CAPITA

Payment Management

Secure Card Portal
Simple Interface Specification

For Portal Version 9.2.x
September 2015

Capita
Software Services, Medtia Court, 70-76 Brunswick Street, Oldham OL1 1BT
Tel 08701 627800   Fax 08701 627801   Email cssenquiries@capita.co.uk
www.capita.co.uk/software

A trading name of Capita Business Services Limited

Document History

Version No.	Date of New Version	Schema Version	Amended by	Reason for Change
2.00	02/04/2014	9.0.0.0	T Dhorajiwala	Following schema changes have been made: - Added *systemCode* element to *additionalInstructions* element. - Added *walletRequest* element to *additionalInstructions* element. - Added *expiryDate* element to *storedCardDetails* element. - Added *cardMnemonic* element to *authDetails* element.
2.10	21/07/2014	9.1.0.0	T Dhorajiwala	Following schema changes have been made: - Added enumeration value of "NONE" to *cardType*. - Added optional attribute *acceptNonCardResponseData* to *scpQueryRequest* element. - Modified *paymentBase* element to allow conditional response of the *authDetails* element or *nonCardPayment* element. - Added *nonCardPayment* element. - Added enumeration value of "NONE" to element *cardDescription*.
2.11	13/02/2015	9.1.0.1	T Dhorajiwala	Following schema changes have been made: - Added *authoriseAndStore* and *authoriseAndAutoStore* enumeration values to the *requestType* element
2.12	26/06/2015	9.2.0.0	T Dhorajiwala	Following schema changes have been made: - Added *recurringPayments* element to the *additionalInstructions* element. - Added *nonPaymentData* element to the *scpInvokeRequest* element

Contents

Purpose and Scope

The purpose of this document is to describe the B2B interfaces for interaction between third-party integrators and the Secure Card Portal (SCP).

External developers should use this document, in conjunction with the SCP WSDL, to write programs that interface to the SCP.

SCP Overview

Requirements

The SCP interface is designed to meet the following requirements:

- Use an industry standard web-service interface.
- Provide a minimal request format for simple payment requests, but allow this to be extended with further information (e.g. line items, delivery details) if required.
- Support Local Authority specific data (e.g. fund code), but do not require it to be provided. This will ensure that existing Portal customers can switch to the SCP without loss of functionality.

Request Formats

This document describes the **simple interface** request format to the SCP.

Simple Interface

The simple interface can be used by customers who wish to process payments through the SCP where the payments are processed using their own merchant account. This interface is typically used by a third-party that has setup a merchant account with a bank.

