FUJ00001378

Document Title:	**Generalised API for OPS/TMS: Appendix A SmartMan Interfaces**
Document Type:	Technical Design Standard
Release:	N/A
Abstract:	This appendix describes the functionality of the Smart Card Manager (SmartMan) developed by ICL Pathway and the interfaces that have been developed within the SmartMan environment to support the use of various types of smart card. The main document provides the information required to plan the development of new applications and describes in more detail the architecture set out in the OPS Architecture Specification. Both documents are supplied under the terms of the Codified Agreement to POCL to facilitate the procurement of applications to run on the Service Infrastructure (interfacing with OPS and TMS). **This document is only available to organisations outside ICL Pathway through formal Non-Disclosure Agreement.**
Document Status:	APPROVED
Author & Dept:	Jon Cruise, Technical Design Authority
Contributors:	Janet Dore, Adrian Goodwin, Tony Hayward
Reviewed By:	ICL Pathway: Terry Austin, Dave Cooke, John Dicks, Adrian Goodwin, Dai Jones, Peter Wiles POCL: Bob Booth
Comments By:	
Comments To:	Document Controller & Authors
Distribution:	ICL Pathway Library and Reviewers

FUJ00001378

0.0 Document control

0.1 Document history

Version No.	Date	Reason for Issue	Associated CP/PinICL No.
0.10	21/1/2000	First version of this appendix.	
0.11	8/2/2000	Second version	
0.12	15/2/2000	Comments from earlier versions incorporated.	
0.13	13/3/2000	Comments from earlier versions incorporated.	
0.14	17/3/2000	Comments from earlier versions incorporated.	
0.15	27/3/2000	Comments from earlier versions incorporated.	
0.16	31/3/2000	Comments from earlier versions incorporated.	
1.0	10/4/2000	Comment form V0.16 incorporated, issued for approval.	

0.2 Approval authorities

Name	Position	Signature	Date
T. Austin	Development Director		
J. Dicks	Customer Requirements Director		
R. Booth	POCL		

FUJ00001378

0.3 Associated documents

Reference	Version	Date	Title	Source
TD/ARC/030	0.4	12/11/99	OPS Architecture Specification	ICL Pathway
TD/ARC/029	0.4	12/11/99	TMS Architecture Specification	ICL Pathway

0.4 Abbreviations and definitions

Abbreviation	Definition
ATR	Answer-To-Reset
DLL	Dynamic Link Library
EOD	End of Day
EPOSS	Electronic Point of Sale Service
GEC	General Electric Company
ISO	International Standards Organisation
OPS	Office Platform System
POCL	Post Office Counters Limited
RIPOSTE	Retail Integrated Point Of Sale system in a Transaction Environment: product from Escher that provides both the infrastructure and the Desktop environment of the Horizon system. The definitions in this manual refer to version 6 onwards.
SPM	Simple Payment Module

FUJ00001378

0.5 Changes in this version

Version	Changes
0.11	Review comments from version 0.10 have been incorporated.
0.12	Review comments from version 0.11 have been incorporated.
0.13	Review comments from version 0.12 have been incorporated.
0.14	Review comments from version 0.13 have been incorporated.
0.15	Review comments from version 0.14 have been incorporated.
0.16	Review comments from version 0.15 have been incorporated.
1.0	Review comments from version 0.16 have been incorporated.

0.6 Changes expected

Changes
This document reflects the current implementation. The provided descriptions and definitions may be subject to change control as determined by technical and/or operational needs.

FUJ00001378

0.7 Table of Contents

FUJ00001378

FUJ00001378

FUJ00001378

A.1 Scope

The information in this appendix is provided to enable the overall planning of new applications that access smart cards. It gives an overview of the software developed by ICL Pathway to overcome the issues caused by the lack of consistent standards for smart cards and tokens, and the use of differing security mechanisms in the smart cards in use in the core Horizon system. The interfaces that have been developed support reading from, and writing to, the smart card, as well as the interpretation of the Token and Product Reference Data involved.

The smart card reader in OPS supports reading from, and writing to, smart cards that comply with ISO 7816 parts 1and 2, or ISO 7816 parts 1, 2, and 3, and are capable of supporting future applications complying with ISO 7816 part 4. The smart card reader and Smart Card Manager software supports the existing APS tokens:

- British Gas Trading SPM Smart Token
- GEC Meters Watercard Smart Token
- British Gas Quantum Smart Token

A number of issues can arise with the introduction of a new smart card, and so a *Smart Card Review* will need to be conducted to identify and quantify any impact. Areas that need to be covered are the smart card technology used, recovery considerations, and the use of menus to allow the clerk to identify a smart card when identification by the system is not possible. See Appendix C, *System Management*, for details.

A.1.1 Content

The technology implications, particularly the security considerations of differing types of smart card are described in section A.2. To deal with the issues caused by differing card types and technology, ICL Pathway has had to develop an extension of the normal Riposte Peripheral Broker interfaces. Section A.2 describes the impact of the use of these interfaces.

The software that provides the smart card interface support for applications is the Smart Card Manager, shortened to SmartMan. The SmartMan interfaces are described in this document. The systems architecture provided by SmartMan for handling smart cards, described in section A.3, identifies those interfaces used by:

- The application that is responsible for implementing one or more smart cards.
- The code that deals with a particular card technology, the CardDLL.
- The SmartMan interfaces that support the application and CardDLLs.

FUJ00001378

All OPS applications are impulse driven. How impulses are handled when smart cards are used is described in section A.4

Because of the security schemes implemented by the various smart card providers, code must be developed for each scheme that can: “unlock” the card, interpret the information on the card, and provide the instructions necessary to update data on the card. This code will vary from card type to card type depending on the technology involved and the business requirement. As an aid to the development of such software, a typical set of interactions that an application will need to implement is described in section A.5.

The Reference Data is used by all applications within the OPS. For smart card applications, the token Reference Data needed to define the smart card, and the Reference Data required by SmartMan to identify the CardDLLs and applications, are covered in section A.6.

The interfaces provided by SmartMan, and those that must be provided by the application and CardDLL, are defined in the remaining sections.

A.1.2 Organisation of this document

The table below summarises how the content of this document is organised into the various sections.

Section	*Contents*
Section A.2	Presents an overview of how differing card technologies are identified and the issue of security code addressed.
Section A.3	This identifies the SmartMan architecture and the responsibilities of the Smart Card Manager, the smart card application and the CardDLLs.
Section A.4	This covers the peripheral impulses from the smart card reader and the impulses received by the application.
Section A.5	This section describes the typical sequence of calls made by an application.
Section A.6	This section describes what needs to be added to Reference Data including the entry in SmartManDLLs.
Section A.7	This section contains a definition of the ISmartMan Interface for CardDLLs.
Section A.8	This section contains a definition of the ISmartCard interface for Applications. This is the interface implemented by the CardDLLs. It is used by SmartMan and the card application.
Section A.9	This section contains a definition of the CardDLL Interfaces for the Application and SmartMan.
Section A.10	This section contains a definition of the CardDLL Interface for EPOSS Transaction Data.

FUJ00001378

Section A.11	This section contains a definition of the ISmartToken Interface.
Section A.12	This section contains a definition of the ISmartProduct Interface.
Section A.13	This section contains a definition of the ISmartApp Interface.

Table A-1 Organisation of content

FUJ00001378

A.2 Smart Card technologies

Smart cards supported by the Horizon system are of two main types or *technologies*: those that have an integrated circuit 'processor' as well as memory, and those with only memory.

The support for these cards is included in the card reader firmware, as well as the associated drivers that handle the input and output between the card and the reader.

Before any projected application is developed, discussions must occur with POCL and ICL Pathway during a *Smart Card Review*, (see Appendix C, *System Management*) to ensure that the type of card intended can be supported, and that any requirement for firmware enhancement is understood and can be quantified.

A.2.1 Processor cards

These are used when there is a requirement for some processing to be carried out on the card itself. For instance, if the data held on the card has to be encrypted, or an applet (a small application) runs on the card and interfaces with an application on the counter desktop.

Directory structures and passwords must conform to the ISO 7816 standard.

A.2.2 Memory cards

These are used to store data relevant to the application, such as the amount of credit for a utility meter application.

A.2.3 Recognising the card technology

For normal tokens (i.e. other than smart cards) the Peripheral Broker uses the Validation Object to identify and validate the token from Reference Data definitions and to invoke the appropriate application to process the token. Smart card processing differs importantly in needing an additional call to the Validation Object.

The first call causes SmartMan to be invoked but at this stage, the smart card cannot be validated. The second call to complete the validation can only take place after the card type has been identified, and for the processor smart cards, the card unlocked by security keys and the directory structure established.

The identification and validation of the smart card data itself can only occur once the data can be accessed. The sequence for identifying smart cards is shown in Figure A-1.

FUJ00001378

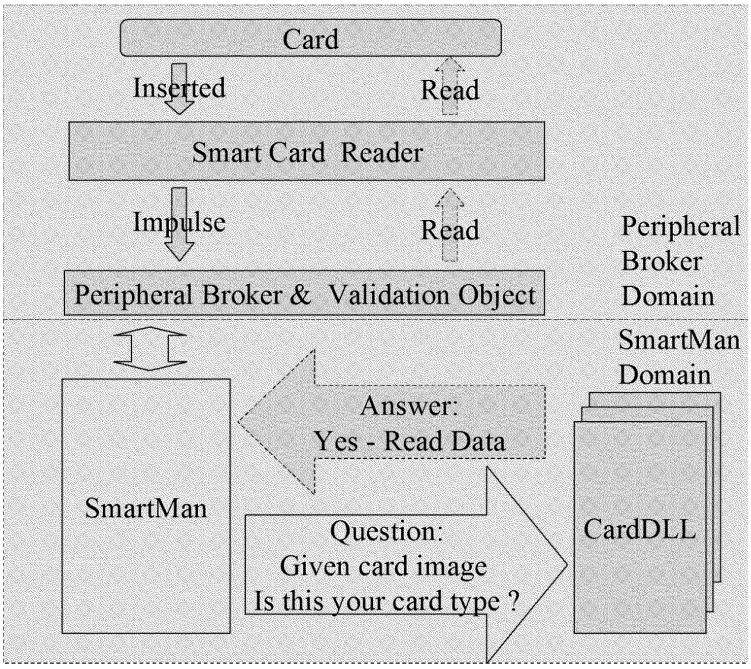


Figure A-1 Identifying the card type

Not all cards can be used in all outlets. Clients, or even products, may be restricted to particular regions of the country, for example. Reference Data for tokens will be sent only to the post offices to which it applies.

SmartMan will inform the clerk if the card is unrecognised at the outlet, but it requires CardDLLs to police card usage and to ignore cards that are presented at the wrong outlet.

The identification of some processor cards can be destructive, in that they would be locked permanently if the card were accessed more than a certain number of times by non-authorised code. If more than one card of this type is in use, and any card cannot be uniquely identified in a non-destructive way, the CardDLL response to SmartMan must be such as to ensure that a clerk selection process is invoked to identify the card type.

