



ePDQ
MPI Integration Guide - XML

Version 4.0 March 2011

Software Version: 5.9 Payment Engine

COPYRIGHT NOTICE

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic or mechanical, including photocopying and recording, for any purpose, without the prior written permission of Product Development, Barclaycard Payment Acceptance, Barclays Bank PLC.

Contents Page

Contents Page.....	2
Audience.....	3
Contacting Us.....	3
Documentation	3
Introduction	4
MPI Overview.....	5
Integration Context.....	6
Connecting to the ePDQ engine.....	6
Example HTML for XML Input Document submission	7
Transaction Documents.....	8
XML OrderFormDoc – Input Document Example.....	8
XML OrderFormDoc – Input Document Example (continued)	9
OrderFormDoc Input Document Field Definitions	10
Table of Field Definitions.....	10
Table of Field Definitions (continued)	11
Internet Authentication Fields.....	11
OrderFormDoc – Output Document Example	12
OrderFormDoc Output Document Field Definitions	14
Report Documents.....	15
XML ReportDoc 1– Input Document Example.....	16
ReportDoc 1 Input Document Field Definitions	17
XML ReportDoc 1 – Output Document Example.....	18
XML ReportDoc 1 – Output Document Example (continued)	19
ReportDoc 1 Output Document Field Definitions	19
XML ReportDoc 2 – Input Document Example	20
ReportDoc 2 Input Document Field Definitions.....	21
XML ReportDoc 2 – Output Document Example	22
XML ReportDoc 2 – Output Document Example (continued).....	23
ReportDoc 2 Output Document Field Definitions.....	23
MPI Integration Checklist	24
Common Error Responses	25
Frequently Asked Questions.....	27
Changes and Amendments	30

Audience

This document is intended to be used by the developer who will perform the integration of the MPI solution. This integration will typically involve the following, to be carried out by the nominated developer:

- Provision of shopping cart
- Provision of secure checkout page for collecting card details
- Provision of script or 'wrapper' for creating the XML authorisation request
- Configuration of server to facilitate transaction request
- Compliance with Payment Card Industry Data Security Standards (PCI DSS)
- All future updates/upgrades dictated by changes to the ePDQ Engine version.

Contacting Us

Telephone Number – 0844 822 2099*
Email Address – epdq@barclaycard.co.uk

Opening hours - 8:00am to midnight, Monday to Sunday.

** Calls may be monitored or recorded to maintain high levels of security and quality of service*

Documentation

Please visit the Barclaycard website to access this document and several others that can help with your integration and store setup. . The documents can be located at the following URL:

<http://www.barclaycard.co.uk/business/existing-customers/activate-epdq-mpi>

Introduction

Welcome to the ePDQ Merchant Payment Interface (MPI) Integration Guide for the XML Input Component.

This document explains the integration process for the ePDQ Merchant Payment Interface (MPI) online transaction processing gateway, with particular reference to the XML Input Component. It also provides answers to commonly asked questions, and includes an XML authorisation request and response document, and a report request and response document, which includes annotation to explain the structure and contents required for a valid authorisation request.

This document should be used in conjunction with the following integration guides:

- ClearCommerce API Reference and Guide
- Document Hierarchy Reference: OrderFormDocs
- Document Hierarchy Reference: ReportDocs
- Using the XML Input Component
- Barclaycard Payment Reference

These documents explain in detail all aspects of the MPI's functionality and usage. They should be referred to in the first instance to address any issues outside of the scope of this document, and should be referred to in conjunction with this document to perform a successful MPI integration.

There are some references to Internet Authentication in this document. These will be relevant to the MPI integration only if the Internet Authentication Software Development Kit (SDK) or an alternative 3D Secure software is also integrated into the merchant's website; the Authentication process generates some values which need to be included in the MPI authorisation request document.

If you do not have copies of any of these documents, please contact us. If you are using a Merchant Development Partner or Web Developer, the URL where documentation can be downloaded from may have been provided direct to your developer.

MPI Overview

With ePDQ Merchant Payment Interface (MPI) an application developer can choose between the C++ API, the Java API, or the XML input component to send transactions, configuration updates and report requests to the ePDQ Engine.

The creation of an XML document, and submission to the XML input component, requires no client software from ClearCommerce. An understanding of the ClearCommerce document hierarchy will be essential in creating a correctly formed XML document bound for the ClearCommerce Engine. (See *Document Hierarchy Reference: OrderFormDocs* for more information.)

- To create an XML document, a developer can use an XML editor, text editor, Java code, or other mechanisms that are capable of creating and formatting XML documents.
- To transport an XML document to the XML input component, the client can utilize a browser, Java code or other tools capable of performing HTTPS POSTs.
- To aid in the development of XML input component client applications, sample Java code and XML documents are provided in the standard ClearCommerce documentation.

Please note, the XML examples provided are U.S. centric, and, if used as a template for your XML Input Documents, must be modified to suit your integration requirements.

To supplement the ClearCommerce provided example XML, this document contains UK-centric XML Input Document examples.

Integration Context

A standard website based MPI integration would usually involve the following steps:

1. Once the cardholder has decided to buy, your website/shopping cart will redirect them to your final 'Checkout' page.
2. This page collects all cardholder and credit card data and sends it to a server side script/application developed by the merchant.
3. This server side script/application will either make calls to the supplied C++ library / Java Class files and use the ClearCommerce supplied API to generate the transaction request 'documents', or will generate XML transaction 'documents' using standard web scripting processes.
4. The transaction request 'document' is posted via an SSL enabled URL directly into the ePDQ engine for the specified transaction type to be performed.
5. The transaction is processed by the ePDQ server. The transaction status results (Success, Declined, etc.) are sent back to the merchant's application in the same format as the transaction request. This all takes place within the same 'post' session.