Example Request

The following example shows a minimal payment request:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<scpSimpleInvokeRequest xmlns="http://www.capita-software-services.com/scp/simple">
     <credentials xmlns="https://support.capita-software.co.uk/selfservice/?commonFoundation">
          <subject>
               <subjectType>CapitaPortal</subjectType>
               <identifier>5</identifier>
                   <systemCode>SCP</systemCode>
          </subject>
          <requestIdentification>
               <uniqueReference>1234</uniqueReference>
               <timeStamp>20110101123445</timeStamp>
          </requestIdentification>
          <signature>
               <algorithm>Original</algorithm>
               <hmacKeyID>1</hmacKeyID>
               <digest>...</digest>
          </signature>
     </credentials>
     <requestType xmlns="http://www.capita-software-services.com/scp/base">payOnly</requestType>
     <requestId xmlns="http://www.capita-software-services.com/scp/base">SR-123456</requestId>
     <routing xmlns="http://www.capita-software-services.com/scp/base">
          <returnUrl>http://mycompany.com/payments/complete.aspx</returnUrl>
          <backUrl>http://mycompany.com/payments/dataentry.aspx</backUrl>
          <siteId>999</siteId>
          <scpId>12</scpId>
     </routing>
     <panEntryMethod xmlns="http://www.capita-software-services.com/scp/base">ECOM</panEntryMethod>
     <sale xmlns="http://www.capita-software-services.com/scp/simple">
          <saleSummary xmlns="http://www.capita-software-services.com/scp/base">
               <description>Goods</description>
                   <amountInMinorUnits>12345</amountInMinorUnits>
          </saleSummary>
     </sale>
</scpSimpleInvokeRequest>
```

The parts of the above request are:

- A <credentials> element. This is used to confirm that the client is authorised to call the SCP.
- A <requestType> element, which identifies the type of request being made.
- A client reference for the request.
- A <routing> element, which identifies the client sending the request and how the SCP is to return a response.
- An indication of whether the transaction was initiated directly by the cardholder (ECOM) or via a third party such as a call centre (CNP).
- Details of what is being purchased. As a minimum, this must include a description and an amount.

Request Types

The different request types; identified by the <*requestType*> element in the schema, can be divided into **authorise**, **authorise and payment capture** and **store**. These request types are detailed below.

Authorise

Request Type	Description
authoriseOnly	This request type is intended for use in the following situations: • The customer is taking payment for goods which cannot be immediately dispatched • The customer system needs to perform further processing once it has been confirmed that the cardholder is legitimate and has sufficient funds, but before accepting payment The SCP performs anti-fraud checks, confirms that the cardholder has sufficient funds to complete the payment and reserves these funds against the cardholder's account. However, the SCP does not initiate the transfer of funds to the customer's account. Instead it returns a key which the client application can subsequently pass to the Capita Common Payments web service (ref [1]) to initiate this transfer (e.g. once goods have been dispatched). Note that the key is valid only for a limited time. If it is used after that time the card issuer may refuse to honour the payment.
authoriseAndStore	An extension of the request type above. The SCP first authorises the payment then, with the user's consent, stores the card details for future use.
authoriseAndAutoStore	As per authorise*AndStore*, but does not prompt the user to confirm whether the card details are to be stored. (It is assumed that the client application has already obtained the user's consent in this case).

Authorise and Payment Capture

Request Type	Description
payOnly	This request type is intended for use in the typical payment scenario. The SCP performs various checks to combat fraudulent card use, confirms that the cardholder has sufficient funds to complete the payment, and then initiates the transfer of funds from the cardholder's account to the customer's account. (This transfer typically takes about three days to complete).
payAndStore	A combination of the two request types above. The SCP first processes the payment then, with the user's consent, stores the card

	details for future use.
payAndAutoStore	As for *payAndStore*, but does not prompt the user to confirm whether the card details are to be stored. (It is assumed that the client application has already obtained the user's consent in this case).

Store

Request Type	Description
storeOnly	This request type allows a client application to store a cardholder's details for future use. (For example, to give users the option of making future payments at a site without entering card details in full every time). The SCP performs anti-fraud checks as above, captures and saves the card details, and returns a key. The customer's systems can subsequently take payments using this card by including the returned key in requests to the SCP or Capita Common Payments web service (ref [1]).

SCP Configurations

A single instance of the SCP application can support multiple customer configurations. A configuration controls:

- How the SCP pages will be displayed. (Fonts, help and other text, etc.)
- Business rules to apply when processing requests. (Which card types are accepted, whether to apply surcharges, etc.)

A particular configuration is uniquely identified by a 'Site Id' and 'SCP Id'. These are numbers that are allocated by Capita. A Site Id typically identifies a particular customer (local authority, business, etc.) Each customer is allocated one or more SCP Ids, so a customer can potentially have more than one SCP configuration. This gives the customer the flexibility to 'brand' the SCP pages according to which part of their organisation the end user is paying, for example.

Terminology

There are various potentially ambiguous terms relating to the SCP. Throughout the remainder of the document the following terminology will be used:

Client/Client Application	A software system (usually a web application), operated by or on behalf of the customer, which calls the SCP web services.
Customer	An organization that is using the SCP to take card payments for goods/services.
Customer Configuration	A particular customer-specific SCP configuration, identified by Site Id and SCP Id (see above).
End User	Someone using the SCP to make a payment – either the cardholder or a third party such as a telesales operative.
Request	Data provided by the client to the SCP when calling a web service method.
Response	Data returned by the SCP in response to a web service method.
SCP	The SCP application.
Transaction	An end-to-end interaction between the client system, SCP and end user. From the point of view of the SCP, a transaction starts when a client makes an *scpInvoke* call and typically ends after the client makes a subsequent *scpQuery* call. (On the client side there is likely to be additional processing before and after this).

Monetary Values

All monetary values that appear in the schemas are expressed in 'minor units' of currency – pence for Sterling transactions, cents for Euro transactions, etc.

E.g. £123.45 would be encoded in an SCP request as 12345 (12,345 pence).

Conventions

The specification for each interface is set out in a table. Each table:

- Lists and describes the interface element.
- Provides a field definition for each element.
- An element name containing [Extended] refers to a complex type that should be extended.
- Specifies whether an element is mandatory (M), optional (O) or conditional (C).
- Where applicable, the XML schema type of the element is identified as per the XML schema specification.
- Where applicable, the length of the element is identified either as a schema restriction or as a business rule restriction.

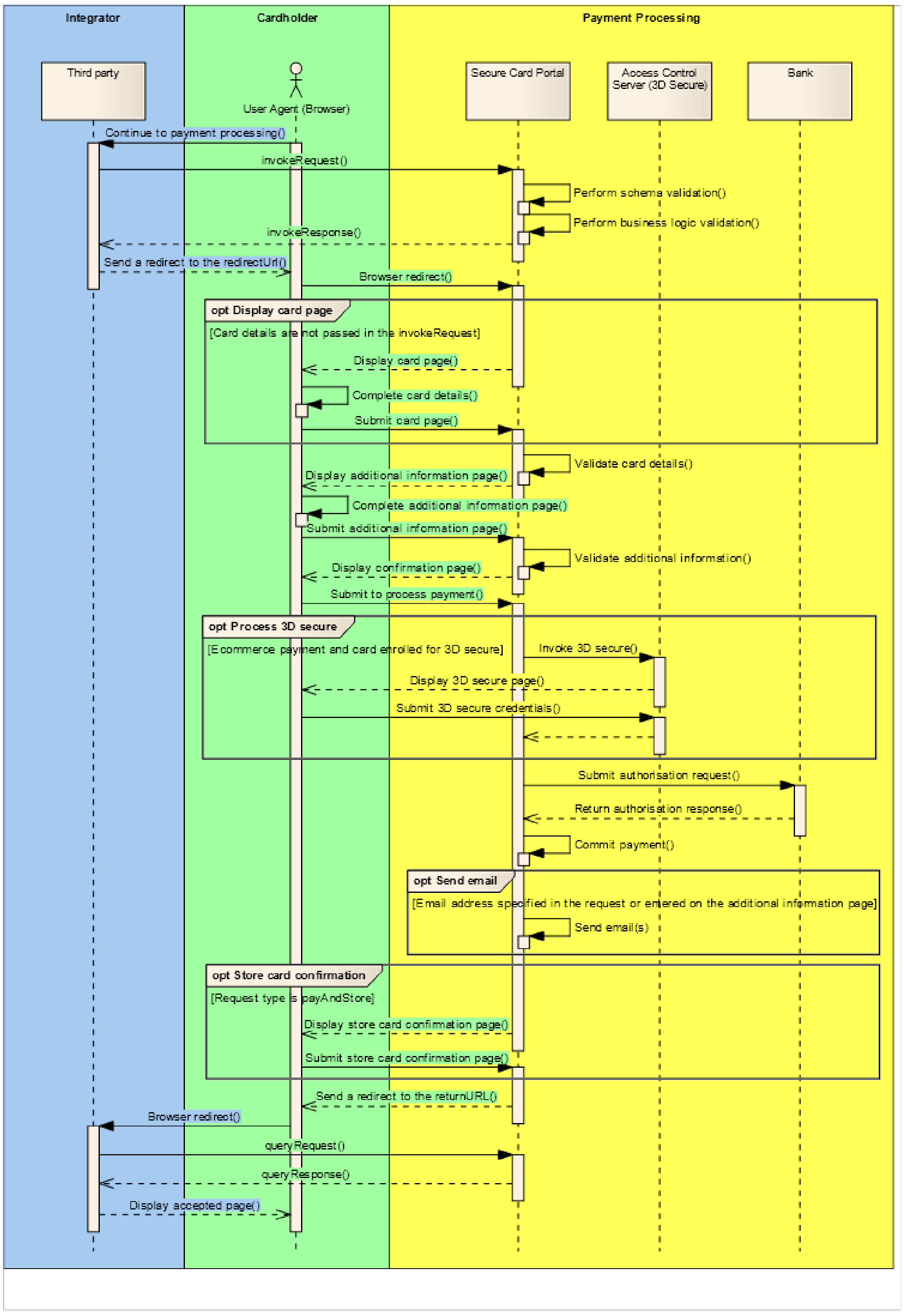
Client-SCP Communication Protocol

General

The interface uses a combination of B2B web service calls and browser redirects to allow data and control of the browser to be passed from the client application to the SCP and back again. The typical flow of a transaction is as follows:

1. The client makes an “invoke” web service call, providing details of a payment and/or store card transaction which the client requires the SCP to process.
2. The SCP validates the request, starts a new transaction, and returns a response containing two items of data:
    - A unique identifier for the transaction.
    - A URL to which the user’s browser should now be redirected. (This URL points to a location within the SCP web application).
3. The client redirects the user’s browser.
4. On receiving the redirect, the SCP:
    - Retrieves the transaction (which is identified by a key within the redirect URL).
    - Processes the transaction, displaying pages on the user’s browser and prompting for input as necessary.
    - Stores the end result of the processing (success, card declined, etc).
    - Redirects the browser back to the client application. (The redirect URL is specified by the <returnUrl> element in the original request).
5. The client makes a “query” web service call to the SCP, passing in the request identifier that was returned in step 2 above.
6. The SCP retrieves the transaction status and returns it.

An illustration of the typical flow is shown in the sequence diagram below.


Integrator
Cardholder
Payment Processing
Third party
User Agent (Browser)
Secure Card Portal
Access Control Server (3D Secure)
Bank
Continue to payment processing()
invokeRequest()
Perform schema validation()
