature Component must match the Trading Role of the Organisation which signed it. If it does not, then it is an error. Valid combinations are given in the table below.

Signature Component Type	Valid Trading Role
OfferResponse	Merchant
PaymentResponse	PaymentHandler

The VerifierOrgRefattribute on the DigSig elements, contains Element References to the Organisation Components of the Organisations that should use the signature to verify that:

- they have a pre-existing relationship with the Organisation that generated the signature,
- the data which is secured by the signature has not been changed,
- the data has been signed correctly, and
- the action they are required to undertake on behalf of the Merchant is therefore authorised.

4.1.3 Symmetric and Asymmetric Cryptography

The Signer of an Action and a Verifier of an Action may have agreed to use cryptography which is understood only by the two organisations involved. This requires that a separate Digital Signature Element for use by the verifier is contained within the Signature Component. This approach is more likely if symmetric cryptography is being used between the Trading Roles.

Equally the same cryptography may be understood by several or all of the Trading Roles. In this case one Digital Signature Element may refer to multiple Verifiers of an Action. This is more likely if public key/asymmetric cryptography is being used.

Note that one transaction may involve use of both symmetric and asymmetric cryptography.

4.1.4 Mandatory and Optional Signatures

OTP does not mandate the use of signatures. For example, if a micro payment is being made for 0.1 cents, then the cost of the cryptography required to generate the signature may be greater than the income generated from the payment. Therefore it is up to the Merchant to decide whether OTP Messages will include signatures, and for the Consumer to decide whether carrying out a transaction without signatures is an acceptable risk. If Merchants discover that transactions without signatures are not being accepted, then they will start using signatures or accept a lower volume and value of business.

Additional optional signatures, over and above the ones required by the Trading Roles may be included, for example, to identify a Customer Care Provider or so that a Merchant can sign an Offer using a certificate issued by a Certificate Authority which offers Merchant "Credentials" or some other warranty on the goods or services being offered.

4.1.5 Using signatures to Prove Actions Complete Successfully

Proving an action completed successfully, is achieved by signing data on Response messages. Specifically:

- on the Offer Response, when a Merchant is making an Offer to the Consumer which can then be sent to either:

  - a Payment Handler to prove that payment is authorised, or
  - a Delivery Handler to prove that Delivery is authorised

- on the Payment Response, when a Payment Handler is generating a Payment Receipt which can be sent to either:
  - a Delivery Handler, in a Delivery Request Block to prove that delivery is authorised, or
  - another Payment Handler, in a second Payment Request, to prove that the second payment in a Value Exchange OTP Transaction is authorised.

This proof of an action may, in future versions of OTP, also be used to prove after the event that the OTP transaction occurred. For example to a Customer Care Provider.

4.2 Checking a Signature is Correctly Calculated

Checking a signature is correctly calculated is part of checking for Message Level Errors (see section 3.3.2). It is included here so that all signature and security related considerations are kept together.

Before a Trading Role can check a signature it must identify which of the potentially multiple digital signature elements should be checked. The steps involved are as follows:

- check that a Signature Block is present and it contains one or more Signature Components
- identify the Organisation Component which contains an `OrgId` attribute for the Organisation which is carrying out the signature check. If no or more than one Organisation Component is found then it is an error
- use the `ID` attribute of the Organisation Component to identify the Digital Signatures Elements which the Trading Role should verify. Note there may be no signatures for a Trading Role to verify.
- verify the Signature Components that contain the Digital Signature Elements as follows:
  - check that the Digital Signature Element correctly signs the Signed Data Element
  - check that the Hash Elements in the Signed Data Element are correctly calculated where Components or Blocks that are hashed have been received by the organisation checking the signature

4.3 Checking a Payment or Delivery can occur

This section describes the processes required for a Payment Handler or Delivery Handler to check that a payment or delivery can occur. This may include checking signatures if this is specified by the Merchant.

In outline the steps are:

- check that the Payment Request or Delivery Request has been sent to the correct organisation
- check that correct OTP components are present in the request, and
- check that the payment or delivery is authorised

For clarity and brevity the following terms or phrases are used in this section:

- a "Request Block" is used to refer to either a Payment Request Block (see section 6.6) or a Delivery Request Block (see section 6.9) unless specified to the contrary
- a "Response Block" is used to refer to either a Payment Response Block (see section 6.8) or a Delivery Response Block (see section 6.10)
- an "Action" is used to refer to an action which occurs on receipt of a Request Block. Actions can be either a Payment or a Delivery
- an "Action Organisation", is used to refer to the Payment Handler or Delivery Handler that carries out an Action
- a "Signer of an Action", is used to refer to the Organisations that sign data about an Action to authorise the Action, either in whole or in part
- a "Verifier of an Action", is used to refer to the Organisations that verify data to determine if they are authorised to carry out the Action
- an `ActionOrgRef`attribute contains Element References which can be used to identify the "Action Organisation" that should carry out an Action

4.3.1 Check the Action Request was sent to the Correct Organisation

Checking the Action Request was sent to the correct Organisation varies depending on whether the Action is a Payment or a Delivery.

4.3.1.1 Payment

In outline a Payment Handler checks if it can accept or make a payment by identifying the Payment Component in the Payment Request Block it has received, then using the `ID` of the Payment Component to track through the Brand List and Brand Selection Components to identify the Organisation selected by the Consumer and then checking that this organisation is itself.

The way data is accessed to do this is illustrated in the figure below.

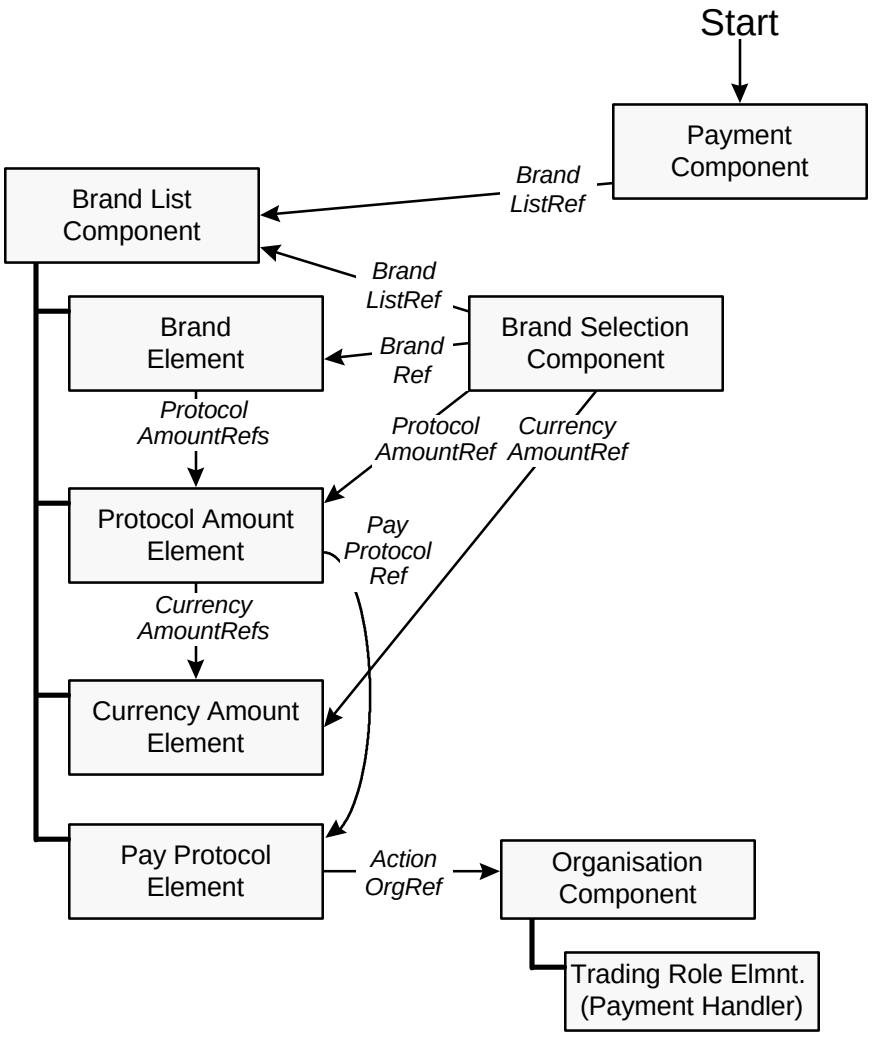


Figure 14 Checking a Payment Handler can carry out a Payment

The following describes the steps involved and the checks which need to be made:

1) Identify the Payment Component (see section 5.8) in the Payment Request Block that was received.

2) Identify the Brand List and Brand Selection Components for the Payment Component. This involves:

   a) identifying the Brand List Component (see section 5.6) where the value of its `ID` attribute matches the `BrandListRef`attribute of the Payment Component. If no or more than one Brand List Component is found there is an error.

   b) identifying the Brand Selection Component (see section 5.7) where the value of its `BrandListRef`attribute matches the `BrandListRef`of the Payment Component. If no or more than one matching Brand Selection Component is found there is an error.

3) Identify the Brand, Protocol Amount, Pay Protocol and Currency Amount elements within the Brand List that have been selected by the Consumer as follows:

   a) the Brand Element (see section 0) selected is the element where the value of its `Id` attribute matches the value of the `BrandRef`attribute in the Brand Selection. If no or more than one matching Brand Element is found then there is an error.

   b) the Protocol Amount Element (see section 0) selected is the element where the value of its `Id` attribute matches the value of the `ProtocolAmountRef`attribute in the Brand Selection Component. If no or more than one matching Protocol Amount Element is found there is an error

   c) the Pay Protocol Element (see section 5.6.4) selected is the element where the value of its `Id` attribute matches the value of the `PayProtocolRef`attribute in the identified Protocol Amount Element. If no or more than one matching Pay Protocol Element is found there is an error

   d) the Currency Amount Element (see section 5.6.3) selected is the element where the value of its `Id` attribute matches the value of the `CurrencyAmountRef`attribute in the Brand Selection Component. If no or more than one matching Currency Amount element is found there is an error

4) Check the consistency of the references in the Brand List and Brand Selection Components:

   a) check that an Element Reference exists in the `ProtocolAmountRefs`attribute of the identified Brand Element that matches the `Id` attribute of the identified Protocol Amount Element. If no or more than one matching Element Reference can be found there is an error

   b) check that the `CurrencyAmountRefs`attribute of the identified Protocol Amount element contains an element reference that matches the `Id` attribute of the identified Currency Amount element. If no or more than one matching Element Reference is found there is an error.

   c) check the consistency of the elements in the Brand List. Specifically, the selected Brand, Protocol Amount, Pay Protocol and Currency Amount Elements are all child elements of the identified Brand List Component. If they are not there is an error.

5) Check that the Payment Handler that received the Payment Request Block is the Payment Handler selected by the Consumer. This involves:

   a) identifying the Organisation Component for the Payment Handler. This is the Organisation Component where its `ID` attribute matches the `ActionOrgRef`attribute in the identified Pay Protocol Element. If no or more than one matching Organisation Component is found there is an error

b) checking the Organisation Component has a Trading Role Element with a `Role` attribute of `PaymentHandler`. If not there is an error

c) finally, if the identified Organisation Component is not the same as the organisation that received the Payment Request Block, then there is an error.

4.3.1.2 Delivery

The way data is accessed by a Delivery Handler in order to check that it may carry out a delivery is illustrated in the figure below.

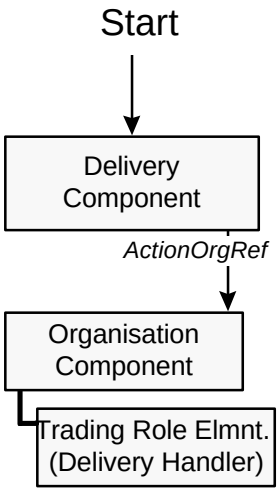


Figure 15 Checking a Delivery Handler can carry out a Delivery

The steps involved are as follows:

1) Identify the Delivery Component in the Delivery Request Block. If there is no or more than one matching Delivery Component there is an error

2) Use the `ActionOrgRef`attribute of the Delivery Component to identify the Organisation Component of the Delivery Handler. If there is no or more than one matching Organisation Component there is an error

3) If the Organisation Component for the Delivery Handler does not have a Trading Role Element with a `Role` attribute of `DeliveryHandler`there is an error

4) Finally, if the organisation that received the Delivery Request Block does not identify the Organisation Component for the Delivery Handler as itself, then there is an error.

4.3.2 Check the Correct Components are present in the Request Block

Check that the correct components are present in the Payment Request Block (see section 6.6) or in the Delivery Request Block (see section 6.9).

If components are missing, there is an error.

4.3.3 Check an Action is Authorised

The previous steps identified the Action Organisation and that all the necessary components are present. This step checks that the Action Organisation is authorised to carry out the Action.

In outline the Action Organisation identifies the Merchant, checks that it has a pre-existing agreement which allows it carry out the Action and that any constraints implied by that agreement are being followed, then, if signatures are required, it checks that they sign the correct data.

The steps involved are as follows:

1) Identify the Merchant. This is the Organisation Component with a Trading Role Element which has a `Role` attribute with a value of `Merchant`. If no or more than one Trading Role Element is found, there is an error

2) Check the Action Organisation's agreements with the Merchant allows the Action to be carried out. To do this the Action Organisation must check that:

    a) the Merchant is known and a pre-existing agreement exists for the Action Organisation to be their agent for the payment or delivery

    b) they are allowed to take part in the type of OTP transaction that is occurring. For example a Payment Handler may have agreed to accept payments as part of a Baseline Purchase, but not make payments as part of a Baseline Refund

    c) any constraints in their agreement with the Merchant are being followed, for example, whether or not an Offer Response signature is required

3) Check the signatures are correct. If signatures are required then they need to be checked. This involves:

    a) Identifying the correct signatures to check. This involves the Action Organisation identifying the Signature Components where the `VerifierOrgRef`attribute of the Digital Signature element points to the Action Organisation's Organisation Component. Depending on the OTP Transaction being carried out (see section 7) either one or two signatures may be identified

    b) checking that the Signature Components are correct. This involves checking that the necessary Trading Components have been hashed (see section 4.3.3.1).

[Note] *Validation that the signature is correct and that the Hash elements within the signature are correctly calculated is described in section 3 OTP Error Handling. This is because errors in the signature or calculation of hashes is considered a Message Level Error and is carried out before the Request Block is processed.*

[Note End]

4.3.3.1 Check the Signatures Sign Correct Data

All Signature Components contained within OTP Messages must always hash:

- the Transaction Id Component (see section 2.3.1) of the OTP message that contains the Signature Component. This binds the globally unique `OtpTransId`to other components which make up the OTP Transaction
- the Transaction Reference Block (see section 2.3) of the first OTP Message that contained the signature. This binds the `OtpTransId` with information about the OTP Message contained inside the Message Id Component (see section 2.3.2).

Checking that each signature signs the correct data, involves checking that hashes of the necessary components are present in the `SignedData`element of the Signature Component.

The hashes that need to be present depend on the Trading Role of the Organisation which generated (signed) the signature:

- if the signer of the signature is a Merchant then:
  - hashes must be present for all the components in the Request Block apart from the Brand Selection Component which is optional
- if the signer of the signature is a Payment Handler then hashes should be present for:
  - the Signature Component signed by the Merchant, and optionally
  - one or more Signature Components signed by the Payment Handler(s) identified by the appropriate `ActionSignerRefs`attribute.

4.4 Data Integrity and Privacy

The overall integrity of data in OTP Messages is ensured by the signing of hashes of Components and Trading Blocks contained in a Signature Component (see section 5.16) in a Signature Block (see section 6.18).

Privacy of information is provided by sending OTP Messages between the various Trading Roles using a secure channel such as [SSL]. Use of a secure channel within OTP is optional.

5. Trading Components

This section describes the Trading Components used within OTP. Trading Components are the child XML elements which occur immediately below a Trading Block as illustrated in the diagram below.

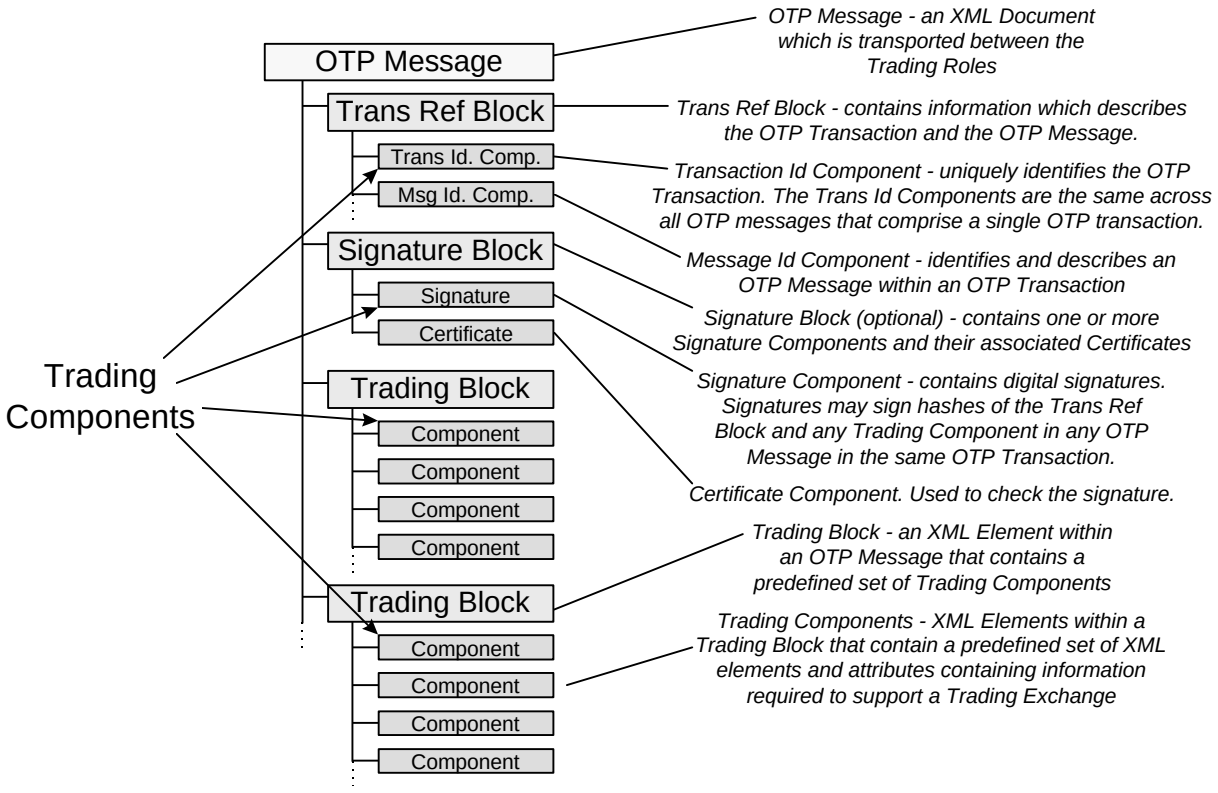


Figure 16 Trading Components

The Trading Components described in this section are listed below in approximately the sequence they are likely to be used:

- Protocol Options Component
- Authentication Data Component
- Authentication Response Component
- Order Component
- Organisation Component
- Brand List Component
- Brand Selection Component
- Payment Component
- Payment Scheme Component
- Payment Receipt Component
- Delivery Component

- Delivery Note Component
- Signature Component
- Certificate Component
- Error Component

Note that the following components are listed in other sections of this specification:

- Transaction Id Component (see section 2.3.1)
- Message Id Component (see section 2.3.2)

5.1 Protocol Options Component

Protocol options are options which apply to the OTP Transaction as a whole. Essentially it is used to identify what type of OTP Transaction is being carried out. For Baseline OTP it will identify one of the "Baseline" OTP Transactions (see section 7. Open Trading Protocol Transactions) by name.

The definition of a Protocol Options Component is as follows.

```
<!ELEMENT ProtocolOptions EMPTY>
<!ATTLIST ProtocolOptions
  ID  ID              #REQUIRED
  xml:lang            NMTOKEN  #REQUIRED
  ShortDesc           CDATA    #REQUIRED
  SenderNetLocn       CDATA    #REQUIRED
  SecureSenderNetLocn CDATA    #REQUIRED
  SuccessNetLocn      CDATA    #REQUIRED
  CancelNetLocn       CDATA    #REQUIRED
  ErrorNetLocn        CDATA    #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Protocol Options Component within the OTP Transaction.
`xml:lang`	Defines the language used by attributes or child elements within this component, unless overridden by an `xml:lang` attribute on a child element. See section 2.9 Identifying Languages.
`ShortDesc`	This contains a short description of the OTP Transaction in the language defined by `xml:lang`. Its purpose is to provide an explanation of what type of OTP Transaction is being conducted by the parties involved. It is used to facilitate selecting an individual transaction from a list of similar transactions , for example from a database of OTP transactions which has been stored by a Consumer, Merchant, etc.
`SenderNetLocn`	This contains the non secured net location of the sender of the TPO Block in which the Protocol Options Component is contained. It is the net location to which the recipient of the TPO block should send a TPO Selection Block if required. The content of this attribute is dependent on the Transport Mechanism

	see the Transport Mechanism Supplement.
`SecureSenderNetLocn`	This contains the secured net location of the sender of the TPO Block in which the Protocol Options Component is contained. The content of this attribute is dependent on the Transport Mechanism see the Transport Mechanism Supplement.
`SuccessNetLocn`	This contains the net location that the should be displayed after the OTP Transaction has successfully completed. The content of this attribute is dependent on the Transport Mechanism see the Transport Mechanism Supplement.
`CancelNetLocn`	This contains the net location that should be displayed by the Consumer after the OTP Transaction has been cancelled by one of the Trading Roles. For this purpose, cancel is defined as sending or receiving a Fail Trading Block (see section 6) where the `FailType` attribute of all the `FailReasons` in the block are set to `Cancel`. The content of this attribute is dependent on the Transport Mechanism see the Transport Mechanism Supplement.
`ErrorNetLocn`	This contains the net location that should be displayed after the OTP Transaction has failed due to an error. For this purpose, an error is defined as sending or receiving an Error Block (see section 6.19) where the `Severity` attribute of at least one of the Error Components (see section 5.17) in the Error Block is set to `HardError`, or there has been an irrecoverable loss of communication. The content of this attribute is dependent on the Transport Mechanism see the Transport Mechanism Supplement.

5.2 Authentication Data Component

This Trading Component contains data about how an Authentication within the OTP Transaction will occur. Its definition is as follows.

```
<!ELEMENT AuthData (PackagedContent)>
<!ATTLIST AuthData
  ID  ID              #REQUIRED
  AuthMethod          NMTOKEN  #REQUIRED
  ContentSoftwareId   CDATA    #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Authentication Data Component within the OTP Transaction.
`AuthMethod`	This identifies the content of the Authentication Data Component. Valid values are: • `sha1` This indicates that the recipient of the Authentication Data Component should generate a hash. See 5.3 Authentication Response Component. • `pay:ppp` A payment protocol specific authentication method. The

"ppp" identifies a payment protocol associated with a payment exchange which is part of the OTP Transaction. In this case the content and format of the AuthData element is defined in the appropriate Payment Scheme supplement. For example if a payment method "xzpay" provided an authentication method, then this attribute would have the value "pay:xzpay"

- x-ddd:nnna user defined authentication scheme type see section (2.7.3 User Defined Codes).

ContentSoftwareId — This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by xml:lang. It must contain, as a minimum:

- the name of the software manufacturer
- the name of the software
- the version of the software, and
- the build of the software

It is recommended that this attribute is included if the software which generated the content cannot be identified from the SoftwareId attribute on the Message Id Component (see section 2.3.2)

Content:

PackagedContent — This contains the challenge data as Packaged Content (see section 2.8) that is to be responded to using the method indicated by AuthMethod

5.3 Authentication Response Component

This Authentication Response Component contains the results of an authentication. Its definition is as follows.

```
<!ELEMENT AuthResp (PackagedContent) >
<!ATTLIST AuthResp
  ID  ID              #REQUIRED
  ContentSoftwareId  CDATA    #IMPLIED >
```

Attributes:

ID — An identifier which uniquely identifies the Authentication Response Component within the OTP Transaction.

ContentSoftwareId — This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by xml:lang. It must contain, as a minimum:

- the name of the software manufacturer
- the name of the software
- the version of the software, and
- the build of the software

It is recommended that this attribute is included if the software which

generated the content cannot be identified from the `SoftwareId` attribute on the Message Id Component (see section 2.3.2)

Content:

`PackagedContent` — This contains the response to the content of the Authentication Data Component see section 5.2 as Packaged Content (see section 2.8).

For a payment specific scheme, it may contain scheme-specific data. Refer to the scheme-specific supplemental documentation.

5.4 Order Component

An Order Component contains information about an order. Its definition is as follows.

```
<!ELEMENT Order (PackagedContent?) >
<!ATTLIST Order
  ID  ID              #REQUIRED
  xml:lang            NMTOKEN  #REQUIRED
  OrderIdentifier     CDATA    #REQUIRED
  ShortDesc           CDATA    #REQUIRED
  OkFrom              CDATA    #REQUIRED
  OkTo                CDATA    #REQUIRED
  ApplicableLaw       CDATA    #REQUIRED
  ContentSoftwareId   CDATA    #IMPLIED >
```

Attributes:

`ID` — An identifier which uniquely identifies the Order Component within the OTP Transaction.

`xml:lang` — Defines the language used by attributes or child elements within this component, unless overridden by an `xml:lang` attribute on a child element. See section 2.9 Identifying Languages.

`OrderIdentifier` — This is a code, reference number or other identifier which the creator of the Order may use to identify the order. It must be unique within an OTP Transaction. If it is used in this way, then it may remove the need to specify any content for the Order element as the reference can be used to look up the necessary information in a database.

`ShortDesc` — A short description of the order in the language defined by `xml:lang`. It is used to facilitate selecting an individual order from a list of orders, for example from a database of orders which has been stored by a Consumer, Merchant, etc.

`OkFrom` — The date and time in [UTC] format after which the offer made by the Merchant lapses.

`OkTo` — The date and time in [UTC] format before which a Value Acquirer may accept the offer made by the Merchant is not valid.

`ApplicableLaw` — A phrase in the language defined by `xml:lang` which describes the state or country of jurisdiction which will apply in resolving problems or disputes.

`ContentSoftwareId`	This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:lang`. It must contain, as a minimum: • the name of the software manufacturer • the name of the software • the version of the software, and • the build of the software It is recommended that this attribute is included if the software which generated the content cannot be identified from the `SoftwareId` attribute on the Message Id Component (see section 2.3.2)
Content:	
`PackagedContent`	An optional description of the order information as Packaged Content (see section 2.8).

5.4.1 Order Description Content

The Packaged Content element will normally be required, however it may be omitted where sufficient information about the purchase can be provided in the `ShortDesc`attribute

The description of the order in the Packaged Content should not include currency and amount information since this is contained in the payment related trading components (Brand List, Brand Selection and Payment).

For interoperability, implementations must support Plain Text as a minimum so that it can be easily displayed.

5.4.2 OkFrom and OkTo Timestamps

Note that:

- the `OkFrom`date may be later than the `OkFrom`date on the Payment Component (see section 5.8) associated with this order, and
- similarly, the `OkTo` date may be earlier that the `OkTo` date on the Payment Component (see section 5.8).

[Note] *Disclaimer. The following information provided in this note does not represent formal advice of the Open Trading Protocol Consortium, any of its members or the authors of this specification. Readers of this specification must form their own views and seek their own legal counsel on the usefulness and applicability of this information.*

The merchant in the context of Internet commerce with anonymous consumers initially frames the terms of the offer on the web page, and in order to obtain the good or service, the consumer must accept them.

If there is to be a time-limited offer, it recommended that merchants communicate this to the consumer and state in the order description in a manner which is clear to the consumer that:
- the offer is time limited

- *the OkFrom and OkTo timestamps specify the validity of the offer*
- *the clock, e.g. the merchant's clock, that will be used to determine the validity of the offer*

[Note End]

5.5 Organisation Component

The Organisation Component provides information about an individual or an organisation. This can be used for a variety of purposes. For example:

- to describe the merchant who is selling the goods,
- to identify who made a purchase,
- to identify who will take delivery of goods,
- to provide a customer care contact,
- to describe who will be the Payment Handler.

Its definition is as follows.

```
<!ELEMENT Org (TradingRole+, ContactInfo?, PersonName?,
      PostalAddress?)>
<!ATTLIST Org
  ID  ID              #REQUIRED
  xml:lang            NMTOKEN  #REQUIRED
  OrgId               CDATA    #REQUIRED
  OtpMsgIdPrefix      NMTOKEN  #REQUIRED
  LegalName           CDATA    #IMPLIED
  ShortDesc           CDATA    #IMPLIED
  LogoNetLocn         CDATA    #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Organisation Component within the OTP Transaction.
`xml:lang`	Defines the language used by attributes or child elements within this component, unless overridden by an `xml:lang` attribute on a child element. See section 2.9 Identifying Languages.
`OrgId`	A code which identifies the organisation described by the Organisation Component. See 5.5.1.1 Organisation IDs, below.
`OtpMsgIdPrefix`	Contains the prefix which must be used for all OTP Messages sent by the Organisation in this OTP Transaction. The values to be used are defined in 2.4.1 OTP Message ID Attribute Definition.
`LegalName`	For organisations which are companies this is their legal name in the language defined by `xml:lang`. It is required for Organisations who have a Trading Role other than `Consumer` or `DeliverTo`.
`ShortDesc`	A short description of the organisation in the language defined by `xml:lang`. It is typically the name by which the organisation is commonly known. For example, if the legal name was "Blue Meadows Financial Services Inc.". Then its short name would likely be "Blue Meadows".

	It is used to facilitate selecting an individual organisation from a list of organisations, for example from a database of organisations involved in OTP Transactions which has been stored by a consumer.
`LogoNetLocn`	The net location which can be used to download the logo for the organisation. See section 8 Retrieving Logos. The content of this attribute must conform to [RFC1738].

Content:

`TradingRole`	See 5.5.2 Trading Role Element below.
`ContactInfo`	See 5.5.3 Contact Information Element below.
`PersonName`	See 5.5.4 Person Name below.
`PostalAddress`	See 5.5.5 Postal Address below.

5.5.1.1 Organisation IDs

Organisation IDs are used by one OTP Trading Role to identify another. In order to avoid confusion, this means that these IDs must be globally unique.

In principle this is achieved in the following way:

- the Organisation Id for all trading roles, apart from the Consumer Trading Role, uses a domain name as their globally unique identifier,
- the Organisation Id for a Consumer Trading Role is allocated by one of the other Trading Roles in an OTP Transaction and is made unique by concatenating it with that other roles' Organisation Id,
- once a Consumer is allocated an Organisation Id within an OTP Transaction the same Organisation Id is used by all the other trading roles in that OTP transaction to identify that Consumer.

Specifically, the content of the Organisation ID is defined as follows:

```
OrgId ::= NonConsumerOrgId | ConsumerOrgId
NonConsumerOrgId ::= DomainName
ConsumerOrgId ::= ConsumerOrgIdPrefix (namechar)+ "/"
                                          NonConsumerOrgId
ConsumerOrgIdPrefix ::= "Consumer:"
```

`ConsumerOrgId`	If the Organisation ID for a `Consumer` consists of: • a standard prefix is to identify that the Organisation Id is for a consumer, followed by • one or more characters which conform to the definition of an XML `"namechar"`. See [XML] specifications, followed by • the `NonConsumerOrgId` for the Organisation which allocated the `ConsumerOrgId` It is normally the Merchant role. Use of upper and lower case is significant.
`NonConsumerOrgId`	If the `Role` is not `Consumer` then this contains the Canonical Name for the non-consumer organisation being described by the Organisation

Component. See [DNS].

Note that a `NonConsumerOrgId` may not start with the `ConsumerOrgIdPrefix`.

Use of upper and lower case is not significant.

Examples of Organisation Ids follow:

- `newjerseybooks.com` - a merchant organisation id
- `westernbank.co.uk` - a payment handler organisation id
- `consumer:1000247ABH/newjerseybooks.com` a consumer organisation id allocated by a merchant

5.5.2 Trading Role Element

This identifies the Trading Role of an individual or organisation in the OTP Transaction. Note, an organisation may have more than one Trading Role and several roles may be present in one organisation element. Its definition is as follows:

```
<!ELEMENT TradingRole EMPTY >
<!ATTLIST TradingRole
  TradingRole        NMTOKEN  #REQUIRED
  ErrorNetLocn       CDATA    #IMPLIED >
```

Attributes:

`TradingRole`

The trading role of the organisation. Valid values are:

- `Consumer`. The person or organisation that is acting in the role of a consumer in the OTP Transaction.
- `Merchant`. The person or organisation that is acting in the role of merchant in the OTP Transaction.
- `PaymentHandler`. The financial institution or other organisation which is a Payment Handler for the OTP Transaction
- `DeliveryHandler`. The person or organisation that is the delivering the goods or services for the OTP Transaction
- `DelivTo`. The person or organisation that is receiving the delivery of goods or services in the OTP Transaction
- `CustCare`. The organisation and/or individual who will provide customer care for an OTP Transaction.
- `x-ddd:nnn` a user defined role (see section 2.7.3 User Defined Codes).

`ErrorNetLocn`

The net location to which OTP messages containing Error Components with a `Severity` of either `HardError` or `TransientError` are sent. See section 5.17.1 Error Processing Guidelines for more details.

This attribute must be set when `TradingRole` is set to `PaymentHandler` or `DeliveryHandler`.

The content of this attribute is dependent on the Transport Mechanism see the Transport Mechanism Supplement.

5.5.3 Contact Information Element

This contains information which can be used to contact an organisation or an individual. All attributes are optional however at least one item of contact information should be present. Its definition is as follows.

```
<!ELEMENT ContactInfo EMPTY >
<!ATTLIST ContactInfo
  xml:lang           NMTOKEN  #IMPLIED
  Tel                CDATA    #IMPLIED
  Fax                CDATA    #IMPLIED
  Email              CDATA    #IMPLIED
  NetLocn            CDATA    #IMPLIED >
```

Attributes:

`xml:lang`	Defines the language used by attributes within this element. See section 2.9 Identifying Languages.
`Tel`	A telephone number by which the organisation may be contacted. Note that this is a text field and no validation is carried out on it.
`Fax`	A fax number by which the organisation may be contacted. Note that this is a text field and no validation is carried out on it.
`Email`	An email address by which the organisation may be contacted. Note that this field should conform to the conventions for address specifications contained in [RFC822].
`NetLocn`	A location on the Internet by which information about the organisation may be obtained that can be displayed using a web browser. The content of this attribute must conform to [RFC1738].

5.5.4 Person Name Element

This contains the name of an individual person. All fields are optional however as a minimum either the `GivenName` or the `FamilyName` should be present. Its definition is as follows.

```
<!ELEMENT PersonName EMPTY >
<!ATTLIST PersonName
  xml:lang           NMTOKEN  #IMPLIED
  Title              CDATA    #IMPLIED
  GivenName          CDATA    #IMPLIED
  Initials           CDATA    #IMPLIED
  FamilyName         CDATA    #IMPLIED >
```

Attributes:

`xml:lang`	Defines the language used by attributes within this element. See section 2.9 Identifying Languages.
`Title`	A distinctive name; personal appellation, hereditary or not, denoting or implying office (e.g. judge, mayor) or nobility (e.g. duke, duchess, earl), or used in addressing or referring to a person (e.g. Mr, Mrs, Miss)
`GivenName`	The primary or main name by which a person is known amongst and identified by their family, friends and acquaintances. Otherwise known as first name or Christian Name.

`Initials`	The first letter of the secondary names (other than the Given Name) by which a person is known amongst or identified by their family, friends and acquaintances.
`FamilyName`	The name by which family of related individuals are known. It is typically the part of an individual's name which is passed on by parents to their children.

5.5.5 Postal Address Element

This contains an address which can be used, for example, for the physical delivery of goods, services or letters. Its definition is as follows.

```
<!ELEMENT PostalAddress EMPTY >
<!ATTLIST PostalAddress
  xml:lang           NMTOKEN  #IMPLIED
  AddressLine1       CDATA    #IMPLIED
  AddressLine2       CDATA    #IMPLIED
  CityOrTown         CDATA    #IMPLIED
  StateOrRegion      CDATA    #IMPLIED
  PostalCode         CDATA    #IMPLIED
  Country            CDATA    #IMPLIED
  LegalLocation      (True|False)  'False' >
```

Attributes:

`xml:lang`	Defines the language used by attributes within this element. See section 2.9 Identifying Languages.
`AddressLine1`	The first line of a postal address. e.g. "The Meadows"
`AddressLine2`	The second line of a postal address. e.g. "Sandy Lane"
`CityOrTown`	The city of town of the address. e.g. "Carpham"
`StateOrRegion`	The state or region within a country where the city or town is placed. e.g. "Surrey"
`Country`	The country for the address. e.g. "UK"
`LegalLocation`	This identifies whether the address is the Registered Address for the Organisation. At least one address for the Organisation must have a value set to `True` unless the Trading Role is either `Consumer` or `DeliverTo`.

5.6 Brand List Component

Brand List Components are contained within the Trading Protocol Options Block (see section 6.1) of the OTP Transaction. They contains lists of:

- payment Brands (see also section 2.6 Brands and Brand Selection),
- amounts to be paid in the currencies that are accepted or offered by the Merchant,
- the payment protocols which can be used to make payments with a Brand, and

- the net locations of the Payment Handlers which accept payment for a payment protocol

The definition of a Brand List Component is as follows.