Please refer to the document 'Using the XML Input Component' (Chapter 2) for a description of the data flow through the ePDQ system.

Connecting to the ePDQ engine

Before you can proceed with the integration you will need the following information. These details will have been sent to the nominated email contact for the merchant when the ePDQ Store/Client ID was initially created:

- Client Id
- Username
- Password
- Access to the MPI Integration documents
- Connection details for submitting transactions to ePDQ; these details, plus links to the MPI Integration documentation are emailed to the nominated merchant email address when the ePDQ store is created.

Using the XML Input Component to test the formatting of your XML you can submit transactions directly via an HTML test page, detailed in the ClearCommerce XML Guide, chapter 3; you will need to change the action of the HTML form to submit transactions to the ePDQ payment gateway.

Please see the following page for a simple HTML example which can be used to test XML Input against the XML Input Component.

Example HTML for XML Input Document submission

```
<html>

<body>

<form action=epdq hostname(*) method="post">

<textarea name="CLRCMRC_XML" rows="50" cols="100"></textarea>

<input type="submit" value="press">

<input type="reset" value="reset">

</form>

</body>

</html>
```

This code should be copied into a standard text editor and saved as a '.HTML' file. When opened in a browser, this webpage will provide a form which allows you to submit an XML Input document directly into the ePDQ engine. The response from ePDQ will be displayed in the browser window. This provides a means of testing the integrity of your XML transaction request documents.

(*) The Hostname and Port Number for ePDQ can be obtained from the 'ePDQ MPI Next Steps' email which was sent to the account owner on creation of their ePDQ account.

Transaction Documents

The following pages contain an example XML Input Document, followed by an example Output Document for performing a typical authorisation request.

The Input Document contains the mandatory fields that must be included (in bold), as well as typical fields that you may wish to include, such as the customer's billing and delivery addresses, Card Security Code and Payer Authentication information (where the Internet Authentication SDK has also been integrated).

Please note that this is an example structure only, there are many other values you can pass that are defined within the ClearCommerce documentation.

It is recommended that if you include an element or node in the XML it should have an appropriate value. Passing value-less fields can result in transaction failure.

Following the Input Document there are explanations of the fields being passed, and references to the ClearCommerce documentation for further reading.

The Output Document is again a typical example, followed by an explanation of the most important fields returned, and again references to the ClearCommerce documentation.

XML OrderFormDoc – Input Document Example

```
<EngineDocList>
  <DocVersion>1.0</DocVersion>
  <EngineDoc>
    <ContentType>OrderFormDoc</ContentType>
    <User>
      <Name>Your Username</Name>
      <Password>Your Password</Password>
      <ClientId DataType="S32">Your Client Id</ClientId>
    </User>
    <Instructions>
      <Pipeline>Payment</Pipeline>
    </Instructions>
    <OrderFormDoc>
      <Mode>P</Mode>
      <Id>Your Order Id</Id>
      <Consumer>
        <Email>Customer's email address</Email>
        <BillTo>
          <Location>
            <Address>
              <FirstName>John</FirstName>
              <LastName>Smith</LastName>
```

XML OrderFormDoc – Input Document Example (continued)

```
<Street1>1 High Street</Street1>
<Street2></Street2>
<Street3></Street3>
<City>Northampton</City>
<StateProv>Northants</StateProv>
<PostalCode>NN1 1NN</PostalCode>
<Country>826</Country>
</Address>
</Location>
</BillTo>
<ShipTo>
  <Location>
    <Address>
      <FirstName>Jane</FirstName>
      <LastName>Smith</LastName>
      <Street1>22 High Street</Street1>
      <Street2></Street2>
      <Street3></Street3>
      <City>Northampton</City>
      <StateProv>Northants</StateProv>
      <PostalCode>NN1 1NN</PostalCode>
      <Country>826</Country>
    </Address>
  </Location>
</ShipTo>
<PaymentMech>
  <CreditCard>
    <Type DataType="S32">1</Type>
    <Number>4111111111111111</Number>
    <Expires DataType="ExpirationDate" Locale="826">01/10</Expires>
    <IssueNum></IssueNum>
    <StartDate DataType="StartDate"></StartDate>
    <Cvv2Indicator>1</Cvv2Indicator>
    <Cvv2Val>999</Cvv2Val>
  </CreditCard>
</PaymentMech>
</Consumer>
<Transaction>
  <Type>Auth</Type>
  <CurrentTotals>
    <Totals>
      <Total DataType="Money" Currency="826">200</Total>
    </Totals>
  </CurrentTotals>
  <CardholderPresentCode DataType="S32"></CardholderPresentCode>
  <PayerSecurityLevel DataType="S32"></PayerSecurityLevel>
  <PayerAuthenticationCode></PayerAuthenticationCode>
  <PayerTxnId></PayerTxnId>
</Transaction>
</OrderFormDoc>
</EngineDoc>
</EngineDocList>
```

OrderFormDoc Input Document Field Definitions

All the fields passed in the Input Document are defined in detail in the ClearCommerce documentation. The following table contains a list of the mandatory and most common values passed.

The data type of each field is "String" unless specified. Any data types other than String must be specified in the input document. Any empty elements with a non-string data type must be removed as they will cause an error in the response document.

The right hand column of the table below contains a reference to the ClearCommerce documentation where further information can be found if required. The page number refers to the 'Document Hierarchy Reference: OrderFormDocs' unless specified.