Perform business logic validation()
invokeResponse()
Send a redirect to the redirectUrl()
Browser redirect()
opt Display card page
[Card details are not passed in the invokeRequest]
Display card page()
Complete card details()
Submit card page()
Validate card details()
Display additional information page()
Complete additional information page()
Submit additional information page()
Validate additional information()
Display confirmation page()
Submit to process payment()
opt Process 3D secure
[Ecommerce payment and card enrolled for 3D secure]
Invoke 3D secure()
Display 3D secure page()
Submit 3D secure credentials()
Submit authorisation request()
Return authorisation response()
Commit payment()
opt Send email
[Email address specified in the request or entered on the additional information page]
Send email(s)
opt Store card confirmation
[Request type is payAndStore]
Display store card confirmation page()
Submit store card confirmation page()
Send a redirect to the returnURL()
Browser redirect()
queryRequest()
queryResponse()
Display accepted page()

The data that flows across the interface is defined in terms of XML schemas. A development team may use tools, such as Visual Studio, that read the SCP WSDL and automatically convert these schemas into constructs in their chosen development language. In this case it is up to the developers to understand how the schema -> code mapping works.

As explained above, interaction with the SCP requires the use of web service methods. The following table lists these methods and the schema elements that define the requests and responses.

Simple Invoke

Namespace	http://www.capita-software-services.com/scp/simple
Method Name[1]	scpSimpleInvoke
Request Schema Element	scpSimpleInvokeRequest *below*
Response Schema Element	scpSimpleInvokeResponse *below*
Asks the SCP to begin the process of taking a card payment and/or storing card details.	

Simple Query

Namespace	http://www.capita-software-services.com/scp/simple
Method Name	scpSimpleQuery
Request Schema Element	scpSimpleQueryRequest *below*
Response Schema Element	scpSimpleQueryResponse *below*
Queries the SCP for the status of a previously initiated transaction.	

Common Methods

Namespace	http://www.capita-software-services.com/scp/base
Method Name	scpVersion
Request Schema Element	scpVersionRequest *below*
Response Schema Element	scpVersionResponse *below*
Queries the SCP for the current software and schema versions.	

Request Validation

The SCP validates a client request in two stages. Initially the request is validated against the XML schema, and then it is checked to ensure that it does not violate any business rules.

[1] This is the method name that is likely to be generated when a code generation tool is run against the WSDL. It is possible that some tools might generate different method names, however.

A failure of schema validation causes a SOAP fault to be returned to the client. However, because the business rules checking takes place at a later stage, errors discovered there are normally reported in the returned <scpInvokeResponse>.

In terms of a typical Java or .NET WSDL binding, this means that failure of an SCP method call may be reported in one of two different ways:

1. The method call throws an exception.
2. The method call completes normally, but the returned object indicates that a problem was identified by the SCP while processing it.

It is the application's responsibility to check for and handle both cases.

Security/Authentication

All web service and browser interaction with the SCP is over https[2]. This ensures that data is sent to and from the SCP securely and that the client can determine that it is communicating with a trusted server.

The client authenticates itself to the server by means of the <credentials> element in the SCP request. (See *Credentials Element* for details).

Timeouts

There are 3 different timeouts settings implemented within the SCP as described below.

Session Timeout

The session timeout indicates the period of inactivity between a user and the SCP.

The session timeout is set to 600 seconds (10 minutes).

Note: Any user activity inside 3D secure will not be regarded as activity within the SCP as this communicates directly between the user and the banks access control server.

Initial Transaction State Timeout

The initial transaction state timeout indicates the period of time the initial transaction state will be valid. The initial transaction state is set when a new request is sent by the third party. The timeout is the period between the *scpInvokeResponse* being passed back to the third party and the third party sending a redirect to the SCP for that request.

The initial transaction state timeout is set to 10 seconds.

After this period, the transaction state will be removed and any further requests will return a transaction state of “INVALID_REFERENCE”.

Transaction State Timeout

The transaction state timeout indicates the period of time the transaction state will be available. During this period the third party may query the transaction state using the *scpQueryRequest*.

The transaction state timeout is set to 1800 seconds (30 minutes).

After this period, the transaction state will be removed and any further requests will return a transaction state of “INVALID_REFERENCE”.

[2] In internal Capita-only test environments http may be used.

Schemas

The schemas can be obtained via your account manager or project manager.

Schema name	Namespace / Description
scpSimple.xsd	http://www.capita-software-services.com/scp/simple
	This schema provides the entry points for making simple requests.
scpBaseTypes.xsd	http://www.capita-software-services.com/scp/base
	This schema provides definitions that are used by the simple schema
commonPaymentTypes.xsd	http://www.capita-software-services.com/portal-api
	This schema provides definitions for various XML data types that are used across Capita Payment Management systems.
commonFoundation.xsd	https://support.capita-software.co.uk/selfservice/?commonFoundation
	This schema provides definitions for the <credentials> element.

ScpSimpleInvokeRequest

A client's initial request to the SCP invoke interface contains details of the items being purchased and/or the card being stored, together with information that controls how the transaction is processed by the SCP.

The *scpSimpleInvokeRequest* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/simple			
Element Name	Description	M/O/C	Type	Length
[Extended]	See *ScpInvokeRequest.*	M		
Sale	See *SimpleSale Type*. Note: This element must not be present in requests which have a requestType of *storeOnly*. (If it is, the SCP will reject the request).	C		

ScpInvokeRequest

The *scpInvokeRequest* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
credentials	See *Credentials Element*.	M		
requestType	See *Request Types*.	M	xs:enum	

requestId	Customer supplied request id. (We recommend that this is unique, but SCP processing does not rely on its uniqueness).	M	xs:token	
routing	See *Routing Element*.	M		
panEntryMethod	Must be set to one of the following: • ECOM – Indicates that the cardholder will be interacting directly with the web pages displayed by the SCP. • CNP – Indicates that the transaction is being processed by a third party (e.g. a telesales operative) on behalf of the cardholder. N.B. In order to comply with bank rules and to ensure that 3-D Secure authentication is processed correctly it is important to set this element correctly.	M	xs:enum	
additionalInstructions	See *AdditionalInstructions Element*. Additional data which may affect how the SCP processes the request, either in terms of the UI displayed or business logic executed.	O		
billing	See *BillingDetails Element*. Card and cardholder details.	O		
nonPaymentData	See *NonPaymentData* Element Additional data to send the MCC6012 data to the bank.	O		

SimpleSale Type

The *simpleSale* type defines the items that are being purchased. It has the following child elements.

Namespace	http://www.capita-software-services.com/scp/simple			
Element Name	Description	M/O/C	Type	Length
[Extended]	See *SaleBase Type*.	M		
postageAndPacking	See NonPaymentData Element This element currently only contains a single child element to capture the MCC 6012 information. This section only applies to transactions that: • Involve a merchant with a MCC 6012 category code. • Use VISA • Process a UK domestic payment. If any of the above three criteria do not apply, ignore the information in this topic. The *nonPaymentData* element has the following child elements.	O		

Namespace		http://www.capita-software-services.com/scp/base
Element Name		Description
mcc6012		
	dateOfBirth	The date of birth for the primary recipient in the format yyyy
	surname	The surname/last name/family name of the primary recipien
	accountNumber	The account number or card number of the primary recipien

		For an account number: • Up to 10 alphanumeric characters. (The interface ac only the first 10 alphanumerics are used) For a card number: • The first 6 and last 4 digits of the actual card numb ‘4444333322221111’ then set this element to ‘4444		
	postcode	A valid UK postcode for the primary recipient.		xs:string

	PostageAndPacking Element.			
surchargeable	See *Surchargeable Element*.	O		
items	Note: This element must not be present in requests which have a requestType of *authoriseOnly*. (If it is, the SCP will reject the request).	C		
item	See *SimpleItem* Element. One or more item elements can be supplied.	M		

SaleBase Type

The *saleBase* type has the following child elements:

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
saleSummary		M		

	description	Brief description of what is being purchased. Note: If individual line items are specified, then the description of the line item will replace this description. In this case, supply a generic value description.	M	xs:string	1-100