When inserted in the card reader, a card generates a different impulse according to its technology. The differing impulses enable SmartMan to recognise the type of card technology and to adjust the way it processes the card according to the type of card involved. For example, memory cards may be encrypted, whereas processor cards may have actual password processing included.

After receiving the initial impulse, SmartMan issues a request to power on the card, the Answer To Reset (ATR). When the response to the ATR has been received, for memory cards SmartMan takes a copy of the data on the card. It passes this copy of the card data to all CardDLLs to identify the appropriate application and code that deal with the card content and security features. For processor cards, where identification could be destructive, only the data

FUJ00001378

returned with the ATR response and any non-protected data on the card is passed to the CardDLLs. If the processor card cannot be uniquely identified by this data, identification of the card type must be undertaken by the clerk being presented with a menu of options.

The issue of identification of processor smart cards can impact existing applications and so must be raised during the SmartMan Review, see Appendix C, *System Management*.

Either type of card are only read from again, or written to after they, and the application that processes them, have been identified and if necessary the card has been “unlocked”. The developers of new applications must supply the security code needed to unlock their particular smart card implementation.

The impulses, and the processing that they initiate, are described in section A.4.

A.2.4 General Smart Card processing flow

The overall scheme of processing smart cards is shown in Figure A-2 below:

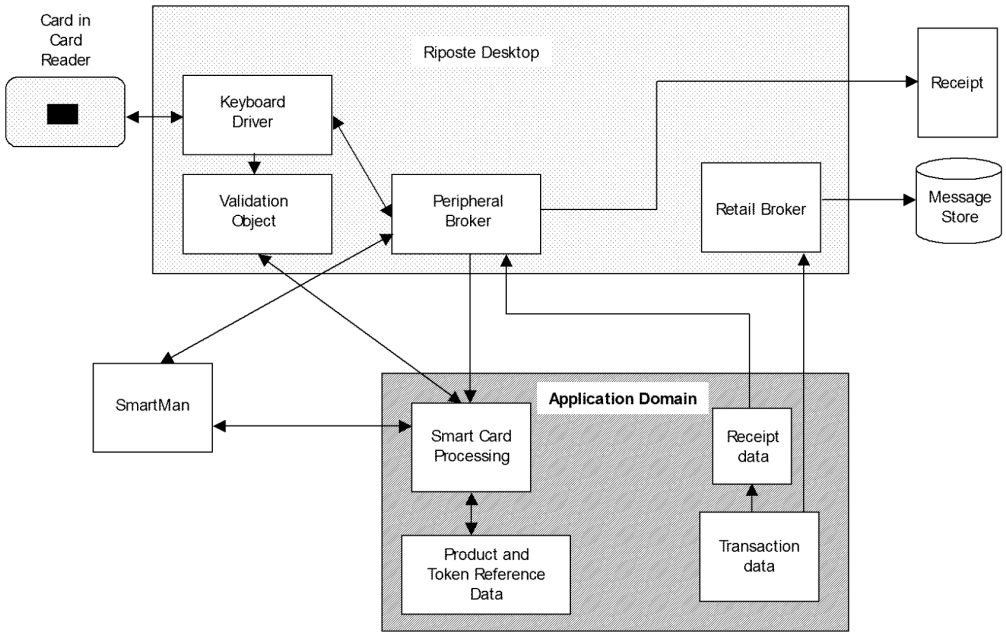


Figure A-2 Processing flows

SmartMan registers event names with the Peripheral Broker for smart card insertion and removal, thus providing an interface capable of supporting both processor and memory smart cards.

The Peripheral Broker uses the Validation Object to identify that a smart card has been used, and invoke SmartMan to process the card. The application is invoked by SmartMan to process the details on the card, and, using the Retail Broker it creates the transaction data on the transaction stack. Finally, the

FUJ00001378

application creates the receipt data for the Peripheral Broker to print the receipt.

The generic sequence thus becomes:

1. When a smart card is inserted into the card reader, the data from the ATR is passed by the device driver to the Peripheral Broker, which passes the data to the Validation Object.
2. The Validation Object matches the data to the event name (insertion or removal) and passes the event name and data to the Peripheral Broker.
3. The Peripheral Broker searches its list of registered events, and for smart cards, calls SmartMan.
4. SmartMan for memory cards takes a copy of the data on the card and passes it to each CardDLL in turn and asks if they support the card. When a CardDLL recognises a card that it can process, it informs SmartMan of this fact. The appropriate application can then be called to process the data from the card. In the case of processor smart cards that could be locked out if a non-authorised CardDLL attempted to read them, the response from the ATR should be used if this can uniquely identify the card. If this is not possible then the clerk will have to be asked to identify the card from a menu identifying all the cards that can be used across all outlets. The application must police whether or not the card can be used at the specific outlet.
5. Where security code is involved, the card is then ‘unlocked’ by the CardDLL, as shown in Figure A-3. The Validation Object is called again to match the card to a registered token definition. The application can then use the CardDLL to read any remaining data from the card, using SmartMan and the Peripheral Broker.
6. The application calls the CardDLL to update the card, via SmartMan and the Peripheral Broker.
7. The application uses the Retail Broker interfaces described in section 5.3, *The Retail Broker*, in the main body, to place the transaction details on the transaction stack, and to commit the transaction to message store. It also passes any receipt data to the Peripheral Broker.

FUJ00001378

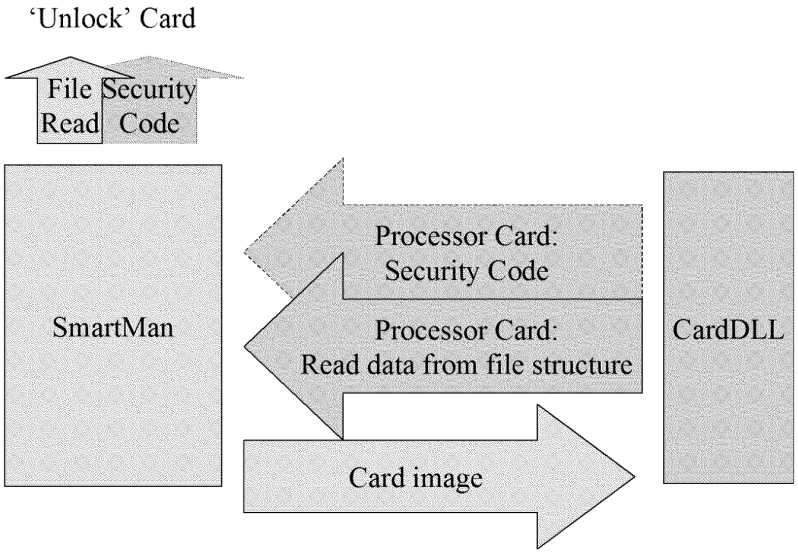


Figure A-3 Unlocking a processor type of smart card

For recovery purposes, it is recommended that the application writes recovery data to the message store itself at the points in the sequence that reflect the business requirements on recovery. See section A.2.5.

A.2.5 Processing Constraints

It is the responsibility of the application to deal with recovery situations as described in section A.2.5.1 as well as the implications of dealing with cards that have specific currency requirements that are not covered by the business rules identified in section 4 of the main document.

A.2.5.1 Recovery

If the application creates recovery data that is committed to the message store as the transaction proceeds but before the final transaction is committed via the Retail Broker, then there are several recovery situations that have to be taken into account. These include clerk and system induced recovery scenarios.

Because of the impact such recovery can have on the performance of the system, ICL Pathway should be consulted about the exact mechanism to be used during the *Performance Review*. The *Performance Review* is described in the *Generalised API for OPS/TMS, Appendix C*: *System Management*, section C.2.3.

FUJ00001378

A.2.5.1.1 Session Swap and Session Transfer

It is possible that the clerk will try to interrupt a session and swap to a new session, or even try to transfer the session to a different counter position. For this reason, when SmartMan receives the impulse indicating a smart card has been inserted into the reader, it locks the desktop buttons to prevent session swap or session transfer. The application must then ensure that these buttons remain locked until after it has told SmartMan that the transaction has been completed, asked SmartMan to get the card removed, and had confirmation back from SmartMan that the card has been removed.

A.2.5.1.2 Interrupted Transactions and Forced Log Out

There are two situations not under the control of the clerk that need to be addressed by any application developed to handle smart cards. These are caused when a counter PC fails part way through a transaction or when a session is ‘settled’ by the system and the clerk logged out when the terminal has been unattended for some while. In these two scenarios it is the responsibility of the application to deal with the recovery of any transactions involved and the implications of the forced settlement. These scenarios are described in section 4.2 of the main body.

A.2.5.1.3 Financial

It is possible that the smart card may, for example, not deal in sterling or may work with amounts defined to a hundredth of a penny. It is the responsibility of the application to deal with any conversions needed to support both the smart card requirement and the Electronic Point Of Sale Service (EPOSS) requirements of the Horizon system.

FUJ00001378

A.3 Architecture and interfaces

ICL Pathway has developed the Smart Card Manager (SmartMan) to provide a single set of consistent interfaces that can be used by applications that run in the OPS to handle smart cards. It allows the Peripheral Broker interfaces (that are used by applications to deal with all the other peripherals that form part of the OPS) to be replaced with interfaces that can deal with the security implications of smart cards. The Peripheral Broker impulses for other devices are described in section 4, *Business Processes*, of the main document.

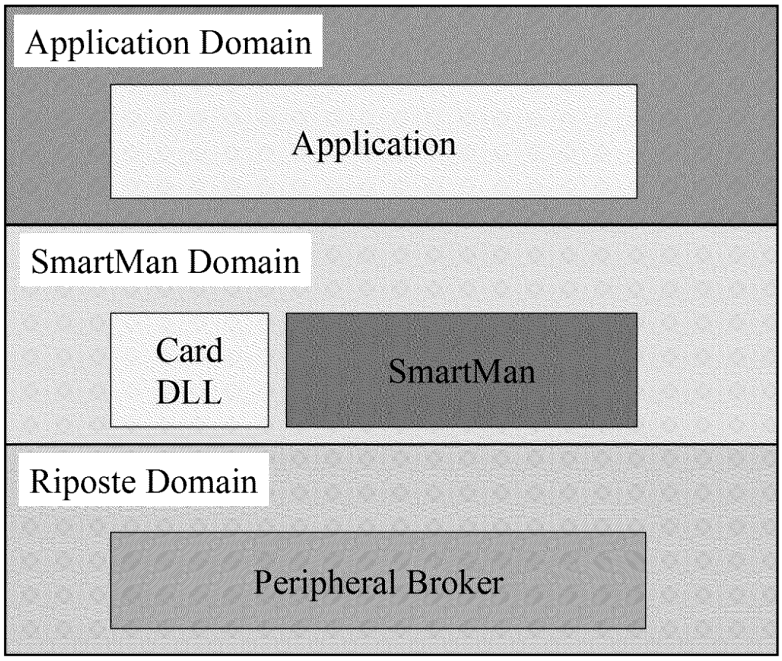


Figure A-4 Application components

Each application that is developed must use these Smart Card Interfaces and provide one or more appropriate CardDLL**s** and if appropriate, security code to deal with the technology for its own smart cards. In turn, these CardDLL**s** must implement the interfaces required by Smart Card Manager.