Table of Field Definitions

Field	Description	DHR_OrderFormDoc page number
<Name></Name> <Password></Password> <ClientId DataType="S32"></ClientId>	The ePDQ username, password and store id. Please ensure the merchant has created a user account with permissions appropriate to the functionality of your MPI integration	46 47 43
<Pipeline>Payment</Pipeline>	If you are using the fraud rules, for instance for Card Security Code checking, or Address Verification, this should always be set to Payment. To bypass the fraud rules use 'PaymentNoFraud'	49
<Mode>P</Mode>	Live transactions can only be submitted using mode P. Other simulation modes available are Y, N, R, FY and FN (T is not available). If you complete a transaction using any of these modes, the order details will appear in the Store Administration	88
<Id></Id>	You can pass your own order id's here. You must use alpha/numeric characters only, to a maximum of 36 characters	87
<Email></Email>	You can pass the customer's email address here	95
<BillTo></BillTo>	In this record you can pass the customer's billing name and address. This is optional, but is required if you are using the Fraud Rules to verify the customer's address	115-127
<ShipTo></ShipTo>	In this record you can pass the customer's delivery address details, which is also optional	193-205

Table of Field Definitions (continued)

<Type DataType="S32">1</Type>	Credit card type, e.g. 1=VISA	159-160
<IssueNum></IssueNum>	UK Maestro/Solo only (1 or 2 digits). These cards may have a start date, or an issue number, or both. Must be included if present on the card.	149
<StartDate DataType="StartDate"></StartDa te>	UK Maestro/Solo only. In the format MM/YY, or MM/DD/YY. Must be included if present on the card	153
<Cvv2Indicator>1</Cvv2Indicator>	Must be 1 if Cvv2Val is submitted	144
<Cvv2Val>999</Cvv2Val>	The card security code. This is not mandatory but highly recommended as a fraud prevention measure when used in conjunction with the Fraud Rules	145
<Type>Auth</Type>	Use Auth to authorise and settle transactions at the same time. Use PreAuth to authorise only, to ship at a later stage. Other transaction types are available, and are listed in the ClearCommerce documentation. The API Guide contains the required structure of the input document according to the transaction type being submitted	382
<Total DataType="Money" Currency="826">200</Total>	Use the correct currency code for the store id being used, e.g. Sterling=826, USD=840, Euro=978	API Guide, Appendix K
<CardholderPresentCode DataType="S32"></CardholderPr esentCode>	This field indicates the type of transaction being processed. The default value is 7, other values are 8 (for recurring billing transactions), and 13 (for orders where Internet Authentication was used)	331
<PayerSecurityLevel DataType="S32"></PayerSecurit yLevel>	(Internet Authentication orders only) This field indicates the results of the Internet Authentication process	355-357 Also API Guide, page 40/41
<PayerAuthenticationCode></Pay erAuthenticationCode>	(Internet Authentication orders only) Value generated during the Internet Authentication process. Proof that the process took place	354 Also API Guide, page 39
<PayerTxnId></PayerTxnId>	(Internet Authentication orders only) The transaction id generated during the authentication process	358 Also API Guide, page 39

Internet Authentication Fields

Please note, the PayerTxnID and PayerAuthenticationCode fields must be correctly encoded to ensure special characters (+, / and =) are not rejected by ePDQ as invalid. Any special characters must be URL/HTML encoded by the application or program you use for creating your authorisation requests – particular attention should be paid to the '+' symbol, as this is used in URL encoding to denote a 'space' character. The appropriate URL/HTML encoding for a '+' character is %2B.

OrderFormDoc – Output Document Example

The following XML is typical of the response you would receive if the Input Document described previously was sent to the ePDQ engine for processing:

```
<?xml version="1.0" encoding="UTF-8"?>
<EngineDocList>
  <DocVersion DataType="String">1.0</DocVersion>
  <EngineDoc>
    <ContentType DataType="String">OrderFormDoc</ContentType>
    <DocumentId DataType="String">445029fd-1a8e-3000-0005-
0003ba65c10f</DocumentId>
    <Instructions>
      <Pipeline DataType="String">Payment</Pipeline>
    </Instructions>
    <MessageList>
      <MaxSev DataType="S32"></MaxSev>
      <Message>
        <Sev DataType="S32"></Sev>
        <Text DataType="String"></Text>
      </Message>
    </MessageList>
    <OrderFormDoc>
      <Consumer>
        <BillTo>
          <Location>
            <Address>
              <City DataType="String">Northampton</City>
              <Country DataType="String">826</Country>
              <FirstName DataType="String">John</FirstName>
              <LastName DataType="String">Smith</LastName>
              <PostalCode DataType="String">NN11NN</PostalCode>
              <StateProv DataType="String">Northants</StateProv>
              <Street1 DataType="String">1 High Street</Street1>
            </Address>
          </Location>
        </BillTo>
        <Email DataType="String">epdq@barclaycard.co.uk</Email>
        <PaymentMech>
          <CreditCard>
            <CardSpan DataType="String">41111111</CardSpan>
            <Type DataType="S32">1</Type>
          </CreditCard>
          <Type DataType="String">CreditCard</Type>
        </PaymentMech>
        <ShipTo>
          <Location>
            <Address>
              <City DataType="String">Northampton</City>
              <Country DataType="String">826</Country>
              <FirstName DataType="String">Jane</FirstName>
              <LastName DataType="String">Smith</LastName>
              <PostalCode DataType="String">NN1 1NN</PostalCode>
```