```
<!ELEMENT BrandList (Brand+, ProtocolAmount+,
        CurrencyAmount+, PayProtocol+) >
<!ATTLIST BrandList
  ID  ID              #REQUIRED
  xml:lang            NMTOKEN  #REQUIRED
  ShortDesc           CDATA    #REQUIRED
  PayDirection (Debit | Credit )   #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Brand List Component within the OTP Transaction.
`xml:lang`	Defines the language used by attributes or child elements within this component, unless overridden by an `xml:lang` attribute on a child element. See section 2.9 Identifying Languages.
`ShortDesc`	A text description in the language defined by `xml:Lang` giving details of the purpose of the Brand List. This information must be displayed to the receiver of the Brand List in order to assist with making the selection. It is of particular benefit in allowing a Consumer to distinguish the purpose of a Brand List when an OTP Transaction involves more than one payment.
`PayDirection`	Indicates the direction in which the payment for which a Brand is being selected is to be made. Its values may be: • `Debit` The sender of the Payment Request Block (e.g. the Consumer) to which this Brand List relates will make the payment to the Payment Handler, or • `Credit` The sender of the Payment Request Block to which this Brand List relates will receive a payment from the Payment Handler.

Content:

`Brand`	This describes a Brand. The sequence of the Brand elements (see section 0) within the Brand List does not indicate any preference. It is recommended that software which processes this Brand List presents Brands in a sequence which the receiver of the Brand List prefers.
`ProtocolAmount`	This links a particular Brand to: • the currencies and amounts in `CurrencyAmount` elements that can be used with the Brand, and • the Payment Protocols and Payment Handlers, which can be used with those currencies and amounts, and a particular Brand
`CurrencyAmount`	This contains a currency code and an amount.
`PayProtocol`	This contains information about a Payment Protocol and the Payment Handler which may be used with a particular Brand.

The relationships between the elements which make up the content of the Brand List is illustrated in the diagram below.

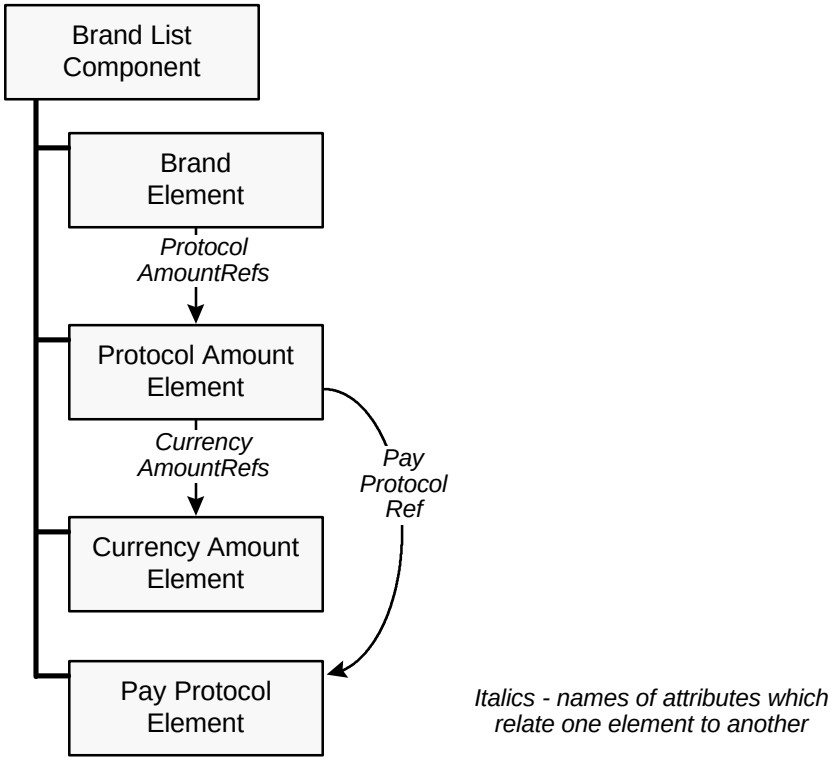


Figure 17 Brand List Element Relationships

Examples of complete Brand Lists are contained in section 9 Brand List Examples.

5.6.1 Brand Element

A Brand Element describes a brand that can be used for making a payment. One or more of these elements is carried in each Brand List Component that has the `PayDirection`attribute set to `Debit`. Exactly one Brand Element may be carried in a Brand List Component that has the `PayDirection` attribute set to `Credit.`

```
<!ELEMENT Brand (PackagedContent?) >
<!ATTLIST Brand
  Id  ID              #REQUIRED
  xml:lang            NMTOKEN  #IMPLIED
  BrandId             NMTOKEN  #REQUIRED
  BrandName           CDATA    #REQUIRED
  BrandLogoNetLocn    CDATA    #REQUIRED
  BrandNarrative      CDATA    #IMPLIED
  ProtocolAmountRefs IDREFS    #REQUIRED
  ContentSoftwareId   CDATA    #IMPLIED >
```

Attributes:

`Id` — Element identifier, potentially referenced in a Brand Selection Component contained in a later Payment Request message and

	uniquely identifies the Brand element within the OTP Transaction.
`xml:lang`	Defines the language used by attributes and content of this element. See section 2.9 Identifying Languages.
`BrandId`	This contains a unique identifier for the brand or promotional brand. It is used to match against a list of Payment Instruments which the Consumer holds to determine whether or not the Consumer can pay with the Brand. The syntax for a `BrandId` is as follows: `BrandId ::= UserDefinedCode | BrandIdDomain ":" BrandValue` Currently the only valid value for the `BrandIdDomain` is `otp` which indicates that the `BrandValue` is registered with OTP. The valid values for `BrandValue` for brands defined within the OTP Brand domain are obtainable from the OTP web site http:www.otp.org. A user defined code follows the conventions defined in section 2.7.3. Uniqueness of a user defined `BrandId` is not guaranteed.
`BrandName`	This contains the name of the brand, for example MasterCard Credit. This is the description of the Brand which is displayed to the consumer in the Consumers language defined by `xml:lang`. For example it might be "American Airlines Advantage Visa". Note that this attribute is not used for matching against the payment instruments held by the Consumer.
`BrandLogoNetLocn`	The net location which can be used to download the logo for the organisation. See section Retrieving Logos (see section 8). The content of this attribute must conform to [RFC1738].
`BrandNarrative`	This optional attribute is designed to be used by the Merchant to indicate some special conditions or benefit which would apply if the Consumer selected that brand. For example "5% discount", "free shipping and handling", "free breakage insurance for 1 year", "double air miles apply", etc.
`ProtocolAmountRefs`	Identifies the protocols and related currencies and amounts which can be used with this Brand. Specified as a list of ID's of Protocol Amount Elements (see section 5.6.2) contained within the Brand List.
`ContentSoftwareId`	This optional attribute contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:lang`. It must contain, as a minimum: • the name of the software manufacturer • the name of the software • the version of the software, and • the build of the software It is recommended that this attribute is included if the software which generated the content cannot be identified from the `SoftwareId` attribute on the Message Id Component (see section 2.3.2)

Content:

`PackagedContent`	Optional Packaged Content (see section 2.8) containing information about the brand which may be used by the payment protocol. The content of this information is defined in the supplement for a payment protocol which describes how the payment protocol works with OTP.

Examples Brand Elements are contained in section 10 Brand List Examples.

5.6.2 Protocol Amount Element

The Protocol Amount element links a Brand to:

- the currencies and amounts in Currency Amount Elements (see section 5.6.3) that can be used with the Brand, and
- the Payment Protocols and Payment Handlers defined in a Pay Protocol Element (see section 5.6.4), which can be used with those currencies and amounts.

Its definition is as follows:

```
<!ELEMENT ProtocolAmount (PackagedContent?) >
<!ATTLIST ProtocolAmount
  Id  ID             #REQUIRED
  PayProtocolRef     IDREF    #REQUIRED
  CurrencyAmountRefs IDREFS   #REQUIRED
  ContentSoftwareId  CDATA    #IMPLIED >
```

Attributes:

`Id`	Element identifier, potentially referenced in a Brand element; or in a Brand Selection Component contained in a later Payment Request message which uniquely identifies the Protocol Amount element within the OTP Transaction.
`PayProtocolRef`	Contains an Element Reference (see section 2.5) that refers to the Pay Protocol Element (see section 5.6.4) that contains the Payment Protocol and Payment Handlers that can be used with the Brand.
`CurrencyamountRefs`	Contains a list of Element References (see section 2.5) that refer to the Currency Amount Element (see section 5.6.3) that describes the currencies and amounts that can be used with the Brand.
`ContentSoftwareId`	This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:lang`. It must contain, as a minimum: • the name of the software manufacturer • the name of the software • the version of the software, and • the build of the software It is recommended that this attribute is included if the software which generated the content cannot be identified from the `SoftwareId` attribute on the Message Id Component (see section 2.3.2)

Content:

`PackagedContent`	Optional Packaged Content (see section 2.8) containing information about the protocol amount which may be used by the payment protocol. The content of this information is defined in the supplement for a payment protocol which describes how the payment protocol works with OTP.

Examples Protocol Amount Elements are contained in9 Brand List Examples.

5.6.3 Currency Amount Element

A Currency Amount element contains:

- a currency code (and its type), and
- an amount.

One or more of these elements is carried in each Brand List Component. Its definition is as follows:

```
<!ELEMENT CurrencyAmount EMPTY >
<!ATTLIST CurrencyAmount
  Id  ID              #REQUIRED
  Amount              CDATA    #REQUIRED
  CurrCodeType        NMTOKEN  'ISO4217'
  CurrCode            CDATA    #REQUIRED >
```

Attributes:

`Id`	Element identifier, potentially referenced in a Brand element; or in a Brand Selection Component contained in a later Payment Request message which uniquely identifies the Currency Amount Element within the OTP Transaction.
`Amount`	Indicates the amount to be paid in whole and fractional units of the currency. For example $245.35 would be expressed `"245.35"`. Note that values smaller than the smallest denomination are allowed. For example one tenth of a cent would be `"0.001"`.
`CurrCodeType`	Indicates the domain of the `CurrCode`. This attribute is included so that the currency code may support non-standard "currencies" such as frequent flyer points, trading stamps, etc. Its values may be: • `ISO4217` indicates the currency code conforms to [ISO 4217] • `x-ddd:nnn`a user defined currency code type (see section 2.7.3 User Defined Codes).
`CurrCode`	A code which identifies the currency to be used in the payment. The domain of valid currency codes is defined by `CurrCodeType`

Note that `Amount`, `CurrCodeType`and `CurrCode`have identical meanings to the attributes of the same name on the Payment Component (see section 5.8).

Examples Currency Amount Elements are contained in 9 Brand List Examples.

5.6.4 Pay Protocol Element

A Pay Protocol element specifies details of a Payment Protocol and the Payment Handler that can be used with a Brand. One or more of these elements is carried in each Brand List.

```
<!ELEMENT PayProtocol (PackagedContent?) >
<!ATTLIST PayProtocol
  Id  ID              #REQUIRED
  xml:lang            NMTOKEN  #IMPLIED
  ProtocolId          NMTOKEN  #REQUIRED
  ProtocolName        CDATA    #REQUIRED
  ActionOrgRef        NMTOKEN  #REQUIRED
  PayReqNetLocn       CDATA    #IMPLIED
  SecPayReqNetLocn    CDATA    #IMPLIED
  ContentSoftwareId   CDATA    #IMPLIED >
```

Attributes:

`Id`	Element identifier, potentially referenced in a Brand element; or in a Brand Selection Component contained in a later Payment Request message which uniquely identifies the Pay Protocol element within the OTP Transaction.
`xml:lang`	Defines the language used by attributes and content of this element. See section 2.9 Identifying Languages.
`ProtocolId`	Consists of a protocol name and version. For example `"SETv1.0"`. The value used for the `ProtocolId`is defined in the payment supplement for the payment method.
`ProtocolName`	A narrative description of the payment protocol and its version in the language identified by `xml:lang`. For example "Secure Electronic Transaction Version 1.0". Its purpose is to help provide information on the payment protocol being used if problems arise.
`ActionOrgRef`	An Element Reference (see section 2.5) to the Organisation Component for the Payment Handler for the Payment Protocol.
`PayReqNetLocn`	The Net Location indicating where an unsecured Payment Request message should be sent if this protocol choice is used. The content of this attribute is dependent on the Transport Mechanism (such must conform to [RFC1738].
`SecPayReqNetLocn`	The Net Location indicating where a secured Payment Request message should be sent if this protocol choice is used. A secured payment involves the use of a secure channel such as [SSL] in order to communicate with the Payment Handler. The content of this attribute must conform to [RFC1738]. See also See section 2.10 Secure and Insecure Net Locations.
`ContentSoftwareId`	This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:`lang. It must contain, as a minimum: • the name of the software manufacturer

- the name of the software
- the version of the software, and
- the build of the software

It is recommended that this attribute is included if the software which generated the content cannot be identified from the `SoftwareId` attribute on the Message Id Component (see section 2.3.2)

Content:

`PackagedContent` — Optional Packaged Content information (see section 2.8) about the protocol which is used by the payment protocol. The content of this information is defined in the supplement for a payment protocol which describes how the payment protocol works with OTP. An example of its use could be to include a payment protocol message.

Examples Pay Protocol Elements are contained in section 5.6 Brand List Component.

5.7 Brand Selection Component

A Brand Selection Component identifies the choice of payment brand, payment protocol and the Payment Handler. This element is used:

- in Payment Request messages within Baseline Purchase and Baseline Value OTP Transactions to identify the brand, protocol and payment handler for a payment, or
- to, optionally, inform a merchant in a purchase of the payment brand being used so that the offer and order details can be amended accordingly.

In Baseline OTP, the integrity of Brand Selection Components is not guaranteed. However, modification of Brand Selection Components can only cause denial of service if the payment protocol itself is secure against message modification, duplication, and swapping attacks.

The definition of a Brand Selection Component is as follows.

```
<!ELEMENT BrandSelection (BrandSelBrandInfo?,
        BrandSelProtocolAmountInfo?, BrandSelCurrencyAmountInfo?)
      >
<!ATTLIST BrandSelection
  ID  ID              #REQUIRED
  BrandListRef        NMTOKEN  #REQUIRED
  BrandRef            NMTOKEN  #REQUIRED
  ProtocolAmountRef   NMTOKEN  #REQUIRED
  CurrencyAmountRef   NMTOKEN  #REQUIRED >
```

Attributes:

`ID` — An identifier which uniquely identifies the Brand Selection Component within the OTP Transaction.

`BrandListRef` — The Element Reference (see section 2.5) of the Brand List Component from which a Brand is being selected

`BrandRef` — The Element Reference of a Brand element within the Brand List Component that is being selected that is to be used in the payment.

`ProtocolAmountRef`	The Element Reference of a Protocol Amount element within the Brand List Component which is to be used when making the payment.
`CurrencyAmountRef`	The Element Reference of a Currency Amount element within the Brand List Component which is to be used when making the payment.

Content:

`BrandSelBrandInfo`, `BrandSelProtocolAmountInfo`, `BrandSelCurrencyAmountInfo`	This contains any additional data that may be required by a particular payment brand or protocol. See sections 5.7.1, 5.7.2, and 5.7.3.

The following rules apply:

- the `BrandListRef`must contain the ID of a Brand List Component in the same OTP Transaction
- every Brand List Component in the Trading Protocol Options Block must be referenced by one and only one Brand Selection Component
- the `BrandRef`must refer to the ID of a Brand contained within the Brand List Component referred to by `BrandListRef`
- the `ProtocolAmountRef`must refer to one of the Element IDs listed in the `ProtocolAmountRefs`attribute of the Brand element identified by `BrandRef`
- the `CurrencyAmountRef`must refer to one of the Element IDs listed in the `CurrencyAmountRefs`attribute of the Protocol Amount Element identified by `ProtocolAmountRef`.

An example of a Brand Selection Component is included in 9 Brand List Examples.

5.7.1 Brand Selection Brand Info Element

The Brand Selection Brand Info Element contains any additional data that may be required by a particular payment brand. See the OTP payment method supplement for a description of how and when it used.

```
<!ELEMENT BrandSelBrandInfo (PackagedContent) >
<!ATTLIST BrandSelBrandInfo
  Id  ID              #REQUIRED
  ContentSoftwareId  CDATA    #IMPLIED >
```

Attributes:

`ContentSoftwareId`	This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:`lang. It must contain, as a minimum: • the name of the software manufacturer • the name of the software • the version of the software, and • the build of the software It is recommended that this attribute is included if the software which generated the content cannot be identified from the `SoftwareId`

attribute on the Message Id Component (see section 2.3.2)

Content:

`PackagedContent` Packaged Content information (see section 2.8) that contains additional data that may be required by a particular payment brand. See the payment method supplement for OTP for rules on how this is used.

5.7.2 Brand Selection Protocol Amount Info Element

The Brand Selection Protocol Amount Info Element contains any additional data that is payment protocol specific that may be required by a particular payment brand or payment protocol. See the OTP payment method supplement for a description of how and when it used.

```
<!ELEMENT BrandSelProtocolAmountInfo (PackagedContent) >
<!ATTLIST BrandSelProtocolAmountInfo
  Id  ID              #REQUIRED
  ContentSoftwareId  CDATA    #IMPLIED >
```

Attributes:

`ContentSoftwareId` See section 5.7.1 Brand Selection Brand Info Element.

Content:

`PackagedContent` Packaged Content information (see section 2.8) that contains any additional data that may be required by a particular payment brand. See the payment method supplement for OTP for rules on how this is used.

5.7.3 Brand Selection Currency Amount Info Element

The Brand Selection Currency Amount Info Element contains any additional data that is payment brand and currency specific that may be required by a particular payment brand. See the OTP payment method supplement for a description of how and when it used.

```
<!ELEMENT BrandSelCurrencyAmountInfo (PackagedContent) >
<!ATTLIST BrandSelCurrencyAmountInfo
  Id  ID              #REQUIRED
  ContentSoftwareId  CDATA    #IMPLIED >
```

Attributes:

`ContentSoftwareId` See section 5.7.1 Brand Selection Brand Info Element.

Content:

`PackagedContent` Packaged Content information (see section 2.8) that contains any additional data relating to the payment brand and currency. See the payment method supplement for OTP for rules on how this is used.

5.8 Payment Component

A Payment Component contains information used to control how a payment is carried out. Its provides information on:

- the times within which a Payment with a Payment Handler may be started
- a reference to the Brand List (see section 5.6) which identifies the Brands, protocols, currencies and amounts which can be used to make a payment
- whether or not an authentication of the consumer is required as part of the payment
- whether or not a payment receipt will be provided
- whether another payment precedes this payment.

Its definition is as follows.

```
<!ELEMENT Payment (PackagedContent?) >
<!ATTLIST Payment
  ID  ID              #REQUIRED
  OkFrom              CDATA    #REQUIRED
  OkTo                CDATA    #REQUIRED
  BrandListRef        NMTOKEN  #REQUIRED
  SignedPayReceipt    ('True'|'False') #REQUIRED
  AuthDataRef         NMTOKEN  #IMPLIED
  StartAfter          NMTOKENS #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Payment Component within the OTP Transaction.
`OkFrom`	The date and time in [UTC] format after which a Payment Handler may accept for processing a Payment Request Block (see section 6.6) containing the Payment Component.
`OkTo`	The date and time in [UTC] format before which a Payment Handler may for processing accept a Payment Request Block containing the Payment Component.
`BrandListRef`	An Element Reference (see section 2.5) of a Brand List Component (see section 5.6) within the TPO Trading Block for the OTP Transaction. The Brand List identifies the alternative ways in which the payment can be made.
`AuthDataRef`	An element reference (see section 2.4) of an Authentication Data Component (see section 5.2) which is to be used for authentication of the Trading Role which sends the Payment Request Block containing the Payment Component to the Payment Handler. If not present, then no authentication is to take place.
`SignedPayReceipt`	Indicates whether or not the Payment Response Block (6.8) generated by the payment handler for the payment must be digitally signed.
`StartAfter`	Contains Element References (see section 2.5) of other Payment Components which describe payments which must be complete before this payment can start. If no `StartAfter` attribute is present then there are no dependencies and the payment can start immediately

Contents

`PackagedContent`	This optional element contains "user" data defined by the Merchant which is required by the Payment Handler. See section 2.8 Packaged Content Element.

5.9 Payment Scheme Component

A Payment Scheme Component contains payment protocol information for a specific payment scheme which is transferred between the parties involved in a payment for example a [SET] message. Its definition is as follows.

```
<!ELEMENT PaySchemeData (PackagedContent) >
<!ATTLIST PaySchemeData
  ID  ID             #REQUIRED
  ConsumerPaymentId  CDATA    #IMPLIED
  PaymentHandlerPayId CDATA   #IMPLIED
  ContentSoftwareId  CDATA    #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Payment Scheme Component within the OTP Transaction.
`ConsumerPaymentId`	An identifier specified by the Consumer which, if returned by the Payment Handler in another Payment Scheme Component or by other means, will enable the Consumer to identify which payment is being referred to.
`PaymentHandlerPayId`	An identifier specified by the Payment Handler which, if returned by the Consumer in another Payment Scheme Component, or by other means, will enable the Payment Handler to identify which payment is being referred to. It is required on every Payment Scheme Component apart from the one contained in a Payment Request Block.
`ContentSoftwareId`	This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:lang`. It must contain, as a minimum: • the name of the software manufacturer • the name of the software • the version of the software, and • the build of the software It is recommended that this attribute is included if the software which generated the content cannot be identified from the `SoftwareId` attribute on the Message Id Component (see section 2.3.2)

Content:

`PackagedContent`	Contains the payment scheme protocol information as Packaged Content (see section 2.8). See the payment scheme supplement for the definition of its content.

5.10 Payment Receipt Component

A Payment Receipt Component contains payment scheme specific data which can be used after the payment has completed to verify the payment occurred. Its definition is as follows.

```
<!ELEMENT PayReceipt (PackagedContent) >
<!ATTLIST PayReceipt
  ID  ID              #REQUIRED
  PaymentRef          NMTOKEN  #REQUIRED
  ContentSoftwareId   CDATA    #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Payment Receipt Component within the OTP Transaction.
`PaymentRef`	Contains an Element Reference (see section 2.5) to the Payment Component (see section 5.8) to which this payment receipt applies
`ContentSoftwareId`	This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:lang`. It must contain, as a minimum: • the name of the software manufacturer • the name of the software • the version of the software, and • the build of the software It is recommended that this attribute is included if the software which generated the content cannot be identified from the `SoftwareId` attribute on the Message Id Component (see section 2.3.2)

Content:

`PackagedContent`	Contains the payment scheme specific record of the payment which can be used for receipt purposes as Packaged Content (see section 2.8). Each payment scheme defines in its supplement the structure of the content.

5.11 Delivery Component

The Delivery Element contains information required to deliver goods or services. Its definition is as follows.

```
<!ELEMENT Delivery (DeliveryData?, PackagedContent?) >
<!ATTLIST Delivery
  ID  ID              #REQUIRED
  xml:lang            NMTOKEN  #REQUIRED
  DelivExch (True|False)      #REQUIRED
  DelivAndPayResp (True|False) #REQUIRED
  ActionOrgRef        NMTOKEN  #IMPLIED
  ConsumerDeliveryId CDATA    #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Delivery Component within the OTP Transaction.
`xml:lang`	Defines the language used by attributes or child elements within this component, unless overridden by an `xml:lang` attribute on a child element. See section 2.9 Identifying Languages.
`DelivExch`	Indicates if this OTP Transaction includes the messages associated with a Delivery Exchange. Valid values are: • `True` indicates it does include a Delivery Exchange • `False` indicates it does not include a Delivery Exchange If set to true then a `DeliveryData` element must be present. If set to false it may be absent.
`DelivAndPayResp`	Indicates if the Delivery Response Block (see section 6.10) and the Payment Response Block (see section 6.8) are combined into one OTP Message. Valid values are: • `True` indicates both blocks will be in the same OTP Message, and • `False` indicates each block will be in a different OTP Message `DelivAndPayResp` should not be true if `DelivExch` is `False`. In practice combining the Delivery Response Block and Payment Response Block is only likely to be practical if the Merchant, the Payment Handler and the Delivery Handler are the same organisation since: • the Payment Handler must have access to Order Component information so that they know what to deliver, and • the Payment Handler must be able to carry out the delivery
`ActionOrgRef`	An Element Reference to the Organisation Component of the Delivery Handler for this delivery.
`ConsumerDeliveryId`	An identifier specified by the Consumer which, if returned by the Delivery Handler in another Delivery Component, or by other means, will enable the Consumer to identify which Delivery is being referred to.

Content:

`DeliveryData`	Contains details about how the delivery will be carried out. See 5.11.1 Delivery Data Element below.
`PackagedContent`	This optional element contains "user" data defined for the Merchant which is required by the Delivery Handler. See section 2.8 Packaged Content Element.

5.11.1 Delivery Data Element

The `DeliveryData` element contains information about where and how goods are to be delivered. Its definition is as follows.

```
<!ELEMENT DeliveryData (PackagedContent?) >
<!ATTLIST DeliveryData
  xml:lang           NMTOKEN  #IMPLIED
  OkFrom             CDATA    #REQUIRED
```

```
  OkTo                CDATA    #REQUIRED
  DelivMethod         NMTOKEN  #REQUIRED
  DelivToRef          NMTOKEN  #REQUIRED
  DelivReqNetLocn     CDATA    #REQUIRED
  SecDelivReqNetLocn CDATA     #REQUIRED
  ContentSoftwareId   CDATA    #IMPLIED >
```

Attributes:

`xml:lang`	Defines the language used by attributes within this component. See section 2.9 Identifying Languages.
`OkFrom`	The date and time in [UTC] format after which the Delivery Handler may accept for processing a Delivery Request Block (see section 6.9).
`OkTo`	The date and time in [UTC] format before which the Delivery Handler may accept for processing a Delivery Request Block.
`DelivMethod`	Indicates the method by which goods or services may be delivered. Valid values are: • `Post` the goods will be delivered by post or courier • `Web` the goods will be delivered electronically in the Delivery Note Component • `Email` the goods will be delivered electronically by e-mail • `x-ddd:nnna` user defined delivery method see 2.7.3 User Defined Codes.
`DelivToRef`	The Element Reference (see section 2.4) of an Organisation Component within the OTP Transaction which has a role of `DelivTo`. The information in this block is used to determine where delivery is to be made. It must be compatible with `DelivMethod` Specifically if the `DelivMethod` is: • `Post`, then the there must be a Postal Address Element containing sufficient information for a postal delivery, • `Web`, then there are no specific requirements. The information will be sent in a web page back to the Consumer • `Email`, then there must be Contact Information Element with a valid e-mail address
`DelivReqNetLocn`	This contains the Net Location to which an unsecured Delivery Request Block (see section 6.9) which contains the Delivery Component should be sent. The content of this attribute is dependent on the Transport Mechanism must conform to [RFC1738].
`SecDelivReqNetLocn`	This contains the Net Location to which a secured Delivery Request Block (see section 6.9) which contains the Delivery Component should be sent. A secured delivery request involves the use of a secure channel such as [SSL] in order to communicate with the Payment Handler. The content of this attribute is dependent on the Transport Mechanism must conform to [RFC1738]. See also Section 2.10 Secure and Insecure Net Locations.
`ContentSoftwareId`	This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve

interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:lang`. It must contain, as a minimum:

- the name of the software manufacturer
- the name of the software
- the version of the software, and
- the build of the software

It is recommended that this attribute is included if the software which generated the content cannot be identified from the `SoftwareId` attribute on the Message Id Component (see section 2.3.2)

Content:

`PackagedContent` Optional additional information about the delivery as Packaged Content (see section 2.8). provided to the Delivery Handler by the merchant.

5.12 Delivery Note Component

A Delivery Note contains delivery instructions about the delivery of goods or services or potentially the actual Delivery Information itself. It is information which the person or organisation receiving the Delivery Note can use when delivery occurs.

```
<!ELEMENT DeliveryNote (PackagedContent) >
<!ATTLIST DeliveryNote
  ID  ID              #REQUIRED
  xml:lang            NMTOKEN  #REQUIRED
  DelivHandlerDelivId         CDATA    #IMPLIED
  ContentSoftwareId  CDATA    #IMPLIED >
```

Attributes:

`ID` An identifier which uniquely identifies the Delivery Note Component within the OTP Transaction.

`xml:lang` Defines the language used by attributes or child elements within this component, unless overridden by an `xml:lang` attribute on a child element. See section 2.9 Identifying Languages.

`DelivHandlerDelivId` An optional identifier specified by the Delivery Handler which, if returned by the Consumer in another Delivery Component, or by other means, will enable the Delivery Handler to identify which Delivery is being referred to. It is required on every Delivery Component apart from the one contained in a Delivery Request Block.

An example use of this attribute is to contain a delivery tracking number.

`ContentSoftwareId` This contains information which identifies the software which generated the content of the element. Its purpose is to help resolve interoperability problems that might occur as a result of incompatibilities between messages produced by different software. It is a single text string in the language defined by `xml:lang`. It must contain, as a minimum:

- the name of the software manufacturer

- the name of the software
- the version of the software, and
- the build of the software

It is recommended that this attribute is included if the software which generated the content cannot be identified from the `SoftwareId` attribute on the Message Id Component (see section 2.3.2)

Content:

`DeliveryNote` — Contains the actual delivery note information as Packaged Content (see section 2.8).

[Note] *If the content of the Delivery Message is a Mime message then the Delivery Note may trigger an application which causes the actual delivery to occur.*
[Note End]

5.13 Payment Method Information Component

A Payment Method Information Component contains data which describes the Payment Method which initiated the Payment Instrument Customer Care Transaction and the software which generated the message. Its definition is as follows.

```
<!ELEMENT PayMethodInfoData EMPTY >
<!ATTLIST PayMethodInfoData
  ID  ID              #REQUIRED
  BrandId             NMTOKEN  #REQUIRED
  PayProtocolId       NMTOKEN  #IMPLIED >
```

Attributes:

`ID` — An identifier which uniquely identifies the Payment Method Information Component within the OTP Transaction.

`BrandId` — The Brand Identifier attribute copied from the `BrandId` attribute of the Brand Element (see section 5.6.1)of the Payment Instrument which needs customer care.

`PayProtocolId` — The `ProtocolId` attribute copied from the Pay Protocol Element (see section 5.6.4) of the Brand being used. This may not be required for all types of Payment Instrument. See the OTP Supplement for the Payment Method to determine if this is to be used.

5.14 Status Component

A Status Component contains status information about the business success or failure (see section 3.2) of a process.

Its definition is as follows.

```
<!ELEMENT Status EMPTY >
<!ATTLIST Status
```

```
  ID  ID              #REQUIRED
  xml:lang            NMTOKEN  #REQUIRED
  StatusType (Offer|Payment|Delivery) #REQUIRED
  ElRef               NMTOKEN  #REQUIRED
  ProcessState (NotYetStarted|InProgress|
      CompletedOk|Failed|ProcessError) #REQUIRED
  CompletionCode      NMTOKEN  #IMPLIED
  ProcessReference    CDATA    #IMPLIED
  StatusDesc          CDATA    #IMPLIED >
```

Attributes:

`ID`

An identifier which uniquely identifies the Status Component within the OTP Transaction.

`xml:lang`

Defines the language used by attributes within this component. See section 2.9 Identifying Languages.

`StatusType`

Indicates the type of process which the Status is reporting on. It may be set to either `Offer`, `Payment` or `Delivery`.

`ElRef`

Contains an Element Reference (see section 2.5) to the Component for which the Status is being described. It must refer to either:

- a Trading Protocol Options Block (see section 6.1), if the `StatusType` is `Offer`,
- a Payment Component (see section 5.8), if the `StatusType` is `Payment`, or
- a Delivery Component (see section 5.11), if the `StatusType` is `Delivery`.

`ProcessState`

Contains a State Code which indicates the current state of the process being carried out. Valid values for `ProcessState` are:

- `NotYetStarted` A Request Block has been received but the process has not yet started
- `InProgress`. Processing of the Request Block has started but it is not yet complete
- `CompletedOk` The processing of the Request Block has completed successfully without any errors
- `Failed`. The processing of the Request Block has failed because of a business error (see section 3.2)
- `ProcessError`. This value is only used when the Status Component is being used in connection with an Inquiry Request Trading Block (see section 6.14). It indicates there was a Technical Error (see section 3.1) in the Request Block which is being processed or some internal processing error.

Note that this code reports on the processing of a Request Block. Further, asynchronous processing may occur after the Response Block associated with the Process has been sent to the Consumer.

`CompletionCode`

Indicates how the process completed. Valid values for the `CompletionCode` are given below together with the conditions when it must be present.

A `CompletionCode` is a maximum of 14 characters long.

`ProcessReference`

This optional attribute holds a reference for the process whose status is being reported. It may hold the following values:

- when `StatusType` is set to `Offer`, it should contain the

	`OrderIdentifier` from the Order Component • when `StatusType` is set to `Payment`, it should contain the `PaymentHandlerPayId` from the Payment Scheme Data Component • when `StatusType` is set to `Delivery`, it should contain the `DelivHandlerDelivId` from the Delivery Note Component This attribute should be absent in the Inquiry Request message when the Consumer has not been given such a reference number by the OTP Service Provider. This attribute can be used in an Inquiry Response Block (see section 6.15) to give the Consumer the reference number for a transaction which has previously been unavailable. For example, the package tracking number might not be assigned at the time of a delivery response was received. However, if the Consumer issues a Baseline Transaction Status Inquiry later, the Delivery Handler can put the package tracking number into this attribute in the Inquiry Response message and send it back to the Consumer.
`StatusDesc`	An optional textual description of the current status of the payment in the language identified by `xml:lang`.

5.14.1.1 Offer Completion Codes

The Completion Code is only required if the `ProcessState` attribute is set to `Failed`. The following table contains the valid values for the `CompletionCode` that may be used. It is recommended that the `StatusDesc` attribute is used to provide further explanation where appropriate.

Value	Description
`AuthError`	**Authentication Error**. The authentication check which was carried out has failed.
`OfferDecl`	**Offer Declined**. The Merchant declines to generate an offer for some reason.
`Unspecified`	**Unspecified error**. There is some unknown problem or error which does not fall into one of the other `CompletionCodes`

5.14.1.2 Payment Completion Codes

The `CompletionCode` is only required if the `ProcessState` attribute is set to `Failed`. The following table contains the valid values for the `CompletionCode` that may be used. It is recommended that the `StatusDesc` attribute is used by individual payment schemes to provide further explanation where appropriate.

Value	Description
`BrandNotSupp`	**Brand not supported**. The payment brand is not supported by the Payment Handler.
`CurrNotSupp`	**Currency not supported**. The currency in which the payment is to be made is not supported by either the Payment Instrument or the Payment Handler.
`AuthError`	**Authentication Error**. The Payment Scheme specific authentication check which was carried out has failed.

InsuffFunds	**Insufficient funds**. There are insufficient funds available for the payment to be made.
InstBrandInvalid	**Payment Instrument not valid for Brand**. A Payment Instrument is being used which does not correspond with the Brand selected. For example a Visa credit card is being used when MasterCard was selected as the Brand.
PaymntDecl	**Payment declined**. The Payment Handler declines to accept the payment for some reason.
InstNotValid	**Payment instrument not valid for trade**. The Payment Instrument cannot be used for the proposed type of trade, for some reason.
BadInstrumenat	**Bad instrument**. There is a problem with the Payment Instrument being used which means that it is unable to be used for the payment.
Unspecified	**Unspecified error**. There is some unknown problem or error which does not fall into one of the other CompletionCodes The StatusDesc attribute should provide the explanation of the cause.

5.14.1.3 Delivery Completion Codes

The following table contains the valid values for the CompletionCodeattribute for a Delivery. It is recommended that the StatusDescattribute is used to provide further explanation where appropriate.