The set of APIs provided by SmartMan cover separate interfaces to the application and CardDLLs, and between the application and the CardDLL. Figure A-4, above, illustrates the general relationship of these components.

FUJ00001378

A.3.1 Architecture

The Smart Card Manager (SmartMan) is responsible for identifying the CardDLL associated with a card when a card is inserted into the card reader, and notifying the interested application whenever a smart card impulse is received. The SmartMan software acts as a buffer between the Smart Card reader, the Peripheral Broker, the application and the CardDLL.

The technology dependent DLLs that have to be provided need to perform those functions specific to a particular type of card, such as updating credit details, extracting meter readings and encryption. Each of these CardDLLs must implement the complete interface defined for the Smart Card Manager as the ISmartCard interface. The CardDLL**s** communicate with SmartMan through the ISmartMan interface.

Applications communicate with the Smart Card Manager and CardDLL via the ISmartMan and ISmartCard interfaces. They also need to provide a set of specific application interfaces that deal with Reference Data: ISmartToken, ISmartProduct, and ISmartApp. The actual sequence of interaction between the application and SmartMan and the CardDLL depends on the smart card technology involved and the business rules involved. Typical sequences are described in section A.5. The impulses that the application receives are described in section A.4.2.

As with all applications, the following are defined in Reference Data:

- The EPOSS product that must be transacted if the card is used
- Any additional data that must be captured during the smart card transaction

The structure of the data on a particular smart card can also be defined in Reference Data if this is appropriate and does not compromise security. Sufficient information must be defined to allow the Validation Object to check that the card is valid. In the case of processor cards that need to be accessed by authorised CardDLLs, these checks can only be made after the card is unlocked.

The Smart Card Manager also requires definitions to be provided of the CardDLLs and application to be called to deal with a particular smart card. These definitions are also held in Reference Data in record collections private to the Smart Card Manager. The Reference Data is described in more detail in section A.6.

The software functions provided by SmartMan to support the handling of smart cards can be described in terms of a number of components as shown in Figure A-5.

FUJ00001378

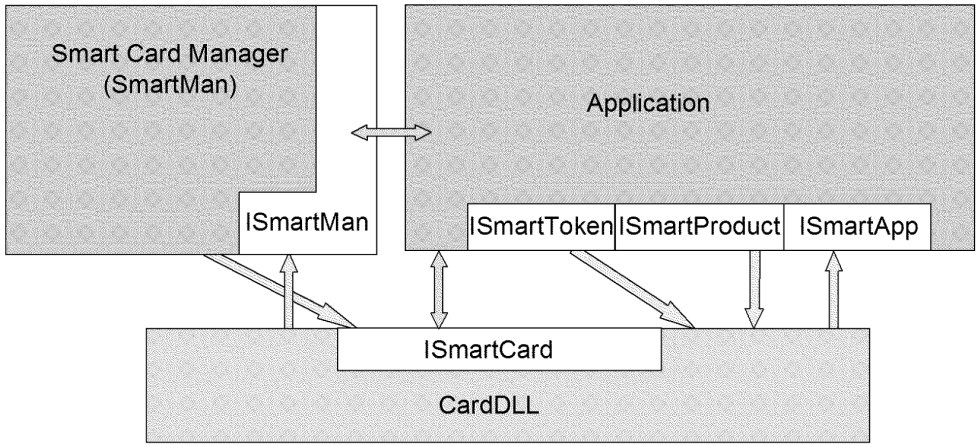


Figure A-5 Smart Card system architecture

For each new smart card application one or more CardDLLs must be developed, in addition to any security code and the new application itself, unless a suitable CardDLL or security code object already exists.

The interfaces that have been developed by ICL Pathway for applications that support smart cards cover:

- The interface that deals with a smart card itself, whether it is a memory or a processor card. This interface is ISmartMan. The ISmartMan interface for applications and the CardDLL support the reading from and writing to the smart card and allow the application to know which CardDLL has been used for such access.
- The interface that deals with the data on the card. This interface is ISmartCard and it ensures that there is a consistent approach to recovery and that the correct card is being used. The interface ensures also that the requirements of the EPOSS transaction can be supported.
- Those interfaces that deal with the functionality are defined in POCL token Reference Data. These interfaces are ISmartToken, ISmartProduct and ISmartApp.

Because of the potential interactions with existing smart card usage, the *Smart Card Review* (see Appendix C*, System Management*) will need to determine what impact any new smart card will have on the existing SmartMan implementation.

FUJ00001378

A.3.2 Component responsibilities

The responsibilities of each component are as follows:

A.3.2.1 Application

The responsibilities of the application are to:

- Use the CardDLL that was identified by ISmartMan to read from, and write to, the card. This is provided by *Card Definition methods* and is described in section A.8.
- Deal with recovery situations, and ensure that the correct card is used at all points in the dialogue with the clerk. This is provided by ISmartCard *Card Content methods*, and is described in section A.9.
- Deal with the additional data requirements of Reference Data for the EPOSS transaction. This is provided by ISmartCard *Card Content methods*, and is described in section A.10.
- Ensure that the appropriate Product and Token Reference Data is used and actioned, and that the required additional data is extracted from the card. These are individual *Reference Data* interfaces and are given in sections, A.11, A.12 and A.13.

A.3.2.2 SmartMan

The responsibilities of SmartMan are to:

- Identify that a card is inserted into the smart card reader, removed from the reader, and re-inserted.
- Provide the CardDLL with the data access requested, via the ISmartMan interface described in section A.7. This is provided by ISmartMan *Card Definition methods.*
- Identify the CardDLL and application that deal with the card via the ISmartMan interface described in section A.8. These are provided by ISmartMan *Card Definition methods.*
- Support the recovery situations and ensure that the correct card is used at all points in the dialogue with the clerk, provided that the ISmartCard *Card Content methods* in section A.9 are used.

FUJ00001378

A.3.2.3 CardDLL

The responsibilities of the CardDLL are to:

- Implement the ISmartMan *Card Definition methods* defined in section A.7 for card access.
- Implement the recovery situations and ensure that the correct card is used at all points in the dialogue with the clerk. These are ISmartCard *Card Content methods*, and are described in section A.9.
- Support application interfaces for EPOSS transaction data. These are *Card Content* ISmartCard methods, and are described in section A.10.

A.3.3 SmartMan API interface structure

The functionality provided by each of the Smart Card Manager interfaces shown in Figure A-2 is identified below.

Interface	*Type*	*Used by*	*Reference*	*Supports*
ISmartMan	CD	CardDLL	A.7	ISmartMan_ChkCod ISmartMan_RdFil ISmartMan_RdMem ISmartMan_ReadCard ISmartMan_SelDir ISmartMan_SelFil ISmartMan_UpdFil ISmartMan_WriteCard
ISmartMan	CD	Application	A.8	ISmartMan_GiveCardDLL ISmartMan_GiveCardDLLDetails ISmartMan_GiveTokenTypes ISmartMan_RequestSMAction
ISmartCard	CC	Application SmartMan	A.9	ISmartCard_AdjustCredit ISmartCard_ApplyUpdates ISmartCard_CancelTransaction ISmartCard_CardUpdate ISmartCard_CheckAdditionalInfoValue ISmartCard_CheckCardType ISmartCard_CheckForCardChange ISmartCard_GiveReceiptCheckDigits ISmartCard_IdentifyCard ISmartCard_Prepare ISmartCard_StartRecovery ISmartCard_StartReversal ISmartCard_ValidateCredit ISmartCard_VerifyCardIntegrity
ISmartCard	CC	Application	A.10	ISmartCard_CardClientID ISmartCard_CardTech

FUJ00001378

				ISmartCard_CurrentCredit ISmartCard_CustomerID ISmartCard_MeterSerial ISmartCard_ReturnData ISmartCard_ReversalAllowed ISmartCard_TransactionValid
ISmartToken	RD	CardDLL	A.11	ISmartToken_PFData ISmartToken_ProductNo ISmartToken_SmartData ISmartToken_TokenName ISmartToken_TokenType
ISmartProduct	RD	CardDLL	A.12	ISmartProduct_AdditionalData ISmartProduct_MaxValue ISmartProduct_MinValue ISmartProduct_MultipleValue
ISmartApp	RD	CardDLLs	A.13	ISmartApp_WriteTransactionData

Key: CD = Card Definition

CC = Card Content

RD = Reference Data

Table A-2 Functionality of SmartMan interfaces

All the methods of an interface must be implemented, even if they are not used. A stub must be provided in this case.

FUJ00001378

A.4 Impulses

A.4.1 Peripheral impulses

After a card has been identified, all further impulses are assumed to relate to that CardDLL and application until such point as it is removed, or SmartMan and the CardDLL identify that a different card is being used. Figure A-6, below, illustrates the impulses involved, including those initially dealt with by SmartMan, such as the initial impulse that starts the processing and the impulse that results in the card being re-inserted. The arrows indicate what has happened to the card.

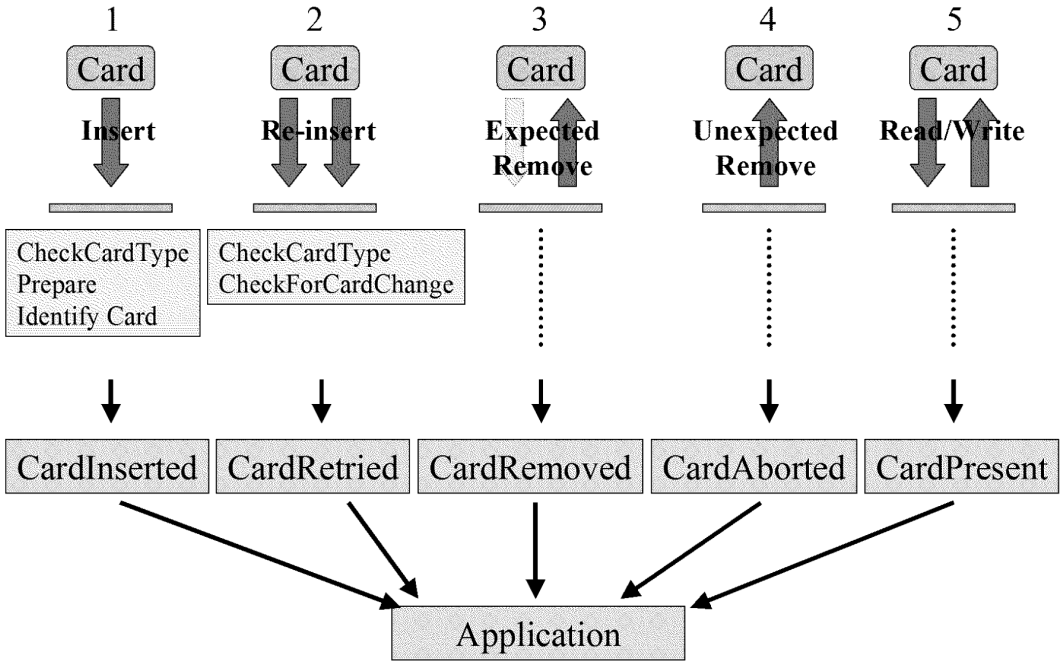


Figure A-6 Peripheral impulses and transactions

Impulses are received under a number of different circumstances, and will occur in differing orders, depending on the application logic and the actions taken by the clerk. The impulses supported are:

1. When the card is inserted for the first time, an impulse is received that, when processed by SmartMan, is passed on to the application as a CardInserted impulse. This impulse initiates a transaction.
2. The application receives a CardRetried impulse when a card is re-inserted as part of the dialogue with the clerk, having been removed at the request of the application or SmartMan.