OrderFormDoc – Output Document Example (continued)

```
<StateProv DataType="String">Northants</StateProv>
<Street1 DataType="String">22 High Street</Street1>
</Address>
</Location>
</ShipTo>
</Consumer>
<DateTime DataType="DateTime">1146133410918</DateTime>
<FraudInfo>
<FraudResultCode DataType="S32">0</FraudResultCode>
<TotalScore DataType="Numeric" Precision="0">0</TotalScore>
</FraudInfo>
<GroupId DataType="String">445029fd-1a8f-3000-0005-
0003ba65c10f</GroupId>
<Id DataType="String">445029fd-1a8f-3000-0005-0003ba65c10f</Id>
<Mode DataType="String">P</Mode>
<Transaction>
<AuthCode DataType="String">611287</AuthCode>
<CardProcResp>
<AvsDisplay DataType="String">YY</AvsDisplay>
<AvsRespCode DataType="String">EX</AvsRespCode>
<CcErrCode DataType="S32">1</CcErrCode>
<CcReturnMsg DataType="String">Approved</CcReturnMsg>
<Cvv2Resp DataType="String">1</Cvv2Resp>
<ProcAvsRespCode DataType="String">44</ProcAvsRespCode>
<ProcReturnCode DataType="String">1</ProcReturnCode>
<ProcReturnMsg DataType="String">Approved</ProcReturnMsg>
<Status DataType="String">1</Status>
</CardProcResp>
<CardholderPresentCode DataType="S32">7</CardholderPresentCode>
<CurrentTotals>
<Totals>
<Total DataType="Money" Currency="826">200</Total>
</Totals>
</CurrentTotals>
<Id DataType="String">445029fd-1a90-3000-0005-0003ba65c10f</Id>
<InputEnvironment DataType="S32">4</InputEnvironment>
<SecurityIndicator DataType="S32">7</SecurityIndicator>
<TerminalInputCapability DataType="S32">1</TerminalInputCapability>
<Type DataType="String">Auth</Type>
</Transaction>
</OrderFormDoc>
<User>
<Alias DataType="String"></Alias>
<ClientId DataType="S32">Your Client Id</ClientId>
<EffectiveAlias DataType="String"></EffectiveAlias>
<EffectiveClientId DataType="S32"></EffectiveClientId>
<Name DataType="String">Your Username</Name>
<Password DataType="String">Your Password</Password>
</User>
</EngineDoc>
<TimeIn DataType="DateTime">1146133410910</TimeIn>
<TimeOut DataType="DateTime">1146133411332</TimeOut>
</EngineDocList>
```

OrderFormDoc Output Document Field Definitions

All the fields returned in the XML Output Document are defined in the ClearCommerce documentation. For information, the most relevant fields are defined below. The critical field that can be used to determine whether a transaction has been successful or not, is the CcErrCode. If this field is not present in the response document then the transaction has not been processed successfully. In this case one or more explanatory error messages will typically be returned in the MessageList, and the MaxSev will have a value between 5 and 8.

Field	Description	DHR_OrderFormDoc page number
<MessageList>	This record contains any error messages generated as a result of a transaction attempt	52-65
<MaxSev DataType="S32"></MaxSev>	This field contains a number from 0-8 indicating the most serious error returned in the output document. 5 or greater indicates the transaction has failed	53
<Message>	A record containing the details of an individual error	54-65
<Sev DataType="S32"></Sev>	This field contains a number from 0-8 indicating the severity of an individual error. 5 or greater indicates the transaction has failed	64
<Text DataType="String"></Text>	Text explaining the error	65
<CardProcResponse>	This is the record that contains all the responses from the authorisation request	392-424
<AvsDisplay DataType="String">YY</AvsDisplay >	AVS response from the card issuer, e.g. YY=Yes the address matched and Yes the postcode matched	Payment Reference Guide, page 56-57
<AvsRespCode DataType="String">EX</AvsRespCode>	AVS response from the card issuer	Payment Reference Guide, page 56-57
<CcErrCode DataType="S32">1</CcErrCode>	This code indicates whether the transaction has been successful or not. If it is not present then the transaction has failed	API Guide, Appendix H
<Cvv2Resp DataType="String">1</Cvv2Resp>	This field indicates whether or not the Card Issuing bank matched the CVV2 number supplied by the cardholder	405

Report Documents

In addition to submitting transactions, the XML Input Documents can also be used to submit ReportDocs to perform various reporting functions against the ePDQ transaction database. A full explanation of this facility is given in the API Guide, Chapter 30.

Once the store is live and trading, valid live transactions will be settled (i.e. the totals uploaded and passed through to the merchant's bank account) each night. Each settlement batch is allocated a Settlement Id, and using Report Docs it is possible to retrieve any Settlement Id's for a given time period, then obtain a breakdown of orders that are settled within a given Settlement Id.

The following 2 example ReportDocs show, in the first instance, how to obtain a specific Settlement Id (i.e. the identity of the settlement batch you wish to interrogate), and then how to use that Settlement Id to obtain a breakdown of the individual orders which made up a particular settlement batch.