Value	Description
BackOrdered	**Back Ordered**. The goods to be delivered are on order but they have not yet been received. Shipping will be arranged when they are received. This is only valid if ProcessStateis CompletedOk
PermNotAvail	**Permanently Not Available**. The goods are permanently unavailable and cannot be re-ordered. This is only valid if ProcessStateis Failed.
TempNotAvail	**Temporarily Not Available**. The goods are temporarily unavailable and may become available if they can be ordered. This is only valid if ProcessStateis CompletedOk
ShipPending	**Shipping Pending**. The goods are available and are scheduled for shipping but they have not yet been shipped. This is only valid if ProcessStateis CompletedOk
Shipped	**Goods Shipped**. The goods have been shipped. Confirmation of delivery is awaited. This is only valid if ProcessStateis CompletedOk
ShippedNoConf	**Shipped - No Delivery Confirmation**. The goods have been shipped but it is not possible to confirm delivery of the goods. This is only valid if ProcessStateis CompletedOk
Confirmed	**Confirmed**. All goods have been delivered and confirmation of their delivery has been received. This is only valid if ProcessStateis CompletedOk

5.15 Inquiry Type Component

The Inquiry Component contains the information which indicates what type of process is being inquired upon.

```
<!ELEMENT InquiryType EMPTY >
<!ATTLIST InquiryType
  ID  ID              #REQUIRED
  Type (Offer|Payment|Delivery) #REQUIRED
  ElRef               NMTOKEN  #IMPLIED
  ProcessReference    CDATA    #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Inquiry Type Component within the OTP Transaction.
`Type`	Contains the type of inquiry. Valid values for `Type` are: • `Offer`. The inquiry is about the status of an offer and is addressed to the Merchant. • `Payment`. The inquiry is about the status of a payment and is addressed to the Payment Handler. • `Delivery`. The inquiry is about the status of a delivery and addressed to the Delivery Handler.
`ElRef`	Contains an Element Reference (see section 2.5) to the component to which this Inquiry Type Component applies. That is, • TPO Block when `Type` is `Offer` • Payment Component when `Type` is `Payment` • Delivery Component when `Type` is `Delivery`
`ProcessReference`	Optionally contains a reference to the process being inquired upon. It should be set if the information is available. For the definition of the values it may contain, see the `ProcessReference` attribute of the Status Component (see section 5.14).

5.16 Signature Component

Each Signature Component digitally signs one or more Blocks or Components including other Signature Components.

For a general explanation of signatures see section 4 Security Considerations. Detailed definitions of the XML structures for signatures is described in the paper "Digital Signature for XML - Proposal", see [XMLSIG].

The Signature Component:

- hashes one or more Blocks or Components in one or more OTP Messages within the same OTP Transaction
- concatenates these hashes into a Signed Data element, and
- signs the `SignedData` element using the optional certificate identified in the `CertRef` attribute of the Digital Signature element.

Note that a Signed Data Element may be signed by more than one Digital Signature element.

A Signature Component can be one of two types either:

- an Offer Response Signature, or
- a Payment Response Signature

How these signatures are constructed is described below

5.16.1 Offer Response Signature Component

The Signed Data Element of the Offer Response Signature Component should contain hashes of the following Components:

- the Transaction Id Component (see section 2.3.1) of the OTP message that contains the Offer Response Signature
- the Transaction Reference Block (see section 2.3) of the OTP Message that contains the Offer Response Signature
- from the TPO Block:
  - the Protocol Options Component
  - each of the Organisation Components
  - each of the Brand List Components
- optionally, all the Brand Selection Components if they were sent to the Merchant in a TPO Selection Block
- from the Offer Response Block:
  - the Order Component
  - each of the Payment Components
  - the Delivery Component
  - each of the Authentication Data Components

The Offer Response Signature Component should also contain Digital Signature Elements for each of the organisations that may or will need to verify the signature. This involves:

- if the Merchant has received a TPO Selection Block containing Brand Selection Components, then generate a Digital Signature Element for the Payment Handler identified by the Brand Selection Component and the Delivery Handler identified by the Delivery Component. See section 4.3.1 Check the Action Request was sent to the Correct Organisation for a description of how this can be done.
- if the Merchant is not expecting to receive a TPO Selection Block then generate a Digital Signature Element for the Delivery Handler and all the Payment Handlers that are involved.

5.16.2 Payment Receipt Signature Component

The Signed Data Element of the Payment Receipt Signature Component should contain hashes of the following Components:

- the Transaction Id Component (see section 2.3.1) of the OTP message that contains the Payment Receipt Signature

- the Transaction Reference Block (see section 2.3) of the OTP Message that contains the Payment Receipt Signature
- the Offer Response Signature Component
- the Payment Receipt Component
- the Status Component
- the Brand Selection Component.

5.16.3 Ping Signature Components

If the Ping Response Transaction is generating a signature (see section 7.9), the Signed Data Element of the Ping Response or Ping Request Signature Components should contain hashes of the following Components:

- all the Organisation Components.

5.17 Error Component

The Error Component contains information about Technical Errors (see section 3.1) in an OTP Message which has been received by one of the Trading Roles involved in the trade.

For clarity two phrases are defined which are used in the description of an Error Component:

- *message in error*. An OTP message which contains or causes an error of some kind
- *message reporting the error*. An OTP message that contains an Error Component that describes the error found in a *message in error.*

The definition of the Error Component is as follows.

```
<!ELEMENT ErrorComp (ErrorLocation+, PackagedContent*) >
<!ATTLIST ErrorComp
  ID  NMTOKEN         #REQUIRED
  xml:lang            NMTOKEN  #REQUIRED
  ErrorCode           NMTOKEN  #REQUIRED
  ErrorDesc           CDATA    #REQUIRED
  Severity (Warning|TransientError|HardError)    #REQUIRED
  MinRetrySecs        CDATA    #IMPLIED
  SwVendorErrorRef    CDATA    #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Error Component within the OTP Transaction.
`xml:lang`	Defines the language used by attributes or child elements within this component, unless overridden by an `xml:lang` attribute on a child element. See section 2.9 Identifying Languages.
`ErrorCode`	Contains an error code which indicates the nature of the error in the *message in error*. Valid values for the `ErrorCode` are given in section 5.17.2 Error Codes.

`ErrorDesc`	Contains a narrative description of the error in the language defined by `xml:lang`. The content of this attribute is defined by the vendor/developer of the software which generated the Error Component
`Severity`	Indicates the severity of the error. Valid values are: • `Warning`. This indicates that although there is a *message in error* the OTP Transaction can still continue. • `TransientError`. This indicates that the error in the *message in error* may be recovered if the *message in error* that is referred to by the `ErrorLocation`element is resent • `HardError`. This indicates that there is an unrecoverable error in the *message in error* and the OTP Transaction must stop.
`MinRetrySecs`	This attribute should be present if `Severity`is set to `TransientError`. It is the minimum number of whole seconds which the OTP aware application which received the *message reporting the error* should wait before re-sending the *message in error* identified by the `ErrorLocation`element. If `Severity`is not set to `TransientError`then the value of this attribute is ignored.
`SwVendorErrorRef`	This attribute is a reference whose value is set by the vendor/developer of the software which generated the Error Component. It should contain data which enables the vendor to identify the precise location in their software and the set of circumstances which caused the software to generate a *message reporting the error*. See also the `SoftwareId` attribute of the Message Id element in the Transaction Reference Block (section 2.3).

Content:

`ErrorLocation`	This identifies the OTP Transaction Id of the *message in error* and, where possible, the element and attribute in the *message in error* that caused the Error Component to be generated. If the `Severity`of the error is not `TransientError`, more than one `ErrorLocation`may be specified as appropriate depending on the nature of the error (see section 5.17.2 Error Codes) and at the discretion of the vendor/developer of the OTP Aware Application.
`PackagedContent`	This contains additional data which can be used to understand the error. Its content may vary as appropriate depending on the nature of the error (see section 5.17.2 Error Codes) and at the discretion of the vendor/developer of the OTP Aware Application. For a definition of `PackagedContent`see section 2.8.

5.17.1 Error Processing Guidelines

If there is more than one Error Component in a *message reporting the error*, carry out the actions appropriate for the Error Component with the highest severity. In this context, `HardError`has a higher severity than `TransientError`, which has a higher severity than `Warning`.

5.17.1.1 Severity - Warning

If an OTP aware application is generating a *message reporting the error* with an Error Component where the `Severity` attribute is set to `Warning`, then if the *message reporting the error* does not contain another Error Component with a severity higher than `Warning`, the OTP Message must also include the Trading Blocks and Trading Components that would have been included if no error was being reported.

If a *message reporting the error* is received with an Error Component where `Severity` is set to `Warning`, then:

- it is recommended that information about the error is either logged, or otherwise reported to the user,
- the implementer of the OTP aware application must either, at their or the user's discretion:
  - continue the OTP transaction as normal, or
  - fail the OTP transaction by generating a *message reporting the error* with an Error Component with `Severity` set to `HardError` (see section 5.17.1.3).

If the intention is to continue the OTP transaction then, if there are no other Error Components with a higher severity, check that the necessary Trading Blocks and Trading Components for normal processing of the transaction to continue are present. If they are not then generate a *message reporting the error* with an Error Component with `Severity` set to `HardError`.

5.17.1.2 Severity - Transient Error

If an OTP Aware Application is generating a *message reporting the error* with an Error Component where the `Severity` attribute is set to `TransientError`, then there should be only one Error Component in the *message reporting the error*. In addition, the `MinRetrySecs` attribute should be present.

If a *message reporting the error* is received with an Error Component where `Severity` is set to `TransientError` then:

- if the `MinRetrySecs` attribute is present and a valid number, then use the `MinRetrySecs` value given. Otherwise if `MinRetrySecs` is missing or is invalid, then:
  - generate a *message reporting the error* containing an Error Component with a `Severity` of `Warning` and send it on the next OTP message (if any) to be sent to the Trading Role which sent the *message reporting the error* with the invalid `MinRetrySecs`, and
  - use a value for `MinRetrySecs` which is set by the vendor/developer of the OTP Aware Application.
- check that only one `ErrorLocation` element is contained within the Error Component and that it refers to an OTP Message which was sent by the recipient of the Error Component with a `Severity` of `TransientError`. If more than one `ErrorLocation` is present then generate a *message reporting the error* with a `Severity` of `HardError`.

5.17.1.3 Severity - Hard Error

If an OTP Aware Application is generating a *message reporting the error* with an Error Component where the `Severity` attribute set to `HardError`, then there should be only one Error Component in the *message reporting the error*.

If a *message reporting the error* is received with an Error Component where `Severity` is set to `HardError` then terminate the OTP Transaction.

5.17.2 Error Codes

The following table contains the valid values for the `ErrorCode` attribute of the Error Component. The first sentence of the description contains the text that should be used to describe the error when displayed or otherwise reported. Individual implementations may translate this into alternative languages at their discretion.

An Error Code must not be more that 14 characters long.

Value	Description
`Reserved`	**Reserved**. This error is reserved by the vendor/developer of the software. Contact the vendor/developer of the software for more information See the `SoftwareId` attribute of the Message Id element in the Transaction Reference Block(section 2.3).
`XmlNotWellFrmd`	**XML not well formed**. The XML document is not well formed. See [XML] for the meaning of "well formed". Even if the XML is not well formed, it should still be scanned to find the Transaction Reference Block so that a properly formed Error Response may be generated.
`XmlNotValid`	**XML not valid**. The XML document is well formed but the document is not valid. See [XML] for the meaning of "valid". Specifically: • the XML document does not comply with the constraints defined in the OTP document type declaration (see section 11 Open Trading Protocol Data Type Definition), and • the XML document does not comply with the constraints defined in the document type declaration of any additional [XML Namespace] that are declared. As for XML not well formed, attempts should still be made to extract the Transaction Reference Block so that a properly formed Error Response may be generated.
`ElUnexpected`	**Unexpected element**. Although the XML document is well formed and valid, an element is present that is not expected in the particular context according to the rules and constraints contained in this specification.
`ElNotSupp`	**Element not supported**. Although the document is well formed and valid, an element is present that: • is consistent with the rules and constraints contained in this specification, but • is not supported by the OTP Aware Application which is processing the OTP Message.
`ElMissing`	**Element missing**. Although the document is well formed and valid, an element is missing that should have been present if the rules and constraints contained in this specification are followed. In this case set the `PackagedContent` of the Error Component to the type of the missing element.
`ElContIllegal`	**Element content illegal**. Although the document is well formed and valid, the element `PackagedContent` contains values which do not conform to the rules and constraints contained in this specification.
`EncapProtErr`	**Encapsulated protocol error**. Although the document is well formed

Value	Description
	and valid, the `PackagedContent`of an element contains data from an encapsulated protocol which contains errors.
`AttUnexpected`	**Unexpected attribute**. Although the XML document is well formed and valid, the presence of the attribute is not expected in the particular context according to the rules and constraints contained in this specification.
`AttNotSupp`	**Attribute not supported**. Although the XML document is well formed and valid, and the presence of the attribute in an element is consistent with the rules and constraints contained in this specification, it is not supported by the OTP Aware Application which is processing the OTP Message.
`AttMissing`	**Attribute missing**. Although the document is well formed and valid, an attribute is missing that should have been present if the rules and constraints contained in this specification are followed. In this case set the `PackagedContent`of the Error Component to the type of the missing attribute.
`AttValIllegal`	**Attribute value illegal**. The attribute contains a value which does not conform to the rules and constraints contained in this specification.
`AttValNotRecog`	**Attribute Value Not Recognised**. The attribute contains a value which the OTP Aware Application generating the *message reporting the error* could not recognise even though it should have been able to since the information had been provided in an earlier OTP message.
`MsgTooLarge`	**Message too large**. The message is too large to be processed by the OTP Aware Application.
`ElTooLarge`	**Element too large**. The element is too large to be processed by the OTP Aware Application
`ValueTooSmall`	**Value too small or early**. The value of all or part of the `PackagedContent`of an element or an attribute, although valid, is too small.
`ValueTooLarge`	**Value too large or in the future**. The value of all or part of the `PackagedContent`of an element or an attribute, although valid, is too large.
`ElInconsistent`	**Element Inconsistent**. Although the document is well formed and valid, according to the rules and constraints contained in this specification: • the content of an element is inconsistent with the content of other elements or their attributes, or • the value of an attribute is inconsistent with the value of one or more other attributes. In this case create `ErrorLocation`elements which identify all the attributes or elements which are inconsistent.
`TransportError`	**Transport Error**. This error code is used to indicate that there is a problem with the Transport Mechanism which is preventing the message from being received. It is typically associated with a Transient Error. Explanation of the Transport Error is contained within the `ErrorDesc`attribute. The values which can be used inside `ErrorDesc`

Value	Description
	with a `TransportError`is specified in the OTP supplement for the Transport mechanism.

5.17.3 Error Location Element

An Error Location Element identifies an element and optionally an attribute in the *message in error* which is associated with the error. It contains a reference to the OTP Message, Trading Block, Trading Component, element and attribute, which is in error.

```
<!ELEMENT ErrorLocation EMPTY>
<!ATTLIST ErrorLocation
  ElementType        NMTOKEN  #REQUIRED
  OtpMsgRef          NMTOKEN  #REQUIRED
  BlkRef             NMTOKEN  #IMPLIED
  CompRef            NMTOKEN  #IMPLIED
  ElementRef         NMTOKEN #IMPLIED
  AttName            NMTOKEN  #IMPLIED >
```

Attributes:

`ElementType`	This is the "type" (see [XML]) of the Element in the *message in error* where the error is located.
`OtpMsgRef`	This is the value of the `ID` attribute of the of the Message Id Component (see section 2.3.2) of the *message in error* to which this Error Component applies.
`BlkRef`	If the error is associated with a specific Trading Block, then this is the value of the `ID` attribute of the Trading Block where the error is located.
`CompRef`	If the error is associated with a specific Trading Component, then this is the value of the `ID` attribute of the Trading Component where the error is located.
`ElementRef`	If the error is associated with a specific element within a Trading Component then, if the element has an attribute with an "attribute type" (see [XML]) of "ID", then this is the value of that attribute.
`AttName`	If the error is associated with the value of an attribute, then this is the name of that attribute. In this case the `PackagedContent`of the Error Component should contain the value of the attribute.

Note that as many as the attributes as possible should be included. For example if an attribute in a child element of a Trading Component contains an incorrect value, then all the attributes of `ErrorLocation` should be present.

6. Trading Blocks

Trading Blocks consist of one or more **Trading Components** and optionally one or more **Signature Components**. One or more **Trading Blocks** may be contained within the **OTP Messages** which are physically sent in the form of [XML] documents between the different organisations that are taking part in a trade.

This is illustrated in the diagram below.

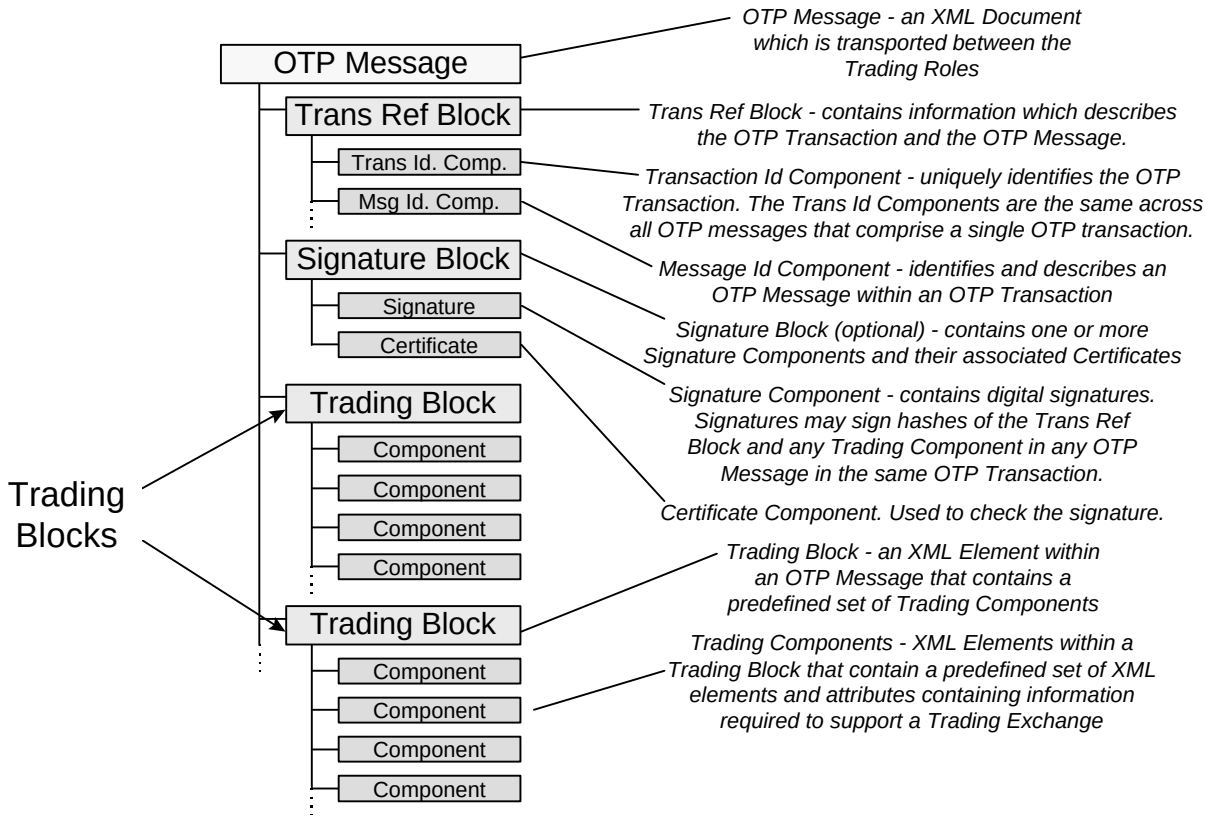


Figure 18 Trading Blocks

Trading Blocks are defined as part of the definition of an OTP Message (see section 2.1.1). The definition of an OTP Message element is repeated here:

```
<!ELEMENT OtpMessage (TransRefBlk, SigBlk?, ErrorBlk?,
(       AuthReqBlk |
        AuthRespBlk |
        DeliveryReqBlk |
        DeliveryRespBlk |
        InquiryReqBlk |
        InquiryRespBlk |
        OfferRespBlk |
        PayExchBlk |
        PayReqBlk |
        PayInstCCExchBlk |
        PayInstCCReqBlk |
        PayInstCCRespBlk
        PayRespBlk |
        PingReqBlk |
        PingRespBlk |
        TpoBlk |
        TpoSelectionBlk |
        )*
) >
```

The remainder of this section defines the Trading Blocks in this version of OTP. They are:

- Authentication Request Block
- Authentication Response Block
- Delivery Request Block
- Delivery Response Block
- Error Block
- Inquiry Request Block
- Inquiry Response Block
- Offer Response Block
- Payment Exchange Block
- Payment Request Block
- Payment Response Block
- Payment Instrument Customer Care Exchange Block
- Payment Instrument Customer Care Request Block
- Payment Instrument Customer Care Response Block
- Signature Block
- Trading Protocol Options Block
- TPO Selection Block

The Transaction Reference Block is described in section 2.3.

6.1 Trading Protocol Options Block

The TPO Trading Block contains options which apply to the OTP Transaction. The definition of a TPO Trading Block is as follows.

```
<!ELEMENT TpoBlk ( ProtocolOptions, BrandList*, Org* ) >
<!ATTLIST TpoBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Trading Protocol Options Block within the OTP Transaction (see section 2.4 ID Attributes).

Content:

`ProtocolOptions`	The Protocol Options Component (see section 5.1)defines the options which apply to the whole OTP Transaction (see section 7).
`BrandList`	This Brand List Component contains one or more payment brands and protocols which may be selected (see section 5.6).
`Org`	The Organisation Components (see section 5.5) identify the organisations and their roles in the OTP Transaction. The roles and organisations which must be present will depend on the particular type of OTP Transaction. See the definition of each transaction in section 7. Open Trading Protocol Transactions.

The TPO Block should contain:

- the Protocol Options Component
- the Organisation Component with the Trading Role of `Merchant`
- the Organisation Component with the Trading Role of `Consumer`
- optionally, the Organisation Component with the Trading Role of `DeliverTo`, if there is a Delivery included in the OTP Transaction
- Brand List Components for each payment in the OTP Transaction
- Organisation Components for all the Payment Handlers involved
- optionally, Organisation Components for the Delivery Handler (if any) for the transaction
- additional Organisation Components that the Merchant may want to include. For example
  - a Customer Care Provider
  - an Certificate Authority that offers Merchant "Credentials" or some other warranty on the goods or services being offered.

6.2 TPO Selection Block

The TPO Selection Block contains the results of selections made from the options contained in the Trading Protocol Options Block (see section 6.1).The definition of a TPO Selection Block is as follows.

```
<!ELEMENT TpoSelectionBlk (BrandSelection+) >
<!ATTLIST TpoSelectionBlk
```

```
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the TPO Selection Block within the OTP Transaction.

Content:

`BrandSelection`	This identifies the choice of payment brand and payment protocol to be used in a payment within the OTP Transaction. There is one Brand Selection Component (see section 5.7) for each payment to be made in the OTP Transaction.

The TPO Selection Block should contain one Brand Selection Component for each Brand List in the TPO Block.

6.3 Offer Response Block

The Offer Response Block contains details of the goods, services, amount, delivery instructions or financial transaction which is to take place. Its definition is as follows.

```
<!ELEMENT OfferRespBlk (AuthData*, Order?, Payment*, Delivery?,
      Status ) >
<!ATTLIST OfferRespBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Offer Response Block within the OTP Transaction.

Content:

`AuthData`	The Authentication Data Component contains information about how Authentication associated with the Offer will occur. See section 5.2.
`Order`	The Order Component contains details about the goods, services or financial transaction which is taking place see section 5.4. The Order Component must be present unless the `ProcessState` attribute of the Status Component is set to `Failed`.
`Payment`	The Payment Components contain information about the payments which are to be made see section 5.8.
`Delivery`	The Delivery Component contains details of the delivery to be made (see section 5.11).
`Status`	Contains status information about the business success (see section 3.2) or failure of the generation of the Offer. Note that in an Offer Response Block, a `ProcessState`of `NotYetStarted`or `InProgress` are illegal values.

The Offer Response Block should contain:

- the Order Component for the OTP Transaction

- Payment Components for each Payment in the OTP Transaction
- the Delivery Component for OTP Transaction requires (if any)
- the Authentication Data Component (if required) for each Payment

6.4 Authentication Request Block

This Authentication Request Block contains the challenge data which is used to obtain information about and optionally authenticate a Consumer by another Trading Role. Its definition is as follows.

```
<!ELEMENT AuthReqBlk (AuthData?) >
<!ATTLIST AuthReqBlk
  ID  ID              #REQUIRED >
```

Attributes

`ID`	An identifier which uniquely identifies the Authentication Request Block within the OTP Transaction.

Content

`AuthData`	If the Authentication Data Component is not present it means that the Authentication Request Block is just requesting the return of Organisation Components which describe the Consumer. If the optional Authentication Data Component (see section 5.2) is present it contains data which describes what additional Authentication the consumer must provide.

6.5 Authentication Response Block

The Authentication Response Block contains the response which results from processing the Authentication Request Block. Its definition is as follows.

```
<!ELEMENT AuthRespBlk (AuthResp, Org+) >
<!ATTLIST AuthRespBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Authentication Response Block within the OTP Transaction.

Content:

`AuthResp`	The Authentication Response Component which contains the results of processing the challenge data in the Authentication Data Component - see section 5.3.
`Org`	Organisation Components which contain information corresponding to the `Consumer` and `DelivTo` Trading Roles.

6.6 Payment Request Block

The Payment Request Block contains information which requests that a payment is started. Its definition is as follows.

```
<!ELEMENT PayReqBlk (AuthData?, BrandList, BrandSelection,
      Payment, PaySchemeData?, Org*) >
<!ATTLIST PayReqBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Payment Request Block within the OTP Transaction.

Content:

`AuthData`	The optional Authentication Data Component contains data about how Authentication associated with the payment, if any, will occur. See section 5.2.
`BrandList`	The Brand List Component contains a list of one or more payment brands and protocols which may be selected (see section 5.6).
`BrandSelection`	This identifies the choice of payment brand, the payment protocol and the payment handler to be used in a payment within the OTP Transaction. There is one Brand Selection Component (see section 5.7) for each payment to be made in the OTP Transaction.
`Payment`	The Payment Components contain information about the payment which is being made see section 5.8.
`PaySchemeData`	The Payment Scheme Component contains payment scheme specific data see section 5.9.
`Org`	The Organisation Component contains details of organisations involved in the payment (see section 5.5). The Organisations present are dependent on the OTP Transaction and the data which is to be signed. See section 4 Security Considerations for more details.

The Payment Request Block should contain:

- the Organisation Component with a Trading Role of `Merchant`
- the Organisation Component with the Trading Role of `Consumer`
- the Payment Component for the Payment
- the Brand List Component for the Payment
- the Brand Selection Component for the Brand List
- the Organisation Component for the Payment Handler of the Payment
- the Organisation Component (if any) for the Organisation which carried out the previous step, for example another Payment Handler
- the Organisation Component for the organisation which is to carry out the next step, if any. This may be, for example, either a Delivery Handler or a Payment Handler.
- the Organisation Components for any additional Organisations that the Merchant has included in the Offer Response Block

- an Optional Payment Scheme Data Component, if required by the Payment Method as defined in the OTP supplement for the payment method.

6.7 Payment Exchange Block

The Payment Exchange Block contains payment scheme specific data which is exchanged between two of the roles in a trade. Its definition is as follows.

```
<!ELEMENT PayExchBlk (PaySchemeData) >
<!ATTLIST PayExchBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Payment Exchange Block within the OTP Transaction.

Content:

`PaySchemeData`	This Trading Component contains payment scheme specific data see section 5.9 Payment Scheme Component.

6.8 Payment Response Block

This Payment Response Block contains a information about the Payment Status, a Payment Receipt, and an optional payment protocol message. Its definition is as follows.

```
<!ELEMENT PayRespBlk (Status, PayReceipt, PaySchemeData?) >
<!ATTLIST PayRespBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Payment Response Block within the OTP Transaction.

Content:

`Status`	Contains status information about the business success (see section 3.2) or failure of the payment. Note that in a Pay Response Block, a `ProcessState`of `NotYetStarted`or `InProgress`are illegal values.
`PayReceipt`	Contains payment scheme specific data which can be used to verify the payment occurred. See section 5.10 Payment Receipt Component.
`PaySchemeData`	Contains payment scheme specific data see section, for example a payment protocol message. See 5.9 Payment Scheme Component.

6.9 Delivery Request Block

The Delivery Request Block contains details of the goods or services which are to be delivered together with a signature which can be used to check that delivery is authorised. Its definition is as follows.

```
<!ELEMENT DeliveryReqBlk (Order, Org*, Delivery) >
<!ATTLIST DeliveryReqBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Delivery Request Block within the OTP Transaction.

Content:

`Order`	The Order Component contains details about the goods, services or financial transaction which is taking place see section 5.4.
`Org`	The Organisation Components (see section 5.5) identify the organisations and their roles in the OTP Transaction. The roles and organisations which must be present will depend on the particular type of OTP Transaction. See the definition of each transaction in section 7. Open Trading Protocol Transactions.
`Delivery`	The Delivery Component contains details of the delivery to be made (see section 5.11).

The Delivery Request Block contains:

- the Organisation Component with a Trading Role of `Merchant`
- the Organisation Component for the `Consumer` and `DeliverTo` Trading Roles
- the Delivery Component for the Delivery
- the Organisation Component for the Delivery Handler. Specifically the Organisation Component identified by the `ActionOrgRef` attribute on the Delivery Component
- the Organisation Component (if any) for the Organisation which carried out the previous step, for example a Payment Handler
- the Organisation Components for any additional Organisations that the Merchant has included in the Offer Response Block

6.10 Delivery Response Block

The Delivery Response Block contains a Delivery Note containing details on how the goods will be delivered. Its definition is as follows. Note that in a Delivery Response Block a Delivery Status Element with a `DeliveryStatusCode` of `NotYetStarted` or `InProgress` is invalid.

```
<!ELEMENT DeliveryRespBlk (Status, DeliveryNote) >
<!ATTLIST DeliveryRespBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Delivery Response Block within the OTP Transaction.

Content:

`Status`	Contains status information about the business success (see section 3.2) or failure of the delivery. Note that in a Delivery Response Block, a `ProcessState`of `NotYetStarted`or `InProgress`are illegal values.
`DeliveryNote`	The Delivery Note Component contains details about how the goods or services will be delivered (see section 5.12).

6.11 Payment Instrument Customer Care Request Block

The Payment Instrument Customer Care Request Block contains information which requests that an OTP Payment Instrument Customer Care Transaction is started in order to provide Customer Care for the Consumer's Payment Instrument. Its definition is as follows.

```
<!ELEMENT PayInstCCReqBlk (PaymethodInfo, PaySchemeData*) >
<!ATTLIST PayInstCCReqBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Payment Instrument Customer Care Request Block within the OTP Transaction.

Content:

`PayMethodInfo`	The Payment Method Information Component (see section 5.13) contains data which describes the Payment Method which initiated the Payment Instrument Customer Care Transaction
`PaySchemeData`	Optional Payment Scheme Components (see section 5.9) that contain payment scheme specific data. The sequence of the Payment Scheme Components in the Block is the sequence in which they should be processed by the Payment Scheme software which receives this message.

6.12 Payment Instrument Customer Care Exchange Block

The Payment Instrument Customer Care Exchange Block contains payment scheme specific data which is exchanged between the Payment Instrument User and the Payment Scheme Customer Care Provider. Its definition is as follows.

```
<!ELEMENT PayInstCCExchBlk (PaySchemeData) >
<!ATTLIST PayInstCCExchBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Payment Instrument Customer Care Exchange Block within the OTP Transaction.

Content:

`PaySchemeData`	Optional Payment Scheme Components (see section 5.9) that contain payment scheme specific data. The sequence of the Payment Scheme Components in the Block is the sequence in which they should be processed by the Payment Scheme software which receives this message.

6.13 Payment Instrument Customer Care Response Block

The Payment Instrument Customer Care Response Block contains the final Payment Scheme Component of the OTP Payment Instrument Customer Care Transaction. Its definition is as follows.

```
<!ELEMENT PayInstCCRespBlk (PaySchemeData) >
<!ATTLIST PayInstCCRespBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Payment Instrument Customer Care Response Block within the OTP Transaction.

Content:

`PaySchemeData`	Optional Payment Scheme Components (see section 5.9) that contain payment scheme specific data. The sequence of the Payment Scheme Components in the Block is the sequence in which they should be processed by the Payment Scheme software which receives this message.

6.14 Inquiry Request Trading Block

The Inquiry Request Trading Block contains an Inquiry Type Component and an optional Payment Scheme Component to contain payment scheme specific inquiry messages.

```
<!ELEMENT InquiryReqBlk ( InquiryType, PaySchemeData? ) >
<!ATTLIST InquiryReqBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Inquiry Request Trading Block within the OTP Transaction.

Content:

`InquiryType`	Inquiry Type Component (see section 5.15) that contains the type of inquiry.
`PaySchemeData`	Payment Scheme Component (see section 5.9) that contains payment scheme specific inquiry messages for inquiries on payments. This is present when the `Type` attribute of Inquiry Type Component is `Payment`.

6.15 Inquiry Response Trading Block

The Inquiry Response Trading Block contains a Status Component and an optional Payment Scheme Component to contain payment scheme specific inquiry messages. Its purpose is to enquire on the current status of an OTP transaction at a server.

```
<!ELEMENT InquiryRespBlk (Status, PaySchemeData?) >
<!ATTLIST InquiryRespBlk
  ID  ID              #REQUIRED
  LastReceivedOtpMsgRef NMTOKEN#IMPLIED
  LastSentOtpMsgRef  NMTOKEN  #IMPLIED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Inquiry Response Trading Block within the OTP Transaction.
`LastReceivedOtpMsgRef`	Contains an Element Reference (see section 2.5) to the Message Id Component (see section 2.3.2) of the last message this server has received from the Consumer. If there is no previously received message from the Consumer in the pertinent transaction, this attribute should be contain the value `Null`. This attribute exists for debugging purposes.
`LastSentOtpMsgRef`	Contains an Element Reference (see section 2.5) to the Message Id Component (see section 2.3.2) of the last message this server has sent to the Consumer. If there is no previously sent message to the Consumer in the pertinent transaction, this attribute should contain the value `Null`. This attribute exists for debugging purposes.

Content:

`Status`	Contains status information about the business success (see section 3.2) or failure of a certain trading exchange (i.e., Offer, Payment, or Delivery).
`PaySchemeData`	Payment Scheme Component (see section 5.9) that contains payment scheme specific inquiry messages for inquiries on payments. This is present when the Type attribute of `StatusType`attribute of the Status Component is set to `Payment`.

6.16 Ping Request Block

The Ping Request Block is used to determine if a Server is operating and whether or not cryptography is compatible.

The definition of a Ping Request Block is as follows.

```
<!ELEMENT PingReqBlk (Org*)>
<!ATTLIST PingReqBlk
  ID  ID              #REQUIRED>
```

Attributes:

`ID` — An identifier which uniquely identifies the Ping Request Trading Block within the OTP Transaction.

Content:

`Org` — Optional Organisation Components (see section 5.5).

If no Organisation Component is present then the Ping Request is anonymous and simply determines if the server is operating.

However if Organisation Components are present, then it indicates that the sender of the Ping Request wants to verify that digital signatures can be handled.

In this case the sender includes:

- an Organisation Component that identifies itself specifying the Trading Role(s) it is taking in OTP transactions (Merchant, Payment Handler, etc)
- an Organisation Component that identifies the intended recipient of the message.

These are then used to generate a signature over the Ping Response Block.

6.17 Ping Response Block

The Ping Response Trading Block provides the result of a Ping Request.

It contains an Organisation Component that identifies the sender of the Ping Response.

If the Ping Request to which this block is a response contained Organisation Components, then it also contains those Organisation Components.

```
<!ELEMENT PingRespBlk (Org+)>
<!ATTLIST PingRespBlk
  ID  ID              #REQUIRED
  PingStatusCode (Ok|Busy|Down) #REQUIRED
  SigVerifyStatusCode (Ok|NotSupported|Fail) #IMPLIED
  xml:lang           NMTOKEN  #IMPLIED
  PingStatusDesc     CDATA    #IMPLIED>
```

Attributes:

`ID`	An identifier which uniquely identifies the Ping Request Trading Block within the OTP Transaction.
`PingStatusCode`	Contains a code which shows the status of the sender software which processes OTP messages. Valid values are: • `Ok`. Everything with the service is working normally, including the signature verification. • `Busy`. Things are working normally but there may be some delays. • `Down`. The server is not functioning fully but can still provide a Ping response.
`SigVerifyStatusCode`	Contains a code which shows the status of signature verification. This is present only when the message containing the Ping Request Block also contains a Signature Block. Valid values are: • `Ok`. The signature has successfully been verified and proved compatible. • `NotSupported`The receiver of this Ping Request Block does not support validation of signatures. • `Fail`. Signature verification failed.
`Xml:lang`	Defines the language used in `PingStatusDesc` This is present when `PingStatusDesc`is present.
`PingStatusDesc`	Contains a short description of the status of the server which sends this Ping Response Block. Servers, if their designers want, can use this attribute to send more refined status information than `PingStatusCode` which can be used for debugging purposes, for example.

Content:

`Org`	These are Organisation Components (see section 5.5). The Organisation Components of the sender of the Ping Response is always included in addition to the Organisation Components sent in the Ping Request.

[Note] *Ping Status Code values do not include a value such as `Fail`, since, when the software receiving the Ping Request message is not working at all, no Ping Response message will be sent back.*

[Note End]

6.18 Signature Block

The Signature Block contains one or more Signature Components and associated Certificates which sign data associated with the OTP Transaction. For a general discussion and introduction to how OTP uses signatures, see section 4 Security Considerations. The definition of the Signature Component and certificates is contained in the paper "Digital Signature for XML - Proposal", see [XMLSIG].