	amountInMinorUnits	Total transaction amount, including postage and packing and VAT/sales tax, but excluding any card surcharge. Note: If individual line items are specified (via an *items items* below) then this must be equal to the sum of the line item amounts plus the postage and packing amount (if any).	M	xs:int	
	reference	Customer reference for the transaction. Note: If individual line items are specified, then the reference of the line item will replace this reference. In this case, supply a generic value reference.	O	xs:string	1-50
	displayableReference	Used to display a different/differently formatted reference to the end user. Note: If individual line items are specified, then the displayableReference of the line item will replace this displayableReference. In this case, supply a generic value displayableReference.	O	xs:string	1-50
deliveryDetails		Describes where and how items are to be delivered.	O		
	name	See *ThreePartName Element*.	O		
	address	See *Address Element*.	O		
	contact	See *Contact Element*.	O		
lgSaleDetails		Provides additional information which is likely to be of use chiefly to Local Government organisations.	O		

	areaCode	Area code	O	xs:string	1-5
	locationCode	Location code	O	xs:string	1-5
	sourceCode	Source code	O	xs:string	1-5
	userName	User name – identifies the user making the payment. Typically populated when panEntryMethod is CNP.	O	xs:string	1-25
	userCode	User code – identifies the user making the payment. Typically populated when panEntryMethod is CNP.	O	xs:string	1-5

Routing Element

The *routing* element defines how the request is to be routed.

It has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
returnUrl	A URL within the client application to which the user's browser is to be redirected once SCP processing is complete. (Note: if a client application can have many calls to the SCP outstanding at once then this URL may need to include an identifier which the client application can use to resume processing of the appropriate session once control is returned by the SCP). The special URL 'scp:close' may be used in place of a normal URL. In this case at the end of a transaction the browser is redirected to a page which attempts to close itself (using JavaScript) and displays a message asking the end user to close the window if	M	xs:token	

	this fails. (This is intended chiefly to support desktop applications which create a browser window to display the SCP pages). Must begin with *http://* or *https://.*			
backUrl	A URL within the client application to which user's browser is to be redirected if the user selects 'Back' on the first page of the SCP application. If this element is omitted then the 'Back' button is not displayed. Must begin with *http://* or *https://.*	O	xs:token	
errorUrl	Reserved for ***future use***.	O	xs:token	
siteId	An integer id allocated by Capita Payment Management.	M	xs:int	
scpId	An integer id allocated by Capita Payment Management. As explained above, *siteId* and *scpId* are used by the SCP to look up customer-specific configuration options when processing a request.	M	xs:int	

AdditionalInstructions Element

This element provides additional information to the SCP that affects the way in which the request is processed. The *additionalInstructions* element itself, and all of its child elements are optional. Where necessary, defaults will be provided when setting up the customer's SCP configuration.

The *additionalInstructions* has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
merchantCode	A code which effectively maps to a customer's bank account. (The mapping between	O	xs:string	1-5

	merchant codes and bank accounts is configured by Capita Software Services).			
countryCode	An ISO 3166-1 numeric code for the country in which the payment is originating or *000* for non UK. The value is also used to perform postcode validation of the cardholders billing address. Currently only supports *826* (UK) and *000* (non-UK). Defaults to *826* if not provided.	O	xs:string	3
currencyCode	An ISO 4217 numeric code for the currency to be used. Currently only supports *826* (GBP) and *978* (EUR). Defaults to *826* if not provided. Support for other currencies will require prior arrangement with Capita Software Services.	O	xs:string	3
acceptedCards	See *AcceptedCards Element*. Identifies the card types that can be used for this transaction.	O		
language	A 2-character code for the language in which the SCP UI should be displayed. Currently supported language codes are *en* (English), *cy* (Welsh), *pl* (Polish). Defaults to *en* if not provided. Support for other languages will require prior arrangement with Capita Software Services.	O	xs:string	2
stageIndicator	Indicates that a ‘progress indicator’ is to be shown on the portal UI pages. (The way in which the indicator is actually displayed is specified in the customer configuration).	O		
firstPortalStage	The number of the ‘processing stage’ at which the SCP is first entered. This allows the SCP progress indicator to follow on from a progress indicator in the client application.	M	xs:byte	1
totalStages	The total number of processing stages that the user must complete.	M	xs:byte	1

responseInterface	Reserved for ***future use*** – must be omitted.	O		
cardholderID	Used when storing a card as part of a storeOnly, payAndStore or payAndAutoStore requestType. This should be the user name that the card holder has used to sign on to the client system, it will be used to ensure that cards are not duplicated within the set of cards stored by a particular card holder.	O	xs:string	1-50
integrator	An integer id allocated by Capita Payment Management.	O	xs:int	
styleCode	If present, the SCP will perform a lookup on the code and include any CSS files to the customer framework. Configuration is required by Capita Software Services.	O	xs:string	1-5
frameworkCode	If present, the SCP will perform a lookup on the code and display the SCP form inside the identified framework. Configuration is required by Capita Software Services.	O	xs:string	1-5
systemCode	Reserved for **internal use only** – must be omitted.	O		
walletRequest	Reserved for **internal use only** – must be omitted.	O		
recurringPayments		O		
paymentAuthorityType	Must be set to one of the following: • notArrangedPayment – The storage of the card is for the purpose of making subsequent ad-hoc payments. • initialRecurringPayment – The storage of the card is for the purpose of making subsequent recurring payments.	O	xs:enum	

BillingDetails Element

This element is used to provide details of the card and/or cardholder. It has a single optional attribute named 'editable', which indicates whether or not the end user is able to edit this data within the SCP application before authorising the card[3]. Default is 'true'.

The *billingDetails* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
card	These will specify either full card details (card number, expiry date, etc.) or a reference to a previously stored card. If it is omitted then the end user is prompted for card details.	O		
cardDetails		C		
cardNumber	Full valid card number.	M	xs:string	12-19
expiryDate	Expiry date in MMYY format (e.g. 0925).	M	xs:string	4
startDate	Start date in MMYY format. If it is omitted and bank rules indicate that a start is required when authorising cards of this type then authorisation may fail.	O	xs:string	4
issueNumber	Issue number – either a one or two digit string (note that '01' and '1' are different issue numbers). If it is omitted and bank rules indicate that an issue number is required when authorising cards of this type then authorisation may fail.	O	xs:string	2
cardSecurityCode	CSC code – three or four digits depending on card type. If it is omitted and CSC	O	xs:string	3-4

[3] In the current SCP implementation this attribute is ignored. Instead, if card details are supplied then the card page is not displayed, which effectively means that the card data is not editable. However, all of the data under <cardHolderDetails> *can* be edited (on the Additional Information page).

	checking is enabled in the customer configuration then authorisation may fail.			
storedCardKey	See *StoredCardKey Element*. Data identifying a stored card (typically returned by a previous call to the SCP).	C		
paymentGroupCode	A code which can be used when storing a card to identify a group of customers who may subsequently take payments using this card. If it is omitted then only the customer storing the card can take payments with it.	O	xs:string	1-5
cardHolderDetails	Cardholder name, address and contact information.	O		
cardHolderName	The name as it appears on the card. This value is displayed on the SCP UI. The user may modify the passed in values.	O	xs:string	1-50
address	See *Address Element*. The address is displayed on the SCP UI and will be used to perform AVS address and postcode validation with the bank. The user may modify the passed in values.	O		
contact	See *Contact Element*. The email address in the contact element will be displayed on the SCP UI. The user may modify the passed in values.	O		

NonPaymentData Element

This element currently only contains a single child element to capture the MCC 6012 information. This section only applies to transactions that:

- Involve a merchant with a MCC 6012 category code.

- Use VISA
- Process a UK domestic payment.

If any of the above three criteria do not apply, ignore the information in this topic.

The *nonPaymentData* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
mcc6012				
dateOfBirth	The date of birth for the primary recipient in the format yyyy-MM-dd.	O	xs:date	
surname	The surname/last name/family name of the primary recipient.	O	xs:string	1-50