XML ReportDoc 1– Input Document Example

Used to obtain the Settlement Id for a particular days trading. All mandatory fields are in bold type

```
<EngineDocList>
  <DocVersion>1.0</DocVersion>
  <EngineDoc>
    <User>
      <Name>Your Username</Name>
      <Password>Your Password</Password>
      <ClientId DataType="S32">Your Client Id</ClientId>
    </User>
    <ContentType>ReportDoc</ContentType>
    <Instructions>
      <RoutingList>
        <Routing>
          <Name>CcxReports</Name>
        </Routing>
      </RoutingList>
    </Instructions>
    <ReportDoc>
      <CompList>
        <Comp>
          <Name>CcxReports</Name>
          <ReportActionList>
            <ReportAction>
              <ReportName>CCE_Settle_List_All</ReportName>
              <Start DataType="S32">1</Start>
              <Count DataType="S32">25</Count>
              <ValueList>
                <Value>
                  <ClientId DataType="S32">Your Client Id</ClientId>
                  <TransactionBeginDate
DataType="DateTime">1130803200000</TransactionBeginDate>
                  <TransactionEndDate
DataType="DateTime">1133395200000</TransactionEndDate>
                </Value>
              </ValueList>
            </ReportAction>
          </ReportActionList>
        </Comp>
      </CompList>
    </ReportDoc>
  </EngineDoc>
</EngineDocList>
```

ReportDoc 1 Input Document Field Definitions

As with the OrderFormDoc any data types other than "String" must be specified. The page reference numbers are for the Document Hierarchy Reference; ReportDoc, unless specified.

Field	Description	DHR_Report Doc page number
<Name></Name> <Password></Password> <ClientId DataType="S32"></ClientId>	The ePDQ username, password and store id.	46 47 43
<ReportName> CCE_Settle_List_All</ReportName>	This is where you define the name of the report that you wish to run.	56 API Guide, Chapter 30
<Start DataType="S32">1</Start>	The returned record number where the results should start.	57
<Count DataType="S32">25</Count>	Number of results to be returned. (ClearCommerce recommend a maximum of 500)	55
<ValueList><Value> <ClientId DataType="S32"></ClientId>	ePDQ store id	133
<TransactionBeginDate DataType="DateTime">1130803200000</ TransactionBeginDate>	Timestamp in milliseconds since 1 st January 1970.	321
<TransactionEndDate DataType="DateTime">1133395200000</ TransactionEndDate> </Value></ValueList>	Timestamp in milliseconds since 1 st January 1970.	323

XML ReportDoc 1 – Output Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<EngineDocList>
  <DocVersion DataType="String">1.0</DocVersion>
  <EngineDoc>
    <ContentType DataType="String">ReportDoc</ContentType>
    <DocumentId DataType="String">44a57865-e7a5-3000-0006-
0003ba65c10f</DocumentId>
    <Instructions>
      <RoutingList>
        <Routing>
          <Name DataType="String">CcxReports</Name>
        </Routing>
      </RoutingList>
    </Instructions>
    <MessageList>
    </MessageList>
    <ReportDoc>
      <CompList>
        <Comp>
          <Name DataType="String">CcxReports</Name>
          <ReportActionList>
            <ReportAction>
              <Count DataType="S32">25</Count>
              <ReportName DataType="String">CCE_Settle_List_All</ReportName>
              <Start DataType="S32">1</Start>
              <Total DataType="S32">2</Total>
              <ValueList>
                <Value>
                  <CreditApprovedCount DataType="S32">3</CreditApprovedCount>
                  <CreditDeclinedCount DataType="S32">0</CreditDeclinedCount>
                  <CreditTotal DataType="Money" Currency="826">300</CreditTotal>
                  <DepositApprovedCount DataType="S32">24</DepositApprovedCount>
                  <DepositDeclinedCount DataType="S32">0</DepositDeclinedCount>
                  <DepositTotal DataType="Money" Currency="826">99809</DepositTotal>
                  <SettlementDate DataType="DateTime">1090592493000</SettlementDate>
                  <SettlementDescription DataType="String"></SettlementDescription>
                  <SettlementId DataType="S32">1</SettlementId>
                  <SettlementProcessorName
DataType="String">BARCLAYS_GBP</SettlementProcessorName>
                  <SettlementStateCode DataType="String">A</SettlementStateCode>
                </Value>
              </ValueList>
            </ReportAction>
          </ReportActionList>
        </Comp>
      </CompList>
    </ReportDoc>
    <User>
      <Alias DataType="String"></Alias>
      <ClientId DataType="S32">Your Client Id</ClientId>
      <EffectiveAlias DataType="String"></EffectiveAlias>
      <EffectiveClientId DataType="S32"></EffectiveClientId>
      <Name DataType="String">Your Username</Name>
      <Password DataType="String">Your Password</Password>
    </User>
  </EngineDoc>
</EngineDocList>
```

XML ReportDoc 1 – Output Document Example (continued)

```

</User>
</EngineDoc>
<TimeIn DataType="DateTime">1151934940325</TimeIn>
<TimeOut DataType="DateTime">1151934940336</TimeOut>
</EngineDocList>

```

ReportDoc 1 Output Document Field Definitions

Field	Description	DHR_Report Doc page number
<CreditTotal DataType="Money" Currency="826">300</CreditTotal>	Total value of credits settled	152
<DepositTotal DataType="Money" Currency="826">99809</DepositTotal>	Total value of sales settled	160
<SettlementId DataType="S32">1</SettlementId>	Unique identifier for a day's settlement total	273
<SettlementProcessorName DataType="String">BARCLAYS_GBP</SettlementProcessorName>	One of: BARCLAYS_GBP (Sterling stores) BARCLAYS_MC (Multicurrency stores) AMEX_BRIGHTON (American Express transactions)	
<SettlementStateCode DataType="String">A</SettlementStateCode>	The status of settlement, e.g. "A" = Approved	277

XML ReportDoc 2 – Input Document Example

Used to obtain a list of transactions for a specific Settlement ID. All mandatory fields are in bold type