The definition of a Signature Block is as follows:

```
<!ELEMENT SigBlk (OtpSig+, OtpCert*) >
<!ATTLIST SigBlk
  ID  ID             #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Signature Block within the OTP Transaction.

Content:

`OtpSig`	Contains a Digital Signature. See the paper "Digital Signature for XML - Proposal" [XMLSIG], for its definition
`OtpCert`	Contains a Digital Certificate. See the paper "Digital Signature for XML - Proposal" [XMLSIG], for its definition

The contents of a Signature Block depends on the Trading Block that is contained in the same OTP Message as the Signature Block.

6.18.1 Offer Response

A Signature Block which is in the same message as an Offer Response Block contains just an Offer Response Signature Component (see section 5.16.1).

6.18.2 Payment Request

A Signature Block which is in the same message as a Payment Request Block contains:

- an Offer Response Signature Component (see section 5.16.1), and
- if the Payment is dependent on an earlier step (as indicated by the `StartAfter`attribute on the Payment Component), then the Payment Receipt Signature Component (see section 5.16.2) generated by the previous step

6.18.3 Payment Response

A Signature Block which is in the same message as a Payment Response Block contains just a Payment Receipt Signature Component (see section 5.16.2) generated by the step.

6.18.4 Delivery Request

A Signature Block which is in the same message as a Delivery Request Block contains:

- an Offer Response Signature Component (see section 5.16.1), and
- the Payment Receipt Signature Component (see section 5.16.2) generated by the previous step.

6.19 Error Block

The Error Trading Block contains one or more Error Components (see section 5.17) which contain information about Technical Errors (see section 3.1) in an OTP Message which has been received by one of the Trading Roles involved in the trade.

For clarity two phrases are defined which are used in the description of an Error Trading Block:

- *message in error*. An OTP message which contains or causes an error of some kind
- *message reporting the error*. An OTP message that contains an Error Trading Block that describes the error found in a *message in error.*

An Error Trading Block may be contained in any *message reporting the error*. The action which then follows depends on the severity of the error. See the definition of an Error Component, for an explanation of the different types of severity and the actions which can then occur.

[Note] *Although, an Error Trading Block can report multiple different errors using multiple Error Components, there is no obligation on a developer of an OTP Aware Application to do so.*
[Note End]

The structure of an Error Trading Block is as follows.