accountNumber	The account number or card number of the primary recipient. For an account number: • Up to 10 alphanumeric characters. (The interface accepts up to 50 characters but only the first 10 alphanumerics are used) For a card number: • The first 6 and last 4 digits of the actual card number. (E.g. if the card number is ‘4444333322221111’ then set this element to ‘4444331111’).	O	xs:string	1-50
postcode	A valid UK postcode for the primary recipient.	O	xs:string	1-10

PostageAndPacking Element

A *postageAndPacking* element has the following child elements.

Namespace		http://www.capita-software-services.com/scp/base			
Element Name		Description	M/O/C	Type	Length
pnpSummary		A summary of the postage and packing	M		
	description	Brief description of the postage and packing.	M	xs:string	1-100
	amountInMinorUnits	Postage and packing amount to be applied.	M	xs:int	
	reference	Postage and packing reference.	O	xs:string	1-50
	displayableReference	Used to display a different/differently formatted reference to the end user.	O	xs:string	1-50
tax		See *TaxItem Element*.	O		
lgPnpDetails		Provides additional information which is likely to be of use chiefly to Local Government organisations.	O		
	fundCode	Postage and packing fund code.	O	xs:string	1-5
	pnpCode	Postage and packing customer code.	O	xs:string	1-5
	pnpOptionCode	A customer code for the level of service required.	O	xs:string	1-5
	pnpOptionDescription	Descriptive text for pnpOptionCode.	O	xs:string	1-50

Surchargeable Element

The *surchargeable* element contains a single child element from the following 2 options.

Namespace	http://www.capita-software-services.com/scp/simple			
Element Name	Description	M/O/C	Type	Length
applyScpConfig	An empty element identifies the sale as surchargeable	C	xs:anyType	
surcharge	See *SurchargeItemDetails Element*. Specifies advanced surcharge options that can be applied to the transaction.	C		

SimpleItem Element

A *simpleItem* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/simple			
Element Name	Description	M/O/C	Type	Length
[Extended]	See *ItemBase Type*.	M		
surchargeable	Boolean value to define whether the item is surchargeable or not.	O	xs:boolean	

ItemBase Type

An *itemBase* type has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
itemSummary	Brief details regarding the item.	M		
description	Description of item.	M	xs:string	1-100
amountInMinorUnits	Amount including any sales tax.	M	xs:int	
reference	Customer reference for the item.	O	xs:string	1-50
displayableReference	Can be used to display a different/differently formatted reference to the end user if required. E.g. reference could be 12345678 and displayable reference 12-345-678.	O	xs:string	1-50
tax	See *<u>TaxItem Element</u>*.	O		
quantity	Quantity of item being purchased. The value must be between 1 and 32767. Note: There is not calculation performed on the quantity and amount. Default is 0.	O	xs:short	1-32767
notificationEmails		O		
email	List of up to 5 email addresses to which an email is to be sent with details of this purchase. (This is intended chiefly to allow people within the customer's organisation to be notified – e.g. so that they can start an order fulfilment process).	M	xs:string	1-255
additionalEmailMessage	Message to be included on the 'notification emails'	O	xs:string	1-2048

lgItemDetails	Provides additional information which is likely to be of use chiefly to Local Government organisations.	O		
fundCode	Fund code for the item.	O	xs:string	1-5
isFundItem	Indicates whether this item is a fund item (e.g. council tax) or a miscellaneous item (e.g. purchase of bin bags).	O	xs:boolean	
additionalReference	A secondary reference for the item.	O	xs:string	1-50
narrative	Narrative text.	O	xs:string	1-50
accountName	See *ThreePartName Element*.	O		
accountAddress	See *Address Element*.	O		
contact	See *Contact Element*.	O		
customerInfo	Provides additional customer information.	O		
customerString1	String value	O	xs:string	1-50
customerString2	String value	O	xs:string	1-50
customerString3	String value	O	xs:string	1-50
customerString4	String value	O	xs:string	1-50
customerString5	String value	O	xs:string	1-50
customerNumber1	Integer value	O	xs:int	

	customerNumber2	Integer value	O	xs:int	
	customerNumber3	Integer value	O	xs:int	
	customerNumber4	Integer value	O	xs:int	
lineId		Used to identify the line item. ***For response mapping only***	M	xs:token	1-50

ThreePartName Element

A *threePartName* element is used to provide name details. It has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
surname	Surname	M	xs:string	1-50
title	Title	O	xs:string	1-50
forename	Forename	O	xs:string	1-50

Address Element

An *address* element is used to provide address details. It has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base

Element Name	Description	M/O/C	Type	Length
address1	Address 1. This is typically the first line of the address.	O	xs:string	1-50
address2	Address 2. This is optionally the second line of the address.	O	xs:string	1-50
address3	Address 3. This is typically the town/city.	O	xs:string	1-50
address4	Address 4	O	xs:string	1-50
county	County	O	xs:string	1-50
country	Country	O	xs:string	1-50
postcode	Postcode	O	xs:string	1-10

Contact Element

A contact element is used to provide contact details. It has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
telephone	Telephone contact number. Must be prefixed with “+”. ***For future use***	O	xs:string	1-16
mobile	Mobile contact number. ***For future use***	O	xs:string	1-16

email	Email address.	O	xs:string	1-255

AcceptedCards Element

The *acceptedCards* element (below *additionalInstructions*) identifies the card types that will be accepted for a transaction. If supplied, this element overrides the default customer configuration of accepted cards. (This may only restrict the default list of accepted cards for a customer).

The acceptedCards element has the following child elements.

Namespace		http://www.capita-software-services.com/scp/base			
Element Name		Description	M/O/C	Type	Length
includes		List of cards to be accepted	O		
	card	See *AcceptedCardType Element*. 1 or more instances of the card element can be specified to be accepted. An empty card element below an includes element matches any card type that is supported by the SCP.	M		
excludes		List of cards to be rejected	O		
	card	See *AcceptedCardType Element*. 1 or more instances of the card element can be specified to be rejected. Note: a card which matches both an includes element and an excludes element is rejected. An empty card element below an excludes element has no effect. (i.e. no additional card types are excluded)	M		

AcceptedCardType Element

The acceptedCardTypes element identifies a single card type that will be accepted/rejected for a transaction.

Each acceptedCardType element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
cardDescription	See *Card Description* for a list of valid codes. A description that identifies a particular group of cards (e.g. VISA). If this is omitted then any cards in any groups are matched.	O	xs:enum	
cardType	See *Card Type* for a list of valid types. If omitted then any card type is matched.	O	xs:enum	

TaxItem Element

A *taxItem* element has the following child elements.

Namespace		http://www.capita-software-services.com/scp/base			
Element Name		Description	M/O/C	Type	Length
vat			O		
	vatCode	A customer-defined vat code.	O	xs:string	1-5
	vatRate	A customer-defined percentage rate. For example for a 20% rate to be applied the value	M	xs:decimal	2, 4

	should be 20.0			
vatAmountInMinorUnits	Amount in pence of the VAT applied to the item.	M	xs:int	

SurchargeItemDetails Element

A *surchargeItemDetails* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
fundCode	Surcharge fund code.	O	xs:string	1-5
reference	Surcharge reference.	O	xs:string	1-50
surchargeInfo	See ***SurchargeInfo Element***.	M		
tax	See ***TaxSurcharge Type***.	O		

SurchargeInfo Element

The *surchargeInfo* element identifies the surcharge that will be applied for a transaction. Below each of these are one or more *cardSurchargeRate* elements, each of which identifies one or more surcharge data.

A surchargeInfo element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length

cardSurchargeRate			Surcharge fund code.	O	xs:string	1-5
	surchargeCardType		See *AcceptedCardType Element*. Specifies the card mnemonic and or card type that apply to this surcharge element.	O		
	surchargeBasis		Specifies whether to apply a fixed or percentage surcharge or both.	M		
		surchargeFixed	A fixed surcharge amount to apply. Integer number of 'minor units' of currency (e.g. pence). Must be greater than zero.	C	xs:int	
		surchargeRate	A percentage rate to apply. E.g. for a 2.5% rate to be applied the value should be 2.5. Must be greater than zero.	C	xs:decimal	2, 4

Note: if the surchargeBasis element is supplied, then at least one of surchargeFixed or surchargeRate must be provided.

TaxSurcharge Type

A *taxSurcharge* type has the following child elements.

Namespace		http://www.capita-software-services.com/scp/base			
Element Name		Description	M/O/C	Type	Length
vat			O		
	vatCode	A customer-defined vat code.	O	xs:string	1-5
	vatRate	A customer-defined percentage rate. For example for a 20% rate to be applied the value	M	xs:decimal	2, 4

		should be 20.0			

ScpSimpleInvokeResponse

A simple invoke response contains status of processing the invoke request.

The *scpSimpleInvokeRequest* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/simple			
Element Name	Description	M/O/C	Type	Length
[Extended]	See *ScpInvokeRequest*.	M		

ScpInvokeResponse

The response to an *scpSimpleInvoke* method call is defined by the *scpInvokeResponse* schema element. This has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
requestId	Matches the values sent in the original invoke request.	O	xs:token	