FUJ00001378

3. Should the clerk remove the card when requested to by the application or SmartMan, then the application is advised this has occurred by a CardRemoved impulse.
4. Should the clerk remove the card when there is no request from the application or SmartMan to do so, then the application is advised this has occurred by a CardAborted impulse.
5. When the card is read from or written to, the application is informed that the card is present by the CardPresent impulse.

Each of these scenarios is described in more detail in the following sections.

A.4.1.1 Card Inserted

When a card is inserted into the card reader an impulse is generated and passed to SmartMan. SmartMan issues an asynchronous reset to the card reader to power on the card, and then waits for the next impulse.

SmartMan, when it receives an ATR (Answer-To-Reset) impulse from the Peripheral Broker, initiates in each registered CardDLL a sequence of method calls to determine which CardDLLs are capable of supporting the card. The methods used are:

- CheckCardType
- Prepare
- IdentifyCard

Once a CardDLL has been identified, or selected via a menu displayed to the clerk, as capable of processing the card, SmartMan passes a CardInserted impulse to the application (case 1 in Figure A-6). See section A.4.2.

For memory cards, the data on the card is used by the CardDLLs to identify that they can support the card. If more than one CardDLL and its associated application indicate that it can process the memory card at the outlet, the clerk is asked to select the required card from a menu. It is the responsibility of the application and CardDLL to police whether or not a card can be used at a particular outlet from the Product and Token Reference Data sent to all outlets. The menu is generated from the MenuDesc attribute values in the SmartCardDLLs reference data for the card type, see section A.6.1 for details.

Where the card is a processor card that can have security features that may be destructive, the data passed to the CardDLLs is that received in the response to the ATR. If this is not sufficient to identify the card a menu of card types must be presented to the clerk for selection. In this case, only after the clerk has identified the card type can the card be ‘unlocked’ for further processing. The CardDLL must instigate the unlocking of cards that have security keys. The CardDLL is used to extract data from the appropriate files within the directory structure, for processor cards, before any dialogue with the user.

FUJ00001378

A.4.1.2 Card Re-inserted

If SmartMan receives an ATR impulse because a card has been re-inserted during a transaction, (e.g. it had been removed for cleaning), SmartMan initiates a sequence of CheckCardType and CheckForCardChange ISmartCard method calls to the CardDLL for the card. These checks are to determine whether the same card has (or has not) been re-inserted.

If the card has been re-inserted, SmartMan passes a CardRetried impulse to the associated application to allow it to continue processing (case 2 in Figure A-6). However, if a new card has been inserted, SmartMan passes a CardAborted impulse to the application so that the transaction can be aborted, or the application can request the clerk to re-insert the original card and start again. Where more than one processor cards with security features exists, and the card cannot be uniquely identified by the response from the ATR, the clerk must be asked again to confirm that the same card has been re-inserted.

A.4.1.3 Card Removed

If the impulse is an expected CardRemoved impulse (e.g. as the result of a cleaning request) SmartMan waits for the next impulse. SmartMan informs the application by issuing a CardRemoved impulse (case 3 in Figure A-6).

A.4.1.4 Card Unexpectedly Removed

If the impulse is an unexpected CardRemoved impulse, (e.g. the Clerk has removed the card prematurely), SmartMan passes a CardAborted impulse to the application so that the transaction can be aborted, or the application can request the clerk to re-insert the original card and start again (case 4 in Figure A-6).

A.4.1.5 Card Write

For any other card read or write requests from the CardDLL that are passed through the SmartMan, a CardPresent impulse is sent to the application (case 5 in Figure A-6).

A.4.1.6 Recovery

Any application dealing with smart cards must provide for the recovery of information should the customer session be terminated by equipment failure before the session has been settled. The mechanism must take account of what might have been written to the smart card. The recommended approach is to write transient messages at appropriate points in the dialogue.

FUJ00001378

Because of the impact such recovery can have on the performance of the system, ICL Pathway must be consulted about the exact mechanism to be used during the *Performance Review*. The *Performance Review* is described in the *Generalised API for OPS/TMS, Appendix C*: *System Management*, section C.2.3.

A.4.1.7 Reversal transaction

The actual order of impulses processed by the application depends on whether or not the normal or reversal transaction is being undertaken. The processing sequence is different in reversal situations. In reversal cases, the application must ask SmartMan to get the card inserted using the RequestSMAction method. This differs from the situation where the card is inserted by the clerk to start a new transaction. In particular, unless SmartMan has been asked to get the card inserted, it checks there isn't another application running before it attempts to start the process of powering on the card and identifying the card and associated application. SmartMan uses the setting of the Riposte CurrentApp property to determine if any other application is running. If another application is running, SmartMan simply ignores the card impulses.

The application therefore needs to deal with reversals in such a way that the context is correctly set before the clerk is asked to deal with a reversal.

A.4.2 Application impulses

The structure of the impulse message received by the application is shown below. The interface defines the CardDLL, and the Card Technology involved, as well as the relationship between a particular type of card and the POCL Clients that can use it:

FUJ00001378

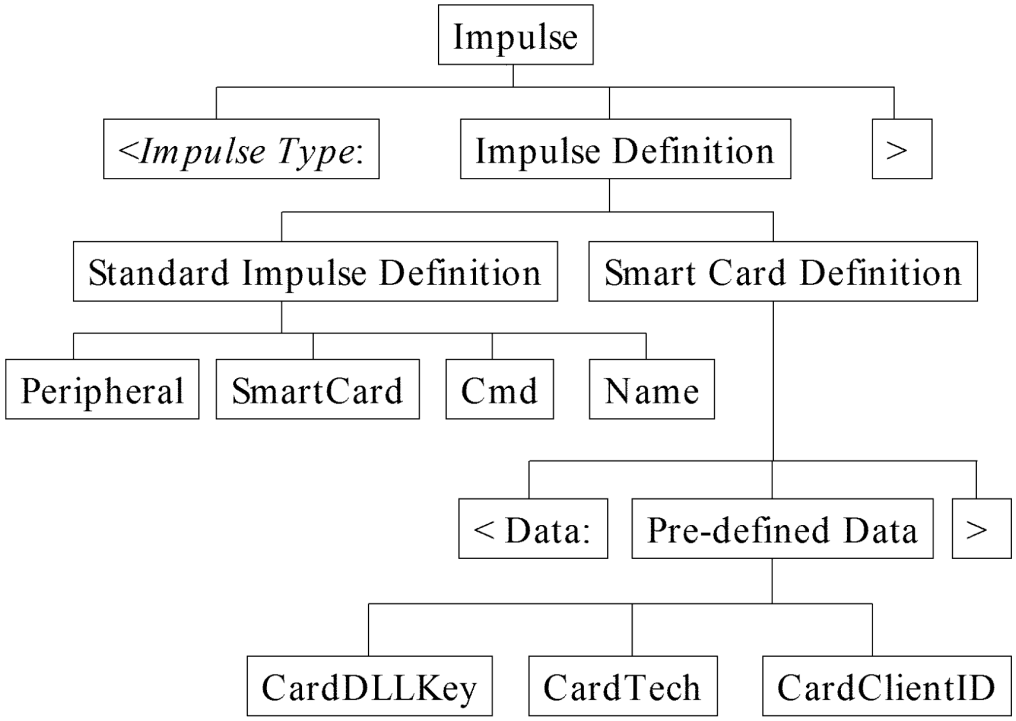


Figure A-7 Smart card Impulse definition

Note that an attribute value is either a string or an attribute grammar string and can contain blanks (an 'empty' string).

Attribute	***Value***
CardClientID	Read from the Smart card. This is the item CID in Reference Data. This attribute may have an empty string value when the impulse is generated as a result of the card application asking SmartMan to ask the clerk to get a card inserted and the card could not be identified.
CardDLLKey	Token Type of the smart card, as defined in token Reference Data. This attribute may have an empty string value when the impulse is generated as a result of the card application asking SmartMan to ask the clerk to get a card inserted and the card could not be identified.

FUJ00001378

CardTech	The same value as SmartCard above. Value of the item CardTech in the specific CardDLL Reference Data for handling the inserted card. This attribute may have an empty string value when the impulse is generated as a result of the card application asking SmartMan to ask the clerk to get a card inserted and the card could not be identified.
Cmd	This has the value: CardAborted, CardInserted, CardPresent, CardRemoved, CardRetried
Impulse Type	This is always set to CmdStr.
Name	Token Type of the Smart card, as defined in token Reference Data, item TT. See section 4.1.5.6, *Interface definition*, in the main document. This is the ObjectName in the SmartManDLL collection that identifies the CardDLL. This attribute may have an empty string value when the impulse is generated as a result of the card application asking SmartMan to ask the clerk to get a card inserted and the card could not be identified.
Peripheral	The device used to read the card. This is always ICLLiftKB as this incorporates the smart card reader.
SmartCard	This indicates that the card is a smart card. See CardTech above. This attribute may have an empty string value when the impulse is generated as a result of the card application asking SmartMan to ask the clerk to get a card inserted and the card could not be identified.

FUJ00001378

An example impulse might be:

```
<CmdStr:
    <Peripheral:ICLLiftKB>
    <SmartCard:JCPetrol>
    <Cmd:CardInserted>
    <Name:JPC>
    <Data:
        <CardDLLKey:JPC>
        <CardTech:JCPetrol>
        <CardClientID:0007>
    >
>
```

FUJ00001378

A.5 Application Processes

The interaction between certain ISmartCard interfaces is outlined here in terms of the sequence of interface calls made by a typical application. The logic and processing involved depends on the specific requirements of the card application and the particular smart card.

A.5.1 Example of Overall Sequence

The application used to process a particular type of smart card is informed, by SmartMan, when such a card is inserted into the card reader. SmartMan does this after it has successfully used the CheckCardType, Prepare and IdentifyCard interfaces to identify the CardDLL and hence the application that deals with the card. The application then makes a sequence of calls on the identified CardDLL using the appropriate ISmartCard interface.