```
<!ELEMENT ErrorBlk (ErrorComp+, PaySchemeData*) >
<!ATTLIST ErrorBlk
  ID  ID              #REQUIRED >
```

Attributes:

`ID`	An identifier which uniquely identifies the Error Trading Block within the OTP Transaction.

Content:

`ErrorComp`	An Error Component (see section 5.17) that contains information about an individual Technical Error.
`PaySchemeData`	An optional Payment Scheme Component (see section 5.9) which contains a Payment Scheme Message. See the appropriate payment scheme supplement to determine whether or not this component needs to be present and for the definition of what it must contain.

7. Open Trading Protocol Transactions

The Baseline Open Trading Protocol supports the following types of Baseline OTP Transactions:

- Authentication
- Deposit
- Purchase
- Refund
- Withdrawal
- Baseline Value Exchange
- Payment Instrument Customer Care
- Transaction Status Inquiry, and
- Ping

Each of these transactions are described in more detail in the following sections providing descriptions of:

- the Trading Blocks in each OTP Transaction
- the Trading Components in each Trading Block, and
- how the Trading Components are signed

[Note] *There are many similarities between the transactions within OTP. This is because there is a lot of reuse of the Trading Blocks between the different transactions.*

This means that there should be significant opportunity for software re-use. For example, from an OTP perspective, the Deposit, Refund and Withdrawal transactions are essentially the same, although the processing which will occur, especially at the server end, will differ.

[Note End]

7.1 Baseline Authentication OTP Transaction

The Baseline Authentication OTP Transaction supports:

- the remote authentication of a Consumer by another Trading Role using a variety of authentication methods, and
- the provision of Organisation Component about a Consumer to another Trading Role.

Typical use includes:

- when the Baseline Authentication OTP Transaction takes place as an early part of a session where strong continuity exists. For example, a Financial Institution could:
  - set up a secure channel (e.g. using SSL) with a customer
  - authenticate the customer using the Baseline Authentication OTP Transaction, and then
  - provide the customer with access to account information and other services with the confidence that they are communicating with a bona fide customer.

- as a means of providing a Merchant role with Organisation Components that contain information about `Consumer` and `DelivTo` Trading Roles.

The Baseline Authentication OTP Transaction consists of just the Authentication Trading Exchange (see section 1.2.4).

The Authentication Exchange is implemented by a set of predefined OTP Messages (see section 0) which are exchanged between the Trading Roles (see section 1.1). Each OTP Message contains Trading Blocks (see section 6) which contain the Trading Components (see section 5) which are required by the Trading Exchanges.

The Trading Blocks used by the Baseline Authentication OTP Transaction are:

- Trading Protocol Options Block
- Authentication Request Block, and
- Authentication Response Block

There are no variations of the Baseline Authentication OTP Transaction.

The OTP Messages used in a Baseline Authentication are illustrated in the diagram below.

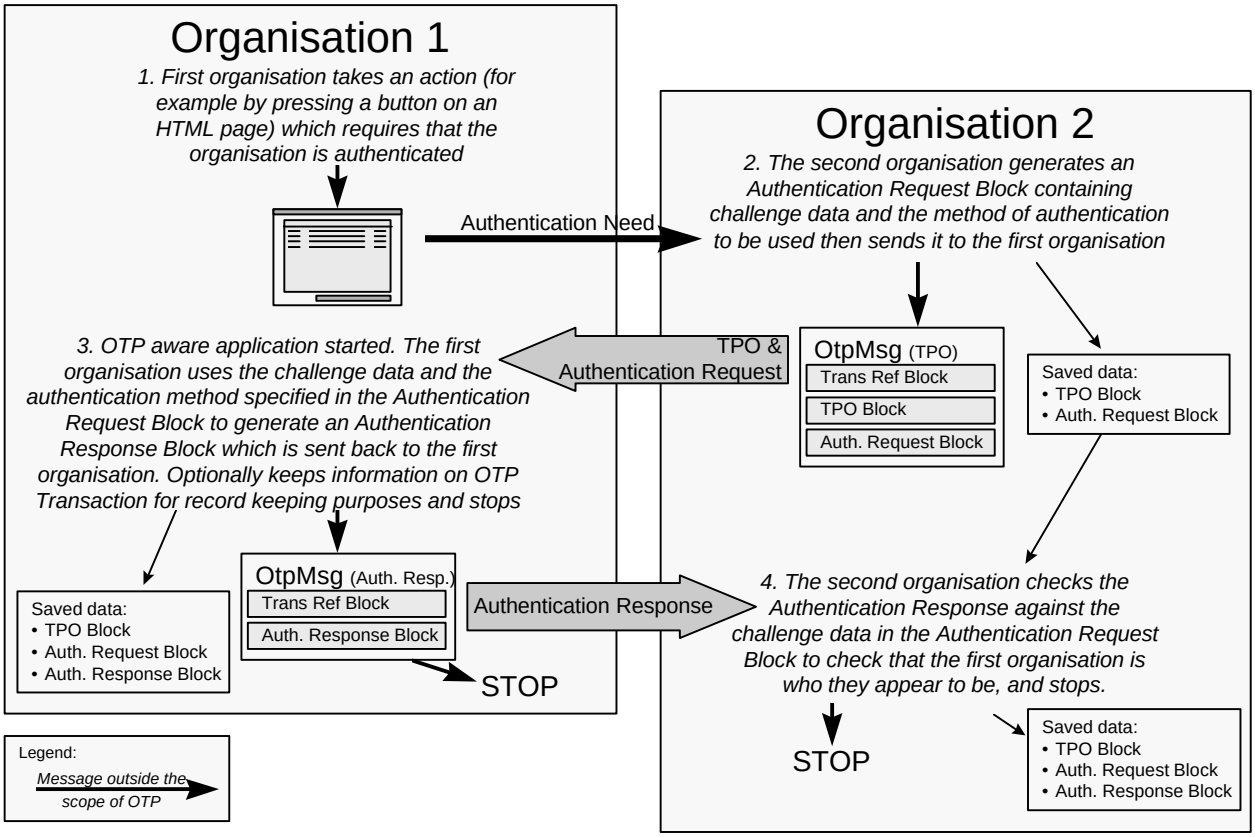


Figure 19 Baseline Authentication

The remainder of this sub-section on the Baseline Authentication OTP Transaction defines the contents of each Trading Block.

7.1.1 Trading Protocol Options Block

The TPO (Trading Protocol Options) Block (see section 7.3.2) must contain the following Trading Component:

- one Protocol Options Component which defines the options which apply to the whole OTP Transaction. See Section 5.1.

7.1.2 Authentication Request Block

The Authentication Request Block (see section 6.4) must contain the following Trading Component:

- one Authentication Data Component (see section 5.2)

7.1.3 Authentication Response Block

The Authentication Response Block (see section 6.5) must contain the following Trading Component:

- one Authentication Response Component (see section 5.3).

7.2 Baseline Deposit OTP Transaction

The Baseline Deposit OTP Transaction supports the deposit of electronic cash with a Financial Institution.

[Note] *The Financial Institution has, in OTP terminology, a role of merchant in that a service (i.e. a deposit of electronic cash) is being offered in return for a fee, for example bank charges of some kind. The term "Financial Institution" is used in the diagrams and in the text for clarity.*
[Note End]

The Baseline Deposit OTP Transaction consists of the following Trading Exchanges:

- an optional Authentication Exchange (see section 1.2.4),
- an Offer Exchange (see section 1.2.1), and
- a Payment Exchange (see section 1.2.2).

These Trading Exchanges are implemented by a set of predefined OTP Messages (see section 0) which are exchanged between the Trading Roles (see section 1.1). Each OTP Message contains Trading Blocks (see section 0) which contain the Trading Components (see section 5) which are required by the Trading Exchanges.

The Trading Blocks used by the Baseline Purchase OTP Transaction are:

- Trading Protocol Options Block
- TPO Selection Block
- Authentication Request Block
- Authentication Response Block

- Offer Response Block
- Payment Request Block
- Payment Exchange Block
- Payment Response Block
- Signature Block

7.2.1 Baseline Deposit Variations

The Baseline Deposit OTP Transaction occurs in two basic forms:

- Baseline Deposit with Authentication. Where the Consumer making the deposit is authenticated before the deposit is made, and
- Baseline Deposit without Authentication. Where the Consumer is not authenticated before the deposit is made.

7.2.2 Baseline Deposit Authentication

In Baseline Deposit with Authentication an Authentication Exchange occurs before the Offer Exchange containing the details of the deposit is provided by the Financial Institution.

In Baseline Deposit without Authentication, there is no Authentication Exchange and the Financial Institution provides details about the deposit immediately at the start of the OTP Transaction.

These two alternatives are illustrated in the two diagrams below. The first diagram illustrates the case when an Authentication Exchange is included.

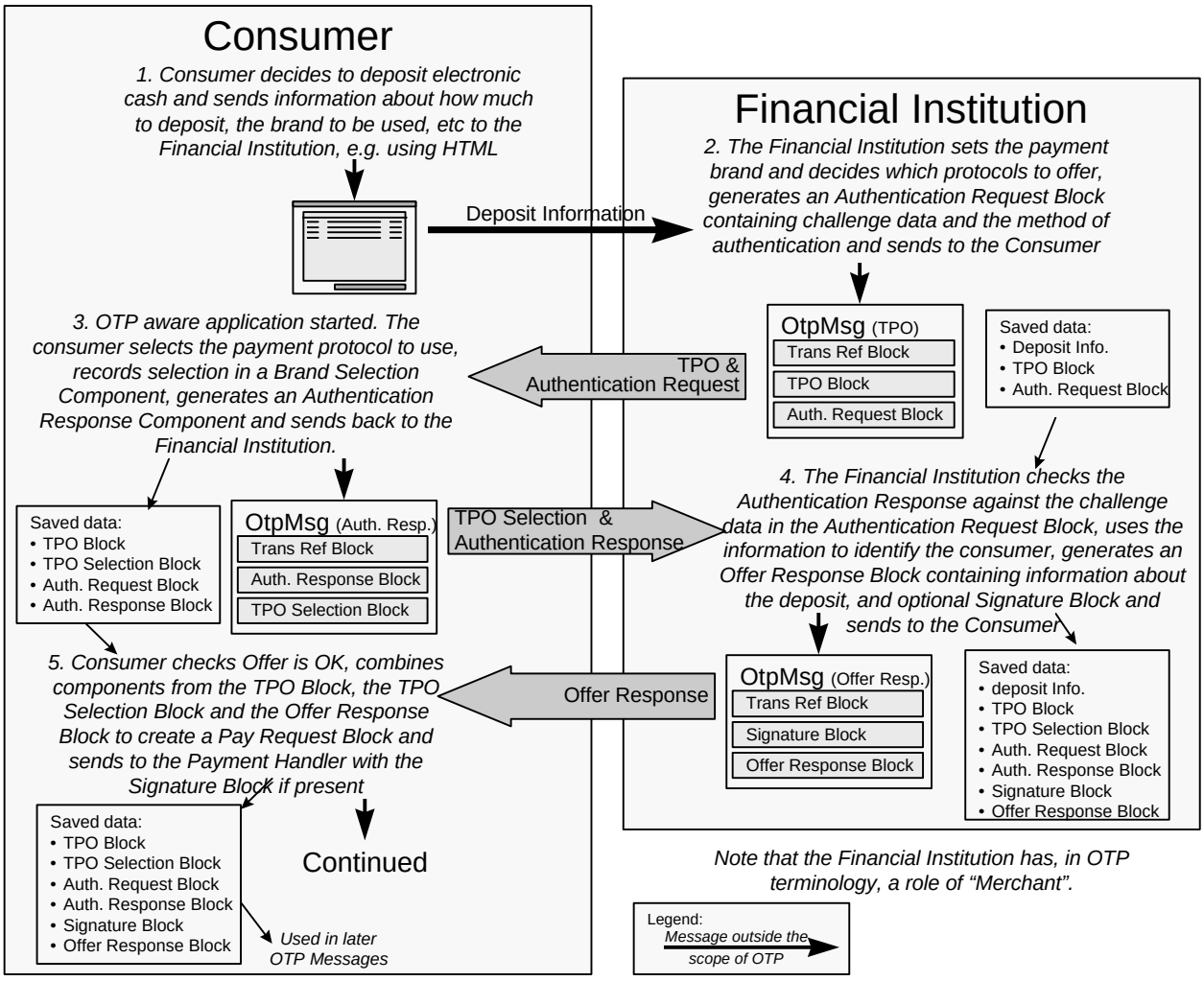


Figure 20 Baseline Deposit with Authentication

Note that the above diagram:

- describes the general case where a Merchant can accept a deposit in several different types of electronic cash. In practice usually only one form of electronic cash may be accepted. However, there may be several different protocols which may be used for the same "brand" of electronic cash.
- the financial institution may use the results of the authentication to identify not only the consumer but also the account to which the payment is to be deposited. If no single account can be identified, then it must be obtained by other means. For example:
  - the consumer could specify the account number in the initial dialogue (see step 1), or
  - the consumer could have been identified earlier, for example using a Baseline Authentication OTP Transaction, and an account selected from a list provided by the Financial Institution.

The second diagram illustrates the case when an Authentication Exchange is not included.

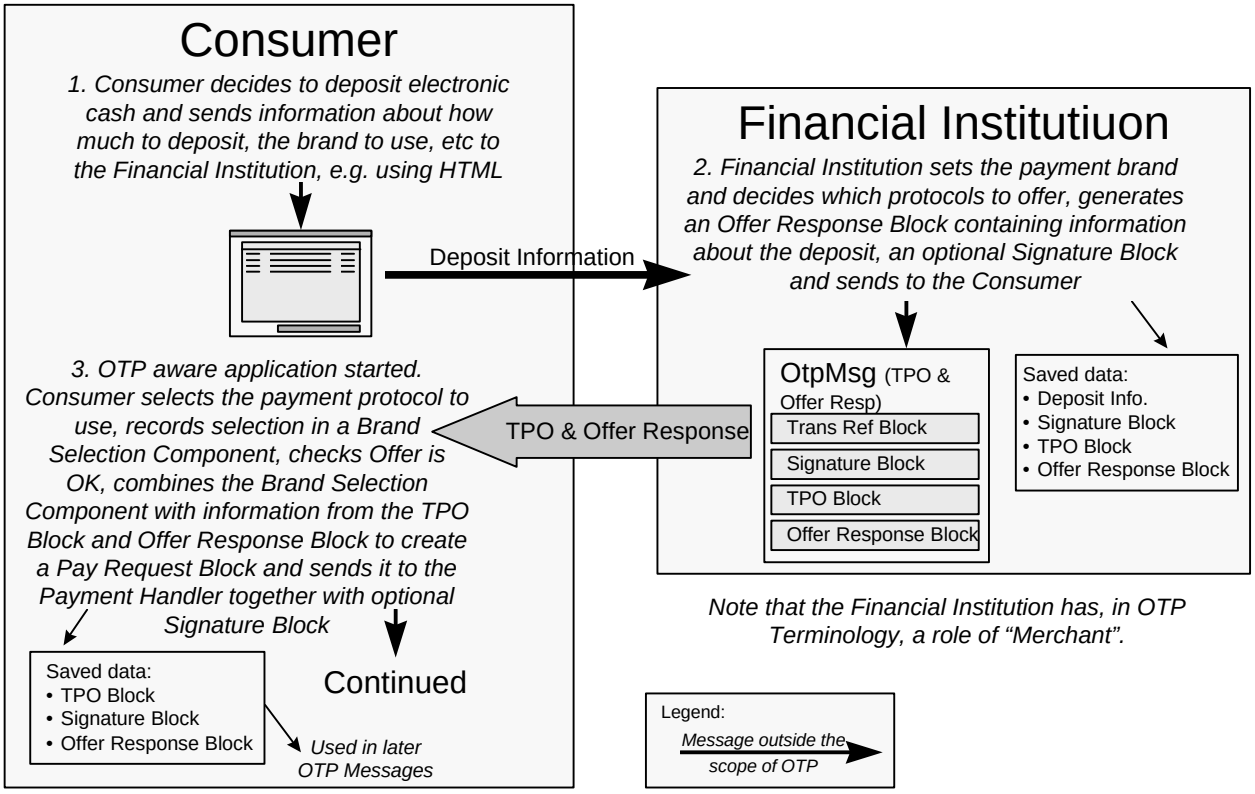


Figure 21 Baseline Deposit without Authentication

The Baseline Deposit without authentication might be used:

- if a previous OTP transaction, for example a Baseline Withdrawal or a Baseline Authentication, authenticated the consumer, and a secure channel has been maintained, therefore the authenticity of the consumer is known
- if authentication is achieved as part of a proprietary payment protocol and is therefore included in the Payment Exchange
- if authentication of the consumer has been achieved by some other means outside of the scope of OTP, for example, by using a pass phrase.

OTP aware applications supporting the Consumer Trading Role must check for the existence of an Authentication Request Block in the first OTP Message to determine whether the Baseline Deposit includes an Authentication Exchange or not.

7.2.3 Baseline Deposit Payment Messages

Once the Offer Response Trading Block has been received, the sequence of OTP Messages illustrated in Figure 22 occurs. These are the same whether or not an Authentication of the Consumer has occurred. Note that these continue where the previous diagrams (Figure 20 and Figure 21) finish.

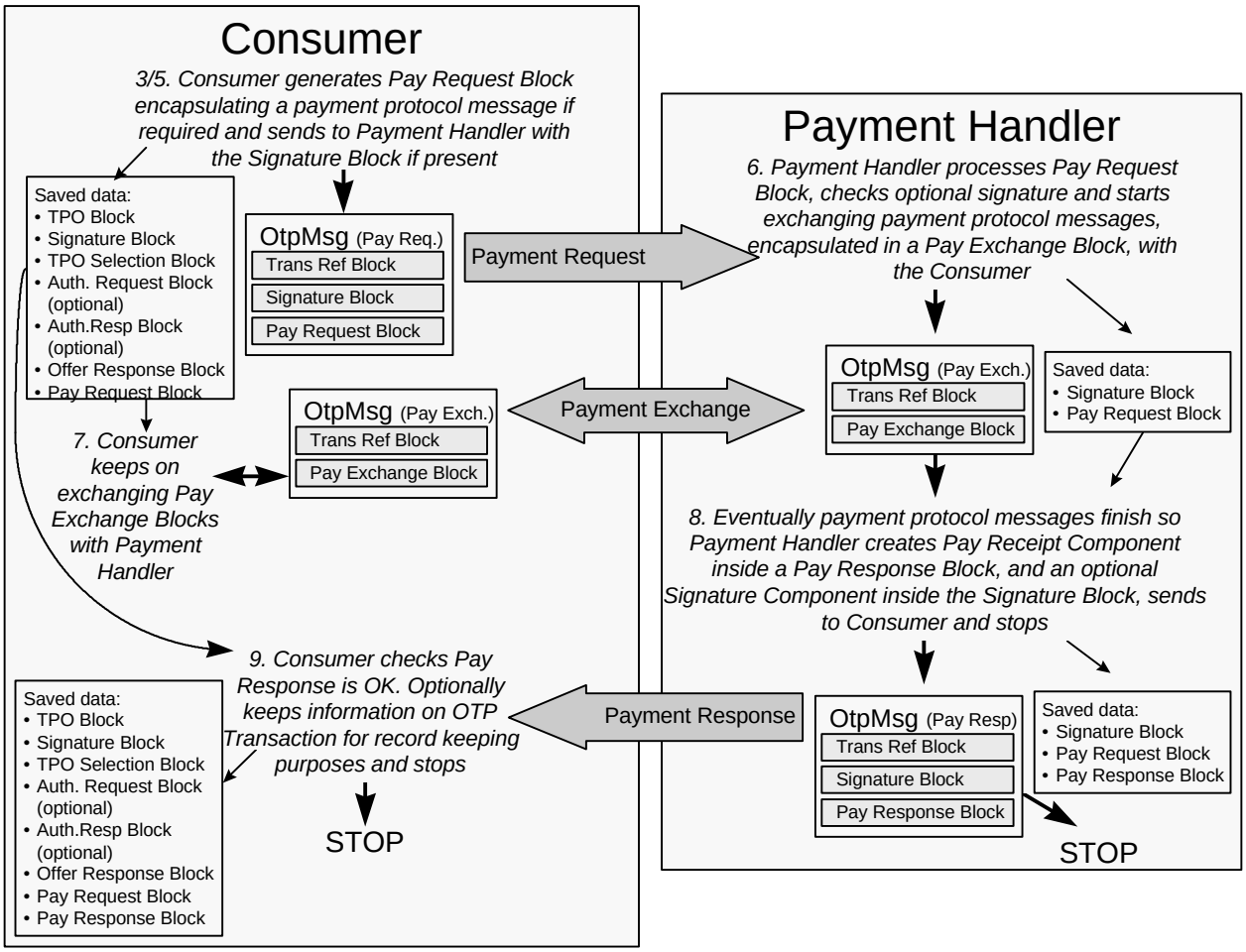


Figure 22 Baseline Deposit Payment Messages

The remainder of this sub-section on the Baseline Deposit OTP Transaction defines the contents of each Trading Block. For most Trading Blocks, the content does not alter with the variations described above. Where differences apply, these are stated.

7.2.4 TPO (Trading Protocol Options) Block

The TPO (Trading Protocol Options) Block (see section 7.3.2) must contain the following Trading Components:

- one Protocol Options Component which defines the options which apply to the whole OTP Transaction. See Section 5.1.
- one Brand List Component (see section 5.6) which contains the payment brand and protocols which may be selected for use in the Payment Exchange.

7.2.5 TPO Selection Block

The TPO Selection Block (see section 6.2) is only used by Baseline Deposit with Authentication. It contains:

- one Brand Selection Component (see section 5.7) for use in the Payment Exchange. It contains the results of the consumer selecting a Payment Brand and Payment Protocol from the list provided in the Brand List Component.

7.2.6 Authentication Request Block

The Authentication Request Block (see section 6.4) must contain the following Trading Component:

- one Authentication Data Component (see section 5.2)

7.2.7 Authentication Response Block

The Authentication Response Block (see section 6.5) must contain the following Trading Component:

- one Authentication Response Component (see section 5.3).

7.2.8 Offer Response Block

The Offer Response Block (see section 6.3) must contain the following components:

- zero or one Authentication Data Component (see section 5.2) An Authentication Data Component is required for each Payment Exchange, where its Payment Component contains an `AuthDataRef`attribute
- one Order Component (see section 5.4) which contains details about the deposit, for example the amount of value being deposited and any fees which might apply
- one Payment Component (see section 5.8) which contains information about the payment which is to be made
- Organisation Components (see section 5.5) with the following roles:
  - the `Merchant`who is accepting the deposit
  - the `Consumer`who is making the deposit
  - the `PaymentHandler`for the payment. The "ID" of the Payment Handler Organisation Component is contained within the `VaOrgRef` attribute of the Payment Component (see section 5.8)
- one Delivery Component (see section 5.11) with the `DelivExch`attribute set to `False`.

[Note] *A role of Merchant is used in the above description since a service (i.e. a deposit of electronic cash) is being offered in return for a fee, for example bank charges of some kind. The term "Financial Institution" is used in the diagrams and in the text for clarity.*

[Note End]

7.2.9 Signature Block (Offer Response)

If the Baseline Deposit Offer Response is being digitally signed then a Signature Block must be included in the same OTP message that contains an "Offer Response" Signature Component (see section 5.16). The Signature Component contains hashes of the following XML elements:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Offer Response Block within the OTP Transaction. It contains information that identifies the OTP Message and OTP Transaction
- the Transaction Id Component (see section 2.3.1) which globally uniquely identifies the OTP Transaction
- the following components of the Offer Response Block:
  - the Authentication Data Component if present
  - the Order Component
  - the Payment Component
  - all the Organisation Components present, and
  - the Delivery Component,
- the following components of the TPO Block :
  - the Protocol Options Component, and
  - the Brand List Component

If the Baseline Deposit is a Baseline Deposit with Authentication then the Signature Component additionally contains a hash of the following:

- the Brand Selection Component contained in the TPO Selection Block.

7.2.10 Payment Request Block

The Payment Request Block (see section 6.6) contains:

- the following components copied from the Offer Response Block:
  - the Authentication Data Component if present
  - the Payment Component
  - the Organisation Components with the roles of: `Merchant` and `PaymentHandler`
- the following component from the TPO Block:
  - the Brand List Component
- one Brand Selection Component either:
  - copied from the Offer Response Block if the deposit is a Baseline Deposit with Authentication, or
  - created by the Consumer, containing the payment brand and payment protocol selected, if the deposit is a Baseline Deposit without Authentication
- one Payment Scheme Component (see section 5.9) if required by the payment method used (see the Payment Method supplement to determine if this is needed).

Payment Handlers should check that they are authorised to carry out the Payment (see section 4 Security Considerations).

7.2.11 Signature Block (Payment Request)

If the Baseline Deposit Offer Response Block was signed then the OTP Message that contains the Payment Request Block must also contain a Signature Block with a copy of the "Offer Response" Signature Component.

7.2.12 Payment Exchange Block

The Payment Exchange Block (see section 6.7) contains:

- one Payment Scheme Component (see section 5.9) which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain.

7.2.13 Payment Response Block

The Payment Response Block (see section 6.8) contains:

- one Payment Receipt Component(see section 5.10) which contains scheme specific data which can be used to verify the payment occurred
- one Payment Scheme Component (see section 5.9) if required which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain
- the "Offer Response" Signature Component (see section 5.16) from the Payment Request Block if present.

7.2.14 Signature Block (Payment Response)

If a signed Payment Receipt is being provided, indicated by the `SignedPayReceipt`attribute of the Payment Component of the Offer Response Block being set to `True`, then the OTP Message that contains the Payment Response Block must also contain a Signature Block with a "Payment Receipt" Signature Component which contains hashes of the following:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Payment Response Block,
- the Transaction Id Component (see section 2.3.1) within the Transaction Reference Block that globally uniquely identifies the OTP Transaction,
- the Payment Receipt Component from the Payment Response Block and
- the "Offer Response" Signature Component from the Payment Request Block if present.

7.3 Baseline Purchase OTP Transaction

The Baseline Purchase OTP Transaction supports the purchase of goods or services using any payment method. It consists of the following Trading Exchanges:

- an Offer Exchange (see section 1.2.1),

- a Payment Exchange (see section 1.2.2), and
- an optional Delivery Exchange (see section 1.2.3)

These Trading Exchanges are implemented by a set of predefined OTP Messages (see section 0) which are exchanged between the Trading Roles (see section 1.1). Each OTP Message contains Trading Blocks (see section 0) which contain the Trading Components (see section 5) which are required by the Trading Exchanges.

The Trading Blocks used by the Baseline Purchase OTP Transaction are:

- Trading Protocol Options Block
- TPO Selection Block
- Offer Response Block
- Payment Request Block
- Payment Exchange Block
- Payment Response Block
- Delivery Request Block
- Delivery Response Block
- Signature Block

7.3.1 Baseline Purchase Variations

The Baseline Purchase OTP Transaction occurs in two basic forms:

- Brand Dependent Purchase. Where the content of the offer, e.g. the order details, amount, delivery details, etc., are dependent on the payment brand and protocol selected by the consumer, and
- Brand Independent Purchase. Where the content of the offer is not dependent on the payment brand and protocol selected.

Further variation is supported in that:

- the Delivery Exchange is optional, and
- the Delivery Response Block may be sent to the consumer either:
  - at the same time as the Payment Response Block, or
  - after the Payment Response Block as the result of the Consumer sending the Delivery Handler a Delivery Request Block.

7.3.1.1 Brand Dependent Purchases

In a Brand Dependent Purchase the TPO Block and the Offer Response Block are sent separately by the Merchant to the Consumer, i.e.:

- the Brand List Component is sent to the Consumer in a TPO Block,
- the Consumer selects a Payment Brand, Payment Protocol and optionally a Currency and amount from the Brand List Component

- the Consumer sends the selected brand, protocol and currency/amount back to the Merchant in a TPO Selection Block, and
- the Merchant uses the information received to define the content of and then send the Offer Response Block to the Consumer.

In a Brand Independent Purchase the TPO Block and the Offer Response Block are sent together by the Merchant to the Consumer in the same OTP Message at the start of the OTP Transaction.

These two alternatives are illustrated in the two diagrams below. The first diagram illustrates a Brand Dependent Purchase.

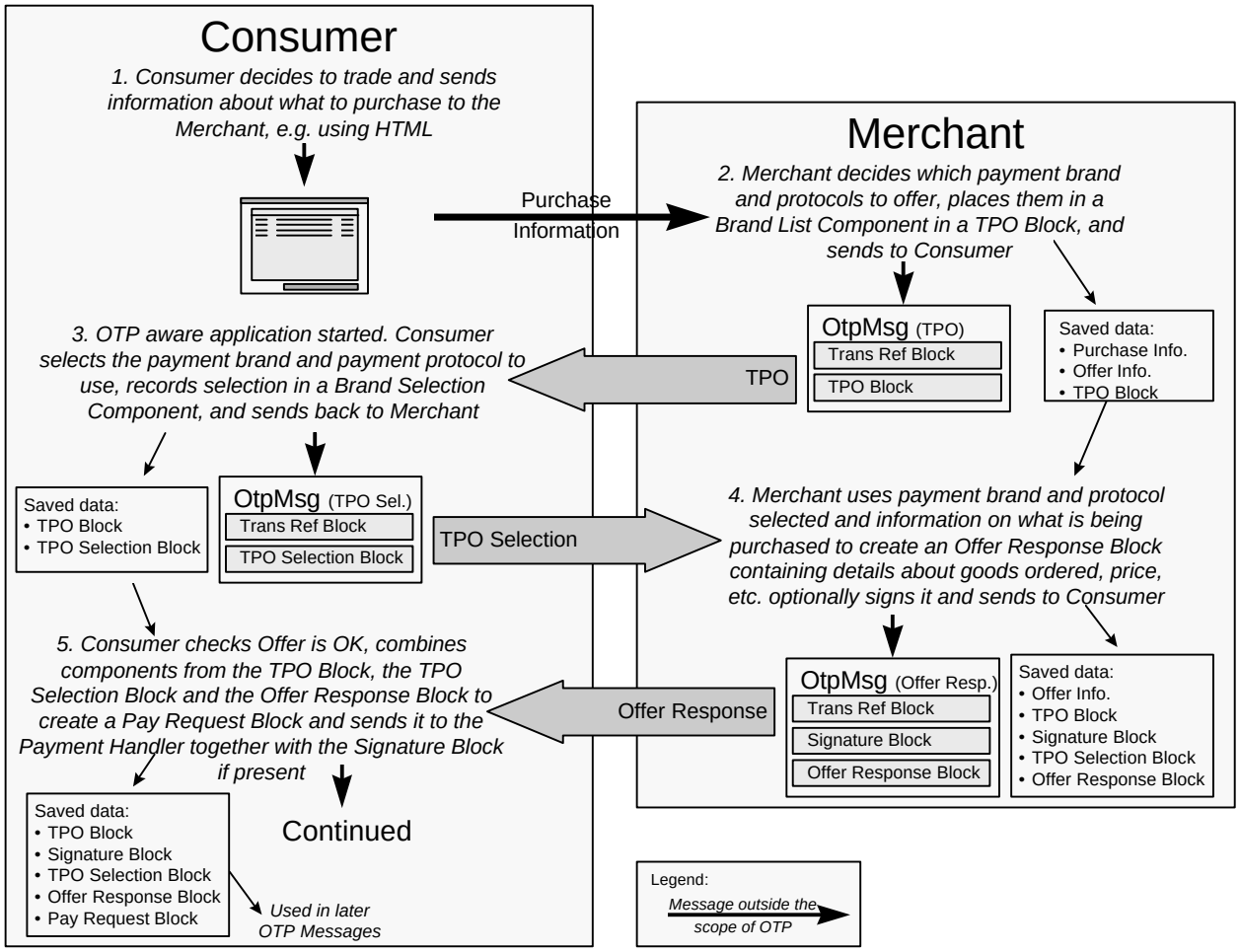


Figure 23 Brand Dependent Baseline Purchase

The second diagram illustrates the Brand Independent Purchase.

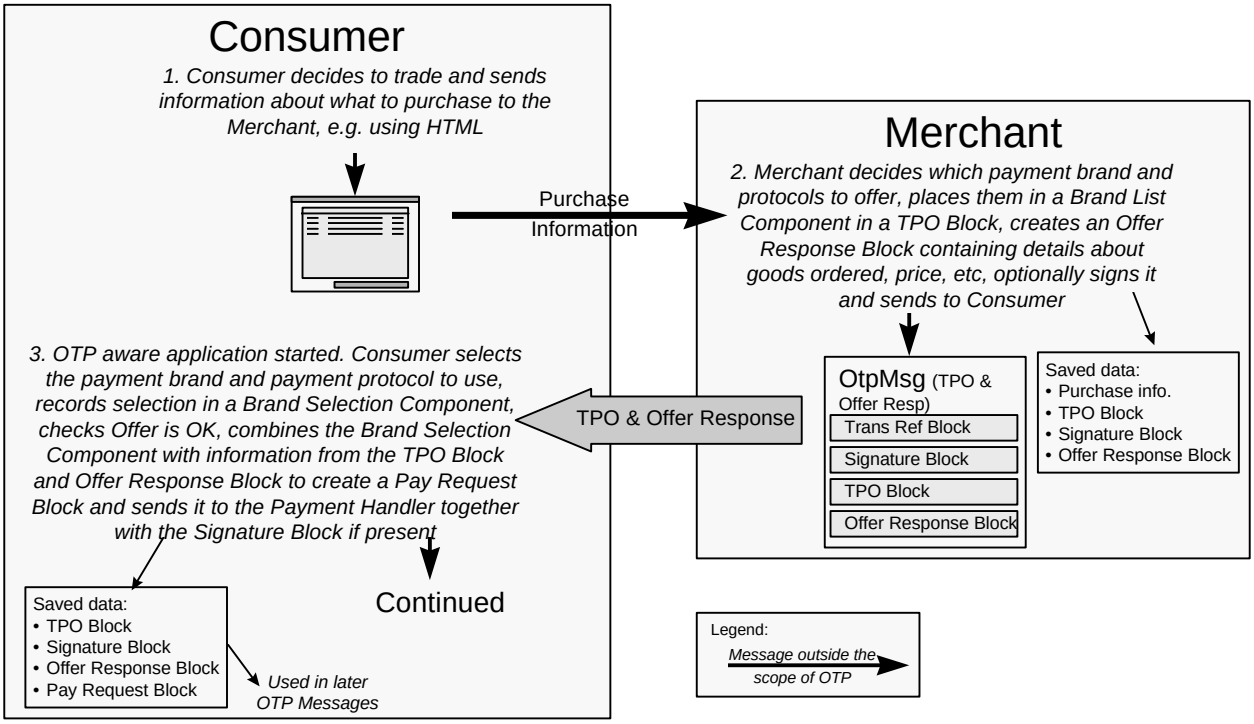


Figure 24 Brand Independent Baseline Purchase

A Brand Independent Purchase always occurs when only one payment brand and protocol is being offered to the Consumer by the Merchant. It is also likely to, but will not necessarily, occur when multiple brands are being offered, the Payment Handler is the same, and all brands use the same set of protocols.

Note that the TPO Block and the Offer Response Block may be sent in separate OTP messages even if the Offer Response Block does not change. However this increases the number of messages in the transaction and is therefore likely to increase transaction response times.

OTP aware applications supporting the Consumer Trading Role must check for the existence of an Offer Response Block in the first OTP Message to determine whether the Baseline Purchase is brand dependent or not.

7.3.1.2 Combining Delivery Response Block and Payment Response Block

The Delivery Response Block and the Payment Response Block may be sent:

- separately by the Payment Handler to the Consumer, i.e.:
  - the Payment Response Block containing a Payment Receipt and optional signature for the payment is sent by the Payment Handler to the Consumer,
  - the Consumer combines these components from the Payment Response Block with components from the Offer Response Block, to create a Delivery Request Block
  - the Consumer sends the Delivery Request Block to the Delivery Handler
  - the Delivery Handler processes the Delivery Request Block and sends a Delivery Response Block back to the Consumer, or

- together, from the Payment Handler to the Consumer, when the Payment Exchange is complete.

These two alternatives are illustrated in the two diagrams below.

The first diagram illustrates when the Delivery Response Block and the Payment Response Block are sent to the Consumer in separate OTP Messages. Note, these diagrams continue where the previous diagrams (Figure 23 and Figure 24) finish.

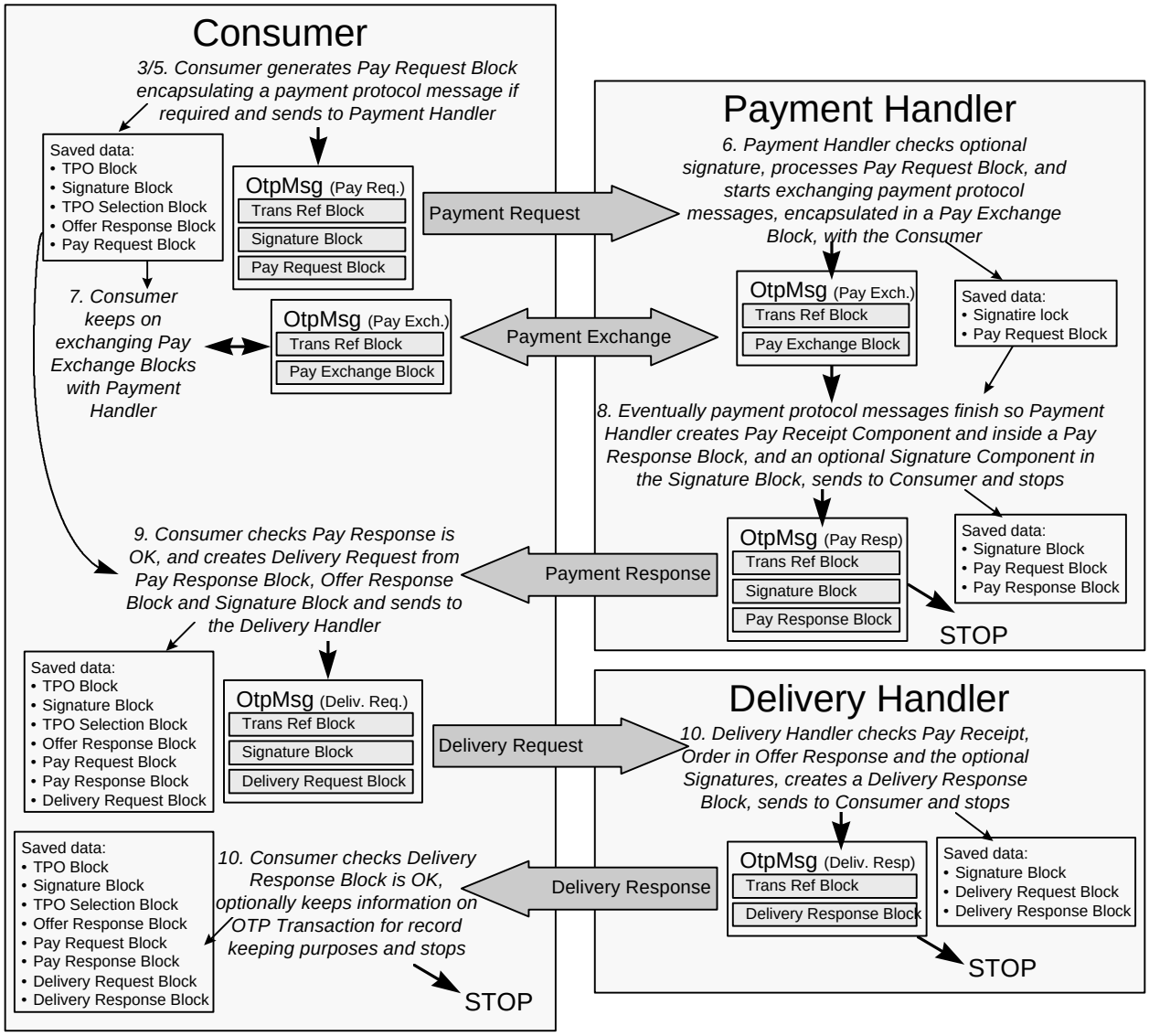


Figure 25 Baseline Purchase, Delivery Response Block and Payment Response Blocks Not Combined

The second diagram illustrates the case when the Delivery Response Block and the Payment Response Block are combined into one OTP Message.

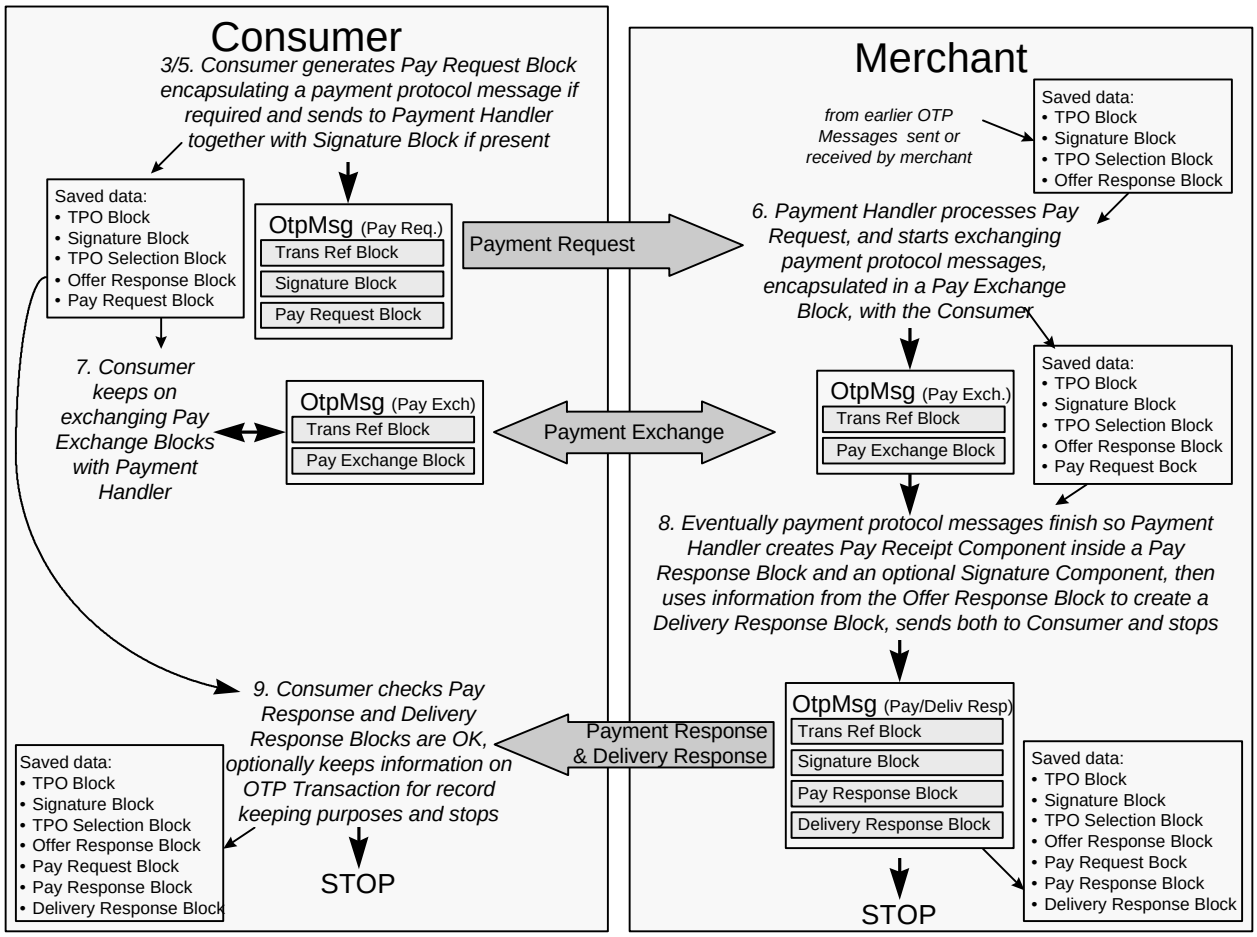


Figure 26 Baseline Purchase, Delivery Response Block and Payment Response Block Combined

The Delivery Response Block and the Payment Response Block may be combined into the same OTP Message only if the Payment Handler has the information available so that she can send the Delivery Response Block. This is likely to, but will not necessarily, occur when the Merchant, the Payment Handler and the Delivery Handler Roles are combined.

The `DelivAndPayResp`attribute of the Delivery Component (see section 5.11) contained within the Offer Response Block (see section 6.3) is set to `True` if the Delivery Response Block and the Payment Response Block are combined into the same OTP Message and is set to `False` if the Delivery Response Block and the Payment Response Block are sent in separate OTP Messages.

7.3.1.3 Optional Delivery Exchange

The final variation of the Baseline Purchase OTP Transactions is a purchase without a delivery step. This is illustrated in the following diagram which continues where the earlier diagrams (Figure 23 and Figure 24) finish.

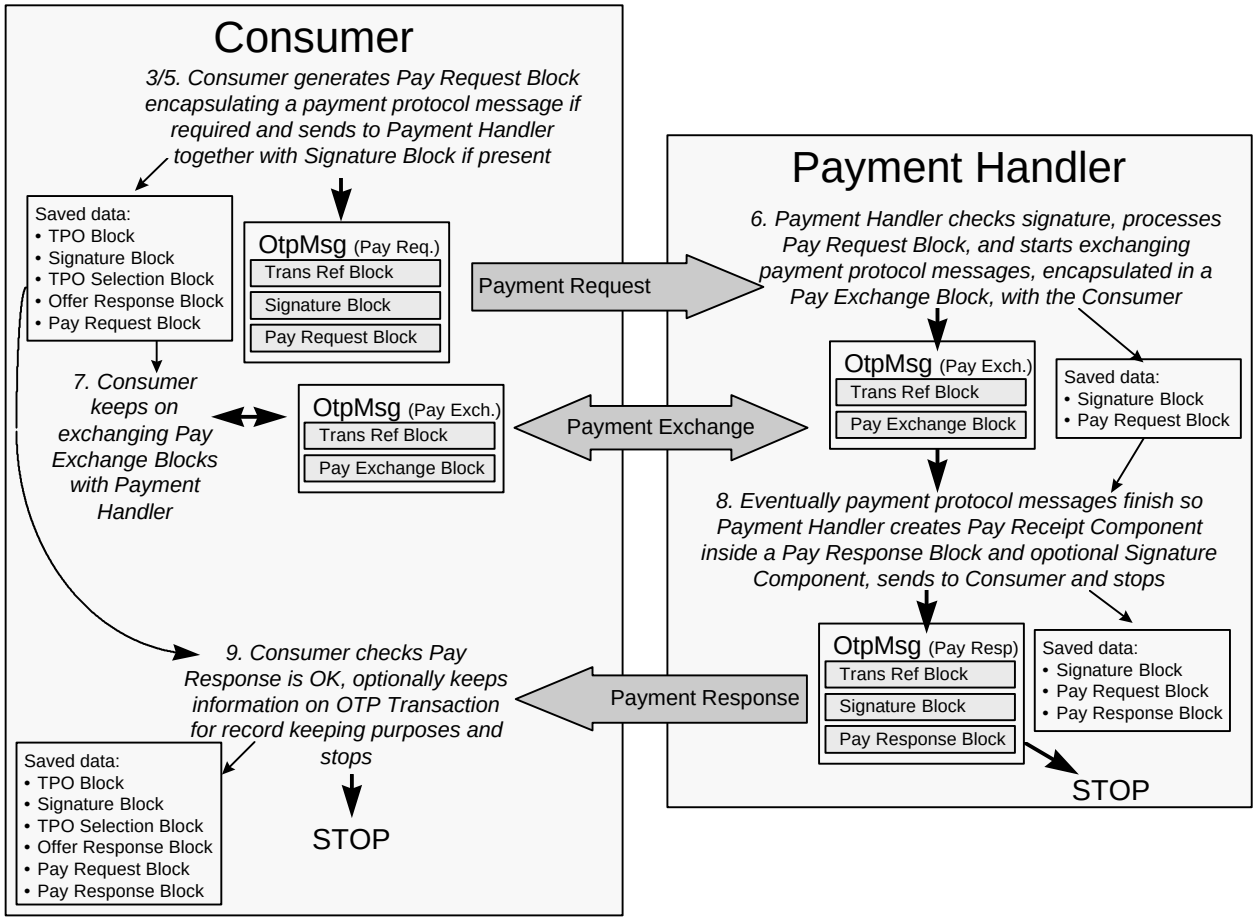


Figure 27 Baseline Purchase, Purchase without Delivery Exchange

The `DelivExch` attribute of the Delivery Component (see section 5.11) contained in the Offer Response Block (see section 6.3) is set to `False` if the Delivery Exchange is omitted and is set to `True` if the Delivery Exchange is included.

7.3.1.4 Combining Variations

The diagram below shows how the different variations in the Baseline Purchase Transaction may be combined.

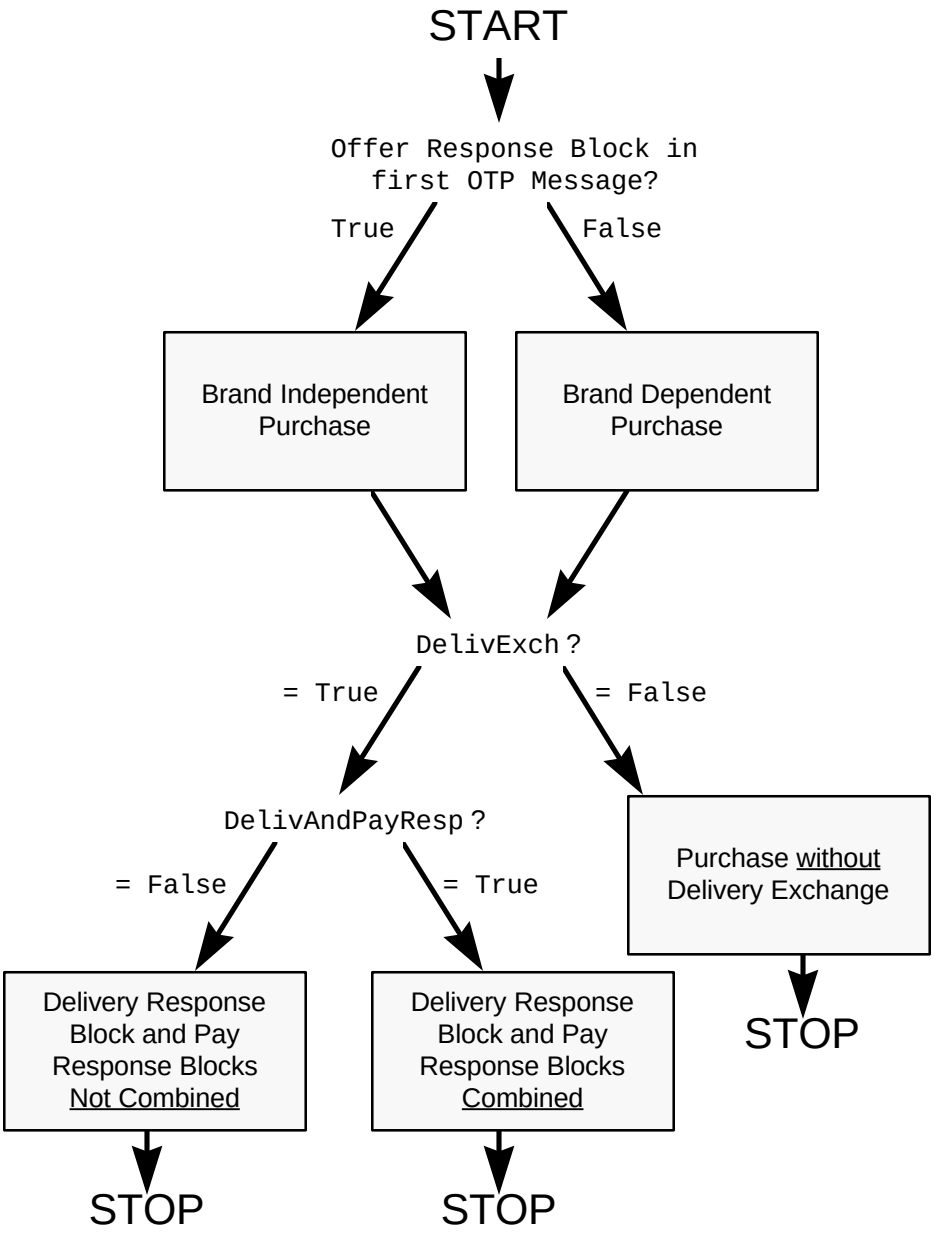


Figure 28 Baseline Purchase Variations

The remainder of this sub-section on the Baseline Purchase OTP Transaction defines the contents of each Trading Block. For most Trading Blocks, the content does not alter with the variations described above. Where differences apply, these are stated.

7.3.2 TPO (Trading Protocol Options) Block

The TPO (Trading Protocol Options) Block (see section 7.3.2) must contain the following Trading Components:

- one Protocol Options Component which defines the options which apply to the whole OTP Transaction. See Section 5.1.
- one Brand List Component (see section 5.6) which contains one or more payment brands and protocols which may be selected for use in the Payment Exchange.

7.3.3 TPO Selection Block

The TPO Selection Block (see section 6.2) is only used by Brand Dependent Purchase. It contains:

- one Brand Selection Component (see section 5.7) for use in the Payment Exchange. It contains the results of the consumer selecting a Payment Brand and Payment Protocol from the list provided in the Brand List Component.

7.3.4 Offer Response Block

The Offer Response Block (see section 6.3) contains the following components:

- zero or one Authentication Data Component (see section 5.2) An Authentication Data Component is required for each Payment Exchange, where its Payment Component (see section 5.8) contains an `AuthDataRef`attribute.
- one Order Component (see section 5.4) which contains details about the goods, services which are being purchased
- one Payment Component (see section 5.8) which contains information about the payment which is to be made
- Organisation Components (see section 5.5) with the following roles:
  - `Merchant`who is providing the goods or services
  - `Consumer`who is making the purchase
  - `PaymentHandler`for the payment. The "ID" of the Payment Handler Organisation Component is contained within the `VaOrgRef` attribute of the Payment Component
- one Delivery Component (see section 5.11) which contains details of the delivery to be made.

If the Baseline Purchase includes a Delivery Exchange then the Offer Response Block must also contain:

- Organisation Components with the following roles:
  - `DeliveryHandler`who will be delivering the goods or services
  - `DelivTo`i.e. the person or organisation which is to take delivery

7.3.5 Signature Block (Offer Response)

If the Baseline Purchase Offer Response is being digitally signed then a Signature Block must be included in the same OTP message that contains an "Offer Response" Signature Component (see section 5.16). The Signature Component contains hashes of the following XML elements:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Offer Response Block within the OTP Transaction. It contains information that identifies the OTP Message and OTP Transaction
- the Transaction Id Component (see section 2.3.1) which globally uniquely identifies the OTP Transaction
- the following components of the Offer Response Block:
  - the Authentication Data Component if present
  - the Order Component
  - the Payment Component
  - all the Organisation Components present, and
  - the Delivery Component,
- the following components of the TPO Block :
  - the Protocol Options Component, and
  - the Brand List Component

If the Baseline Purchase is a Brand Dependent Purchase then the Signature Component additionally contains a hash of the following:

- the Brand Selection Component contained in the TPO Selection Block.

7.3.6 Payment Request Block

The Payment Request Block (see section 6.6) contains:

- the following components copied from the Offer Response Block:
  - the Authentication Data Component if present
  - the Payment Component
  - the Organisation Components with the roles of: `Merchant` and `PaymentHandler`
- the following component from the TPO Block:
  - the Brand List Component
- one Brand Selection Component either:
  - copied from the Offer Response Block if the purchase is a Brand Dependent Purchase, or
  - created by the Consumer, containing the payment brand and payment protocol selected, if the purchase is a Brand Independent Purchase
- one Payment Scheme Component (see section 5.9) if required by the payment method used (see the Payment Method supplement to determine if this is needed).

Payment Handlers should check that they are authorised to carry out the Payment (see section 4 Security Considerations).

7.3.7 Signature Block (Payment Request)

If the Baseline Purchase Offer Response Block was signed then the OTP Message that contains the Payment Request Block must also contain a Signature Block with a copy of the "Offer Response" Signature Component.

7.3.8 Payment Exchange Block

The Payment Exchange Block (see section 6.7) contains:

- one Payment Scheme Component (see section 5.9) which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain.

7.3.9 Payment Response Block

The Payment Response Block (see section 6.8) contains:

- one Payment Receipt Component (see section) which contains scheme specific data which can be used to verify the payment occurred
- one Payment Scheme Component (see section 5.9) if required which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain
- the "Offer Response" Signature Component (see section 5.16) from the Payment Request Block if present.

7.3.10 Signature Block (Payment Response)

If a signed Payment Receipt is being provided, indicated by the `SignedPayReceipt`attribute of the Payment Component of the Offer Response Block being set to `True`, then the OTP Message that contains the Payment Response Block must also contain a Signature Block with a "Payment Receipt" Signature Component which contains hashes of the following:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Payment Response Block,
- the Transaction Id Component (see section 2.3.1) within the Transaction Reference Block that globally uniquely identifies the OTP Transaction,
- the Payment Receipt Component from the Payment Response Block and
- the "Offer Response" Signature Component from the Payment Request Block if present.

7.3.11 Delivery Request Block

The Delivery Request Block (see section 6.9) contains:

- the following components copied from the Offer Response Block:
  - the Order Component (see section 5.4)

- the Organisation Component (see section 5.5) with the roles of: `Merchant`, `DeliveryHandler` and `DeliverTo`
- the Delivery Component (see section 5.11)

Payment Handlers should check that they are authorised to carry out the Payment (see section 4 Security Considerations).

7.3.12 Signature Block (Delivery Request)

If the Baseline Purchase Offer Response or Payment Response Blocks were signed then the OTP Message that contains the Delivery Request Block must also contain a Signature Block with a copy of:

- the "Offer Response" Signature Component if present, and/or
- the "Payment Receipt" Signature Component, if present

7.3.13 Delivery Response Block

The Delivery Response Block contains:

- one Delivery Note Component (see section 5.12) which contains delivery instructions about the delivery of goods or services

7.4 Baseline Refund OTP Transaction

In business terms the refund process typically consists of:

- a request for a refund being made by the Consumer to the Merchant, typically supported by evidence to demonstrate:
  - the original trade took place, for example by providing a receipt for the original transaction
  - using some type of authentication, that the consumer requesting the refund is the consumer, or a representative of the consumer, who carried out the original trade
  - the reason why the merchant should make the refund
- the merchant agreeing (or not) to the refund. This may involve some negotiation between the Consumer and the Merchant, and, if the merchant agrees,
- a refund payment by the Merchant to the Consumer.

The Baseline Refund OTP Transaction supports a subset of the above, specifically it supports:

- the optional authentication of the Consumer using an Authentication Exchange (see section 1.2.4), and
- the refund payment by the Merchant to the Consumer using the following two Trading Exchanges:
  - an Offer Exchange (see section 1.2.1), and
  - a Payment Exchange (see section 1.2.2).

These Trading Exchanges are implemented by a set of predefined OTP Messages (see section 0) which are exchanged between the Trading Roles (see section 1.1). Each OTP Message contains Trading

Blocks (see section 0) which contain the Trading Components (see section 5) which are required by the Trading Exchanges.

The Trading Blocks used by the Baseline Purchase OTP Transaction are:

- Trading Protocol Options Block
- TPO Selection Block
- Authentication Request Block
- Authentication Response Block
- Offer Response Block
- Payment Request Block
- Payment Exchange Block
- Payment Response Block
- Signature Block

7.4.1 Baseline Refund Variations

The Baseline Refund OTP Transaction occurs in two basic forms:

- Baseline Refund with Authentication. Where the Consumer requesting the refund is authenticated before the refund is made, and
- Baseline Refund without Authentication. Where the Consumer is not authenticated before the refund is made.

7.4.2 Baseline Refund Authentication

In Baseline Refund with Authentication an Authentication Exchange occurs before the Offer Exchange containing the details of the refund is provided by the Merchant.

In Baseline Refund without Authentication, there is no Authentication Exchange and the Merchant provides details about the refund immediately at the start of the OTP Transaction.

These two alternatives are illustrated in the two diagrams below. The first diagram illustrates the case when an Authentication Exchange is included.

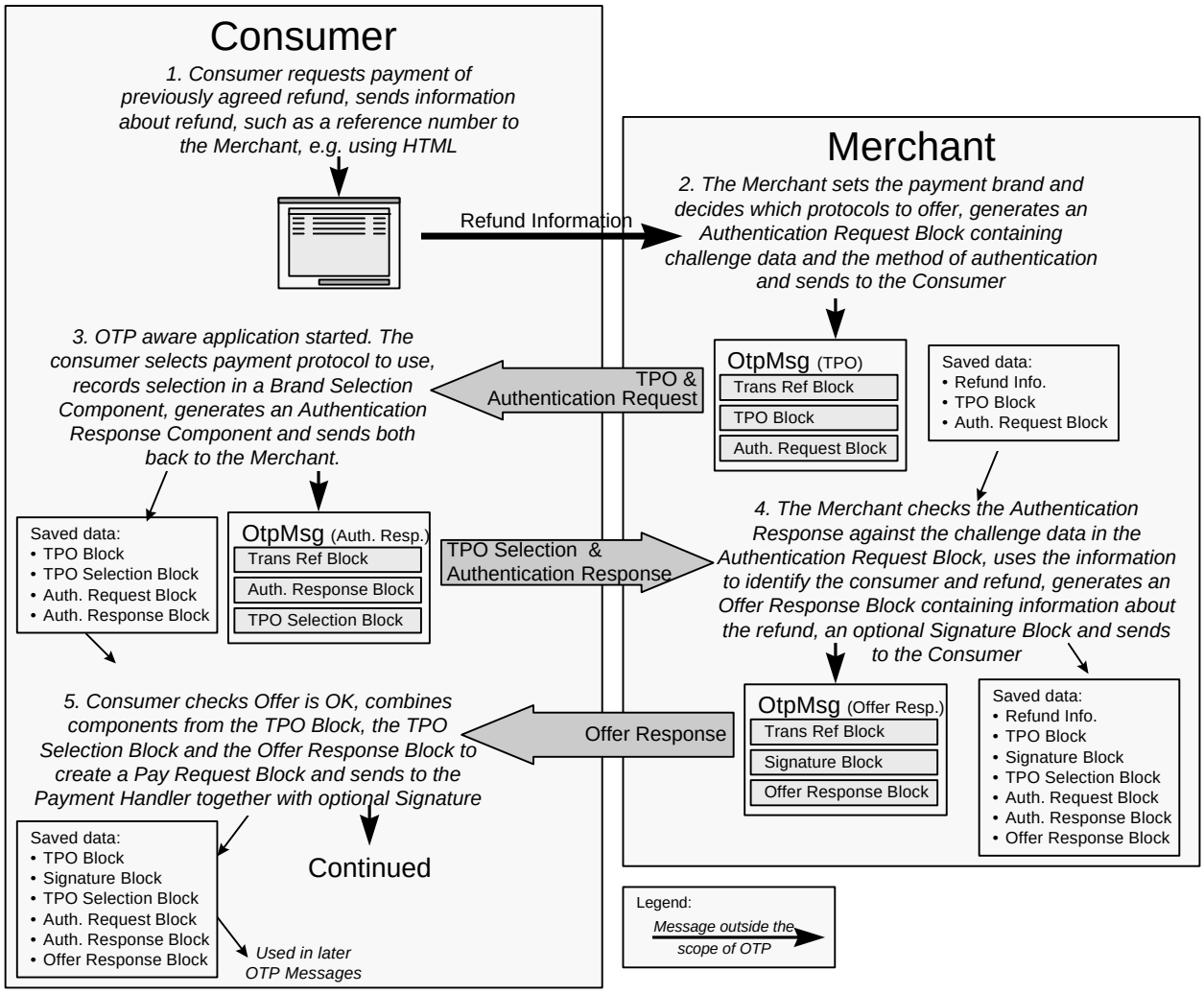


Figure 29 Baseline Refund with Authentication

The second diagram illustrates the case when an Authentication Exchange is not included.

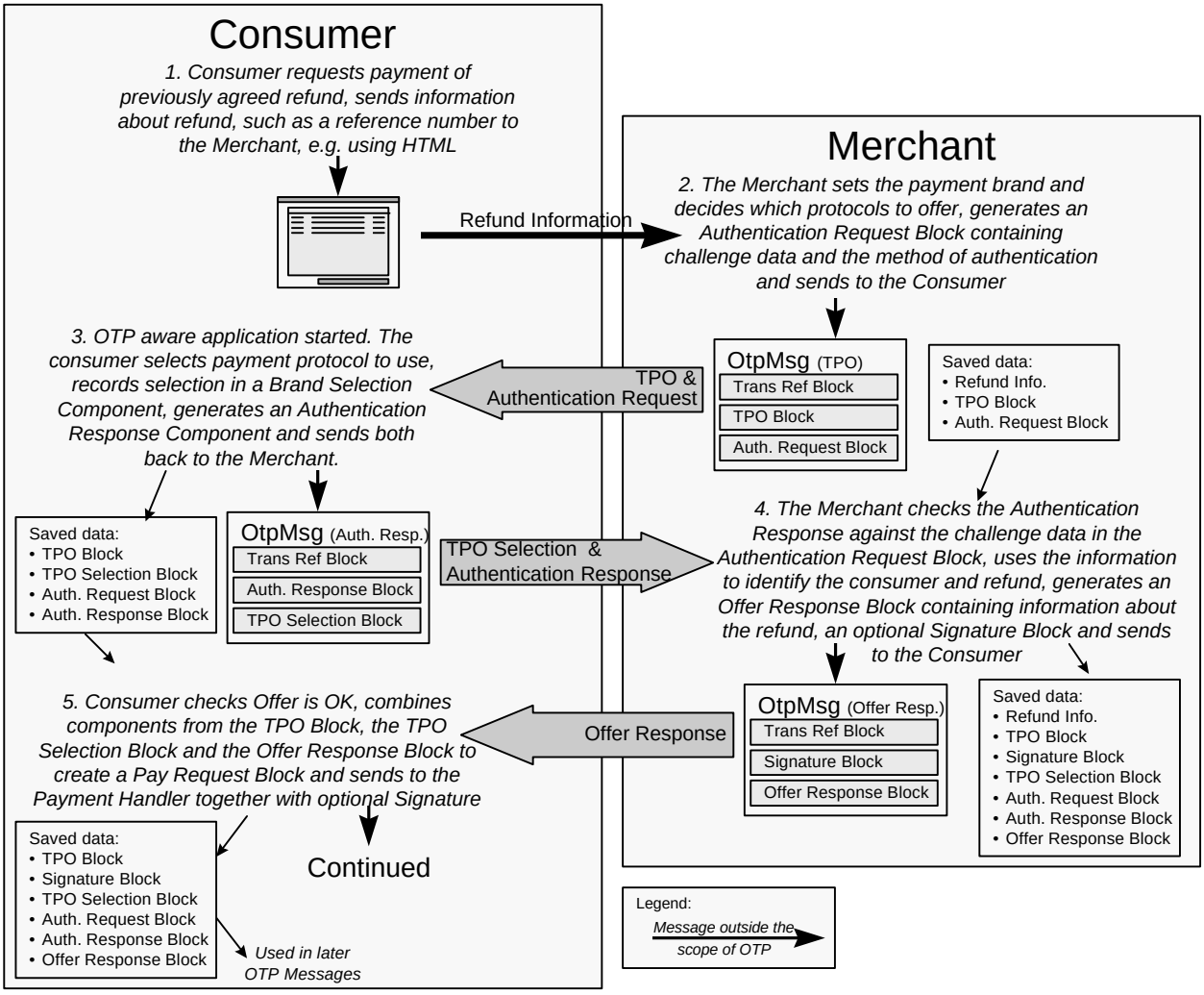


Figure 30 Baseline Refund without Authentication

The Baseline Refund without authentication might be used:

- when authentication of the consumer has been achieved by some other means, for example, the consumer has entered some previously supplied code in order to identify herself and the refund to which the code applies. The code could be supplied, for example on a web page or by e-mail.
- when a previous OTP transaction, for example a Baseline Authentication, authenticated the consumer, and a secure channel has been maintained, therefore the authenticity of the consumer is known and therefore the previously agreed refund can be identified.

OTP aware applications supporting the Consumer Trading Role must check for the existence of an Authentication Request Block in the first OTP Message to determine whether the Baseline Refund includes an Authentication Exchange or not.

7.4.3 Baseline Refund Payment Messages

Once the Offer Response Trading Block has been received, the sequence of OTP Messages illustrated in Figure 31 occurs. These are the same whether or not an Authentication of the Consumer has occurred. Note that these continue where the previous diagrams (Figure 29 and Figure 30) finish.

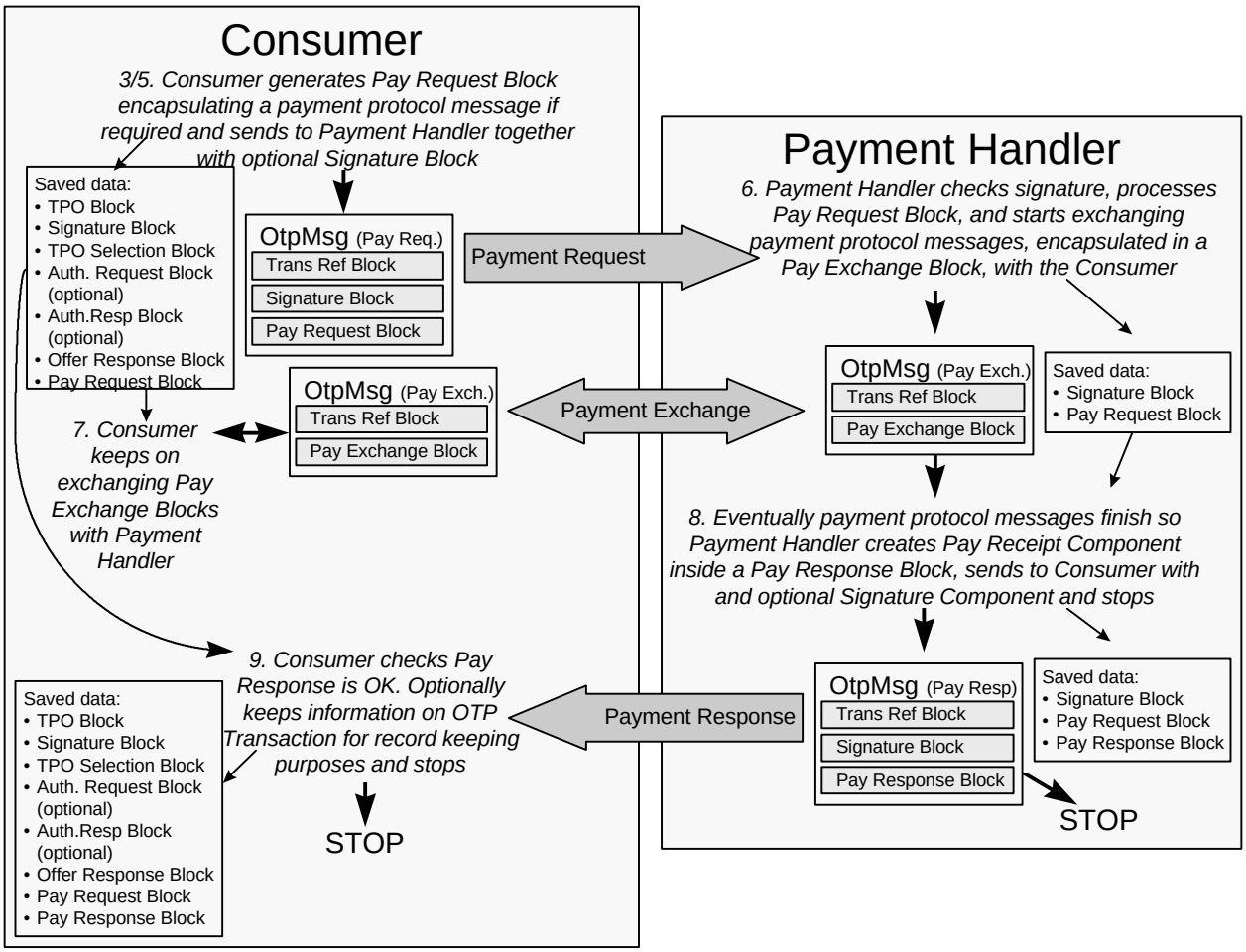


Figure 31 Baseline Refund Payment Messages

The remainder of this sub-section on the Baseline Refund OTP Transaction defines the contents of each Trading Block. For most Trading Blocks, the content does not alter with the variations described above. Where differences apply, these are stated.

7.4.4 TPO (Trading Protocol Options) Block

The TPO (Trading Protocol Options) Block (see section 7.3.2) must contain the following Trading Components:

- one Protocol Options Component which defines the options which apply to the whole OTP Transaction. See Section 5.1.

- one Brand List Component (see section 5.6) which contains the payment brand and protocols which may be selected for use in the Payment Exchange.

7.4.5 TPO Selection Block

The TPO Selection Block (see section 6.2) is only used by Baseline Refund with Authentication. It contains:

- one Brand Selection Component (see section 5.7) for use in the Payment Exchange. It contains the results of the consumer selecting a Payment Brand and Payment Protocol from the list provided in the Brand List Component.

7.4.6 Authentication Request Block

The Authentication Request Block (see section 6.4) must contain the following Trading Component:

- one Authentication Data Component (see section 5.2)

7.4.7 Authentication Response Block

The Authentication Response Block (see section 6.5) must contain the following Trading Component:

- one Authentication Response Component (see section 5.3).

7.4.8 Offer Response Block

The Offer Response Block (see section 6.3) must contain the following components:

- zero or one Authentication Data Component (see section 5.2) An Authentication Data Component is required for each Payment Exchange, where its Payment Component contains an `AuthDataRef` attribute
- one Order Component (see section 5.4) which contains details about the refund, for example the amount being refunded and any conditions which might apply
- one Payment Component (see section 6.2) which contains information about the payment which is to be made
- Organisation Components (see section 5.5) with the following roles:
  - the `Merchant` who is making the refund
  - the `Consumer` who is requesting the refund
  - the `PaymentHandler` for the payment. The "ID" of the Payment Handler Organisation Component is contained within the `VaOrgRef` attribute of the Payment Component
- one Delivery Component (see section 5.11) with the `DelivExch` attribute set to `False`.

7.4.9 Signature Block (Offer Response)

If the Baseline Refund Offer Response is being digitally signed then a Signature Block must be included in the same OTP message that contains an "Offer Response" Signature Component (see section 5.16). The Signature Component contains hashes of the following XML elements:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Offer Response Block within the OTP Transaction. It contains information that identifies the OTP Message and OTP Transaction
- the Transaction Id Component (see section 2.3.1) which globally uniquely identifies the OTP Transaction
- the following components of the Offer Response Block:
  - the Authentication Data Component if present
  - the Order Component
  - the Payment Component
  - all the Organisation Components present, and
  - the Delivery Component,
- the following components of the TPO Block :
  - the Protocol Options Component, and
  - the Brand List Component

If the Baseline Refund is a Baseline Refund with Authentication then the Signature Component additionally contains a hash of the following:

- the Brand Selection Component contained in the TPO Selection Block.

7.4.10 Payment Request Block

The Payment Request Block (see section 6.6) contains:

- the following components copied from the Offer Response Block:
  - the Authentication Data Component if present
  - the Payment Component
  - the Organisation Components with the roles of: `Merchant` and `PaymentHandler`
- the following component from the TPO Block:
  - the Brand List Component
- one Brand Selection Component either:
  - copied from the Offer Response Block if the refund is a Baseline Refund with Authentication, or
  - created by the Consumer, containing the payment brand and payment protocol selected, if the refund is a Baseline Refund with Authentication
- one Payment Scheme Component (see section 5.9) if required by the payment method used (see the Payment Method supplement to determine if this is needed).

Payment Handlers should check that they are authorised to carry out the Payment (see section 4 Security Considerations).

7.4.11 Signature Block (Payment Request)

If the Baseline Refund Offer Response Block was signed then the OTP Message that contains the Payment Request Block must also contain a Signature Block with a copy of the "Offer Response" Signature Component.

7.4.12 Payment Exchange Block

The Payment Exchange Block (see section 6.7) contains:

- one Payment Scheme Component (see section 5.9) which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain.

7.4.13 Payment Response Block

The Payment Response Block (see section 6.8) contains:

- one Payment Receipt Component (see section 5.10) which contains scheme specific data which can be used to verify the payment occurred
- one Payment Scheme Component (see section 5.9) if required which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain
- the "Offer Response" Signature Component (see section 5.16) from the Payment Request Block if present.

7.4.14 Signature Block (Payment Response)

If a signed Payment Receipt is being provided, indicated by the `SignedPayReceipt`attribute of the Payment Component of the Offer Response Block being set to `True`, then the OTP Message that contains the Payment Response Block must also contain a Signature Block with a "Payment Receipt" Signature Component which contains hashes of the following:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Payment Response Block,
- the Transaction Id Component (see section 2.3.1) within the Transaction Reference Block that globally uniquely identifies the OTP Transaction,
- the Payment Receipt Component from the Payment Response Block and
- the "Offer Response" Signature Component from the Payment Request Block if present.

7.5 Baseline Withdrawal OTP Transaction

The Baseline Withdrawal OTP Transaction supports the withdrawal of electronic cash from a Financial Institution.

[Note] *The Financial Institution has, in OTP terminology, a role of merchant in that a service (i.e. a withdrawal of electronic cash) is being offered in return for a fee, for example bank charges of some kind. The term "Financial Institution" is used in the diagrams and in the text for clarity.*

[Note End]

The Baseline Withdrawal OTP Transaction consists of the following Trading Exchanges:

- an optional Authentication Exchange (see section 1.2.4),
- an Offer Exchange (see section 1.2.1), and
- a Payment Exchange (see section 1.2.2).

These Trading Exchanges are implemented by a set of predefined OTP Messages (see section 0) which are exchanged between the Trading Roles (see section 1.1). Each OTP Message contains Trading Blocks (see section 0) which contain the Trading Components (see section 5) which are required by the Trading Exchanges.

The Trading Blocks used by the Baseline Purchase OTP Transaction are:

- Trading Protocol Options Block
- TPO Selection Block
- Authentication Request Block
- Authentication Response Block
- Offer Response Block
- Payment Request Block
- Payment Exchange Block
- Payment Response Block
- Signature Block

7.5.1 Baseline Withdrawal Variations

The Baseline Withdrawal OTP Transaction occurs in two basic forms:

- Baseline Withdrawal with Authentication. Where the Consumer making the withdrawal is authenticated before the withdrawal is made, and
- Baseline Withdrawal without Authentication. Where the Consumer is not authenticated before the withdrawal is made.

7.5.2 Baseline Withdrawal Authentication

In Baseline Withdrawal with Authentication an Authentication Exchange occurs before the Offer Exchange containing the details of the withdrawal is provided by the Financial Institution.

In Baseline Withdrawal without Authentication, there is no Authentication Exchange and the Financial Institution provides details about the withdrawal immediately at the start of the OTP Transaction.

These two alternatives are illustrated in the two diagrams below. The first diagram illustrates the case when an Authentication Exchange is included.

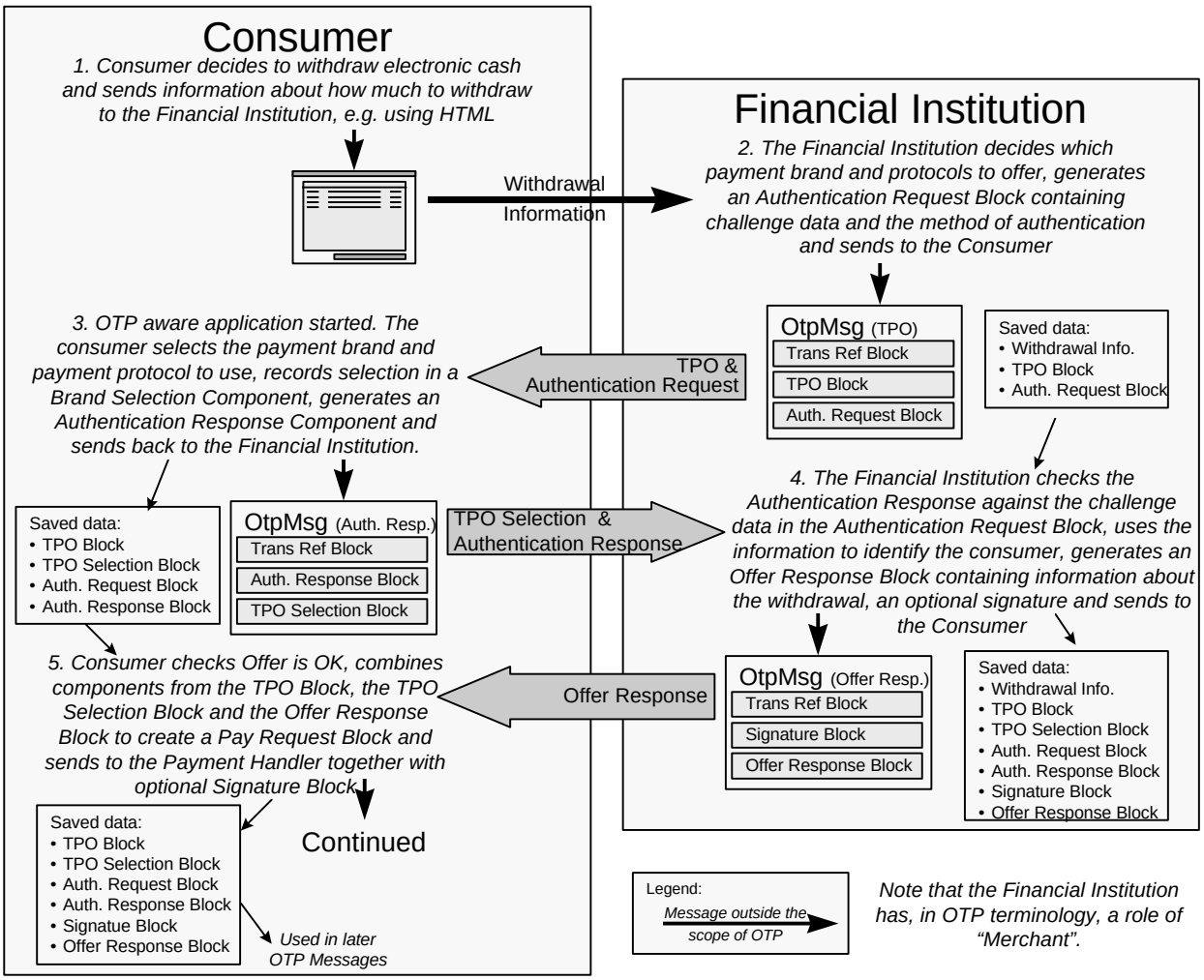


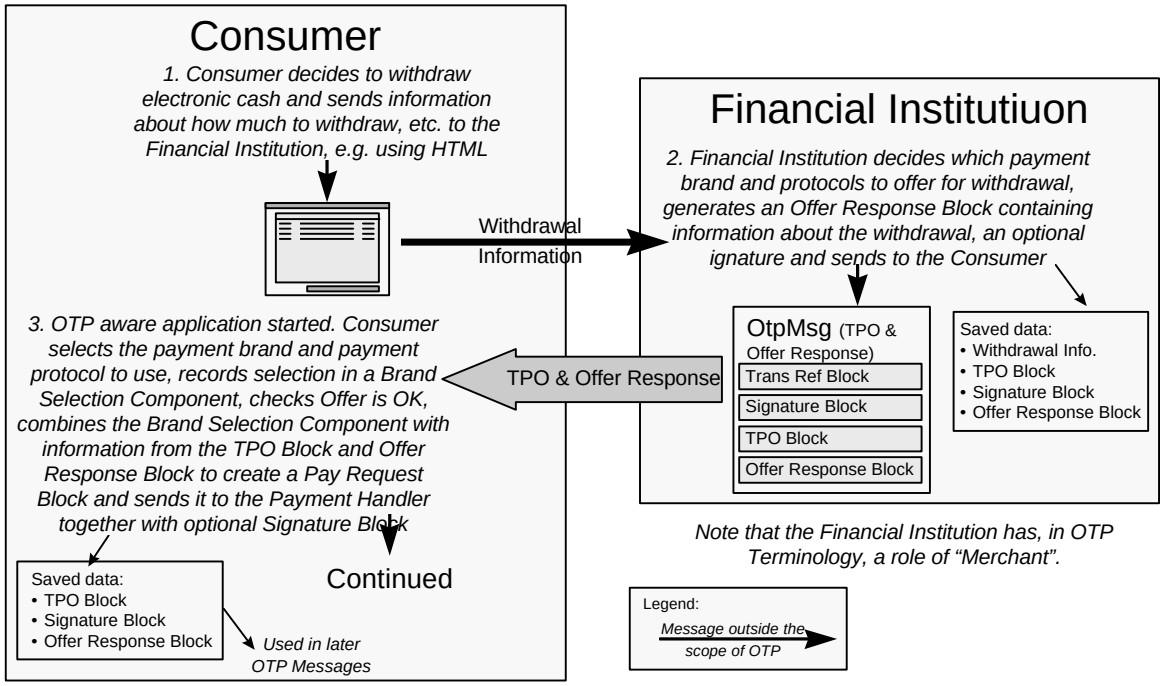
Figure 32 Baseline Withdrawal with Authentication

Note that the above diagram:

- describes the general case where a Financial Institution can offer withdrawal of several different types of electronic cash. In practice usually only one form of electronic cash may be offered. However, there may be several different protocols which may be used for the same "brand" of electronic cash

- the financial institution may use the results of the authentication to identify not only the consumer but also the account from which the withdrawal is to be made. If no single account can be identified, then it must be obtained by other means. For example:
  - the consumer could specify the account number in the initial dialogue (see step 1), or
  - the consumer could have been identified earlier, for example using a Baseline Authentication OTP Transaction, and an account selected from a list provided by the Financial Institution.

The second diagram illustrates the case when an Authentication Exchange is not included.



Baseline Withdrawal without Authentication

Figure 33 Baseline Withdrawal without Authentication

The Baseline Withdrawal without Authentication might be used:

- when a previous OTP transaction, for example a Baseline Deposit or a Baseline Authentication, authenticated the consumer, and a secure channel has been maintained, therefore the authenticity of the consumer is known
- when authentication is achieved as part of a proprietary payment protocol and is therefore included in the Payment Exchange
- when authentication of the consumer has been achieved by some other means, for example, by using a pass phrase, or a proprietary banking software solution.

OTP aware applications supporting the Consumer Trading Role must check for the existence of an Authentication Request Block in the first OTP Message to determine whether the Baseline Withdrawal includes an Authentication Exchange or not.

7.5.3 Baseline Withdrawal Payment Messages

Once the Offer Response Trading Block has been received, the sequence of OTP Messages illustrated in Figure 22 occurs. These are the same whether or not an Authentication of the Consumer has occurred. Note that these continue where the previous diagrams (Figure 20 and Figure 21) finish.

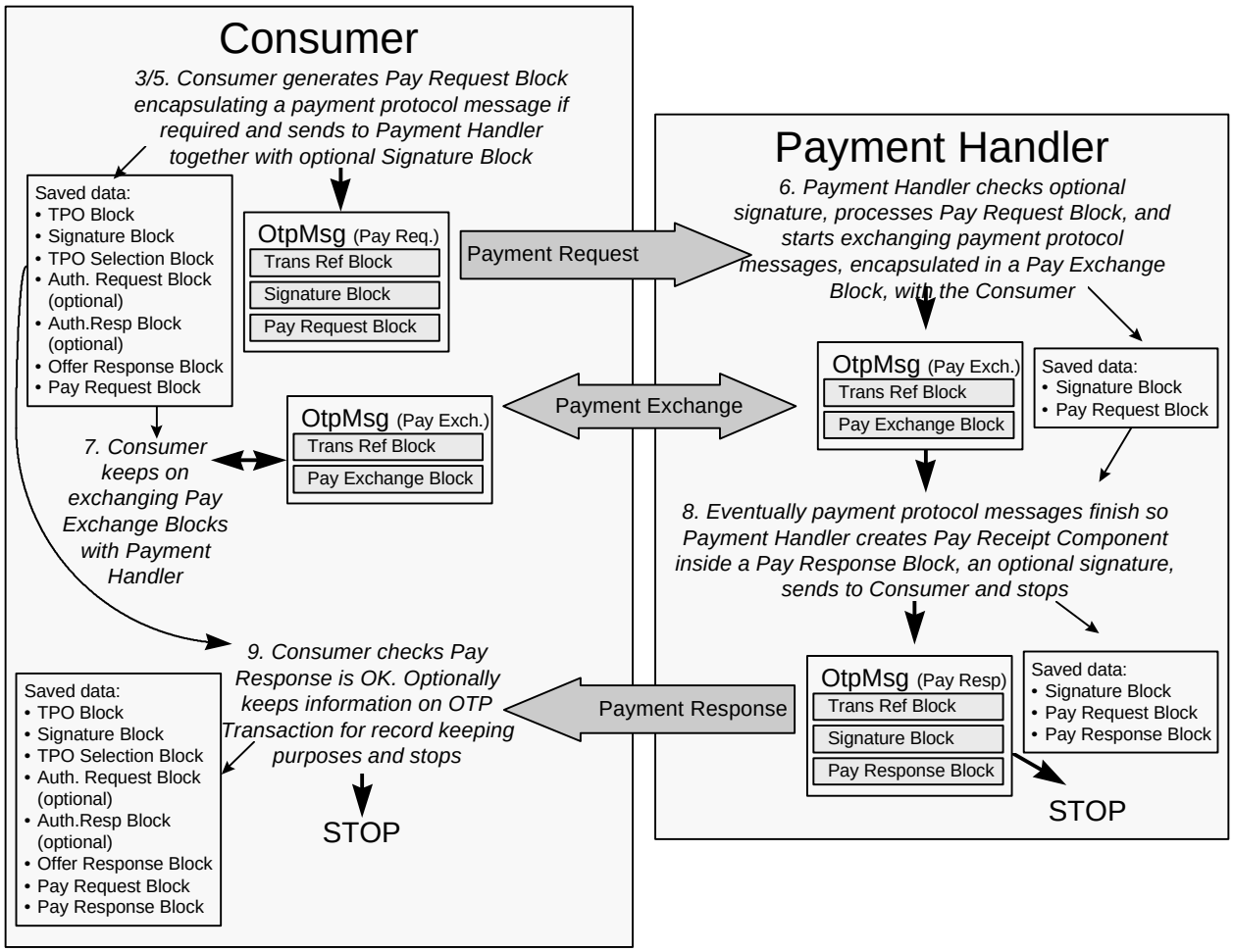


Figure 34 Baseline Withdrawal Payment Messages

The remainder of this sub-section on the Baseline Withdrawal OTP Transaction defines the contents of each Trading Block. For most Trading Blocks, the content does not alter with the variations described above. Where differences apply, these are stated.

7.5.4 TPO (Trading Protocol Options) Block

The TPO (Trading Protocol Options) Block (see section 7.3.2) must contain the following Trading Components:

- one Protocol Options Component which defines the options which apply to the whole OTP Transaction. See Section 5.1.

- one Brand List Component (see section 5.6) which contains the payment brand and protocols which may be selected for use in the Payment Exchange.

7.5.5 TPO Selection Block

The TPO Selection Block (see section 6.2) is only used by Baseline Withdrawal with Authentication. It contains:

- one Brand Selection Component (see section 5.7) for use in the Payment Exchange. It contains the results of the consumer selecting a Payment Brand and Payment Protocol from the list provided in the Brand List Component.

7.5.6 Authentication Request Block

The Authentication Request Block (see section 6.4) must contain the following Trading Component:

- one Authentication Data Component (see section 5.2)

7.5.7 Authentication Response Block

The Authentication Response Block (see section 6.5) must contain the following Trading Component:

- one Authentication Response Component (see section 5.3).

7.5.8 Offer Response Block

The Offer Response Block (see section 6.3) must contain the following components:

- zero or one Authentication Data Components (see section 5.2) An Authentication Data Component is required for each Payment Exchange, where its Payment Component contains an `AuthDataRef`attribute
- one Order Component (see section 5.4) which contains details about the withdrawal, for example the amount being withdrawn and any fees which might apply
- one Payment Component (see section 6.2) which contains information about the payment which is to be made
- Organisation Components (see section 5.5) with the following roles:
  - the `Merchant`who is accepting the withdrawal
  - the `Consumer`who is making the withdrawal
  - the `PaymentHandler`for the payment. The "ID" of the Payment Handler Organisation Component is contained within the `VaOrgRef` attribute of the Payment Component
- one Delivery Component (see section 5.11) with the `DelivExch`attribute set to `False`.

[Note] *In the above an Organisation with a role of Merchant is used in that a service (i.e. a withdrawal of electronic cash) is being offered in return for a fee, for example bank charges of some kind. The term "Financial Institution" is used in the diagrams and in the text for clarity.*

[Note End]

7.5.9 Signature Block (Offer Response)

If the Baseline Withdrawal Offer Response is being digitally signed then a Signature Block must be included in the same OTP message that contains an "Offer Response" Signature Component (see section 5.16). The Signature Component contains hashes of the following XML elements:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Offer Response Block within the OTP Transaction. It contains information that identifies the OTP Message and OTP Transaction
- the Transaction Id Component (see section 2.3.1) which globally uniquely identifies the OTP Transaction
- the following components of the Offer Response Block:
  - the Authentication Data Component if present
  - the Order Component
  - the Payment Component
  - all the Organisation Components present, and
  - the Delivery Component,
- the following components of the TPO Block :
  - the Protocol Options Component, and
  - the Brand List Component

If the Baseline Withdrawal is a Baseline Withdrawal with Authentication then the Signature Component additionally contains a hash of the following:

- the Brand Selection Component contained in the TPO Selection Block.

7.5.10 Payment Request Block

The Payment Request Block (see section 6.6) contains:

- the following components copied from the Offer Response Block:
  - the Authentication Data Component if present
  - the Payment Component
  - the Organisation Components with the roles of: `Merchant` and `PaymentHandler`
- the following component from the TPO Block:
  - the Brand List Component
- one Brand Selection Component either:
  - copied from the Offer Response Block if the withdrawal is a Baseline Withdrawal with Authentication, or
  - created by the Consumer, containing the payment brand and payment protocol selected, if the withdrawal is a Baseline Withdrawal without Authentication
- one Payment Scheme Component (see section 5.9) if required by the payment method used (see the Payment Method supplement to determine if this is needed).

Payment Handlers should check that they are authorised to carry out the Payment (see section 4 Security Considerations).

7.5.11 Signature Block (Payment Request)

If the Baseline Withdrawal Offer Response Block was signed then the OTP Message that contains the Payment Request Block must also contain a Signature Block with a copy of the "Offer Response" Signature Component.

7.5.12 Payment Exchange Block

The Payment Exchange Block (see section 6.7) contains:

- one Payment Scheme Component (see section 5.9) which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain.

7.5.13 Payment Response Block

The Payment Response Block (see section 6.8) contains:

- one Payment Receipt Component (see section 5.10) which contains scheme specific data which can be used to verify the payment occurred
- one Payment Scheme Component (see section 5.9) if required which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain
- the "Offer Response" Signature Component (see section 5.16) from the Payment Request Block if present.

7.5.14 Signature Block (Payment Response)

If a signed Payment Receipt is being provided, indicated by the `SignedPayReceipt`attribute of the Payment Component of the Offer Response Block being set to `True`, then the OTP Message that contains the Payment Response Block must also contain a Signature Block with a "Payment Receipt" Signature Component which contains hashes of the following:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Payment Response Block,
- the Transaction Id Component (see section 2.3.1) within the Transaction Reference Block that globally uniquely identifies the OTP Transaction,
- the Payment Receipt Component from the Payment Response Block and
- the "Offer Response" Signature Component from the Payment Request Block if present.

7.6 Baseline Value Exchange OTP Transaction

The Baseline Value Exchange Transaction uses Payment Exchanges (see section 1.2.2) to support the exchange of value in one currency obtained using one payment method with value in the same or another currency using the same or another payment method. Examples of its use include:

- *electronic cash advance on a credit card*. For example the first payment could be a dollar SET Payment Exchange on a credit card with the second Payment Exchange being a download of DigiCash e-cash in dollars.
- *foreign exchange using the same payment method*. For example the payment could be an upload of Mondex value in French Francs and the second a download of Mondex value in British Pounds
- *foreign exchange using different payment methods*. For example the first payment could be a SET payment in Euros followed a download of GeldKarte in Deutchmarks.

The Baseline Value Exchange uses three Trading Exchanges:

- an Offer Exchange (see section 1.2.1) which provides details of what values and currencies will be exchanged, and
- two Payment Exchanges (see section 1.2.2) which carry out the two payments involved

These Trading Exchanges are implemented by a set of predefined OTP Messages (see section 0) which are exchanged between the Trading Roles (see section 1.1). Each OTP Message contains Trading Blocks (see section 0) which contain the Trading Components (see section 5) which are required by the Trading Exchanges.

The Trading Blocks used by the Baseline Value Exchange OTP Transaction are:

- Trading Protocol Options Block
- TPO Selection Block
- Offer Response Block
- Payment Request Block
- Payment Exchange Block
- Payment Response Block
- Signature Block

7.6.1 Baseline Value Exchange Variations

The Baseline Value Exchange OTP Transaction occurs in two basic forms:

- Brand Dependent Value Exchange. Where the content of the offer, for example the rate at which one form of value is exchanged for another, is dependent on the payment brands and protocols selected by the consumer, and
- Brand Independent Value Exchange. Where the content of the offer is not dependent on the payment brands and protocols selected.

In Brand Dependent Value Exchange the TPO Block and the Offer Response Block are sent separately by the Merchant to the Consumer, i.e.:

- the Brand List Components for the two payments are sent to the Consumer in a TPO Block,
- the Consumer selects a Payment Brand and Payment Protocol from the Brand List Component for each of the payments in the Value Exchange
- the Consumer sends the selected brands and protocols back to the Merchant in a TPO Selection Block, and

- the Merchant Uses the information received to define the content of the Offer Response Block and then sends it to the Consumer.

[Note] *In the above the role is a Merchant even though the organisation carrying out the Value Exchange may be a Bank or some other Financial Institution. This is because the Bank is acting as a merchant in that they are making an offer which the Consumer can either accept or decline.*

[Note End]

In Brand Independent Value Exchange the TPO Block and the Offer Response Block are sent together by the Merchant to the Consumer in the same OTP Message at the start of the OTP Transaction.

These two alternatives are illustrated in the two diagrams below. The first diagram illustrates a Brand Dependent Value Exchange.

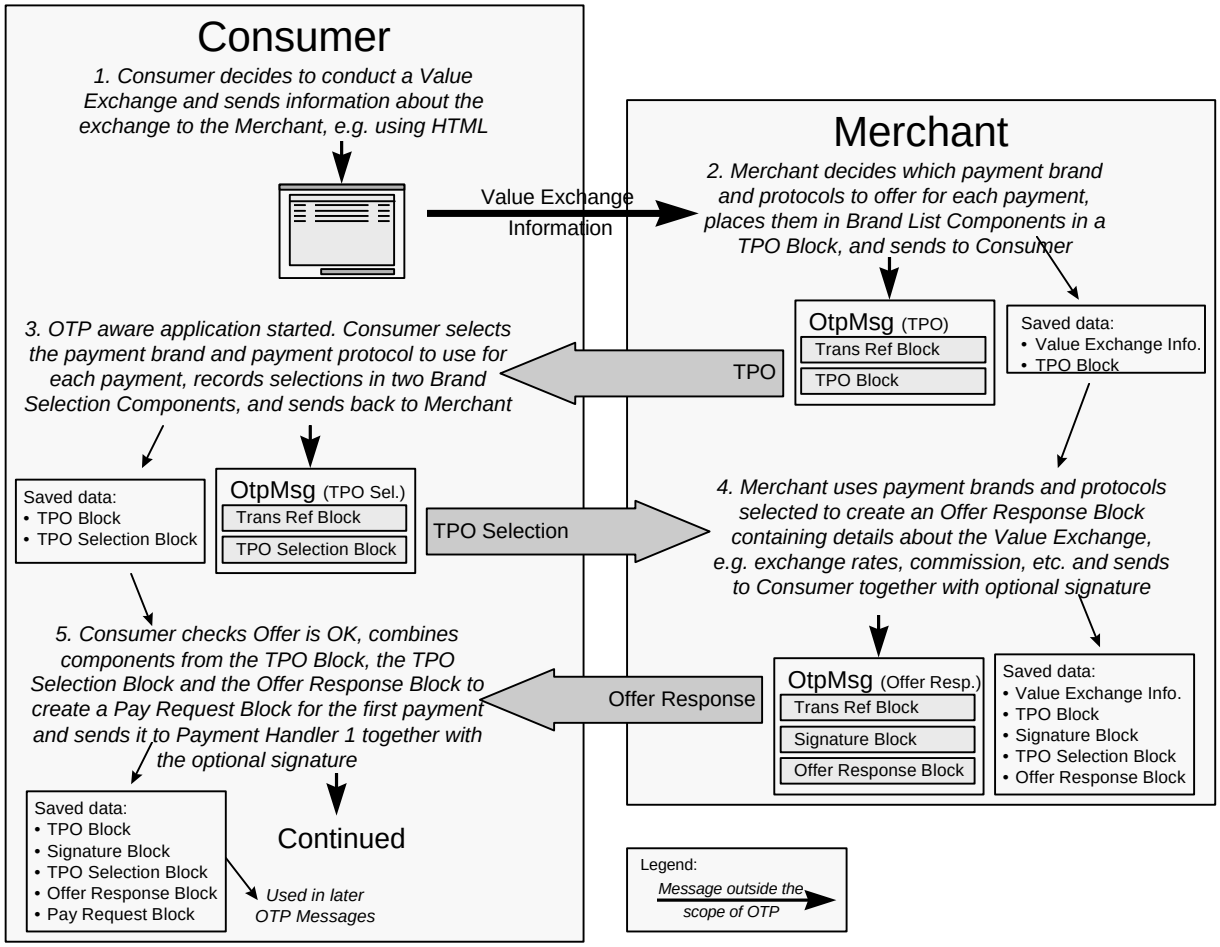


Figure 35 Brand Dependent Value Exchange

The second diagram illustrates the a Brand Independent Value Exchange.

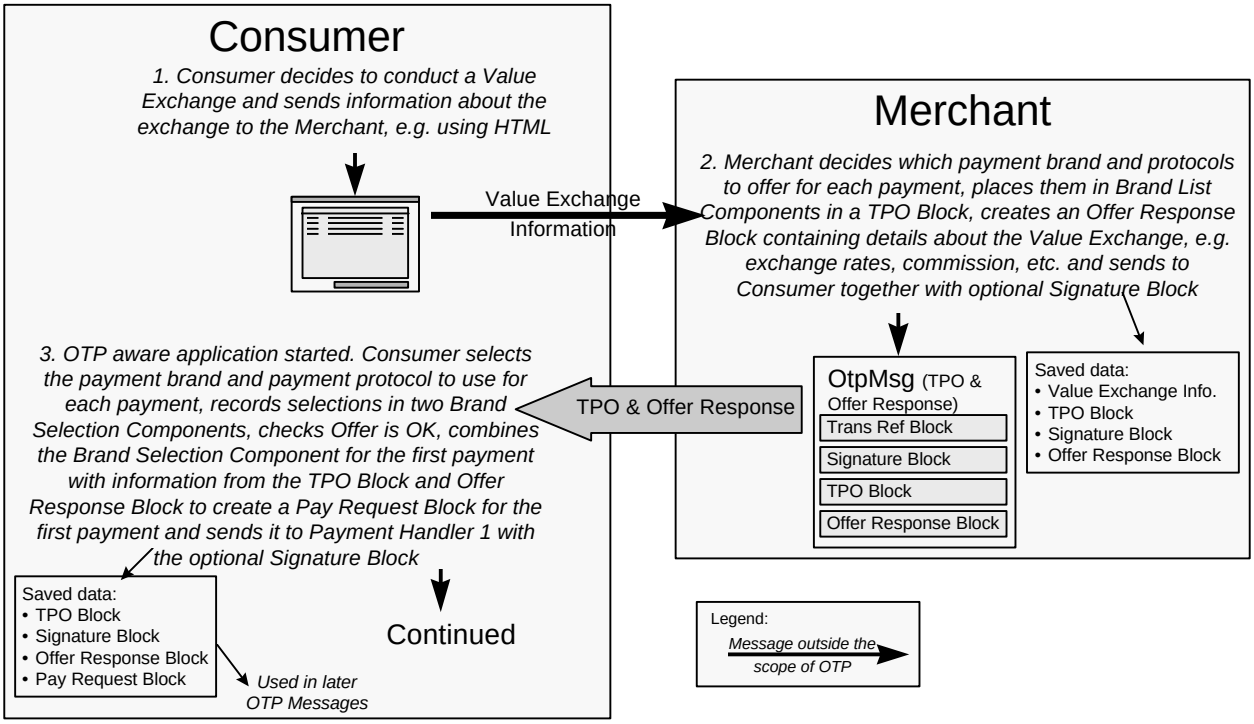


Figure 36 Brand Independent Value Exchange

The TPO Block and Offer Response Block may only be combined into the same OTP Message if the content of the Offer Response Block does not change as a result of selecting the payment brands and payment protocols to be used in the Value Exchange.

Note that the TPO Block and the Offer Response Block may be sent in separate OTP messages even if the Offer Response Block does not change. However this increases the number of messages in the transaction and is therefore likely to increase transaction response times.

OTP aware applications supporting the Consumer Trading Role must check for the existence of an Offer Response Block in the first OTP Message to determine whether the Baseline Value Exchange is brand dependent.

Whether or not the Value Exchange is brand dependent, the exchange of Trading Blocks between the Consumer and the Payment Handlers are the same. This is illustrated in the diagram below. Note that this diagram continues where the previous diagrams (Figure 35 and Figure 36) finish.

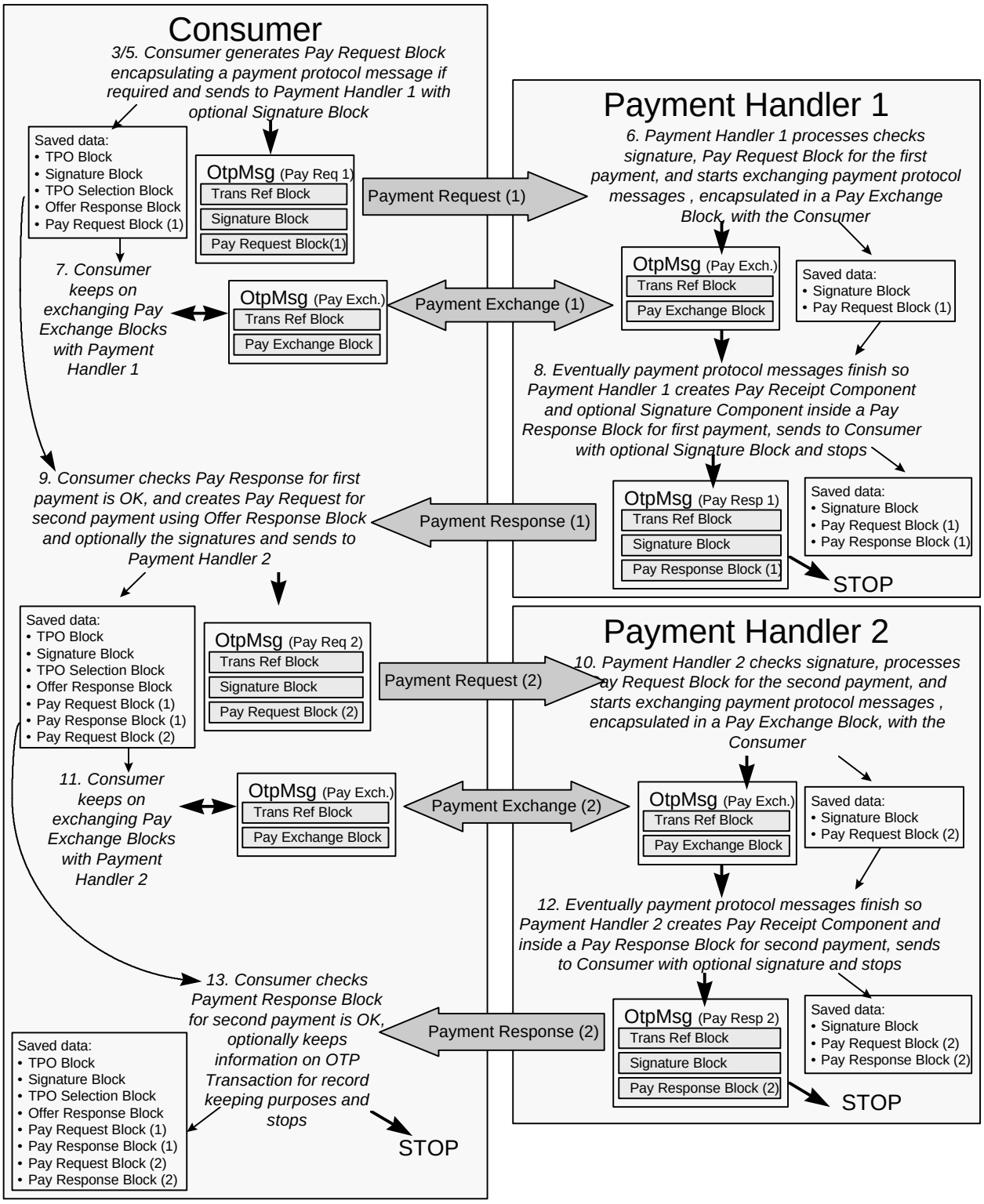


Figure 37 Baseline Value Exchange Payment Messages

The remainder of this sub-section on the Baseline Value Exchange OTP Transaction defines the contents of each Trading Block. The content does not alter with the variations described above.

7.6.2 TPO (Trading Protocol Options) Block

The TPO (Trading Protocol Options) Block (see section 7.3.2) must contain the following Trading Components:

- one Protocol Options Component which defines the options which apply to the whole OTP Transaction. See Section 5.1.
- two Brand List Components (see section 5.6) one for each Payment Exchange where each Brand List Component contains one or more payment brands and protocols which may be selected for use in the Payment Exchange.

7.6.3 TPO Selection Block

The TPO Selection Block (see section 6.2) is only used by Brand Dependent Value Exchange. It contains:

- two Brand Selection Components (see section 5.7). One for each of the Payment Exchanges. Each Brand Selection Component contains the results of the consumer selecting a Payment Brand and Payment Protocol from the list provided in the Brand List Component.

7.6.4 Offer Response Block

The Offer Response Block (see section 6.3) contains the following components:

- zero, one or two Authentication Data Component (see section 5.2). An Authentication Data Component is required for each Payment Exchange, where its Payment Component contains an `AuthDataRef` attribute.
- one Order Component (see section 5.4) which contains details about the Value Exchange, for example, exchange rates, commission, etc.
- two Payment Components (see section 5.8) which contain information about each of the two payments which are to be made
- Organisation Components (see section 5.5) with the following roles:
  - `Merchant` who is providing the goods or services
  - `Consumer` who is making the purchase
  - the `PaymentHandlers` for the payments. The "ID" of a Payment Handler Organisation Component is contained within the `VaOrgRef` attribute of each of the Payment Components

7.6.5 Signature Block (Offer Response)

If the Baseline Value Exchange Offer Response is being digitally signed then a Signature Block must be included in the same OTP message that contains an "Offer Response" Signature Component (see section 5.16). The Signature Component contains hashes of the following XML elements:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Offer Response Block within the OTP Transaction. It contains information that identifies the OTP Message and OTP Transaction
- the Transaction Id Component (see section 2.3.1) which globally uniquely identifies the OTP Transaction

- the following components of the Offer Response Block:
  - the Authentication Data Component if present
  - the Order Component
  - the two Payment Components
  - all the Organisation Components present, and
- the following components of the TPO Block :
  - the Protocol Options Component, and
  - the Brand List Component

If the Baseline Value Exchange is a Brand Dependent Value Exchange then the Signature Component additionally contains a hash of the following:

- the two Brand Selection Components contained in the TPO Selection Block.

7.6.6 Payment Request Block (first payment)

The Payment Request Block (see section 6.6) for the first payment contains:

- the following components copied from the Offer Response Block:
  - the Authentication Data Component for the first payment if required
  - the Payment Component for the first payment
  - the Organisation Components with the roles of: `Merchant` and `PaymentHandler` for the first payment
- the following component copied from the TPO Block:
  - the Brand List Component for the first payment
- one Brand Selection Component for the first payment which is either:
  - copied from the Offer Response Block if the purchase is a Brand Dependent Value Exchange, or
  - created by the Consumer, containing the payment brand and payment protocol selected, if the purchase is a Brand Independent Value Exchange
- one Payment Scheme Component (see section 5.9) if required by the payment method used (see the Payment Method supplement to determine if this is needed).

Note that:

- the Payment Component for the first payment is the one within the Offer Response Block that contains no `StartAfter` attribute (see section 5.8)
- the Authentication Data Component to include is identified by the `AuthDataRef` attribute of the Payment Component for the first payment. If no `AuthDataRef` attribute is present then no Authentication Data Component is required
- the Payment Handler to include is identified by the Brand Selection Component (see section 5.7) for the first payment. Also see section 4.3.1 Check the Action Request was sent to the Correct Organisation for an explanation on how Payment Handlers are identified
- the Brand List Component to include is the one identified by the `BrandListRef` attribute of the Payment Component for the first payment
- the Brand Selection Component to include from the Offer Response Block is the one that contains an Element Reference (see section 2.5) which identifies the Brand List Component for the first payment

7.6.7 Signature Block (Payment Request - first payment)

If the Baseline Value Exchange Offer Response Block was signed then the OTP Message that contains the Payment Request Block for the first payment must also contain a Signature Block with a copy of the "Offer Response" Signature Component.

7.6.8 Payment Exchange Block (first payment)

The Payment Exchange Block (see section 6.7) for the first payment contains:

- one Payment Scheme Component (see section 5.9) which contains payment method specific data for the payment method being used by the first payment. See the Payment Method supplement for the payment method being used to determine what this should contain.

7.6.9 Payment Response Block (first payment)

The Payment Response Block for the first payment (see section 6.8) contains:

- one Payment Receipt Component (see section 5.10) which contains scheme specific data which can be used to verify the first payment occurred
- one Payment Scheme Component (see section 5.9) if required by the payment method used by the first payment which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain
- the Signature Component (see section 5.16) from the Payment Request Block for the first payment if present.

7.6.10 Signature Block (Payment Response - first payment)

If a signed Payment Receipt is being provided for the first payment, indicated by the `SignedPayReceipt` attribute of the Payment Component for the first payment in the Offer Response Block being set to `True`, then the OTP Message that contains the Payment Response Block for the first payment must also contain a Signature Block with a "Payment Receipt" Signature Component which contains hashes of the following:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Payment Response Block for the first payment,
- the Transaction Id Component (see section 2.3.1) within the Transaction Reference Block that globally uniquely identifies the OTP Transaction,
- the Payment Receipt Component from the Payment Response Block for the first payment and
- the "Offer Response" Signature Component from the Payment Request Block for the first payment, if present.

7.6.11 Payment Request Block (second payment)

The Payment Request Block (see section 6.6) for the second payment contains:

- the following components copied from the Offer Response Block:
  - the Authentication Data Component for the second payment if required
  - the Payment Component for the second payment
  - the Organisation Components with the roles of: `Merchant` and `PaymentHandler` for the second payment
- the following component copied from the TPO Block:
  - the Brand List Component for the second payment
- one Brand Selection Component for the second payment which is either:
  - copied from the Offer Response Block if the purchase is a Brand Dependent Value Exchange, or
  - created by the Consumer, containing the payment brand and payment protocol selected, if the purchase is a Brand Independent Value Exchange
- one Payment Scheme Component (see section 5.9) if required by the payment method used (see the Payment Method supplement to determine if this is needed)

Note that:

- the Payment Component for the second payment is the one within the Offer Response Block that contains a `StartAfter` attribute (see section 5.8) that identifies the Payment Component for the first payment
- the Authentication Data Component to include is identified by the `AuthDataRef` attribute of the Payment Component for the second payment. If no `AuthDataRef` attribute is present then no Authentication Data Component is required
- the Payment Handler to include is identified by the Brand Selection Component (see section 5.7) for the second payment. Also see section 4.3.1 Check the Action Request was sent to the Correct Organisation for an explanation on how Payment Handlers are identified
- the Brand List Component to include is the one identified by the `BrandListRef` attribute of the Payment Component for the second payment
- the Brand Selection Component to include from the Offer Response Block is the one that contains an Element Reference (see section 2.5) which identifies the Brand List Component for the second payment

7.6.12 Signature Block (Payment Request - second payment)

If the Baseline Value Exchange Offer Response Block or the Payment Response Block for the first payment was signed then the OTP Message that contains the Payment Request Block for the second payment must also contain a Signature Block with a copy of:

- the "Offer Response" Signature Component, if present, and/or
- the "Payment Receipt" Signature Component copied from the Payment Response Block for the first payment, if present.

7.6.13 Payment Exchange Block (second payment)

The Payment Exchange Block (see section 6.7) for the second payment contains:

- one Payment Scheme Component (see section 5.9) which contains payment method specific data for the payment method being used by the second payment. See the Payment Method supplement for the payment method being used to determine what this should contain.

7.6.14 Payment Response Block (second payment)

The Payment Response Block for the second payment (see section 6.8) contains:

- one Payment Receipt Component (see section 5.10) which contains scheme specific data which can be used to verify the second payment occurred
- one Payment Scheme Component (see section 5.9) if required by the payment method used by the second payment which contains payment method specific data. See the Payment Method supplement for the payment method being used to determine what this should contain
- all the Signature Components (see section 5.16) from the Payment Request Block for the second payment if present.

7.6.15 Signature Block (Payment Response - second payment)

If a signed Payment Receipt is being provided for the second payment, indicated by the `SignedPayReceipt`attribute of the Payment Component for the second payment in the Offer Response Block being set to `True`, then the OTP Message that contains the Payment Response Block for the second payment must also contain a Signature Block with a "Payment Receipt" Signature Component which contains hashes of the following:

- the Transaction Reference Block (see section 2.3) for the OTP Message which contains the first usage of the Payment Response Block for the second payment,
- the Transaction Id Component (see section 2.3.1) within the Transaction Reference Block that globally uniquely identifies the OTP Transaction,
- the Payment Receipt Component from the Payment Response Block for the second payment
- the "Offer Response" Signature Component from the Payment Request Block for the second payment, if present, and
- the "Payment Receipt" Signature Component from the Payment Request Block for the first payment, if present.

7.6.16 Baseline Value Exchange Signatures

The use of signatures to ensure the integrity of a Baseline Value Exchange is illustrated by the diagram below.

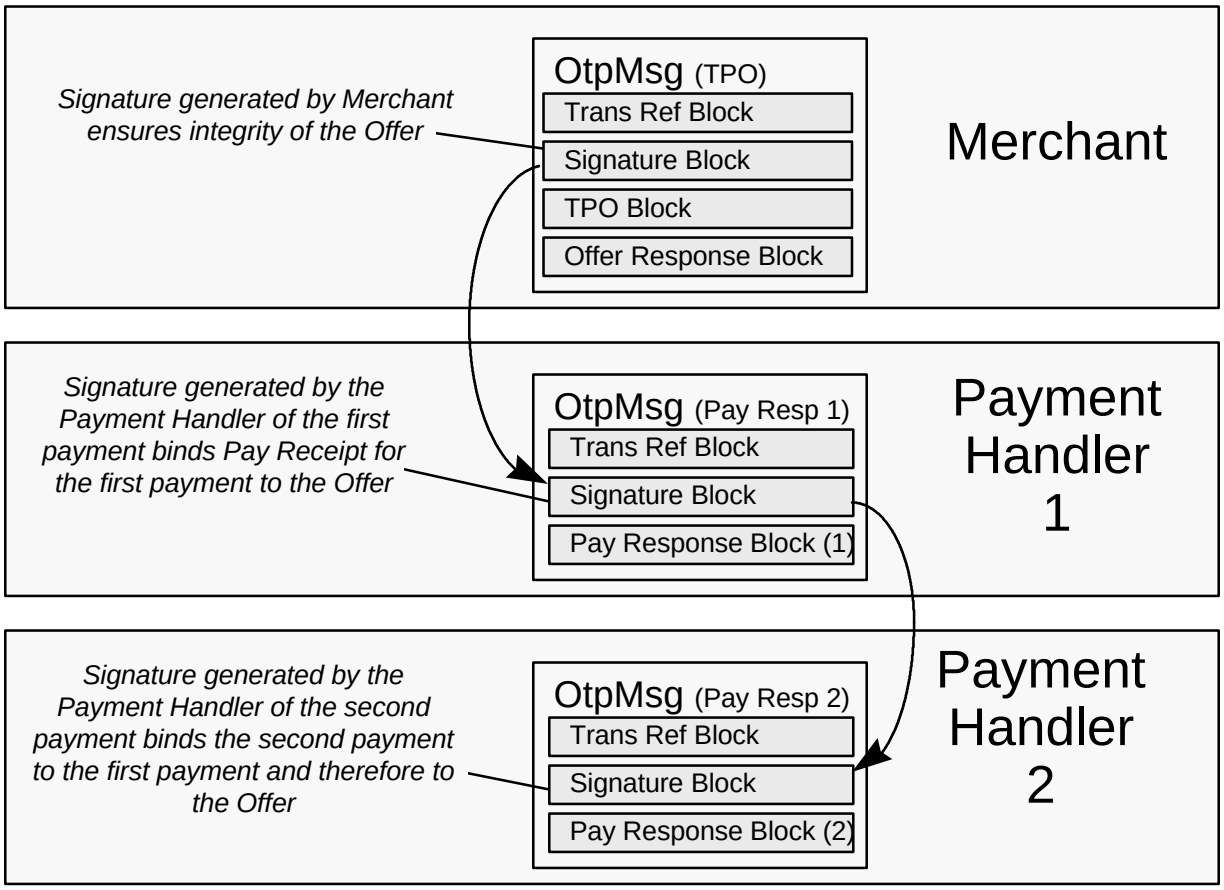


Figure 38 Baseline Value Exchange Signatures

If signatures are used then the Payment Handlers should check that all Signature Components they receive are valid (see section 4 Security Considerations).

7.7 Payment Instrument Customer Care OTP Transaction

An OTP Payment Instrument Customer Care Transaction is used to provide Payment Brand or Payment Method specific customer care. It allows Consumer Payment Brand software to exchange information with a Payment Instrument Customer Care Provider.

The circumstances under which this transaction is used, if any, is defined in the OTP Supplement for the Payment Brand.

Note that the OTP Payment Instrument Customer Care Transaction:

- is initiated by the Consumer Payment Brand software which must identify the need for the transaction to occur. Note that in other OTP Transactions, the transaction is initiated by the Merchant
- has no TPO Block, as it is initiated by the Consumer
- relies on the Consumer Payment Brand software to identify the net location of the Payment Instrument Customer Care Provider to which the first message in the transaction must be sent
- ends when the Payment Scheme Customer Care Service determines that the exchange of messages with the Consumer is to stop.

Note that a Payment Instrument Customer Care Transaction can be initiated at any time by a Consumer including in the middle of another OTP Transaction. In this case, the transaction shall establish a different transport session from the ongoing transaction. See the Mapping to Transport for the Transport Mechanism being used.

The transaction consists of three types of OTP messages as illustrated in the figure below.

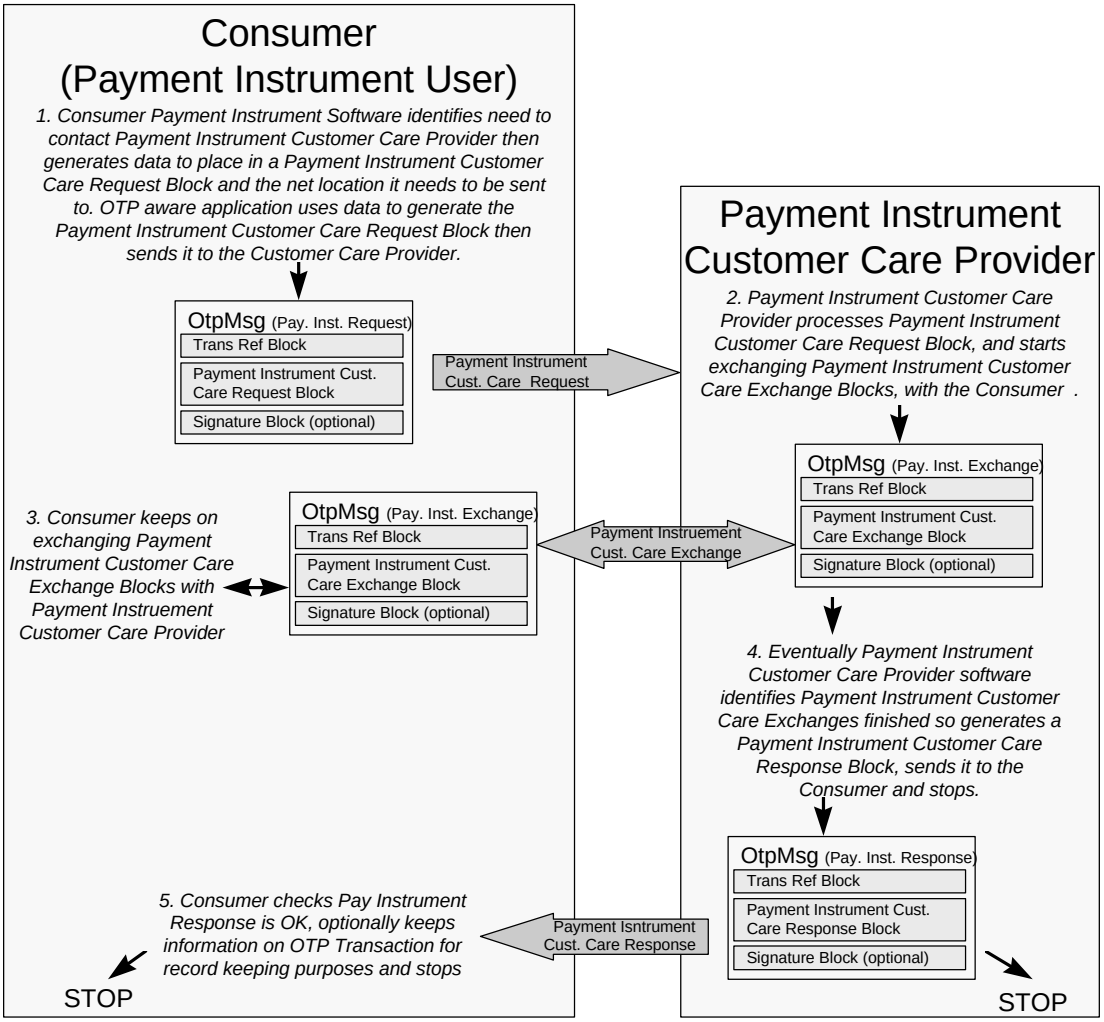


Figure 39 OTP Payment Instrument Customer Care Transaction Message Flows

The remainder of this sub-section on the Payment Instrument Customer Care Transaction defines the contents of each Trading Block.

7.7.1 Payment Instrument Customer Care Request Block

The Payment Instrument Customer Care Request Block contains:

- a Payment Method Information Component (see section 5.13) which describes the Payment Method for which Customer Care is requested, and
- zero or more optional Payment Scheme Components (see section 5.9) which contain optional Payment scheme data

7.7.2 Payment Instrument Customer Care Exchange Block

The Payment Instrument Customer Care Exchange Block contains:

- a Payment Method Information Component (see section 5.13) which describes the Payment Method for which Customer Care is being provided, and
- zero or more optional Payment Scheme Components (see section 5.9) which contain optional Payment scheme data

7.7.3 Payment Instrument Customer Care Response Block

The Payment Instrument Customer Care Response Block contains:

- a Payment Method Information Component (see section 5.13) which describes the Payment Method for which Customer Care is complete, and
- zero or more optional Payment Scheme Component (see section 5.9) which contains optional Payment scheme data

7.7.4 Signature Block

Any of the OTP Messages which contain Payment Instrument Customer care blocks may also include a Signature Block (see section 6.18) containing a Signature Component (see section 5.16). How these are used and what it signs is dependent on the Payment Brand and Payment method being used. See the OTP Payment Supplement for the payment method.

7.8 Baseline Transaction Status Inquiry OTP Transaction

The Baseline OTP Transaction Status Inquiry provides a Consumer with information on the status of an existing or complete OTP transaction.

The Trading Blocks used by the Baseline Transaction Status Inquiry Transaction are:

- an Inquiry Request Trading Block (see section 6.14), and

- an Inquiry Response Trading Block (see section 6.15).

[Note] *Note that:*

- *Consumer Inquiries on Authentication transaction are not supported.*
- *Authentication of Consumers as part of an inquiry is not supported in the Baseline version of OTP.*

[Note End]

7.8.1 Which Trading Roles can receive Inquiry Requests

The Consumer can send a Transaction Status Inquiry Block to the appropriate Trading Role after the following events have occurred:

- to the Merchant, after sending TPO Selection Block,
- to the Payment Handler, after sending Payment Request Block,
- to the Delivery Handler, after sending Delivery Request Block.

[Note] *OTP does not support sending Inquiry Requests to the Consumer since the consumer may not be on-line to receive and process them.*

[Note End]

If the Consumer is inquiring on transaction that is not yet complete, it should send the Inquiry Request Block to the Trading Role to which it sent the last OTP message. If the Consumer is inquiring on a transaction which is complete, there are two alternatives in deciding the Trading Roles that the Inquiry Request Block should be sent to:

- the Consumer OTP software can ask the end user to determine the type of inquiry they want to make, or
- the Consumer OTP software can send the inquiry request message to all the Trading Roles that were involved in the OTP transaction.

For the second case above, how the Consumer OTP Aware Application displays the inquiry response data received from each Trading Role is up to each implementation.

7.8.2 Transaction Status Inquiry Transport Session

For a Transaction Status Inquiry on an ongoing transaction, the Consumer shall establish with a Trading Role, a different transport session from the ongoing transaction. For a Transaction Status Inquiry on a past transaction, how the OTP module on the software at the Trading Role is started upon the receipt of Inquiry Request message is defined in each Mapping to Transport supplement for OTP.

7.8.3 Transaction Status Inquiry Error Handling

Errors in a Transaction Status Inquiry can be categorised into one the following three cases:

- Business errors (see section 3.2) in the original (inquired) messages
- Technical errors (see section 3.1) - both OTP and payment scheme specific ones - in the original OTP (inquired) messages

- Technical errors in the message containing the Inquiry Request Block itself

The following outlines what the software should do in each case

Business errors in the original messages

Return an Inquiry Response Block containing the Status Component which was last sent to the Consumer.

Technical errors in the original messages

Return an Inquiry Response Block containing a Status Component. The Status Component should contain a `ProcessState`attribute set to `ProcessError`. In this case send back an Error Block indicating where the error was found in the original message.

Technical errors in the Inquiry Request Block

Return an Error message. That is, send back an Error Block containing the Error Code (see section 5.17.2) which describes the nature of the error in the Inquiry Request message.

7.8.4 Inquiry Transaction Messages

The following Figure outlines the Baseline OTP Transaction Status Inquiry processes on both Consumer and Service Provider sides.

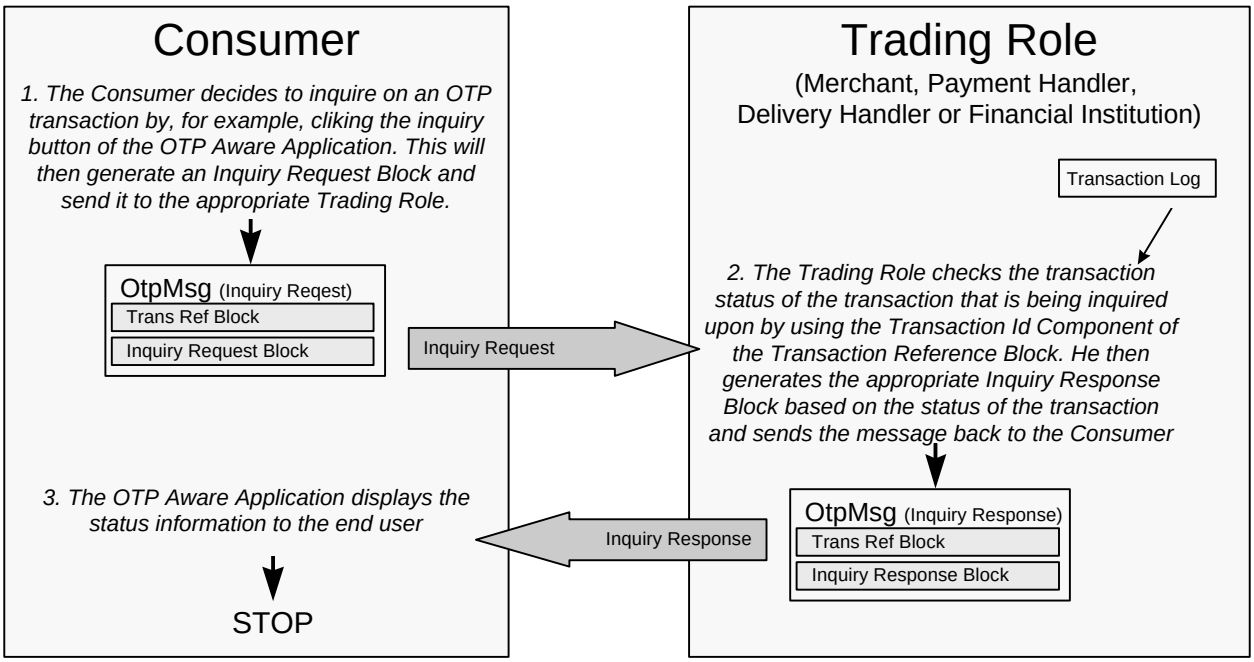


Figure 40 Baseline Transaction Status Inquiry

The remainder of this sub-section on the Baseline Transaction Status Inquiry OTP Transaction defines the contents of each Trading Block.

7.8.5 Transaction Reference Block

The Consumer must use the same Transaction Id Component (see section 2.3.1) as in the inquired transaction. The `OtpTransId`attribute in this component serves as the key in querying the transaction logs maintained at the Trading Role's site. The value of the `ID` attribute of the Message Id Component should be different from those of the inquired transaction (see section 2.4.1).

7.8.6 Inquiry Request Block

The Inquiry Request Block (see section 6.14) contains the following components:

- one Inquiry Type Component (see section 5.15). This identifies whether the inquiry is on an offer, payment, or delivery.
- zero or one Payment Scheme Component (see section 5.9). This is for encapsulating payment scheme specific inquiry messages for inquiries on payment.

7.8.7 Inquiry Response Block

The Inquiry Response Block (see section 6.15) contains the following components:

- one Status Component (see section 5.14). This component hold the status information on the inquired transaction,
- zero or one Payment Scheme Components. These contain for encapsulated payment scheme specific inquiry messages for inquiries on payment.

7.9 Baseline Ping OTP Transaction

The purpose of the Baseline OTP Ping Transaction is to enable OTP aware application software to determine if the OTP aware application at another Trading Role is operating and verifying whether or not signatures can be handled.

The Trading Blocks used by the Baseline Ping OTP Transaction are:

- a Ping Request Block (see section 6.16)
- a Ping Response Block (see section 6.17), and
- a Signature Block (see section 6.18).

7.9.1 Ping Messages

The following figure outlines the message flows in the Baseline OTP Ping Transaction.

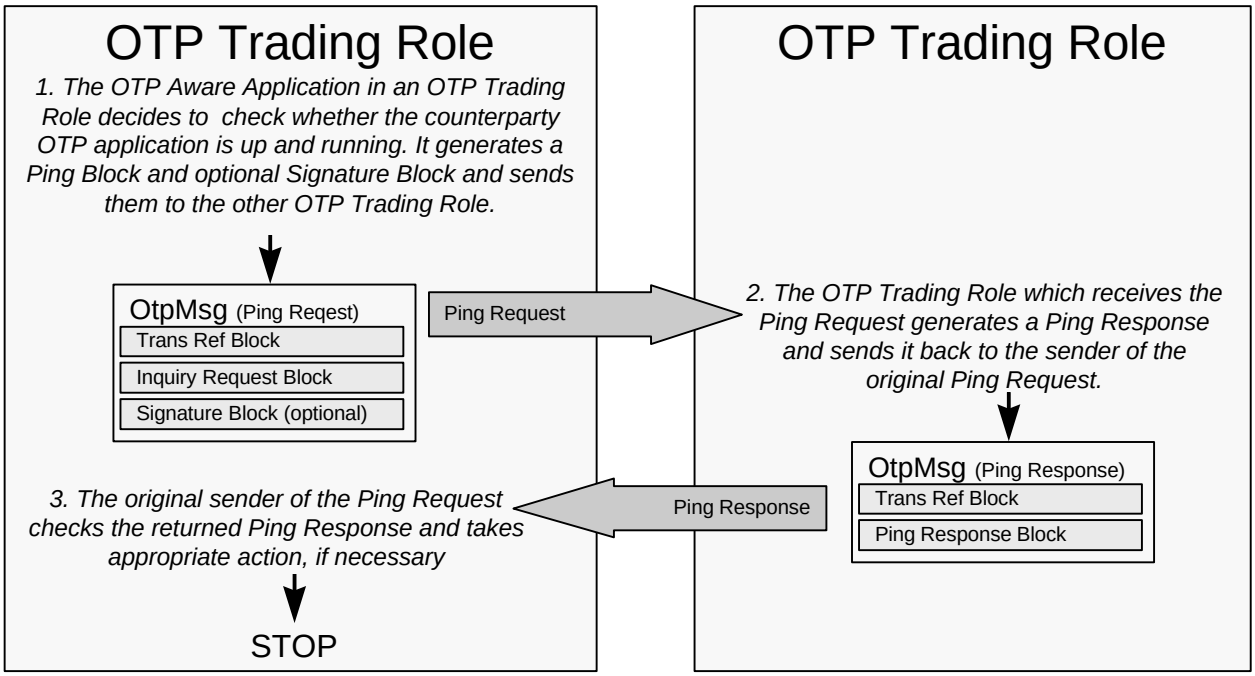


Figure 41 Baseline Ping Messages

The verification that signatures can be handled is indicated by the sender of the Ping Request Block including:

- Organisation Components that identify itself and the intended recipient of the Ping Request Block, and
- a Signature Block that signs data in the Ping Request.

In this way the receiver of the Ping Request:

- knows who is sending the Ping Request and can therefore verify the Signature on the Request, and
- knows who to generate a signature for on the Ping Response.

Note that a Ping Request:

- does not affect any on-going transaction
- does NOT start an OTP aware application, unlike other OTP transaction messages such as TPO or Transaction Status Inquiry.

All OTP aware applications must return a Ping Response message to the sender of a Ping Request message when it is received.

A Baseline OTP Ping request can also contain an optional Signature Block. OTP aware applications can, for example, use the Signature Block to check the recipient of a Ping Request can successfully process and check signatures it has received.

For each Baseline Ping OTP Transaction, each OTP role shall establish a different transport session from other OTP transactions.

Any OTP Trading Role can send a Ping request to any other OTP Trading Role at any time it wants. A Ping message has its own `OtpTransID`, which is different from other OTP transactions.

The remainder of this sub-section on the Baseline Ping OTP Transaction defines the contents of each Trading Block.

7.9.2 Transaction Reference Block

The `OtpTransId`of a Ping transaction should be different from any other OTP transaction.

7.9.3 Ping Request Block

If the Ping Transaction is anonymous then no Organisation Components are included in the Ping Request Block (see section 6.6).

If the Ping Transaction is not anonymous then the Ping Request Block contains Organisation Components for:

- the sender of the Ping Request Block, and
- the verifier of the Signature Component

If Organisation Components are present, then it indicates that the sender of the Ping Request message has generated a Signature Block. The signature block must be verified by the Trading Role that receives the Ping Request Block.

7.9.4 Signature Block (Ping Request)

The Ping Request Signature Block (see section 6.18) contains the following components:

- one Signature Component (see section 5.16)
- one or more Certificate Components, if required.

7.9.5 Ping Response Block

The Ping Response Block (see section 6.17) contains the following component:

- the Organisation Component of the sender of the Ping Response message

If the Ping Transaction is not anonymous then the Ping Response additionally contains:

- copies of the Organisation Components contained in the Ping Request Block.

7.9.6 Signature Block (Ping Response)

The Ping Response Signature Block (see section 6.18) contains the following components:

- one Signature Component (see section 5.16)
- one or more Certificate Components, if required.

8. Retrieving Logos

This section describes how to retrieve logos for display by OTP aware software using the Logo Net Locations attribute contained in the Brand Element (see section 5.6.1) and the Organisation Component (see section 5.5).

The full address of a logo is defined as follows:

```
Logo_address ::= Logo_net_location "/" Logo_size Logo_color_depth
                                          ".GIF"
```

Where:

- `Logo_net_location` is obtained from the `LogoNetLoc` attribute in the Brand Element (see section 5.6.1) or the Organisation Component. Note that:
  - the content of this attribute is dependent on the Transport Mechanism (such as HTTP) that is used. See the Transport Mechanism supplement,
  - implementers should check that if the rightmost character of Logo Net Location is set to right-slash "/" then another, right slash should not be included when generating the Logo Address,
- `Logo_size` identifies the size of the logo,
- `Logo_color_depth` identifies the colour depth of the logo
- `"GIF"` indicates that the logos are in GIF format

`Logo_size` and `Logo_color_depth` are specified by the implementer of the OTP software that is retrieving the logo depending on the size and colour that they want to use.

8.1 Logo Size

There are five standard sizes for logos. The sizes in pixels and the corresponding values for Logo Size are given in the table below.

Size in Pixels	Logo Size Value
32 x 32 or 32 x 20	`exsmall`
53 x 33	`small`
103 x 65	`medium`
180 x 114	`large`
263 x 166	`exlarge`

8.2 Logo Color Depth

There are three standard colour depths. The colour depth (including bits per pixel) and the corresponding value for `Logo_Color_Depth` are given in the table below.

Color Depth (bits per pixel)	Logo Color Depth Value
4 (16 colors)	4
8 (256 colors)	nothing
24 (16 million colors)	24

Note that if Logo Color Depth is omitted then a logo with the default colour depth of 256 colours will be retrieved.

8.3 Logo Net Location Examples

If Logo Net Location was set to `"ftp://logos.xzpay.com"`, then:

- `"ftp://logos.xzpay.com/medium.gif"` would retrieve a medium size 256 colour logo
- `"ftp://logos.xzpay.com/small4.gif"` would retrieve a small size 16 colour logo

[Note] *Organisations which make logos available for use with OTP should always make available "small" and "medium" size logos and use the GIF format.*

[Note End]

9. Brand List Examples

This example contains three examples of the XML for a Brand List Component. It covers:

- a simple credit card based example
- a credit card based brand list including promotional credit card brands, and
- a complex electronic cash based brand list

Note that:

- brand lists can be as complex or as simple as required
- all example techniques described in this appendix can be included in one brand list.

9.1 Simple Credit Card Based Example

This is a simple example involving:

- only major credit card payment brands
- a single price in a single currency
- a single payment handler, and
- a single payment protocol