```
<EngineDocList>
  <DocVersion>1.0</DocVersion>
  <EngineDoc>
    <User>
      <Name>Your Username</Name>
      <Password>Your Password</Password>
      <ClientId DataType="S32">Your Client Id</ClientId>
    </User>
    <ContentType>ReportDoc</ContentType>
    <Instructions>
      <RoutingList>
        <Routing>
          <Name>CcxReports</Name>
        </Routing>
      </RoutingList>
    </Instructions>
    <ReportDoc>
      <CompList>
        <Comp>
          <Name>CcxReports</Name>
          <ReportActionList>
            <ReportAction>
              <ReportName>CCE_Settle_Detail_All</ReportName>
              <Start DataType="S32">1</Start>
              <Count DataType="S32">50</Count>
              <ValueList>
                <Value>
                  <ClientId DataType="S32">Your Client Id</ClientId>
                  <SettlementId DataType="S32">Your target Settlement Id</SettlementId>
                  <SettlementProcessorName>See definition list</SettlementProcessorName>
                </Value>
              </ValueList>
            </ReportAction>
          </ReportActionList>
        </Comp>
      </CompList>
    </ReportDoc>
  </EngineDoc>
</EngineDocList>
```

ReportDoc 2 Input Document Field Definitions

Field	Description	DHR_Report Doc page number
<Name></Name> <Password></Password> <ClientId DataType="S32">*****</ClientId>	The ePDQ username, password and store id.	46 47 43
<ReportName> CCE_Settle_Detail_All </ReportName>	This is where you define the name of the report that you wish to run.	56 API Guide, Chapter 30
<Start DataType="S32">1</Start>	The returned record number where the results should start.	57
<Count DataType="S32">50</Count>	Number of results to be returned. (ClearCommerce recommend a maximum of 500)	55
<SettlementId DataType="S32"></SettlementId>	The Settlement Id to be broken down.	273
<SettlementProcessorName> </SettlementProcessorName>	Where the settlement was processed. One of: BARCLAYS_GBP (Sterling stores) BARCLAYS_MC (Multicurrency stores) AMEX_BRIGHTON (American Express transactions)	

XML ReportDoc 2 – Output Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<EngineDocList>
  <DocVersion DataType="String">1.0</DocVersion>
  <EngineDoc>
    <ContentType DataType="String">ReportDoc</ContentType>
    <DocumentId DataType="String">44c02c6a-3476-3001-0001-
0003ba65f927</DocumentId>
    <Instructions>
      <RoutingList>
        <Routing>
          <Name DataType="String">CcxReports</Name>
        </Routing>
      </RoutingList>
    </Instructions>
    <MessageList>
    </MessageList>
    <ReportDoc>
      <CompList>
        <Comp>
          <Name DataType="String">CcxReports</Name>
          <ReportActionList>
            <ReportAction>
              <Count DataType="S32">50</Count>
              <ReportName DataType="String">CCE_Settle_Detail_All</ReportName>
              <Start DataType="S32">1</Start>
              <Total DataType="S32">2</Total>
              <ValueList>
                <Value>
                  <AuthorizationCode DataType="String">971280</AuthorizationCode>
                  <AuthorizedDate DataType="DateTime">1073309371000</AuthorizedDate>
                  <CapturedAmount DataType="Money"
Currency="826">100</CapturedAmount>
                  <CapturedDate DataType="DateTime">1073309391000</CapturedDate>
                  <CardExpiryDate DataType="DateTime">1138751999000</CardExpiryDate>
                  <CcProcessorErrorCode DataType="S32">0</CcProcessorErrorCode>
                  <ChargeTypeCode DataType="String">S</ChargeTypeCode>
                  <OrderId DataType="String">3ff8f966-217b-3000-0013-
0003ba297d8f</OrderId>
                  <Span DataType="String">41111111</Span>
                  <TransactionDate DataType="DateTime">1073309371000</TransactionDate>
                  <TransactionId DataType="String">3ff8f966-217c-3000-0013-
0003ba297d8f</TransactionId>
                  <TransactionStatus DataType="String">S</TransactionStatus>
                </Value>
                <Value>
                  <AuthorizationCode DataType="String"></AuthorizationCode>
                  <AuthorizedDate DataType="DateTime">1073309800000</AuthorizedDate>
                  <CapturedAmount DataType="Money"
Currency="826">100</CapturedAmount>
                  <CapturedDate DataType="DateTime">1073309800000</CapturedDate>
                  <CardExpiryDate DataType="DateTime">1138751999000</CardExpiryDate>
                  <CcProcessorErrorCode DataType="S32">0</CcProcessorErrorCode>
                </Value>
              </ValueList>
            </ReportAction>
          </ReportActionList>
        </Comp>
      </CompList>
    </ReportDoc>
  </EngineDoc>
</EngineDocList>
```

XML ReportDoc 2 – Output Document Example (continued)

```

    <ChargeTypeCode DataType="String">C</ChargeTypeCode>
    <OrderId DataType="String">3ff8f966-217b-3000-0013-
0003ba297d8f</OrderId>
    <Span DataType="String">41111111</Span>
    <TransactionDate
DataType="DateTime">107330980000</TransactionDate>
    <TransactionId DataType="String">3ff911fc-17e4-3000-0014-
0003ba297d8f</TransactionId>
    <TransactionStatus DataType="String">S</TransactionStatus>
    </Value>
    </ValueList>
    </ReportAction>
    </ReportActionList>
    </Comp>
    </CompList>
    </ReportDoc>
    <User>
    <Alias DataType="String"></Alias>
    <ClientId DataType="S32">Your Client Id</ClientId>
    <EffectiveAlias DataType="String"></EffectiveAlias>
    <EffectiveClientId DataType="S32"></EffectiveClientId>
    <Name DataType="String">Your Username</Name>
    <Password DataType="String">Your Password</Password>
    </User>
    </EngineDoc>
    <Timeln DataType="DateTime">1153492499604</Timeln>
    <TimeOut DataType="DateTime">1153492499757</TimeOut>
    </EngineDocList>