scpReference	A unique transaction reference generated by the SCP. This should be used in a subsequent *scpQuery* request to retrieve the status of the transaction. (It is also included, as an aid to problem investigation, in SCP logs which relate to the transaction).	M	xs:string	1-50
transactionState	See *Transaction State* for a list of valid transaction states. Only the following values from the list are valid for the scpInvoke response.	M	xs:enum	

		• IN_PROGRESS • COMPLETE			
invokeResult			M		
	status	See *Status* for a list of valid status codes. Only the following values from the list are valid for the scpInvoke response. • SUCCESS • INVALID_REQUEST • ERROR	M	xs:enum	
	redirectUrl	The URL to which the end user's browser should now be redirected. Only present if *status* is SUCCESS.	C	xs:token	
	errorDetails	See *ErrorDetails Element*. Provides further information if an error occurred. Only present if *status* is not SUCCESS.	C		

ScpSimpleQueryRequest

After calling *scpInvoke* to initiate SCP processing of a transaction, the client may query the status of the transaction by calling the *scpQuery* method. Normally this should be done when the SCP redirects the user's browser back to the customer's web site, but the client may call *scpQuery* at any time.

The *scpSimpleQueryRequest* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/simple			
Element Name	Description	M/O/C	Type	Length
[Extended]	See *ScpQueryRequest*.	M		

ScpQueryRequest

A scpQueryRequest element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
credentials	See *Credentials Element*.	M		
siteId	Must be the site id that was provided in the scpInvoke request.	M	xs:string	1-5
scpReference	A unique transaction reference generated by the SCP and returned in the scpInvoke response.	M	xs:string	1-50

A scpQueryRequest has a single optional attribute named ' acceptNonCardResponseData'. This indicates whether or not to return the ‘nonCardPayment’ element in the response instead of the ‘authDetails’ element. If this attribute is supplied with the value ‘true’ AND the transaction has been made using a non card payment details (e.g. Barclays Pingit), then the ‘nonCardPayment’ element will be returned. Otherwise the ‘authDetails’ element will be returned.

ScpSimpleQueryResponse

The scpQuery response contains details of the result of processing a transaction.

The *scpSimpleQueryResponse* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/simple			
Element Name	Description	M/O/C	Type	Length
[Extended]	See *ScpQueryResponse*.	M		
paymentResult	Result of processing a payment.	C		
status	See *Status* for a list of valid status codes. Only the following values from the list are valid for the scpQuery response. • SUCCESS • ERROR • CARD_DETAILS_REJECTED • CANCELLED • LOGGED_OUT • NOT_ATTEMPTED	M		
paymentDetails	See *A transaction* may involve making a payment, storing card details, or both, which leads to various possible scenarios.	C		

		For example, in a *payAndStore* transaction, the payment might be successful but storing the card could fail. To ensure that the client can identify and handle cases like this, *scpQueryResponse* contains elements that report the results of the 'pay' and 'store card' operations separately. Either of these may have a NOT_ATTEMPTED status, indicating that the associated operation has not taken place. This could be because: • The transaction type does not require this operation to take place. E.g. for a *storeOnly* transaction, *paymentResult* will have a NOT_ATTEMPTED status. • A previous operation failed. E.g. if during a *payAndStore* transaction the payment fails, then the SCP will not attempt to store the card and so *storeCardResult* will have a NOT_ATTEMPTED status. Additionally, it's possible that the storing of the card is successful, but the email receipt could not be sent. To ensure that the client can identify this case, the results of the 'pay' , 'store card' and 'email' would be reported separately. An error will only be reported if sending of the email has failed. No indication of a successful email will be returned since it cannot be confirmed if an email has been received by the recipient. PaymentDetails Element. Only present if *status* is SUCCESS.			
	errorDetails	See *ErrorDetails Element*. Provides further information if an error occurred. Only present if *status* is not SUCCESS.	C		

ScpQueryResponse

The response to an *scpQuery* method call is defined by the *scpQueryResponse* schema element. This has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base

Element Name	Description	M/O/C	Type	Length
requestId	Matches the values sent in the original invoke request.	O	xs:token	
scpReference	A unique transaction reference generated by the SCP. This should be used in a subsequent *scpQuery* request to retrieve the status of the transaction. (It is also included, as an aid to problem investigation, in SCP logs which relate to the transaction).	M	xs:string	1-50
transactionState	See *Transaction State* for a list of valid transaction states. Only the following values from the list are valid for the scpQuery response. • IN_PROGRESS • COMPLETE • INVALID_REFERENCE Note: If transactionState is *not* COMPLETE then only the mandatory elements *scpReference* and *transactionState* will be present in the response.	M	xs:enum	
storeCardResult	Result of storing a card.	C		
status	See *Status* for a list of valid status codes. Only the following values from the list are valid for the scpQuery response. • SUCCESS • ERROR • NOT_ATTEMPTED	M		
storedCardDetails	Data identifying a stored card.	C		

storedCardKey	See *StoredCardKey Element*.	M		
cardDescription	See *Card Description* for a list of valid codes.	M	xs:enum	
cardType	See *Card Type* for a list of valid types.	M	xs:enum	
expiryDate	Expiry date in MMYY format (e.g. 0925).	O	xs:string	4
errorDetails	See *ErrorDetails Element*. Provides further information if an error occurred. Only present if *status* is not SUCCESS.	C		
emailResult	Only provided if an error occurs whilst sending an email.	C		
status	See *Status* for a list of valid status codes. Only an ERROR status can be returned indicating that sending of the email failed.	M		
errorDetails	See *ErrorDetails Element*. Provides further information if an error occurred.	M		

A transaction may involve making a payment, storing card details, or both, which leads to various possible scenarios.

For example, in a *payAndStore* transaction, the payment might be successful but storing the card could fail. To ensure that the client can identify and handle cases like this, *scpQueryResponse* contains elements that report the results of the 'pay' and 'store card' operations separately. Either of these may have a NOT_ATTEMPTED status, indicating that the associated operation has not taken place. This could be because:

- The transaction type does not require this operation to take place. E.g. for a *storeOnly* transaction, *paymentResult* will have a NOT_ATTEMPTED status.
- A previous operation failed. E.g. if during a *payAndStore* transaction the payment fails, then the SCP will not attempt to store the card and so *storeCardResult* will have a NOT_ATTEMPTED status.

Additionally, it’s possible that the storing of the card is successful, but the email receipt could not be sent. To ensure that the client can identify this case, the results of the ‘pay’ , ‘store card’ and ‘email’ would be reported separately. An error will only be reported if sending of the email has failed. No indication of a successful email will be returned since it cannot be confirmed if an email has been received by the recipient.

PaymentDetails Element

A paymentDetails element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/simple			
Element Name	Description	M/O/C	Type	Length
[Extended]	See *PaymentBase Element*.	M		
saleSummary		M		
items		O		
itemSummary	See *ItemSummaryBase Element*. One or more summary of the items paid.	M		
surchargeDetails	See *SurchargeDetails Element*.	O		

PaymentBase Element

A *paymentBase* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length

paymentHeader		M		
transactionDate	The GMT date of the transaction in the standard XML format. E.g. 2014-04-17T12:14:16	M	xs:dateTime	
machineCode	The machine code set for the customer	M	xs:string	1-5
uniqueTranId	A 12-character reference assigned to the transaction. This will be passed to the bank during settlement and may appear on card statements.	M	xs:string	12
authDetails		C		
authCode	As returned by the acquiring bank.	M	xs:string	1-50
amountInMinorUnits	Total amount authorised, including any card surcharge applied.	M	xs:int	
maskedCardNumber	Card number with all but first six and last four digits masked with asterisks. E.g. 444433******1111	M	xs:string	12-19
cardDescription	See *Card Description* for a list of valid codes. This, together with the cardType element, identifies the type of card used for the payment.	M	xs:enum	
cardType	See *Card Type* for a list of valid types.	M	xs:enum	
merchantNumber	Merchant number against which the payment was made.	M	xs:string	1-50
expiryDate	Expiry date in MMYY format (e.g. 0925).	M	xs:string	4
continuousAuditNumber	A sequential audit number generated by the SCP for the payment record.	O	xs:int	1-5

		Note: This is not returned for *authoriseXXX* transactions.			
	cardMnemonic	As returned in the end of day file.	M	xs:string	4
nonCardPayment			C		
	amountInMinorUnits	Total amount authorised.	M	xs:int	
	continuousAuditNumber	A sequential audit number generated by the SCP for the payment record. Note: This is not returned for *authoriseXXX* transactions.	O	xs:int	1-5
	paymentType	Currently returns the fixed value “PINGIT”	M	xs:string	6
	paymentProviderReference	A 12-character reference assigned to the transaction. This will be passed to Barclays Pingit.	M	xs:string	12