```
<BrandList ID='M1.2'
  XML:Lang='us-en'
  ShortDesc='Purchase book including s&h'
  PayDirection='Debit' >
  <Brand ID ='M1.30'
    BrandId='MC'
    BrandName='MasterCard'
    BrandLogoNetLocn='ftp:otplogos.mastercard.com'
    ProtocolAmountRefs='M1.33'>
  </Brand>
  <Brand ID ='M.31'
    BrandId='Visa'
    BrandName='Visa'
    BrandLogoNetLocn='ftp:otplogos.visa.com'
    ProtocolAmountRefs='M1.33'>
  </Brand>
  <Brand ID ='M1.32'
    BrandId='Amex'
    BrandName='American Express'
    BrandLogoNetLocn='ftp:otplogos.amex.com'
    ProtocolAmountRefs ='M1.33' >
  </Brand >
  <ProtocolAmount ID ='M1.33'
    PayProtocolRef='M1.35'
    CurrencyAmountRefs='M1.34'>
  </ProtocolAmount>
  <CurrencyAmount ID ='M1.34'
```

```
    Amount='10.95'
    CurrCode='USD'/>
  <PayProtocol ID ='M1.35'
    ProtocolId='SCCD1.0'
    ProtocolName='Secure Channel Credit/Debit'
    PayReqNetLocn='http://www.merchant.com/etill/sccd1' >
  </PayProtocol>
</BrandList>
```

9.2 Credit Card Brand List Including Promotional Brands

An example of a Credit Card based Brand List follows. It includes:

- two ordinary card association brands and two promotional credit card brands. The promotional brands consist of one loyalty based (British Airways MasterCard) which offers additional loyalty points and one store based (Walmart) which offers a discount on purchases over a certain amount
- two payment protocols:
  - SET (Secure Electronic Transactions) see [SET], and
  - SCCD (Secure Channel Credit Debit) see [SCCD].