```

ReportDoc 2 Output Document Field Definitions

Field	Description	DHR_ReportDoc page number
<Total DataType="S32">2</Total>	The total number of orders returned by this report	58
<CapturedAmount DataType="Money" Currency="826">100</CapturedAmount>	The value of an order	111
<ChargeTypeCode DataType="String">S</ChargeTypeCode>	Either S for Sale or C for Credit	125
<OrderId DataType="String">3ff8f966-217b-3000-0013-0003ba297d8f</OrderId>	The Order Id	213
41111111	The first and last 4 digits of the card number.	289
<TransactionStatus DataType="String">S</TransactionStatus>	Status of the order, i.e. S = settled.	Payment Reference Guide, page 20

MPI Integration Checklist

There are many considerations to be made whilst integrating any payment solution. Below are included a few 'general' checks which we would recommend are performed prior to completely activating the site:

Store XML Input and Output documents

In order for the ePDQ support team to address any queries relating to responses received from ePDQ, the XML Input and Output documents will be required. We would recommend that these XML documents are stored in line with Visa and MasterCard's PCI DSS guidelines – typically removing card number, CVV2 value and password. In the event of any query relating to a response, please send the relevant XML documents to the ePDQ Support Team.

Ensure Payment Pipeline is correct

Typically the value for the Pipeline node should be set to 'Payment', ensuring the transactions are routed via the suite of Fraud Rules available in the ePDQ Store Administration Interface. If this is set to 'PaymentNoFraud' then the transaction request will bypass all Fraud Rules.

Ensure Mode is set to appropriate value

The correct value for the <Mode> node for live transactions is 'P'. It is recommended that this is checked once the ePDQ Store has been activated to ensure actual live transactions are not processed in a test mode (Y or N).

Transactions processed in any node other than 'P' will not be settled.

If using Internet Authentication, ensure relevant fields are included

For any transaction where Internet Authentication (Verified by Visa or SecureCode) has been performed, please ensure the relevant values are included in your XML Document for the authorisation request. These should include:

```
<PayerSecurityLevel DataType="S32">Authentication status
code</PayerSecurityLevel>
<PayerTxnId>The Authentication Transaction ID, if available</PayerTxnId>
<PayerAuthenticationCode>The UCAF or CAVV value</PayerAuthenticationCode>
<CardholderPresentCode DataType="S32">This must be set to 13 for an
Authentication enabled transaction</CardholderPresentCode>
```

Failure to include these values correctly may lead to loss of liability shift.

Ensure settlement is taking place

This can be checked using the ReportDoc Settlement report mentioned earlier in the documentation, or via the ePDQ Store Administration Interface.

Common Error Responses

Here are some common error messages that may be returned in the output document, and the likely causes:

1) "Insufficient permissions to perform requested operation."

There are a number of circumstances that could generate this error, so you should check the following:

- Your ClientId, Username and Password are correct
- Your username has the necessary permissions to perform the operation you are attempting to perform
- You are submitting transactions to the correct server domain and port number
- All tags are correctly spelt and closed (all tags are case-sensitive).
- All tags are located in the correct place within the document structure
- All tags are correctly populated, e.g. <Type>Auth</Type>, not <Type>auth</Type>

2) "The argument 'currency' in context 'CcaMoney01::operator>=' had value '826'. ('condition' is '!=')"

This error means that you have submitted the incorrect currency code for the ClientId you are using. The currency code that appears in the error message is the code that you should be using in your input document, in the line:

```
<Total DataType="Money" Currency="826">200</Total>
```

3) No error message, but the input is returned unprocessed in the output document

You need to include the Pipeline within the <EngineDoc> record:

```
<Instructions>  
<Pipeline>Payment</Pipeline>  
</Instructions>
```

4) "Missing required key from EngineDoc.User."

This message means that there is an error with one of the mandatory fields in the <User> record; check that all mandatory tags are present, correctly formed and correctly populated.

5) "Routing information is missing or not valid for ClientId (Your ClientId)"

This message doesn't indicate any error with the input document, it occurs when a transaction is attempted using mode "P" or "T", but the store has not been activated.

- 6) **"Rule 27 with expression {AVSDisplay} = "NY" | {AVSDisplay} = "NN" could not be evaluated for client '(Your Client Id)': Key 'AVSDisplay' was not found in the document. The action to take on missing field is '0'."**

This message (or similar) doesn't indicate that the order has failed, but is returned when a fraud rule that has been activated on the store can't be evaluated. In this case the fraud rule requires data to be returned from the card issuer (e.g. CV2 or AVS data), but the order has been processed in a simulation mode (e.g. mode "Y") that doesn't contact the card issuer, therefore this data isn't returned.

- 7) **"Component 'CcxBarclaysGbpAuthTest' specified is not valid"
"Component 'CcxPymtDbUpdater' specified is not valid."
"Component 'CcxFraudShield' specified is not valid"
"Component 'CcxDigitalReceipt' specified is not valid."
"Component 'CcxPeriodicBilling' specified is not valid."
"Component 'CcxPostProcessor' specified is not valid."**

These errors occur if you submit transactions to a live store using mode "T" (see explanation for input field "<Mode>" on page 9 for a list of acceptable values for this field).

If you have checked all of the above and the error is persisting please submit the full XML input and output documents (minus your password and any live card numbers, CVV2 values etc.) to the ePDQ Support Team for further assistance.

Frequently Asked Questions

Q. What is the MPI in simple terms?

A. The ePDQ MPI (Merchant Payment Interface) is a solution for processing internet payments through a server external to the secure Barclaycard ePDQ environment.