Interspersed in the sequence of calls would need to be other non-CardDLL actions to load relevant Token and Product Reference Data, write recovery data to the message store, collect data from the Counter Clerk and write transactions and receipts. Also, the sequence of calls may be 'disturbed' to allow the application to cope with invalid data being entered, the clerk removing the card from the reader etc, but in general, the calls made normally follow the same sequence.

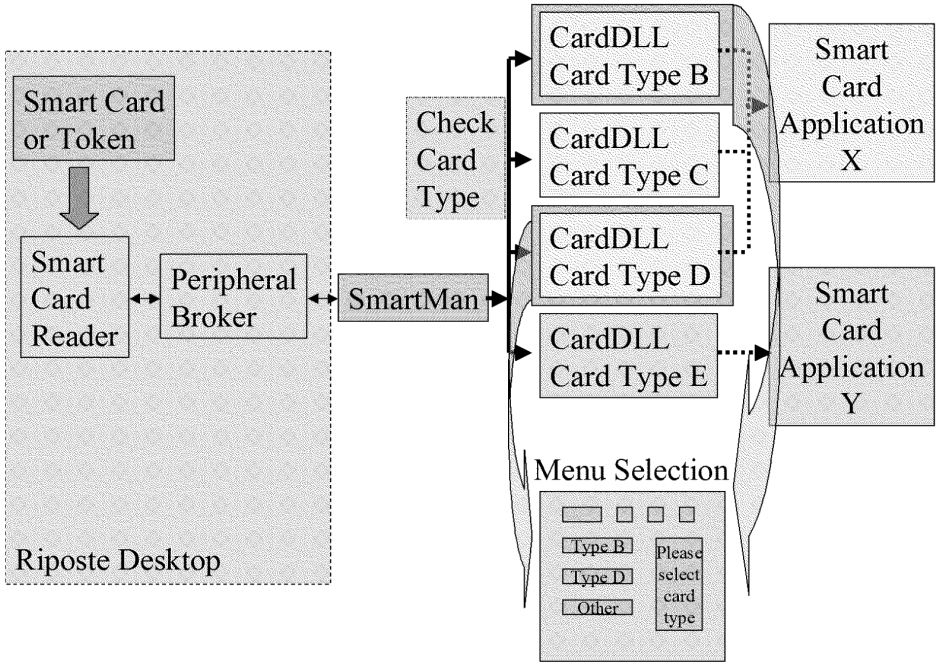


Figure A-8 Processes

In the menu presented to the clerk to identify the card being used, the menu text used reflects the titles given to the smart cards in Reference Data.

FUJ00001378

Other peripherals can be used during a transaction and are dealt with in the normal way by the Peripheral Broker, which provides the application for input devices with the appropriate impulses. It is the responsibility of the application to set the Riposte CurrentApp value to prevent other impulses from other input peripherals interrupting a smart card dialogue.

The construction of these menus can impact other smart card applications and so must be discussed during the *Smart Card Review,* see Appendix C, *System Management*.

A.5.2 Example of normal transaction

Typically, the sequence of calls for normal transactions, would be:

1. ISmartCard_VerifyCardIntegrity
2. ISmartCard_ApplyUpdates
3. ISmartCard_ValidateCredit
4. ISmartCard_AdjustCredit
5. ISmartCard_CardUpdate
6. ISmartCard_GiveReceiptCheckDigits
7. The card application uses the ISmartMan_RequestSMAction method to request SmartMan to ask the clerk to remove the card from the reader.

A.5.3 Example of reversal transaction

Similarly, a typical sequence of calls for reversal transactions, would be:

1. The card application uses the ISmartMan_RequestSMAction method to request SmartMan to ask the clerk to insert the card in the reader.
2. ISmartCard_StartReversal
3. ISmartCard_ApplyUpdates
4. ISmartCard_AdjustCredit
5. ISmartCard_CardUpdate
6. ISmartCard_GiveReceiptCheckDigits
7. The card application uses the ISmartMan_RequestSMAction method to request SmartMan to ask the clerk to remove the card from the reader.

Note that ISmartCard_StartReversal has the same effect in this sequence as ISmartCard_VerifyCardIntegrity has in the typical sequence for a normal transaction.

It is the responsibility of the application to check that the correct card is used during a reversal, and that business rules on reversals are implemented, particularly with respect to reversals over day boundaries as identified by the EOD marker. See section 4.2.3 in the main body for details on EOD.

FUJ00001378

A.5.4 Example of recovery transaction

If an error occurs during any of these sequences, the application may need to attempt some form of recovery action and then attempt to re-start the sequence at the point of failure. Alternatively, if recovery is not possible, the application can call the ISmartCard interface CancelTransaction method to inform the CardDLL the transaction has not been successful, and CardUpdate to allow the CardDLL to update any status information recorded on the card.

A typical sequence of calls for recovery transactions would therefore be:

1. ISmartCard_Prepare
2. ISmartCard_StartRecovery
3. ISmartCard_CheckAdditionalValue
4. ISmartCard_AdjustCredit
5. ISmartCard_CardUpdate
6. ISmartCard_GiveReceiptCheckDigits

A.5.5 Interrupted transactions

Should a transaction be interrupted by the failure of a PC, then the status of the transaction is reflected in the last recovery data recorded. It is therefore recommended that the application writes recovery messages as the transaction proceeds.

Recovery would then take place as identified in the example in section A.5.4.

Because of the impact such recovery can have on the performance of the system, ICL Pathway must be consulted about the exact mechanism to be used during the *Performance Review*. The *Performance Review* is described in *Appendix C: System Management,* section C.2.3.

FUJ00001378

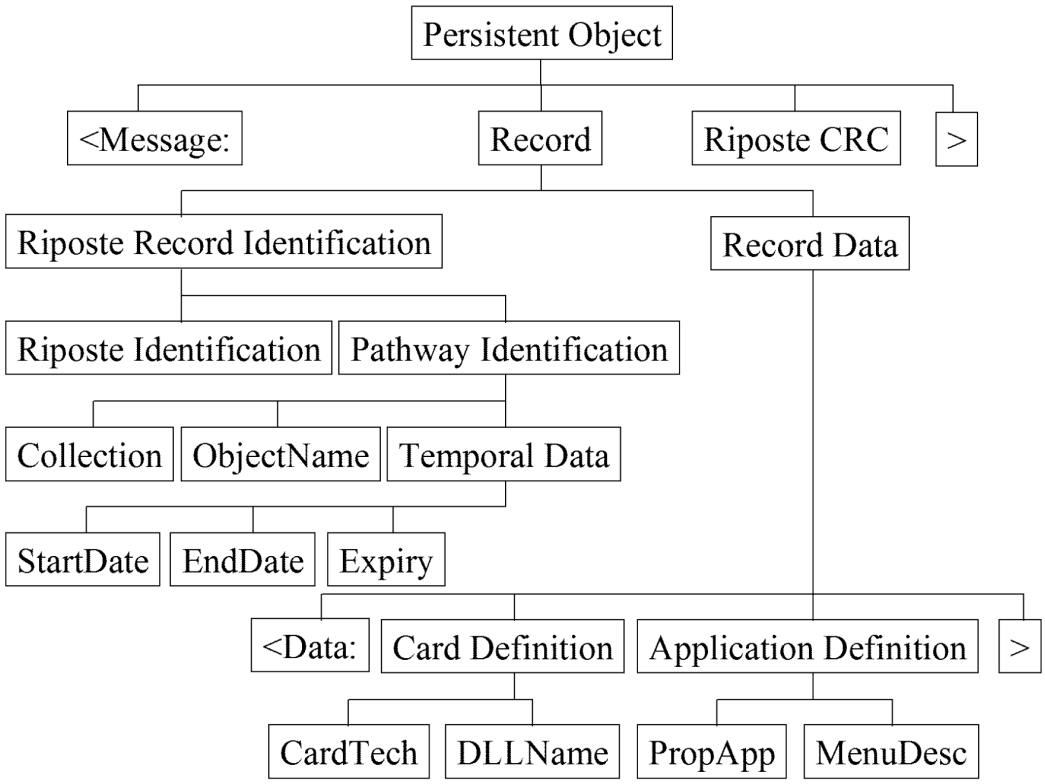
A.6 Smart Card Reference Data

There are a number of Reference Data definitions that apply to smart card applications. These are:

- The definition of the Reference Data that defines the Product associated with each card type, the Product Reference Data, is given in section 4.1.4.3 of the main document. See also section A.12on the interface needed to support Product Reference Data.
- The Reference Data that defines each card type, the Token Reference Data, is defined in section 4.1.5 of the main document. See also section A.6.2 on the smart card specific Token definitions and section A.11 on the interface needed to support Token Reference Data.
- The Reference Data that identifies which CardDLL supports which card or Token definition, the SmartCardDLLs collection. This is defined in section A.6.1.

A.6.1 SmartCardDLLs collection

To enable SmartMan to locate and load the CardDLLs, each CardDLL must have the appropriate Reference Data created in the SmartManDLLs Reference Data collection, with an ObjectName equal to the Token Type, as follows:



FUJ00001378

Figure A-9 Smart card Reference Data

Attribute	***Value***
CardTech	Value of the item CardTech in the specific CardDLL Reference Data for handling the inserted card. See section A.4.2. It is used here only for documentation and readability purposes. During processing, CardTech is supplied by the CardDLL via the ISmartCard_CardTech property.
Collection	SmartManDLLs.
DLLName	This is the DLL for the Card Technology.
EndDate	End date (dd-Mon-yyyy hh:mm:ss).
Expiry	Number of days after which the message is considered for archiving and deletion.
MenuDesc	A short description of the application supported by the CardDLL. This is used to generate a menu selection button if more than one CardDLL is capable of supporting the card.
ObjectName	This *must* be value of the TT attribute in the Token definition associated with this CardDLL. It is used as the unique code to indicate the Token Type on a Smart card transaction receipt associated with this type of card. It is also used by SmartMan to enable associated applications to request details of the CardDLL in use.
PropApp	This application is called when that CardDLL is used. See section 4.1.5.6, *Interface definition*, in the main document.
StartDate	Start date (dd-Mon-yyyy hh:mm:ss)

A typical entry, for example might be:

FUJ00001378


```
<Collection:SmartManDLLs>
<ObjectName:JPC>
<StartDate:01-Jan-2000 00:00:01>
<EndDate:>
<Expiry:34>
<Data:
    <CardTech:JCPetrol>
    <DLLName:JCPetrolCard.JCPetrol>
    <PropApp:APS>
    <MenuDesc:Petrol Pre-Payment>
>
```

A.6.2 Smart card token Reference Data

Smart card transactions are more complex than either magnetic card or bar code transactions, as there may be a multi-phase interaction with smart cards during the course of the transaction. The definitions used for smart cards are defined in section 4.1.5 in the main document and cover how security checks are defined for memory cards. Processor smart cards use the same Reference Data definitions, but the details that define security and card layout are not always included as these likely to be handled by separate security software and the CardDLL itself.

Token definitions for smart cards have Element definitions for some data that uniquely identifies the token. This is the CardClientId (read from the smart card at run time) plus a string of characters that is unique to the CardDLL that will process the smart card. This latter string is CardTech.