ItemSummaryBase Element

An *itemSummaryBase* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
lineId	Used to identify the line item.	M	xs:token	1-50
continuousAuditNumber	A sequential audit number generated by the SCP for the item record.	M	xs:int	1-5

SurchargeDetails Element

An *surchargeDetails* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
fundCode	Surcharge fund code.	M	xs:string	1-5
reference	Surcharge reference.	M	xs:string	1-50
amountInMinorUnits	Surcharge amount applied to the transaction	M	xs:int	
surchargeBasis		M		
surchargeFixed	A fixed surcharge amount applied to the transaction. In 'minor units' of currency (e.g. pence).	C	xs:int	
surchargeRate	A percentage rate applied to the transaction. E.g. for a 2.5% rate to be applied the value will be 2.5.	C	xs:decimal	2, 4
continuousAuditNumber	A sequential audit number generated by the SCP for the surcharge record.	M	xs:int	1-5
vatAmountInMinorUnits	VAT amount for the surcharge applied to the transaction.	M	xs:int	

Note: the surchargeBasis element can return either or both child elements depending on the surcharge applied.

StoredCardKey Element

A storedCardKey element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
token	A token identifying the stored card. This token does not include the card number or any other card details in either plain or encrypted form.	M	xs:string	24
lastFourDigits	The last four digits of the card number.	M	xs:string	4

ErrorDetails Element

An *errorDetails* element has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
errorId	Useful to diagnose the error within the SCP system.	O	xs:string	1-50
errorMessage	Explanatory message of the error.	O	xs:string	1-2048

Card Description

The following is a list of the card descriptions which are recognised by the SCP:

Namespace	http://www.capita-software-services.com/portal-api
Enum Value	Description
AMERICAN EXPRESS	American Express card
VISA	Visa card
MASTERCARD	MasterCard
LASER	Laser card
DINERS	Diners Club card
JCBC	JCBC card
NONE	Non card payment

Card Type

The following is a list of card types which are recognised by the SCP.

Namespace	http://www.capita-software-services.com/scp/base
Enum Value	Description
DEBIT	Debit card
CREDIT	Credit card
NONE	No card type

Transaction State

The following is a list of transactions states returned by the SCP.

Namespace	http://www.capita-software-services.com/scp/base
Enum Value	Description
IN_PROGRESS	Indicates that the SCP has started a transaction but the transaction

	is not yet complete. This is the normal case.
COMPLETE	Indicates that the SCP has finished processing the transaction. At present this can happen only if an error occurred, because if the *scpInvoke* call completes normally the transaction is always put in the IN_PROGRESS state. However, future releases of the SCP may include new types of transaction which can complete immediately without requiring end user input, so this is not a reliable way of checking for errors. (Use the *status* element under *invokeResult* instead).
INVALID_REFERENCE	Indicates that the SCP does not recognize the supplied scpReference. This can happen either because the reference is wrong (i.e. was never returned by the SCP) or because the referenced transaction completed some time ago and has now been deleted.

Status

The following is a list of status codes returned by the SCP.

Namespace	http://www.capita-software-services.com/scp/base
Enum Value	Description
SUCCESS	Payment has been authorised successfully / Card has been stored successfully.
INVALID_REQUEST	Returned if the SCP found that the contents of the *scpInvoke request* were not valid. This probably indicates an issue that needs to be resolved by the client.
CARD_DETAILS_REJECTED	Bank has rejected the card details and processing has returned back to the customer system.
CANCELLED	User selected ‘Cancel’.
LOGGED_OUT	User selected ‘Logout’.
NOT_ATTEMPTED	Payment was not attempted / Store card was not attempted.
ERROR	Returned if the SCP encountered an error while processing the *scpInvoke* request. This probably indicates an issue that needs to be resolved by Capita.

A *status* element can appear in an *scpInvokeResponse* (under the *invokeResult* element) and in an *scpQueryResponse* (under the *paymentResult* and *storeCardResult* elements). However, not all of the enumeration values are applicable in all cases. The following table shows which status values can appear in which contexts:

Status	invokeResult	paymentResult	storeCardResult	emailResult
SUCCESS	✓	✓	✓	✘
INVALID_REQUEST	✓	✘	✘	✘
CARD_DETAILS_REJECTED	✘	✓	✘	✘
CANCELLED	✘	✓	✘	✘
LOGGED_OUT	✘	✓	✘	✘
NOT_ATTEMPTED	✘	✓	✓	✘
ERROR	✓	✓	✓	✓

Transaction State vs. Status Matrix

The following matrix shows the available *transactionState* and *status* combinations.

	Transaction State			
Status		IN_PROGRESS	COMPLETE	INVALID_REFERENCE
	No *status* response element	Payment processing is still in progress	N/A	Invalid scpReference sent in the request
	SUCCESS (paymentResult)	N/A	Payment has been authorised successfully	N/A
	SUCCESS (storeCardResult)	N/A	Card has been stored successfully	N/A
	CARD_DETAILS_REJECTED	N/A	Bank has rejected the card details and processing has returned back to the customer system	N/A
	CANCELLED	N/A	User selected 'Cancel'	N/A
	LOGGED_OUT	N/A	User selected 'Logout'	N/A
	NOT_ATTEMPTED (paymentResult)	N/A	Payment was not attempted (for a *storeOnly* transaction)	N/A
	NOT_ATTEMPTED (storeCardResult)	N/A	Store card was not attempted. This will always be returned for a *payOnly* or *authoriseOnly* transaction. It will be returned for other *payXXX* and *authoriseXXX* transactions if the authorisation/payment fails.	N/A
	ERROR (paymentResult)	N/A	Failed to process the payment. Issue needs to be resolved by Capita.	N/A

	ERROR (storeCardResult)	N/A	Failed to store the card. Issue needs to be resolved by Capita.	N/A
	ERROR (emailResult)	N/A	Failed to send the email(s). Issue needs to be resolved by Capita.	N/A

ScpVersionRequest

A client can query the SCP for the current software and schema versions by sending an *ScpVersionRequest* to the SCP. The scpVersionRequest element has the following child elements. The *subject identifier* in this element may be set to any SCP Id for which the client has an HMAC key.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
credentials	See *Credentials Element*.	M		

ScpVersionResponse

This is the response to an *scpVersionRequest*. It has the following child elements.

Namespace	http://www.capita-software-services.com/scp/base			
Element Name	Description	M/O/C	Type	Length
version	The SCP software version.	M	xs:string	
schemaVersion	The current schema version (i.e. the value of the 'version' attribute in the schema's root element).	M	xs:string	

Credentials Element

This element is used by the SCP to check that the client is authorised to call the SCP web service. It has the following child elements. (Note: all literal elements – e.g. 'CapitaPortal' – are case-sensitive).

Namespace	https://support.capita-software.co.uk/selfservice/?commonFoundation			
Element Name	Description	M/O/C	Type	Length
subject	Identifies the client making the request.	M		
subjectType	Must be set to 'CapitaPortal'.	M	xs:string	12
identifier	Must be the same as the *scpId* specified under the *routing* element. (When this element appears below an *scpQueryRequest* type then the *scpId* from the original *scpInvokeRequest* must be used).	M	xs:int	
systemCode	Must be set to 'SCP'.	M	xs:string	3
requestIdentification	Uniquely identifies this SCP request.	M		
uniqueReference	A 'unique' reference for the request. Formally the uniqueness requirement is that if a client makes a SOAP call to the SCP, specifying *scpId* **S** and unique reference **U,** then no client may make another SOAP call with *scpId* **S** and unique reference **U** for at least 24 hours. Note that the uniqueness requirement is per SOAP call, not per transaction. I.e. you can't use the same unique reference for an *scpInvoke* call and the subsequent *scpQuery* call.	M	xs:string	

	timestamp	Current UTC (Coordinated Universal Time) date and time to the nearest second, in the following format: `YYYYMMDDHHMMSS`	M	xs:string	14
signature		Confirms, by means of a secret key, that this request comes from a trusted client.	M		
	algorithm	Must be set to 'Original'.	M	xs:string	8
	hmacKeyID	Provided by Capita Software Services.	M	xs:int	
	digest	Calculated using the key provided by Capita Software Services.	M		

Digest Generation

The <digest> element contains a keyed hash of the other data under the <credentials> element. The secret key will be provided by Capita Software Services.

The digest is calculated as follows:

1. Concatenate *subjectType*, *identifier*, *uniqueReference*, *timestamp, algorithm* and *hmacKeyID* into a single string, with a '!' inserted between each concatenated value. E.g. the result might be:

   ```
   CapitaPortal!37!X326736B!20110203201814!Original!2
   ```

2. Convert this string to a sequence of bytes, using UTF-8 encoding.
3. Generate an HMAC/SHA256 keyed hash of the resulting byte sequence, using the Base-64 decoded value of the key supplied by Capita Software Services.
4. Base-64 encode the resultant hash.

There will be a separate key for each SCP instance (identified by *scpId*). Therefore a client that interfaces with more than one SCP instance must support multiple keys. A client must also provide facilities to change the *hmacKeyID* and key for a particular scpId if requested by Capita Software Services.

Code samples

The following code samples may be helpful in generating the digest. In all cases, assume the following.

- secretKey = secret key provided by Capita Software Services.
- credentialsToHash = concatenated string of credentials element values as described above.

Java

Timestamp generation