```
12345678901234567890123456789012345678901234567890123456789012

<BrandList ID='M1.2'
  XML:Lang='us-en'
  ShortDesc='Purchase ladies coat'
  PayDirection='Debit' >
  <Brand ID ='M1.3'
    BrandId='MC'
    BrandName='MasterCard'
    BrandLogoNetLocn='ftp:otplogos.mastercard.com'
    ProtocolAmountRefs='M1.7 M1.8'>
  </Brand>
  <Brand ID ='M1.4'
    BrandId='Visa'
    BrandName='Visa'
    BrandLogoNetLocn='ftp:otplogos.visa.com'
    ProtocolAmountRefs='M1.7 M1.8'>
  </Brand>
  <Brand ID ='M1.5'
    BrandId='MC/BritishAirways'
    BrandName='British Airways MasterCard'
    BrandLogoNetLocn='ftp:otplogos.britishairways.co.uk'
    BrandNarrative='Double air miles with British Airways
                                          MasterCard'
    ProtocolAmountRefs ='M1.7 M1.8' >
  </Brand >
  <Brand ID ='M1.6'
    BrandId='Walmart'
    BrandName='Walmart Store Card'
    BrandLogoNetLocn='ftp:otplogos.walmart.com'
    BrandNarrative='5% off with your Walmart Card
                    on purchases over $150'
    ProtocolAmountRefs='M1.7'>
```

```
  </Brand>
  <ProtocolAmount ID ='M1.7'
    PayProtocolRef='M1.10'
    CurrencyAmountRefs='M1.9' >
    <PackagedContent Transform="BASE64">
       238djqw1298erh18dhoire
    <PackagedContent>
  </ProtocolAmount>
  <ProtocolAmount ID ='M1.8'
    PayProtocolRef='M1.11'
    CurrencyAmountRefs='M1.9' >
    <PackagedContent Transform="BASE64">
       238djqw1298erh18dhoire
    <PackagedContent Transform="BASE64">
  </ProtocolAmount>
  <CurrencyAmount ID ='M1.9'
    Amount='157.53'
    CurrCode='USD'/>
  <PayProtocol ID ='M1.10'
    ProtocolId='SET1.0'
    ProtocolName='Secure Electronic Transaction Version 1.0'
    PayReqNetLocn='http://www.merchant.com/etill/set1' >
    <PackagedContent Transform="BASE64">
      8ueu26e482hd82he82
    <PackagedContent Transform="BASE64">
  </PayProtocol>
  <PayProtocol ID ='M1.11'
    ProtocolId='SCCD1.0'
    ProtocolName='Secure Channel Credit/Debit'
    PayReqNetLocn='http://www.merchant.com/etill/sccd1' >
    <PackagedContent Transform="BASE64">
       82hd82he8226e48ueu
    <PackagedContent Transform="BASE64">
  </PayProtocol>
 </BrandList>
```

9.3 Brand Selection Example

In order to pay by 'British Airways' MasterCard using the example above using SET and therefore getting double air miles, the Brand Selection would be:

```
<BrandSelection ID='C1.2'
  BrandListRef='M1.3'
  BrandRef='M1.5'
  ProtocolAmountRef='M1.7'
  CurrencyAmountRef='M1.9' >
</BrandSelection>
```

9.4 Complex Electronic Cash Based Brand List

The following is an fairly complex example which includes:

- payments using either Mondex, GeldKarte, CyberCash or DigiCash
- in currencies including US dollars, British Pounds, Italian Lira, German Marks and Canadian Dollars
- a discount on the price if the payment is made in Mondex using British pounds or US dollars, and
- more than one payment handler is used for payments involving Mondex or CyberCash
- support for more than one version of a CyberCash CyberCoin payment protocol.

```
<BrandList ID='M1.2'
  XML:Lang='us-en'
  ShortDesc='Company report on XYZ Co'
  PayDirection='Debit' >
  <Brand ID ='M1.13'
    BrandId='MX'
    BrandName='Mondex Electronic Cash'
    BrandLogoNetLocn='ftp:otplogos.mondex.com'
    ProtocolAmountRefs='M1.17 M1.18'>
  </Brand>
  <Brand ID ='M1.14'
    BrandId='GK'
    BrandName='GeldKarte Electronic Cash'
    BrandLogoNetLocn='ftp:otplogos.geldkarte.co.de'
    ProtocolAmountRefs='M1.19'>
  </Brand>
  <Brand ID ='M1.15'
    BrandId='CCash'
    BrandName='CyberCoin Eletronic Cash'
    BrandLogoNetLocn='ftp:otplogos.cybercash.com'
    ProtocolAmountRefs ='M1.20' >
  </Brand >
  <Brand ID ='M1.16'
    BrandId='DigiCash'
    BrandName='DigiCash Electronic Cash'
    BrandLogoNetLocn='ftp:otplogos.digicash.com'
    BrandNarrative='5% off with your Walmart Card
                    on purchases over $150'
    ProtocolAmountRefs='M1.22'>
  </Brand>
  <ProtocolAmount ID ='M1.17'
    PayProtocolRef='M1.31'
    CurrencyAmountRefs='M1.25 M1.29'>
  </ProtocolAmount>
  <ProtocolAmount ID ='M1.18'
    PayProtocolRef='M1.32'
    CurrencyAmountRefs='M1.26 M1.27 M1.28 M1.30'>
  </ProtocolAmount>
  <ProtocolAmount ID ='M1.19'
    PayProtocolRef='M1.35'
    CurrencyAmountRefs='M1.28'>
  </ProtocolAmount>
```

```
<ProtocolAmount ID ='M1.20'
  PayProtocolRef='M1.34 M1.33'
  CurrencyAmountRefs='M1.23 M1.24 M1.27 M1.28 M1.29 M1.30'>
</ProtocolAmount>
<ProtocolAmount ID ='M1.21'
  PayProtocolRef='M1.36'
  CurrencyAmountRefs='M1.23 M1.24 M1.27 M1.28 M1.29 M1.30'>
</ProtocolAmount>
<CurrencyAmount ID ='M1.23'
  Amount='20.00'
  CurrCode='USD'/>
<CurrencyAmount ID ='M1.24'
  Amount='12.00'
  CurrCode='GBP'/>
<CurrencyAmount ID ='M1.25'
  Amount='19.50'
  CurrCode='USD'/>
<CurrencyAmount ID ='M1.26'
  Amount='11.75'
  CurrCode='GBP'/>
<CurrencyAmount ID ='M1.27'
  Amount='36.00'
  CurrCode='DEM'/>
<CurrencyAmount ID ='M1.28'
  Amount='100.00'
  CurrCode='FFR'/>
<CurrencyAmount ID ='M1.29'
  Amount='22.00'
  CurrCode='CAD'/>
<CurrencyAmount ID ='M1.30'
  Amount='15000'
  CurrCode='ITL'/>
<PayProtocol ID ='M1.31'
  ProtocolId='MXv1.0'
  ProtocolName='Mondex OTP Protocol Version 1.0'
  PayReqNetLocn='http://www.mxbankus.com/etill/mx' >
</PayProtocol>
<PayProtocol ID ='M1.32'
  ProtocolId='MXv1.0'
  ProtocolName='Mondex OTP Protocol Version 1.0'
  PayReqNetLocn='http://www.mxbankuk.com/vserver' >
</PayProtocol>
<PayProtocol ID ='M1.33'
  ProtocolId='Ccashv1.0'
  ProtocolName='CyberCash Version 1.0'
  PayReqNetLocn='http://www.ccash.com/ccoin' >
</PayProtocol>
<PayProtocol ID ='M1.34'
  ProtocolId='CCashv2.0'
  ProtocolName='CyberCash Version 2.0'
  PayReqNetLocn='http://www.ccash.com/ccoin' >
</PayProtocol>
<PayProtocol ID ='M1.35'
  ProtocolId='GKv1.0'
  ProtocolName='GeldKarte Version 1.0'
  PayReqNetLocn='http://www.merchant.com/pgway' >
</PayProtocol>
<PayProtocol ID ='M1.36'
```

```
    ProtocolId='DCashv1.0'
    ProtocolName='DigiCash Protocol Version 1.0'
    PayReqNetLocn='http://www.merchant.com/digicash' >
  </PayProtocol>
  </BrandList>
```

10. XML Overview

This section contains an overview of [XML]. Its purpose is to provide sufficient explanation so that the XML examples in this document may be understood. This description is not complete. For more detail and the full specification see the reference contained in the Preface to this document.

XML is based on SGML has the goal of enabling "generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML."

XML is designed as a universal, open data format for the Internet and allows the structure of data in messages to be clearly and unambiguously defined.

In the following examples, underlined words are to be replaced by proper names; for example, document name could be `OtpMessage`, `EDIMessage`, and so on.

The structure of data in XML is defined using a number of key components. These are:

- document definitions
- element declarations and
- attribute declarations.

These are described below.

10.1 Document Definition

A document definition has the following form:

```
<!DOCTYPE DocumentName [DocumentStructureDefinition] >
```

`DocumentName`	For OTP this is always `OtpMessage`
`DocumentStructure Definition`	This contains the declarations of elements, attributes and entities.

For example:

```
<!DOCTYPE X [
   <!ELEMENT Y (Y1, Y2) >
   <!ATTLIST Y ('a' | 'b' | 'c') 'a' >
   <!ELEMENT Z (Z1 | Z2) >
   <!ATTLIST Z CDATA #REQUIRED>
]>
```

10.2 Element Declaration

An element declaration has the following form:

```
<!ELEMENT ElementName (ElementContents)>
```

`ElementContent`	This defines the relationships among the elements, the order of occurrences of the elements, and their number of occurrences.

10.2.1 Example 1

An element X consists of elements A, B, and C in that order. This would be declared as follows:

```
<!ELEMENT X (A, B, C) >
```

If the elements A, B, and C can appear in any order, `"&"` is used in place of `","`.

As XML this would be expressed as follows:

```
<X>
   <A> DataA </A>
   <B> DataB </B>
   <C> DataC </C>
</X>
```

In this example

- `"<X>"` is a start tag and `"</x>"` is an end tag
- `"DataA"`, `"DataB"`, and `"DataC"` is the content of the element and can consist of other elements, or character data or may even be empty.

10.2.2 Example 2

An element X consists of one of the elements A, B, or C. This would be declared as follows:

```
<!ELEMENT X ( A | B | C ) >
```

If element A is selected then this would be expressed as:

```
<X>
   <A> DataA </A>
</X>
```

10.2.3 Example 3

An element X consists of element A occurring zero or more times and element B occurring one or more times in that order. This would be declared as follows:

```
<!ELEMENT X ( A*, B+ ) >
```

The "*" indicates zero or more, and the "+" indicates one or more.

If A occurred zero times and b occurred twice then this would be expressed as:

```
<X>
   <B> DataB </B>
   <B> DataB </B>
</X>
```

10.2.4 Data Types used in element declarations

The previous examples described how one element can be defined as having "children" elements. In element declarations XML also supports data types. The data types used in this OTP specification are shown below.

Data Types	Descriptions
#PCDATA	The element content contains data which the XML parser can search to look for tags or entity declarations.
ANY	The element content can contain any element defined in any order.
EMPTY	The element content contains no data.

10.3 Attribute declarations

Attribute declarations describe information about an element. More than one attribute can be defined for one element. Attributes are contained within the start tag of an element. They are defined as follows:

```
<!ATTLIST ElementName AttributeName1 DeclaredValue1 DefaultValue1
            AttributeName2 DeclaredValue2 DefaultValue2
                        ...
            AttributeNameN DeclaredValueN DefaultValueN >
```

10.3.1 Declared value

When the permissible values of an attribute are known, those values are declared as a list in the declared value.

When the list of permissible values are not pre-defined, data types are specified instead. The data types which can be used for attribute declarations are listed below. Only the ones used in this OTP specification are shown.

Attribute Types	Descriptions
CDATA	Character data. Characters other than attribute value delimiters such as (“') can be used. Characters of zero length are allowed.
NMTOKEN	An attribute which conforms with the rules for an XML name. In outline it must start with a letter and be followed by any combination of letters,

	digits, or a few special characters. No spaces are allowed
NMTOKENS	One or more NMTOKEN separated by spaces.
ID	Identifier. This value of this attribute is unique for each element.
IDREF	This value of this attribute matches the value of some ID attribute of an element in the same XML document. It is used to point to that element.
IDREFS	One or more IDREFs separated by spaces.

10.3.2 Default value

Default values indicate whether or not the attribute must be present in an element.

For default values, the following default keywords as well as some concrete values can be specified. Only the default keywords used in this OTP specification are shown.

Values	Descriptions
#REQUIRED	CANNOT abbreviate. Some value must be specified for this attribute.
#IMPLIED	When an attribute with this default value is not specified, the application gives the pre-determined attribute value.
'value'	The 'value specified is the default. Other values may be used.
#FIXED 'value'	The value must and can only be the value specified

Example

An example of an attribute declaration follows:

```
<!ATTLIST X
   Att1 ( A, B ) #REQUIRED >
```

In this example a value for ATT1 must be present as it is "#REQUIRED" and it must be either "A" or "B" for the XML document to be valid. For example:

```
<X Att1='B'> DataX </X>
```

11. Open Trading Protocol Data Type Definition

This section contains a copy of the XML DTD for the Open Trading Protocols for information purposes.

The master copy of the DTD for OTP, which should be relied upon is available for download from the OTP web site (http:www.otp.org).

```
<!--
*****************************************************
*                                                   *
* OPEN TRADING PROTOCOL DTD VERSION 0.9.9           *
*                                                   *
*****************************************************

*****************************************************
* OTP MESSAGE DEFINITION                            *
*****************************************************
 -->

<!ELEMENT OtpMessage (TransRefBlk, SigBlk?, ErrorBlk?,
     ( AuthReqBlk |
       AuthRespBlk |
       DeliveryReqBlk |
       DeliveryRespBlk |
       InquiryReqBlk |
       InquiryRespBlk |
       OfferRespBlk |
       PayExchBlk |
       PayReqBlk |
       PayInstCCExchBlk |
       PayInstCCReqBlk |
       PayInstCCRespBlk
       PayRespBlk |
       PingReqBlk |
       PingRespBlk |
       TpoBlk |
       TpoSelectionBlk |
       )*
     ) >



<!--
*****************************************************
* TRANSACTION REFERENCE BLOCK DEFINITION            *
*****************************************************
 -->

<!ELEMENT TransRefBlk (TransId, MsgId, RelatedTo*) >
<!ATTLIST TransRefBlk
 ID                 ID      #REQUIRED >
```

```
<!ELEMENT TransId EMPTY >
<!ATTLIST TransId
 ID                 ID      #REQUIRED
 Version            NMTOKEN #FIXED '1.0'
 OtpTransId         NMTOKEN #REQUIRED
 OtpTransType       CDATA   #REQUIRED >
 TransTimeStamp     CDATA   #REQUIRED >


<!ELEMENT MsgId EMPTY >
<!ATTLIST MsgId
 ID                 ID      #REQUIRED
 RespOtpMsg         NMTOKEN #IMPLIED
 xml:lang           NMTOKEN #REQUIRED
 SoftwareId         CDATA   #REQUIRED
 TimeStamp          CDATA   #IMPLIED >


<!ELEMENT RelatedTo (PackagedContent) >
<!ATTLIST RelatedTo
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #REQUIRED
 RelationshipType   NMTOKEN #REQUIRED
 Relation           CDATA   #REQUIRED
 RelnKeyWords       NMTOKENS #IMPLIED >


<!--
*******************************************************
* Packaged Content Common Element                     *
*******************************************************
 -->

<!ELEMENT PackagedContent (#PCDATA) >
<!ATTLIST PackagedContent
 Content          NMTOKEN   "PCDATA"
 Transform (NONE|BASE64)    "NONE" >



<!--
*******************************************************
* TRADING COMPONENTS                                  *
*******************************************************
 -->
<!-- PROTOCOL OPTIONS COMPONENT -->
<!ELEMENT ProtocolOptions EMPTY >
<!ATTLIST ProtocolOptions
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #REQUIRED
 ShortDesc          CDATA   #REQUIRED
 SenderNetLocn      CDATA   #REQUIRED
 SecureSenderNetLocn CDATA  #REQUIRED
 SuccessNetLocn     CDATA   #REQUIRED
 CancelNetLocn      CDATA   #REQUIRED
 ErrorNetLocn       CDATA   #REQUIRED >
```

```
<!-- AUTHENTICATION DATA COMPONENT -->
<!ELEMENT AuthData (PackagedContent)>
<!ATTLIST AuthData
 ID                 ID      #REQUIRED
 AuthMethod         NMTOKEN #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >


<!-- AUTHENTICATION RESPONSE COMPONENT -->
<!ELEMENT AuthResp (PackagedContent) >
<!ATTLIST AuthResp
 ID                 ID      #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >


<!-- ORDER COMPONENT -->
<!ELEMENT Order (PackagedContent?) >
<!ATTLIST Order
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #REQUIRED
 OrderIdentifier    CDATA   #REQUIRED
 ShortDesc          CDATA   #REQUIRED
 OkFrom             CDATA   #REQUIRED
 OkTo               CDATA   #REQUIRED
 ApplicableLaw      CDATA   #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >


<!-- ORGANISATION COMPONENT -->
<!ELEMENT Org (TradingRole+, ContactInfo?,
     PersonName?, PostalAddress?)>
<!ATTLIST Org
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #REQUIRED
 OrgId              CDATA   #REQUIRED
 OtpMsgIdPrefix     NMTOKEN #REQUIRED
 LegalName          CDATA   #IMPLIED
 ShortDesc          CDATA   #IMPLIED
 LogoNetLocn        CDATA   #IMPLIED >


<!ELEMENT TradingRole EMPTY >
<!ATTLIST TradingRole
 TradingRole        NMTOKEN #REQUIRED
 ErrorNetLocn       CDATA   #IMPLIED >


<!ELEMENT ContactInfo EMPTY >
<!ATTLIST ContactInfo
 xml:lang           NMTOKEN #IMPLIED
 Tel                CDATA   #IMPLIED
 Fax                CDATA   #IMPLIED
 Email              CDATA   #IMPLIED
 NetLocn            CDATA   #IMPLIED >
```

```
<!ELEMENT PersonName EMPTY >
<!ATTLIST PersonName
 xml:lang           NMTOKEN #IMPLIED
 Title              CDATA   #IMPLIED
 GivenName          CDATA   #IMPLIED
 Initials           CDATA   #IMPLIED
 FamilyName         CDATA   #IMPLIED >


<!ELEMENT PostalAddress EMPTY >
<!ATTLIST PostalAddress
 xml:lang           NMTOKEN #IMPLIED
 AddressLine1       CDATA   #IMPLIED
 AddressLine2       CDATA   #IMPLIED
 CityOrTown         CDATA   #IMPLIED
 StateOrRegion      CDATA   #IMPLIED
 PostalCode         CDATA   #IMPLIED
 Country            CDATA   #IMPLIED
 LegalLocation (True|False) 'False' >

<!-- BRAND LIST COMPONENT -->
<!ELEMENT BrandList (Brand+, ProtocolAmount+,
 CurrencyAmount+, PayProtocol+) >
<!ATTLIST BrandList
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #REQUIRED
 ShortDesc          CDATA   #REQUIRED
 PayDirection (Debit|Credit) #REQUIRED >

<!ELEMENT Brand (PackagedContent?) >
<!ATTLIST Brand
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #IMPLIED
 BrandId            NMTOKEN #REQUIRED
 BrandName          CDATA   #REQUIRED
 BrandLogoNetLocn   CDATA   #REQUIRED
 BrandNarrative     CDATA   #IMPLIED
 ProtocolAmountRefs IDREFS  #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >

<!ELEMENT ProtocolAmount (PackagedContent?) >
<!ATTLIST ProtocolAmount
 ID                 ID      #REQUIRED
 PayProtocolRef     IDREF   #REQUIRED
 CurrencyAmountRefs IDREFS  #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >

<!ELEMENT CurrencyAmount EMPTY >
<!ATTLIST CurrencyAmount
 ID                 ID      #REQUIRED
 Amount             CDATA   #REQUIRED
 CurrCodeType       NMTOKEN 'ISO4217'
 CurrCode           CDATA   #REQUIRED >
```

```
<!ELEMENT PayProtocol (PackagedContent?) >
<!ATTLIST PayProtocol
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #IMPLIED
 ProtocolId         NMTOKEN #REQUIRED
 ProtocolName       CDATA   #REQUIRED
 ActionOrgRef       NMTOKEN #REQUIRED
 PayReqNetLocn      CDATA   #IMPLIED
 SecPayReqNetLocn   CDATA   #IMPLIED
 ContentSoftwareId  CDATA   #IMPLIED >


<!-- BRAND SELECTION COMPONENT -->
<!ELEMENT BrandSelection (BrandSelBrandInfo?,
     BrandSelProtocolAmountInfo?,
     BrandSelCurrencyAmountInfo?) >
<!ATTLIST BrandSelection
 ID                 ID      #REQUIRED
 BrandListRef       NMTOKEN #REQUIRED
 BrandRef           NMTOKEN #REQUIRED
 ProtocolAmountRef  NMTOKEN #REQUIRED
 CurrencyAmountRef  NMTOKEN #REQUIRED >

<!ELEMENT BrandSelBrandInfo (PackagedContent) >
<!ATTLIST BrandSelBrandInfo
 ID                 ID      #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >

<!ELEMENT BrandSelProtocolAmountInfo (PackagedContent) >
<!ATTLIST BrandSelProtocolAmountInfo
 ID                 ID      #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >

<!ELEMENT BrandSelCurrencyAmountInfo (PackagedContent) >
<!ATTLIST BrandSelCurrencyAmountInfo
 ID                 ID      #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >


<!-- PAYMENT COMPONENT -->
<!ELEMENT Payment (PackagedContent?) >
<!ATTLIST Payment
 ID                 ID      #REQUIRED
 OkFrom             CDATA   #REQUIRED
 OkTo               CDATA   #REQUIRED
 BrandListRef       NMTOKEN #REQUIRED
 SignedPayReceipt (True | False) #REQUIRED
 AuthDataRef        NMTOKEN #IMPLIED
 StartAfter         NMTOKENS #IMPLIED >


<!-- PAYMENT SCHEME COMPONENT -->
<!ELEMENT PaySchemeData (PackagedContent) >
<!ATTLIST PaySchemeData
 ID                 ID      #REQUIRED
 ConsumerPaymentId  CDATA   #IMPLIED
 PaymentHandlerPayId CDATA  #IMPLIED
 ContentSoftwareId  CDATA   #IMPLIED >
```

```
<!-- PAYMENT RECEIPT COMPONENT -->
<!ELEMENT PayReceipt (PackagedContent) >
<!ATTLIST PayReceipt
 ID                 ID      #REQUIRED
 PaymentRef         NMTOKEN #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >


<!-- DELIVERY COMPONENT -->
<!ELEMENT Delivery (DeliveryData?, PackagedContent?) >
<!ATTLIST Delivery
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #REQUIRED
 DelivExch          (True|False) #REQUIRED
 DelivAndPayResp    (True|False) #REQUIRED
 ActionOrgRef       NMTOKEN #IMPLIED
 ConsumerDeliveryId CDATA   #IMPLIED >

<!ELEMENT DeliveryData (PackagedContent?) >
<!ATTLIST DeliveryData
 xml:lang           NMTOKEN #IMPLIED
 OkFrom             CDATA   #REQUIRED
 OkTo               CDATA   #REQUIRED
 DelivMethod        NMTOKEN #REQUIRED
 DelivToRef         NMTOKEN #REQUIRED
 DelivReqNetLocn    CDATA   #REQUIRED
 SecDelivReqNetLocn CDATA   #REQUIRED
 ContentSoftwareId  CDATA   #IMPLIED >


<!-- DELIVERY NOTE COMPONENT -->
<!ELEMENT DeliveryNote (PackagedContent) >
<!ATTLIST DeliveryNote
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #REQUIRED
 DelivHandlerDelivId CDATA  #IMPLIED
 ContentSoftwareId  CDATA   #IMPLIED >


<!-- PAYMENT METHOD INFORMATION COMPONENT -->
<!ELEMENT PayMethodInfoData EMPTY >
<!ATTLIST PayMethodInfoData
 ID                 ID      #REQUIRED
 BrandId            NMTOKEN #REQUIRED
 PayProtocolId      NMTOKEN #IMPLIED >
```

```
<!-- STATUS COMPONENT -->
<!ELEMENT Status EMPTY >
<!ATTLIST Status
 ID                 ID      #REQUIRED
 xml:lang           NMTOKEN #REQUIRED
 StatusType (Offer|Payment|Delivery) #REQUIRED
 ElRef              NMTOKEN #REQUIRED
 ProcessState (NotYetStarted|InProgress|
     CompletedOk|Failed|ProcessError) #REQUIRED
 CompletionCode     NMTOKEN #IMPLIED
 ProcessReference   CDATA   #IMPLIED
 StatusDesc         CDATA   #IMPLIED >


<!-- INQUIRY TYPE COMPONENT -->
<!ELEMENT InquiryType EMPTY >
<!ATTLIST InquiryType
 ID                 ID      #REQUIRED
 Type (Offer|Payment|Delivery) #REQUIRED
 ElRef              NMTOKEN #IMPLIED
 ProcessReference   CDATA   #IMPLIED >


<!-- ERROR COMPONENT -->
<!ELEMENT ErrorComp (ErrorLocation+, PackagedContent*) >
<!ATTLIST ErrorComp
 ID                 NMTOKEN #REQUIRED
 xml:lang           NMTOKEN #REQUIRED
 ErrorCode          NMTOKEN #REQUIRED
 ErrorDesc          CDATA   #REQUIRED
 Severity (Warning|TransientError|HardError) #REQUIRED
 MinRetrySecs       CDATA   #IMPLIED
 SwVendorErrorRef   CDATA   #IMPLIED >


<!ELEMENT ErrorLocation EMPTY >
<!ATTLIST ErrorLocation
 ElementType        NMTOKEN #REQUIRED
 OtpMsgRef          NMTOKEN #REQUIRED
 BlkRef             NMTOKEN #IMPLIED
 CompRef            NMTOKEN #IMPLIED
 ElementRef         NMTOKEN #IMPLIED
 AttName            NMTOKEN #IMPLIED >
```

```
<!--
*****************************************************
* TRADING BLOCKS                                    *
*****************************************************
 -->

<!-- TRADING PROTOCOL OPTIONS BLOCK -->
<!ELEMENT TpoBlk ( ProtocolOptions, BrandList*, Org* ) >
<!ATTLIST TpoBlk
 ID                 ID      #REQUIRED >


<!-- TPO SELECTION BLOCK -->
<!ELEMENT TpoSelectionBlk (BrandSelection+) >
<!ATTLIST TpoSelectionBlk
 ID                 ID      #REQUIRED >


<!-- OFFER RESPONSE BLOCK -->
<!ELEMENT OfferRespBlk (AuthData*, Order?, Payment*,
     Delivery?, Status ) >
<!ATTLIST OfferRespBlk
 ID                 ID      #REQUIRED >


<!-- AUTHENTICATION REQUEST BLOCK -->
<!ELEMENT AuthReqBlk (AuthData?) >
<!ATTLIST AuthReqBlk
 ID                 ID      #REQUIRED >


<!-- AUTHENTICATION RESPONSE BLOCK -->
<!ELEMENT AuthRespBlk (AuthResp, Org+) >
<!ATTLIST AuthRespBlk
 ID                 ID      #REQUIRED >


<!-- PAYMENT REQUEST BLOCK -->
<!ELEMENT PayReqBlk (AuthData?, BrandList, BrandSelection,
     Payment, PaySchemeData?, Org*) >
<!ATTLIST PayReqBlk
 ID                 ID      #REQUIRED >


<!-- PAYMENT EXCHANGE BLOCK -->
<!ELEMENT PayExchBlk (PaySchemeData) >
<!ATTLIST PayExchBlk
 ID                 ID      #REQUIRED >


<!-- PAYMENT RESPONSE BLOCK -->
<!ELEMENT PayRespBlk (Status, PayReceipt, PaySchemeData?) >
<!ATTLIST PayRespBlk
 ID                 ID      #REQUIRED >
```

```
<!-- DELIVERY REQUEST BLOCK -->
<!ELEMENT DeliveryReqBlk (Order, Org*, Delivery) >
<!ATTLIST DeliveryReqBlk
 ID                 ID      #REQUIRED >


<!-- DELIVERY RESPONSE BLOCK -->
<!ELEMENT DeliveryRespBlk (Status, DeliveryNote) >
<!ATTLIST DeliveryRespBlk
 ID                 ID      #REQUIRED >


<!-- PAYMENT INSTRUMENT CUSTOMER CARE REQUEST BLOCK -->
<!ELEMENT PayInstCCReqBlk (PaymethodInfo, PaySchemeData*) >
<!ATTLIST PayInstCCReqBlk
 ID                 ID      #REQUIRED >


<!-- PAYMENT INSTRUMENT CUSTOMER CARE EXCHANGE BLOCK -->
<!ELEMENT PayInstCCExchBlk (PaySchemeData) >
<!ATTLIST PayInstCCExchBlk
 ID                 ID      #REQUIRED >


<!-- PAYMENT INSTRUMENT CUSTOMER CARE RESPONSE BLOCK -->
<!ELEMENT PayInstCCRespBlk (PaySchemeData) >
<!ATTLIST PayInstCCRespBlk
 ID                 ID      #REQUIRED >


<!-- INQUIRY REQUEST BLOCK -->
<!ELEMENT InquiryReqBlk ( InquiryType, PaySchemeData? ) >
<!ATTLIST InquiryReqBlk
 ID                 ID      #REQUIRED >


<!-- INQUIRY RESPONSE BLOCK -->
<!ELEMENT InquiryRespBlk (Status, PaySchemeData?) >
<!ATTLIST InquiryRespBlk
 ID                 ID      #REQUIRED
 LastReceivedOtpMsgRef NMTOKEN #IMPLIED
 LastSentOtpMsgRef  NMTOKEN #IMPLIED >


<!-- PING REQUEST BLOCK -->
<!ELEMENT PingReqBlk (Org*)>
<!ATTLIST PingReqBlk
 ID                 ID      #REQUIRED>


<!-- PING RESPONSE BLOCK -->
<!ELEMENT PingRespBlk (Org+)>
<!ATTLIST PingRespBlk
 ID                 ID      #REQUIRED
 PingStatusCode (Ok|Busy|Down) #REQUIRED
 SigVerifyStatusCode (Ok|NotSupported|Fail) #IMPLIED
 xml:lang           NMTOKEN #IMPLIED
 PingStatusDesc     CDATA   #IMPLIED>
```

```
<!-- SIGNATURE BLOCK -->
<!ELEMENT SigBlk (OtpSig+, OtpCert*) >
<!ATTLIST SigBlk
 ID                 ID      #REQUIRED >


<!-- ERROR BLOCK -->
<!ELEMENT ErrorBlk (ErrorComp+, PaySchemeData*) >
<!ATTLIST ErrorBlk
 ID                 ID      #REQUIRED >
```

12. References

This section contains references to related documents identified in this specification.

[Base64] Base64 Content-Transfer-Encoding. A method of transporting binary data defined by MIME. See: RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. N. Freed & N. Borenstein. November 1996.

[DNS] The Internet Domain Name System which allocates Internet names to organisations for example "otp.org", the Domain Name for OTP. See RFC 1034: Domain names - concepts and facilities. P.V. Mockapetris. Nov-01-1987, and RFC 1035: Domain names - implementation and specification. P.V. Mockapetris. Nov-01-1987.

[DSA] The Digital Signature Algorithm (DSA) published by the National Institute of Standards and Technology (NIST) in the Digital Signature Standard (DSS), which is a part of the US government's Capstone project.

[ECCDSA] Elliptic Curve Cryptosystems Digital Signature Algorithm (ECCDSA). Elliptic curve cryptosystems are analogs of public-key cryptosystems such as RSA in which modular multiplication is replaced by the elliptic curve addition operation. See: V. S. Miller. Use of elliptic curves in cryptography. In Advances in Cryptology - Crypto '85, pages 417-426, Springer-Verlag, 1986.

[HTML] Hyper Text Mark Up Language. The Hypertext Mark-up Language (HTML) is a simple mark-up language used to create hypertext documents that are platform independent. See RFC 1866 and the World Wide Web (W3C) consortium web site at: http://www.w3.org/MarkUp/

[HTTP] Hyper Text Transfer Protocol versions 1.0 and 1.1. See RFC 1945: Hypertext Transfer Protocol -- HTTP/1.0. T. Berners-Lee, R. Fielding & H. Frystyk. May 1996. and RFC 2068: Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. January 1997.

[ISO4217] ISO 4217: Codes for the Representation of Currencies. Available from ANSI or ISO.

[MIME] Multipurpose Internet Mail Extensions. See RFC822, RFC2045, RFC2046, RFC2047, RFC2048 and RFC2049.

[OPS] Open Profiling Standard. A proposed standard which provides a framework with built-in privacy safeguards for the trusted exchange of profile information between individuals and web sites. Being developed by Netscape and Microsoft amongst others.

[RFC822] IETF (Internet Engineering Task Force). RFC 822: The Standard for the Format of ARPA Internet Messages

. 13 August 1982, David H Crocker. 13 August 1982.

[RFC1738] IETF (Internet Engineering Task Force). RFC 1738: Uniform Resource Locators (URL), ed. T. Berners-Lee, L. Masinter, M. McCahill. 1994.

[RSA] RSA is a public-key cryptosystem for both encryption and authentication supported by RSA Data Security Inc. See: R. L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2): 120-126, February 1978.

[SCCD] Secure Channel Credit Debit. A method of conducting a credit or debit card

payment where unauthorised access to account information is prevented through use of secure channel transport mechanisms such as SSL. An OTP supplement describing how SCCD works is under development. Author. Jonathan Sowler JCP,

[SET] Secure Electronic Transaction Specification, Version 1.0, May 31, 1997. Supports credit and debit card payments using certificates at the Consumer and Merchant to help ensure authenticity.
Download from: <http://www.mastercard.com/set/specs.html>.

[SHA1] [FIPS-180-1]"Secure Hash Standard", National Institute of Standards and Technology, US Department Of Commerce, April 1995. Also known as: 59 Fed Reg. 35317 (1994).

[UTC] Universal Time Co-ordinated. A method of defining time absolutely relative to Greenwich Mean Time (GMT). Typically of the form: `"CCYY-MM-DDTHH:MM:SS.sssZ+n"` where the `"+n"` defines the number of hours from GMT. See ISO DIS8601.

[UTF16] The Unicode Standard, Version 2.0. The Unicode Consortium, Reading, Massachusetts. See ISO/IEC 10646 1 Proposed Draft Amendment 1

[X.509] ITU Recommendation X.509 1993 | ISO/IEC 9594-8: 1995, Including Draft Amendment 1: Certificate Extensions (Version 3 Certificate)

[XML Namespace] See Design decisions reached at the XML Working Group meeting in Montreal, Canada, August 22, 1987

[XML] Extensible Mark Up Language. See http://www.w3.org/TR/PR-xml-971208 for the 8 December 1997 version.

[XMLSIG] A proposal developed by the OTP consortium describing an approach to signing XML documents such as OTP Messages. It is intended that this document is submitted to W3C for consideration. Author. Richard Brown. GlobeSet. (Under preparation August 1998)

13. Author's Address

The author of this document is:

David Burdett
Development Director
Mondex International Ltd
Advanced Technology Division
111 Pine St
San Francisco, 94111
California
USA

Tel: +1 (415) 645 6973

Email: david.burdett@mondex.com

The author of this document appreciates the following contributors to this protocol (in alphabetic order of company) without which it could not have been developed.

- Phillip Mullarkey, British Telecom plc
- Andrew Marchewka, Canadian Imperial Bank of Commerce
- Brian Boesch, CyberCash Inc.
- Donald Eastlake 3rd, CyberCash Inc.
- Mark Linehan, International Business Machines
- Peter Chang, Hewlett Packard
- Masaaki Hiroya, Hitachi Ltd
- Yoshiaki Kawatsura, Hitachi Ltd
- Jonathan Sowler, JCP Computer Services Ltd
- John Wankmueller, MasterCard International
- Steve Fabes, Mondex International Ltd
- Akihiro Nakano, Plat Home, Inc. (ex Hitachi Ltd)
- Chris Smith, Royal Bank of Canada
- Hans Bernhard-Beykirch, SIZ (IT Development and Coordination Centre of the German Savings Banks Organisation)
- W. Reid Carlisle, Spyrus (ex Citibank Universal Card Services, formally AT&T Universal Card Services)
- Efrem Lipkin, Sun Microsystems
- Terry Allen, Veo Systems

The author would also like to thank the following organisations for their support:

- Amino Communications
- DigiCash

- Fujitsu
- General Information Systems
- Globe Id Software
- Hyperion
- InterTrader
- Nobil I T Corp
- Mercantec
- Netscape
- Nippon Telegraph and Telephone Corporation
- Oracle Corporation
- Smart Card Integrations Ltd.
- Spyrus
- Verifone
- Unisource nv
- Wells Fargo Bank