Two sets of element attributes therefore must be set up for each smart/memory token in the token Reference Data.

Each element of the data on a token is defined in terms if its element identity (EID), the name of the element (Name), given that it starts (Start) at a particular position and has a specific length (Len). The type of the data element (Pic) and the token identity to which it belongs (TID) are included, as is the start position (Start), type (Type) and length of the field to which it is applied (Length) and length of the check digit (Len).

FUJ00001378

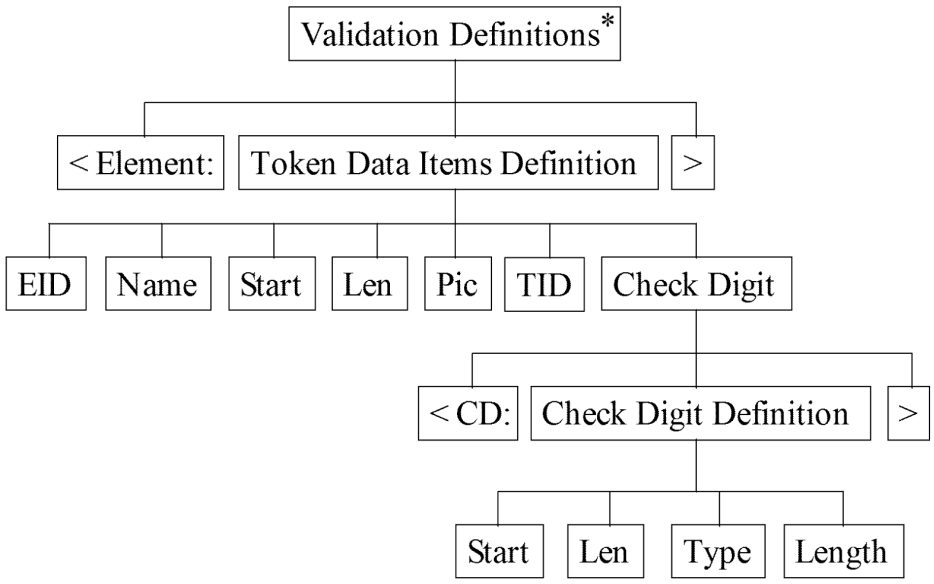


Figure A-10 Element Definitions

Data for specifying **Element** (token data items):

Name	***Description***
EID (Element)	Data element identity
Len (Element)	Data element length
Name (Element)	Data element name
Pic (Element)	Data element type - a quoted string that represents the card technology type e.g. “JCPetrol”, see below*
Start (Element)	Data element start
TID (Element)	Token identity

Data for specifying **CD** (check digits):

Name	***Description***
Len (CD)	Length of check digit
Length (CD)	Length of field to which it is applied
Start (CD)	Check digit start position
Type (CD)	Check digit type

As an example:

FUJ00001378


```
<Element:
 <EID:1>
 <Name:CardTech>
 <Start:1>
    <Pic:”JCPetrol”>
    <Len: 8>
    <CD:>
    <TID:0007>
>
<Element:
    <EID:2>
    <Name:Rref>
    <Start: 9>
    <Len:4>
    <CD:>
    <TID:0007>
>
```

FUJ00001378

A.7 The ISmartMan Card Definition Methods for CardDLLs

These SmartMan methods are used by the CardDLLs. They support simple memory cards and smart cards that include their own processors. The following calls are ISO 7816 wrapped calls.

The simple memory card methods are:

- ISmartMan_ReadCard
- ISmartMan_WriteCard

The more complex processor enabled cards require additional methods to deal with security codes and any directory structures present on the cards:

- Security:
  - ISmartMan_ChkCod
- Directory structures:
  - ISmartMan_RdFil
  - ISmartMan_SelDir
  - ISmartMan_SelFil
  - ISmartMan_UpdFil
- Reading Data:
  - ISmartMan_RdMem

Note: The CardDLLs must only use the functions listed below, and not those provided by the Riposte Peripheral Broker.

In the definitions that follow, eSMResults is an enumeration supplied by the ISmartMan interface definition. The definitions of byte, string, variant, and long are the standard Microsoft VB conventions.

FUJ00001378

A.7.1 ISmartMan_ChkCod

Used to apply the secret code to the Smart card. (The ISmartMan_ChkCod command calls the COS ChkCod command.)

Parameters		
Name	***Type***	***Description***
bytCodeNumber	Byte	The reference number of the code being applied.
vCodeValue	Variant	The array contains the secret code derived from the appropriate security module. It may or may not be readable text.
sCardResponse	String	Status code from card.
Return Value	Long	eSMResults.SM_CARDOK Success. eSMResults.SM_CARDLOCKED Card Locked – Unusable.

A.7.2 ISmartMan_RdFil

Reads a number of bytes from a specified file on the Smart Card

Parameters		
Name	***Type***	***Description***
lFileNo	Long	The number of the file to be read. Range 0 – 255.
lWordAddress	Long	The offset of the first word to be read in the file. Range 0 – 255.
lLength	Long	The number of bytes to be read.
vReadData	Variant	Pointer to the data area where the raw (binary) data is stored.
lBlockSize	Long	The maximum amount of data that can be read in a single transfer block. SmartMan will split large card accesses into several smaller blocks with a maximum length of lBlockSize.
sCardResponse	String	Contains a four digit hex string containing the response code from the smart card. If successful, this will contain eSMResults.APDU_OK. In an error occurs, this will be the failing code.

FUJ00001378

Parameters		
Name	***Type***	***Description***
Return Value	Long	eSMResults.SM_CARDOK Success. Others The requested data could not be read from the card.

A.7.3 ISmartMan_RdMem

Reads a number of bytes from the Smart card.

Parameters		
Name	***Type***	***Description***
lByteAddress	Long	The offset in memory of the first byte to be read.
lLength	Long	The number of bytes to be read.
vReadData	Variant	Pointer to the data area where the raw (binary) data is stored.
lBlockSize	Long	The maximum amount of data that can be read in a single transfer block. SmartMan will split large card accesses into several smaller blocks with a maximum length of lBlockSize.
sCardResponse	String	Contains a four digit hex string containing the response code from the Smart card. If successful this will contain eSMResults.APDU_OK. If an error occurs, this will be the failing code.
Return Value	Long	eSMResults.SM_CARDOK Success. Others The requested data could not be read from the card.

FUJ00001378

A.7.4 ISmartMan_ReadCard

Reads simple memory cards.

Parameters		
Name	***Type***	***Description***
lAddress	Long	The byte address of the start of the required data.
lLength	Long	The number of bytes to read.
vReadData	Variant	A pointer to a fixed-length byte array to hold the card read. The string must be long enough to hold all the requested data. The first byte of data is copied into vReadData[Address].
lBlockSize	Long	The maximum amount of data that can be read in a single transfer block. SmartMan splits large card accesses into several smaller blocks with a maximum length of lBlockSize.
Return Value	Long	eSMResults.SM_CARDOK Success. eSMResults.SM_CARDERR Faulty card or card removed from reader. eSMResults.SM_READCANCEL Clerk removed card & hit Cancel button. eSMResults.SM_READERR Other Error. eSMResults.SM_SECURITYERR Security key problem.

FUJ00001378

A.7.5 ISmartMan_SelDir

Selects a directory on the Smart Card

Parameters		
Name	***Type***	***Description***
bytDirectory	Byte	The number of the required directory (1 – 255). A value of 255 selects the card directory.
vDataArray	Variant	An 8 byte data array containing data received from the card when the directory is selected.
sCardResponse	String	Contains a four digit hex string containing the response code from the smart card. If successful this contains eSMResults.APDU_OK. In an error occurs, this is the failing code.
Return Value	Long	eSMResults.SM_CARDOK Success. eSMResults.SM_READERR Unable to select directory.

A.7.6 ISmartMan_SelFil

Selects a file on the Smart Card.

Parameters		
Name	***Type***	***Description***
bytFileID	Byte	The file to be accessed on the card in the range 1 – 255.
bytSearchMode	Byte	Mode of search in the range 0 – 3, see GemPlus documentation.
bytLength	Byte	Number of bytes expected. This may only be 4 or 8.
sCardResponse	String	Status code from card.
vDataArray	Variant	This 4 or 8 character array contains information regarding descriptor of the file. Returned from the card, see the GemPlus documentation.
Return Value	Long	eSMResults.SM_CARDOK Success. eSMResults.SM_READERR Unable to select file.

FUJ00001378

A.7.7 ISmartMan_UpdFil

Writes (updates) a file on the Smart Card

Parameters		
Name	***Type***	***Description***
lFileNo	Long	Number of the file to be updated (1 – 255).
lWordAddress	Long	Start address (in words).
lLength	Long	Number of bytes to be written.
vWriteData	Variant	Pointer to buffer area for the write data, starting from byte 0. When writing data to the card from a word offset other than 0, the data written to the card file is an 'image' of the vWriteData array. For example, a write of 4 bytes from offset 4 copies the data in vWriteData to the card-file bytes 16 to 19.
lBlockSize	Long	The maximum amount of data that can be written in a single transfer block. SmartMan will split large card accesses into several smaller blocks with a maximum length of lBlockSize.
sCardResponse	String	Contains a four digit hex string containing the response code from the smart card. If successful this contains eSMResults.APDU_OK. In an error occurs, this will be the failing code.
Return Value	Long	eSMResults.SM_CARDOK Success. Other An error code based on the response from Riposte.

FUJ00001378

A.7.8 ISmartMan_WriteCard

Writes to simple memory card.

Parameters		
Name	***Type***	***Description***
lAddress	Long	The byte address of the start of the data area within the card.
lLength	Long	The number of bytes to write.
vWriteData	Variant	A pointer to a fixed-length byte-string to hold the card read data, stored as raw binary. The first byte of data is written from vWriteData[Address]. For example, to write to byte 255 of the card, the data is copied from vWriteData[255].
lBlockSize	Long	The maximum amount of data that can be written in a single transfer block. SmartMan splits large card accesses into several smaller blocks with a maximum length of lBlockSize.
Return Value	Long	eSMResults.SM_CARDOK Success. eSMResults.SM_CARDERR Faulty card or card removed from reader. eSMResults.SM_SECURITYERR Security key problem. eSMResults.SM_WRITECANCEL Clerk removed card and touched the Cancel button. eSMResults.SM_WRITEERR Other Error.

FUJ00001378

A.8 The ISmartMan Card Definition Methods for Applications

SmartMan provides a number of ISmartMan methods with which the application can establish which CardDLLs support the card, the card technology involved, and the token types supported:

- ISmartMan_GiveCardDLL
- ISmartMan_GiveCardDLLDetails
- ISmartMan_GiveTokenTypes

The other ISmartMan method, RequestSMAction, allows the application to request SmartMan to:

- Deal with transaction status using SmartMan actions:
  - COMPLETE
  - ABANDON
  - TERMINATE
- Deal with cards themselves using SmartMan actions:
  - CANCELCARD
  - CHANGECARD
  - INSERTCARD
  - REMOVE
  - RETRY
- Deal with error using SmartMan actions:
  - CARDDLLERROR
  - DISPLAY

In the definitions that follow, eSMResults is an enumeration supplied by the ISmartMan interface definition. The definitions of byte, string, variant, and long are the standard Microsoft VB conventions.