```
    private String formatTimestamp()
    {
         final Calendar cal = Calendar.getInstance();
         cal.clear();
         cal.setTimeZone(TimeZone.getTimeZone("GMT+0"));
         cal.setTime(currentDate.getDate());
         final String result =
              String.format("%04d%02d%02d%02d%02d%02d", cal.get(YEAR), cal.get(MONTH) + 1,
                        cal.get(DAY_OF_MONTH), cal.get(HOUR_OF_DAY), cal.get(MINUTE), cal.get(SECOND));
         return result;
    }
```

Digest generation

```
    private String calculateDigest(final String secretKey, final String credentialsToHash)
    {
           try
           {
                  final byte[] keyBytes = Base64.decode(secretKey);
                  final SecretKey key = new SecretKeySpec(keyBytes, "HmacSHA256");
                  final Mac hmac = Mac.getInstance(key.getAlgorithm());
                  hmac.init(key);
                  final byte[] bytesToHash = credentialsToHash.getBytes("UTF8");
                  final byte[] hash = hmac.doFinal(bytesToHash);
                  return Base64.encodeBytes(hash);
           }
```

```
            catch (final Exception e)
            {
                  throw new Exception(e);
            }
      }
```

PHP

Timestamp generation

```
     gmdate("YmdHis");
```

Digest generation

```
     $key = base64_decode($secretKey);

     $hash = hash_hmac('sha256', $credentialsToHash, $key, true);

     $digest = base64_encode($hash);
```

C#

Timestamp generation

```
        private static string GenerateTimestamp()
        {
            DateTime now = DateTime.UtcNow;
            return string.Format("{0:d4}{1:d2}{2:d2}{3:d2}{4:d2}{5:d2}",
                now.Year,
                now.Month,
                now.Day,
                now.Hour,
                now.Minute,
                now.Second);
        }
```

Digest generation

```
        private static string CalculateDigest(string secretkey, string credentialsToHash)
        {
            byte[] keyBytes = Convert.FromBase64String(secretkey);
            byte[] bytesToHash = (new UTF8Encoding()).GetBytes(credentialsToHash);
            HMACSHA256 hmac = new HMACSHA256(keyBytes);
            byte[] hash = hmac.ComputeHash(bytesToHash);
            return Convert.ToBase64String(hash);
        }
```

Limitations

Please note that the following functionality which was present in the v7 has not yet been implemented in the v8 Payment Portal. Introduction of this functionality may depend on demand. If you are planning a project to migrate from a v7 to a v8 Payment Portal please ensure that all required functionality is present.

Interfaces

The v8 Payment Portal only supports the new web-service interface and the various form post interfaces from v7 are not available

Email Receipts

Receipts are currently only generated as HTML and the plain text receipt option is not available.

PCI Logo

The flag to show the PCI logo (Trustwave) is set at an application level rather than UI based configuration

Switch Refs Functionality

The flag to switch the reference and additionalReference values before passing to the API is not present and references will flow through the system as passed or entered. The Switch Refs option in v7 is used by Planning Portal and ELMS portals.

The Portal no longer requires this functionality as the same result can be achieved by passing the references in the correct fields (as required in the EOD) and passing the other reference in the **displayableReference** element in the request.

Editable Amounts

The ability to edit payment amount in the Portal is not available.

Min / Max Amounts

The ability to define the minimum and maximum amounts acceptable by card type is not available.

Max Declines

The functionality to define in the request the number of attempts to pay (or declines) that a user can make is not fully available.

The number of declines allowed can be configured, but the additional response information is not fully implemented.

Reference Description

The referenceDescription element, often used to pass the ledger description, is not available.

References

The latest WSDLs can be found on the following links:

Simple WSDL

https://sbsctest.e-paycapita.com:443/scp/scpws/scpSimpleClient.wsdl

Please contact your project manager or account manager if you require copies of the following documents:

[1] AXIS Common Payments Interface Specification – Portal & Two-Part Payments