Typically this solution will be integrated on a secure server supplied by the merchant in an environment which adheres to Visa and MasterCard's Payment Card Industry Data Security Standards (PCI DSS), and also operates in line with any other acts or standards covering the acceptance of credit card and customer information across the internet. Defining and supplying this security would be the responsibility of the merchant. Further information on PCI DSS can be obtained from Visa and MasterCard's general websites.

The MPI is generally used in situations where the look or functionality of the CPI is unsuitable, or when the merchant requires a solution to be integrated into an offline database – for example, airline ticket allocation software – to generate automated transactions. The MPI is also used in situations where the merchant needs to store the customer's credit card number.

The integration of the MPI solution requires extensive web programming and web security knowledge to implement successfully. Responsibility for the security of all data will rest with the merchant.

Included with the MPI are the necessary C++ libraries/Java classes, full integration documentation covering all available functions on the ePDQ system, and an overview of the XML version. The web pages collecting the card data, and the application for creating the transaction requests must be provided by the merchant or their developer.

All updates to the MPI integration will need to be performed by the merchant, including the addition of Internet Authentication and any alterations driven by modifications to the ePDQ product itself. A Software Developers Kit (SDK) exists for the purpose of adding the Verified by Visa and MasterCard SecureCode authentication systems. This SDK can be obtained by applying for Internet Authentication – please contact the ePDQ Customer Support Team using the details supplied in the 'Contact' section.

Please note, the MPI is not the standard solution for ePDQ integration. In order to utilise the MPI solution the merchant will need to complete an Internet Security Proforma, available from the ePDQ Customer Support Team.

- Q.** Do I need to install any ePDQ specific software on my server?
- A.** No. The ClearCommerce ePDQ engine is hosted by Barclaycard in a secure environment. The integration of the MPI involves you communicating with this application, posting XML Input via HTTPS, and parsing the XML response.
- Q.** How long will integration take?
- A.** This depends entirely on the complexity of the integration. Typically, the creation of an XML wrapping and parsing script should not take a great deal of time. The duration of integration will be affected by how this XML process interacts with other processes required by the online web-store/merchant application.
- Q.** How do I secure my website?
- A.** Internet security for MPI integration is entirely the responsibility of the merchant and the company hosting the web site. Please refer to Visa and MasterCard's PCI DSS (Payment Card Industry Data Security Standards) for further information – details available on their respective websites.
- Q.** Where should the XML Input documents be sent?
- A.** Please contact the ePDQ Support Team, or refer to the initial set up email sent to you when your ePDQ account was created.
- Q.** What can I use the MPI for?
- A.** The MPI can be used for processing authorisation requests across the internet, plus it also allows you to replicate any Configuration or Reporting function available through the ePDQ Store Administration Interface. For example, you can perform Settlement Reports using the XML Input Component.
- Q.** Do I need to read all the documentation?
- A.** It is advisable to read as much of the documentation as necessary to ensure a complete understanding of the ePDQ MPI facility, although particular attention should be given to the sections of the documentation relevant to your integration. For example, if you have no intention of using the Reporting function on ePDQ, there is no need to review the ReportDoc section of the API guide.
- Q.** How do I integrate Internet Authentication into the ePDQ MPI solution?
- A.** Barclaycard have provided a Software Developers Kit (SDK) for integration of the Internet Security facility. This SDK is typically integrated at the same time as the MPI, providing parameters necessary for the authorisation requests (ECI, CAVV etc.), although it can just as easily be integrated after the MPI.

Q. Can I use the Barclaycard logo on my payment page?

A. The Barclaycard logo must not be present on the page collecting the credit card information, as you cannot imply that Barclaycard have sanctioned the security of your website or transaction processes.

Q. Where can I obtain the Card Scheme Logos?

A. The logos for Visa and MasterCard etc. can all be downloaded from the Barclaycard website (<http://www.barclaycard.co.uk/business/existing-customers/support-materials/>)

Q. Where do I obtain the username and password?

A. The owner of the ePDQ store is responsible for creating a user account for the purposes of ePDQ integration and you will need to contact them for these details. If you are the owner of the ePDQ store you should have received you username via email and to get you password you will need to contact us.

Q. What user level do I need for the integration?

A. This user account should have permissions appropriate to the functionality of the MPI integration –for example, ReportDoc request types require ePDQ Level 3 access. Further information can be found in the ePDQ User Guide.

Changes and Amendments

Please find below details of any forthcoming changes to the ePDQ MPI service, with which you should familiarise yourself in order to fully prepare for any required amendments to new or existing ePDQ MPI integrations:

31st March, 2011:

From Thursday 31st March, 2011 the MPI response document structure will be amended to ensure that the ePDQ MPI service continues to be a safe, PCI-DSS compliant solution for our customers.

The amendments to the response (or output) document structure are as follows:

- Replacement of the '<Number>' element with the '<CardSpan>' element.
- Removal of the '<Cvv2Indicator>' element.
- Removal of the '<Cvv2Val>' element
- Removal of the '<Expires>' element.
- Removal of the '<IssueNum>' element.

This change will not affect the document structure of your payment submissions. Only the response data returned by the ePDQ MPI service will be affected, as detailed above.

For existing ePDQ MPI customers, to ensure you are ready for these changes, please validate your integration's handling of the response data from ePDQ MPI, focusing in particular on whether there is any system or programmatic reliance on the changing response/output-data detailed above.

Most, if not all, customers using the ePDQ MPI service will remain unaffected. However, please ensure that you pass this information on to the relevant technical resource within your business so that they may assess if there will be any impact on your ePDQ MPI integration.