FUJ00001378

A.8.1 ISmartMan_GiveCardDLL

Returns a reference to the CardDLL associated with the Smart Card application

Parameters		
Name	***Type***	***Description***
sCardDLLKey	String	A string containing the key to the CardDLL as contained in the sCardDLLKey attribute of the SmartCard impulse attribute grammar.
Return Value	String	A reference to the associated CardDLL. If the parameter sCardDLLKey matches, the CardDLL object, the reference is returned. e.g. <DLLName:JCPetrolCard.JCPetrol> If the CardDLL object does not exist or is not associated with the current card application the value *Nothing* is returned.

A.8.2 ISmartMan_GiveCardDLLDetails

Returns a reference to the CardDLL and its associated CardTech.

Parameters		
Name	***Type***	***Description***
sCardDLLKey	String	A string containing the key to the CardDLL for which details are required. This is the same key that is contained in the sCardDLLKey attribute of the SmartCard impulse attribute grammar.
sCardTech	String	Returned set to the Card technology supported by the identified CardDLL.

FUJ00001378

Parameters		
Name	***Type***	***Description***
Return Value	String	A reference to the associated CardDLL. If the parameter sCardDLLKey matches, the CardDLL object, the CardDLL name, is returned and its associated CardTech value. E.g. <CardTech:JCPetrol> <DLLName:JCPetrolCard.JCPetrol> If the CardDLL object does not exist or is not associated with the current card application the value *Nothing* is returned.

A.8.3 ISmartMan_GiveTokenTypes

Returns a list of the Token Types associated with the CardDLL.

Parameters		
Name	***Type***	***Description***
SPropApp	String	A string containing the CardDLL application for which a list of token types is to be returned.
Return Value	String	A list of token types, e.g. <TokenTypes:<WC:1><QC:1><SPM:1>>

FUJ00001378

A.8.4 ISmartMan_RequestSMAction

Provides the current Smart card Application with the means to request defined actions from SmartMan, in the following attribute grammar format:

```
<Result:
    <Success:1>
    <SMCallBack: Application >
    <SMAction: eSMActions member >
    <SMResult: eSMResult member >
>
```

Note that SMResult may not always be present.

In the definitions that follow, eSMResults is an enumeration supplied by the ISmartMan interface definition.

Parameters		
Name	***Type***	***Description***
sSMAction	String	A string identifying the action required, see above. eSMActions.SM_ABANDONED Indicates that SmartMan must inform the Counter Clerk the current transaction has been abandoned because of an error and ask him to remove the card. Once the card has been removed, SmartMan then completes its processing of the transaction by generating a CardRemoved impulse to the associated application. eSMActions.SM_CANCELCARD Indicates that the Counter Clerk has cancelled the current transaction and that the card must be powered off. Once the card has been power off, SmartMan then completes its processing of the transaction by generating a CardRemoved impulse to the associated application. eSMActions.SM_CARDDLLERROR Indicates that SmartMan is to inform the Counter Clerk that the CardDLL associated with the current card transaction could not be loaded and the card must be removed. Once the card has been removed, SmartMan sends a CardAborted impulse to the application.

FUJ00001378

		eSMActions.SM_CHANGECARD Indicates that SmartMan is to ask the Counter Clerk to change the card currently in the reader for the card associated with the current transaction. When this has been completed, SmartMan sends a CardPresent impulse to the application. eSMActions.SM_COMPLETE Indicates that SmartMan must inform the Counter Clerk the current transaction has been completed successfully and ask him to remove the card. Once the card has been removed, SmartMan then completes its processing of the transaction by generating a CardRemoved impulse to the associated application. eSMActions.SM_DISPLAY Indicates that SmartMan must display the message identified by the value of SMResult. When this is acknowledged, SmartMan sends a CardAborted impulse to the application. eSMActions.SM_INSERTCARD Indicates that SmartMan is to ask the Counter Clerk to insert a card to allow the current transaction to proceed. When the card is available, SmartMan generates a CardPresent impulse to the associated application. eSMActions.SM_REMOVE Indicates that SmartMan must power off the card, as it should not have been inserted in the first place. No instructions are displayed to ask for the card to be removed. No impulse is generated as a result of this action. eSMActions.SM_RETRY Indicates that SmartMan is to ask the Counter Clerk to remove, clean and re-insert the card. When this has been completed, SmartMan sends a CardRetried impulse to the application.

FUJ00001378

		eSMActions.SM_TERMINATE Indicates that SmartMan must inform the Counter Clerk the current transaction is being terminated and ask him to remove the card. SmartMan then terminates / aborts the transaction. No impulse is generated as a result of this action.

FUJ00001378

A.9 The CardDLL Card Contents Methods for the Application and SmartMan

There are a number of ISmartCard methods that need to be implemented by each CardDLL to support business and system actions:

All methods of this interface must be implemented. If they are not used, a stub must be provided.

- Support the voiding of existing transaction using:
  - ISmartCard_CancelTransaction
- Support the reversal of existing transaction using:
  - ISmartCard_StartReversal
- Support the recovery of transactions after failure using:
  - ISmartCard_CheckAdditionalInfoValue
  - ISmartCard_GiveReceiptCheckDigits
  - ISmartCard_Prepare
  - ISmartCard_StartRecovery
- Support instructions to update values on the card (dependent on business rules in Reference Data) using:
  - ISmartCard_AdjustCredit
  - ISmartCard_ApplyUpdates
  - ISmartCard_ValidateCredit
- Support instructions to deal with smart card access using:
  - ISmartCard_CardUpdate
  - ISmartCard_CheckCardType
  - ISmartCard_CheckForCardChange
  - ISmartCard_IdentifyCard
  - ISmartCard_VerifyCardIntegrity

In the definitions that follow, eSMResults is an enumeration supplied by the ISmartMan interface definition.

If new methods are required, they must be raised at the *SmartMan Review,* see Appendix C, *System Management.*

FUJ00001378

A.9.1 ISmartCard_AdjustCredit

Provides a means of adjusting the amount of credit to be written back into a card.

Parameters		
Name	***Type***	***Description***
cCreditAmount	Currency	The currency value of the adjustment. May be positive to add credit, or negative to make a refund.
Return Value	Long	eSMResults.SM_CARDOK Success. eSMResults.SM_INVALIDCREDIT The cCreditAmount supplied was invalid.

See section A.2.5.1.3 for financial implications.

A.9.2 ISmartCard_ApplyUpdates

Loads any global, private and tariff updates to the cards data image in memory. Used after the card has been successfully inserted and read, but before any credits adjustments are made

Parameters		
Name	***Type***	***Description***
Return Value	Long	eSMResults.SM_CARDOK Success. eSMResults.SM_GLOBALERR Error obtaining Global Data. eSMResults.SM_PRIVATEERR Error obtaining Private Data. eSMResults.SM_TARIFFERR Error loading Tariff Details.

FUJ00001378

A.9.3 ISmartCard_CancelTransaction

Enables the CardDLL to tidy up after a transaction has been cancelled.

Parameters		
Name	***Type***	***Description***
Return Value	Long	eSMResults.SM_CARDOK Success.

A.9.4 ISmartCard_CardUpdate

Writes all updated data to the Smart card.

Parameters		
Name	***Type***	***Description***
Return Value	Long	eSMResults.SM_CARDOK Success. eSMResults.SM_RETRY Prompts the controlling application to display a screen instructing the clerk to remove, clean and reinsert the card. eSMResults.SM_WRITEFAIL Failed to write data to SmartCard.

A.9.5 ISmartCard_CheckAdditionalInfoValue

Requests the CardDLL to examine each card-specific value retrieved from a transaction receipt and identified by the AdditionalData Reference Data attribute name. Used during recovery.

Parameters		
Name	***Type***	***Description***
sName	String	The AdditionalData Reference Data attribute name of the data value to be validated.
sValue	String	The data value (retrieved from a transaction receipt) to be validated.

FUJ00001378

Parameters		
Name	***Type***	***Description***
sCheckDigit	String	The checksum check digit against which the data is to be validated. An EMPTY_STRING will be supplied if a checksum check digit is not available.
Return Value	Boolean	True: the data value and the checksum, if supplied, are valid. False: the data value or the checksum, if supplied, is invalid.

A.9.6 ISmartCard_CheckCardType

SmartMan calls the CheckCardType function of each CardDLL after the card has been inserted. The CardDLL examines the supplied CardType and Answer-To-Reset parameters to determine if it is can process the card.

Parameters		
Name	***Type***	***Description***
lCardType	Long	Contains the basic type of card, as reported by the Card Reader. Currently defined values are: eSMResults.SM_CARDTYPE_ASYNCT0 eSMResults.SM_CARDTYPE_ASYNCT1 eSMResults.SM_CARDTYPE_SYNCMEMORY
sAnswerToReset	String	For T=0 and T=1 Smart Cards, this contains a string of Hex characters containing the card “Answer To Reset” code.

FUJ00001378

Parameters		
Name	***Type***	***Description***
Return Value	Long	Indicates whether the card can be handled by this DLL. Options are: eSMResults.SM_CARDTYPE_INVALID The card cannot be handled. eSMResults.SM_CARDTYPE_VALID_EXCLUSIVE The card can only be processed by this CardDLL. eSMResults.SM_CARDTYPE_VALID_SHARED The card can be processed by this DLL, and is a multipurpose card that may also be used by another CardDLL.

A.9.7 ISmartCard_CheckForCardChange

Check that the correct card is inserted.

Parameters		
Name	***Type***	***Description***
Return Value	Long	eSMResults.SM_CARDOK Success, the card is correct (ie original card still inserted). eSMResults.SM_CARDCHANGED Inserted card has been changed. eSMResults.SM_CARDUNREADABLE Failed to read inserted card. eSMResults.SM_RETRY Prompts the controlling application to display a screen instructing the clerk to remove, clean and reinsert the card.

FUJ00001378

A.9.8 ISmartCard_GiveReceiptCheckDigits

Retrieves check digits generated by the CardDLL.

Parameters		
Name	***Type***	***Description***
Return Value	String	Any check digits the CardDLL has generated for card-specific data values printed on transaction receipts.

A.9.9 ISmartCard_IdentifyCard

Reads identification data from the card; called by the application to the CardDLL.

Parameters		
Name	***Type***	***Description***
bRetry	Boolean	True: The card insertion is a reinsertion of the same card. False: The card insertion is for a different card.
sCardClientID	String	A string to hold the Card Client ID number as read from the card data.
Return Value	Long	eSMResults.SM_CARDOK Success. eSMResults.SM_CARDCORRUPT The card is corrupt and cannot be rebuilt. eSMResults.SM_CARDUNREADABLE Cannot read the card (broken, security etc.). eSMResults.SM_CARDUNSUPPORTED The card variant is not supported. eSMResults.SM_ERROR A non-specific error has occurred. eSMResults.SM_RETRY Prompts SmartMan to display a screen instructing the clerk to remove, clean and reinsert the card. eSMResults.SM_SECURITYERR Security related problem.

FUJ00001378

A.9.10 ISmartCard_Prepare

Called by SmartMan after a CardDLL has been identified, or by the application at the start of a recovery transaction, to load internal variables and data caches.

Parameters		
Name	***Type***	***Description***
ObjSmartMan	ISmartMan	A pointer to the SmartMan class. This can be used as a reference to SmartMan to access any of its public methods and properties.

A.9.11 ISmartCard_StartRecovery

Used to supply details of the token and product associated with the transaction that is about to be recovered, and its timestamp.

Parameters		
Name	***Type***	***Description***
objToken	ISmartToken	A controlling application dependent object through which the token Reference Data can be accessed.
objProduct	ISmartProduct	A controlling application dependent object through which the product Reference Data can be accessed.
sTimeStamp	String	The original timestamp associated with the transaction being recovered formatted as "dd/mm/yy hh:mm:ss".
sAdditionalInfo	String	The AdditionalInfo associated with the transaction being recovered. This may be an empty string.
sCardClientID	String	The CardClientID associated with the transaction being recovered.
sCustomerID	String	The CustomerID associated with the transaction being recovered.
bRecoveringReversal	Boolean	True: the transaction being recovered is a reversal. False: the transaction being recovered is a normal transaction.

FUJ00001378

A.9.12 ISmartCard_StartReversal

Used to supply details of the token and product associated with the transaction that is about to be reversed, and its timestamp.

Parameters		
Name	***Type***	***Description***
objToken	ISmartToken	A controlling application dependent object through which the token Reference Data can be accessed.
objProduct	ISmartProduct	A controlling application dependent object through which the product Reference Data can be accessed.
sTimeStamp	String	The original timestamp associated with the transaction being reversed formatted as "dd/mm/yy hh:mm:ss".
sAdditionalInfo	String	The AdditionalInfo associated with the transaction being reversed.
Return Value	Long	eSMResults.SM_CARDOK The reversal process can proceed. eSMResults.SM_CARDCHANGED The contents of the card does not match the information recorded in the transaction being reversed.

FUJ00001378

A.9.13 ISmartCard_ValidateCredit

Checks that it is permitted to update the card with the specified credit value. The actual credit value on the card should not be altered.

Parameters		
Name	***Type***	***Description***
cCreditAmount	Currency	The currency value of the adjustment. It may be positive to add credit or negative to make a refund.
cMinAllowed	Currency	Should be set by CardDLL to the minimum currency value which may be legally credited to the card.
cMaxAllowed	Currency	Should be set by the CardDLL to indicate the maximum currency amount that can be credited to the card *at this time*.
cMultipleAllowed	Currency	Should be set by the CardDLL to the credit multiple currency value, i.e. if the card may only be credited a whole number of pounds, set this parameter to 1.00.
Return Value	Long	eSMResults.SM_CARDOK Success, the card can be updated with the identified credit. eSMResults.SM_INVALIDCREDIT Failed, the card cannot be updated with the identified credit.

FUJ00001378

A.9.14 ISmartCard_VerifyCardIntegrity

Obtains the client configuration data for the card, and verifies card integrity. Performs card recovery if needed.

Parameters		
Name	***Type***	***Description***
objToken	ISmartToken	A controlling application dependent object through which the token Reference Data can be accessed.
objProduct	ISmartProduct	A controlling application dependent object through which the product Reference Data can be accessed.
sTimeStamp	String	The timestamp associated with the transaction being performed formatted as "dd/mm/yy hh:mm:ss".

FUJ00001378

Return Value	Long	ESMResults.SM_CARDOK Success. eSMResults.SM_CARDINCREDIT Failed, the card already contains credit and cannot therefore be updated. eSMResults.SM_CARDUNREADABLE Cannot read the card (broken, security). eSMResults.SM_CARDUNSUPPORTED The card variant is not supported. eSMResults.SM_CLIENTUNSUPPORTED The client is not supported at this office. eSMResults.SM_CREDITADJUSTONLY Card is corrupt and can only have its credit adjusted. eSMResults.SM_REBUILDNOTAVAIL Rebuild needed but not allowed for this client. eSMResults.SM_RETRY Prompts SmartMan to display a screen instructing the clerk to remove, clean and reinsert the card. eSMResults.SM_TARIFFERR Unable to locate tariff data. eSMResults.SM_WRITEFAIL Card cannot be written to.

FUJ00001378

A.10 The CardDLL Card Definition Properties for EPOSS Transaction Data

There are a number of ISmartCard properties that each CardDLL must provide to support the provision of standard Reference Data values from the card for EPOSS transactions. These are:

- ISmartCard_CardClientID
- ISmartCard_CardTech
- ISmartCard_CurrentCredit
- ISmartCard_CustomerID
- ISmartCard_MeterSerial
- ISmartCard_ReturnData
- ISmartCard_ReversalAllowed
- ISmartCard_TransactionValid

In the definitions that follow, eSMResults is an enumeration supplied by the ISmartMan interface definition.

All methods of this interface must be implemented. If they are not used, a stub must be provided.

A.10.1 ISmartCard_CardClientID

Returns the Card Client ID, as held in the card data.

Parameters		
Name	***Type***	***Description***
Return Value	String	A string containing the Card Client ID number. This should not contain any embedded spaces, but may include leading zeros, e.g. “0095”. This is not the same as the POCL client ID, which the controlling application obtains by searching Reference Data for a match with CardTech and CardClientID.

FUJ00001378

A.10.2 ISmartCard_CardTech

Returns a string containing the card type.

Parameters		
Name	***Type***	***Description***
Return Value	String	A string containing the Card Type.

A.10.3 ISmartCard_CurrentCredit

Returns the amount of credit as a currency value actually on the card, it **does not** include any pending credits or debits.

Parameters		
Name	***Type***	***Description***
Return Value	Currency	Credit value originally on the card as a currency value, plus any adjustments made by updates to the card via ISmartCard_CardUpdate.

A.10.4 ISmartCard_CustomerID

Returns the customer ID, as held in the card data.

Parameters		
Name	***Type***	***Description***
Return Value	String	A string containing the customer ID number. This should not contain any embedded spaces, but may include leading zeros.

A.10.5 ISmartCard_MeterSerial

Returns the serial number of the card's related meter.

Parameters		
Name	***Type***	***Description***
Return Value	String	A string containing the meter serial number.

FUJ00001378

A.10.6 ISmartCard_ReturnData

Obtains all AdditionalInfo data to be returned to the client.

Parameters		
Name	***Type***	***Description***
Return Value	String	An attribute string containing all data to be returned to the client as AdditionalInfo. Binary data should be represented as a hex string, i.e. the 4-byte value &HF032E23A would be represented as the string “F032E23A”.

A.10.7 ISmartCard_ReversalAllowed

Determines whether the previous credit can be reversed.

Parameters		
Name	***Type***	***Description***
Return Value	Boolean	True: the last transaction can be reversed. False: the last transaction cannot be reversed.

The topic of reversals is covered in section 4.2.3 of the main document.

A.10.8 ISmartCard_TransactionValid

Indicates whether processing the card should generate a message store transaction and receipt.

Parameters		
Name	***Type***	***Description***
Return Value	Boolean	True for a valid transaction, otherwise False.

FUJ00001378

A.11 The ISmartToken Interface

This interface is provided to satisfy the need for access to Token Reference Data. This interface is implemented by the application and is not used by SmartMan itself. It provides the interface needed by the CardDLL for access to the POCL EPOSS Transaction Reference Data that applies to the card.-

The ISmartToken interface provides the type of *Token* object supplied by a Smart card Application via the ISmartCard_StartRecovery, ISmartCard_StartReversal and ISmartCard_VerifyCardIntegrity methods, see above.

In the definitions that follow, eSMResults is an enumeration that should be supplied by the ISmartToken interface definition.

All methods of this interface must be implemented. If they are not used, a stub must be provided.

A.11.1 ISmartToken_PFData

Retrieves details of the pre-filled data attributes associated with a token.

Parameters		
Name	***Type***	***Description***
SItem	String	Identifies the Prefilled Data attribute for which the attribute value is to be retrieved. Valid values of sItem are: CA, CID, CN, SG, SVC, TT, TI and TV
Return Value	String	The value of the Prefilled Data attribute identified above or an empty string if the attribute does not exist for the token.

A.11.2 ISmartToken_ProductNo

Retrieves the value of the ProductNo attribute associated with a token.

Parameters		
Name	***Type***	***Description***
Return Value	String	The value of the ProductNo attribute for a token.

FUJ00001378

A.11.3 ISmartToken_SchemeName

Retrieves the value of TN attribute associated with a token.

Parameters		
Name	***Type***	***Description***
Return Value	String	The value of the TN attribute for a token.

A.11.4 ISmartToken_SmartData

Retrieves values of SmartData attributes associated with a token.

Parameters		
Name	***Type***	***Description***
sItem	String	Identifies the specific SmartData attribute for which the value is to be retrieved. Valid values of sItem are: MaxCrLim, MCAv, PBAv, RVAv and CRAv.
Return Value	String	The value of the SmartData attribute for a token.

A.11.5 ISmartToken_TokenName

Retrieves the name of a token (the value of ObjectName attribute associated with the token).

Parameters		
Name	***Type***	***Description***
Return Value	String	The token name.

A.11.6 ISmartToken_TokenType

Retrieves the value of PFData.TT attribute associated with a token.

Parameters		
Name	***Type***	***Description***
Return Value	String	The value of the PFData.TT attribute for a token.

FUJ00001378

A.12 The ISmartProduct Interface

This interface is provided to satisfy the need for access to Product Reference Data. This interface is implemented by the application and is not used by SmartMan itself. It provides the interface needed by the CardDLL for access to the POCL Product Reference Data that applies to the card.

The ISmartProduct interface provides the *Product* object supplied by a Smart card Application via the ISmartCard_StartRecovery, ISmartCard_StartReversal and ISmartCard_VerifyCardIntegrity methods.

In the definitions that follow, eSMResults is an enumeration that should be supplied by the ISmartProduct interface definition.

All methods of this interface must be implemented. If they are not used, a stub must be provided.

A.12.1 ISmartProduct_AdditionalData

Retrieves details of the additional data attributes associated with a product.

Parameters		
Name	***Type***	***Description***
sName	String	Identifies the specific piece of Additional Data, defined by AdditionalData.N, for which the attribute values are to be retrieved.
sItem	String	Identifies the specific Additional Data attribute for which the value is to be retrieved. Valid values of sItem are: A, C, D, F, Max, Min, N, P, S, SD, O and VM.
Return Value	Variant	The value of the Additional Data attribute identified above or an empty string if the attribute does not exist for the product.

A.12.2 ISmartProduct_MaxValue

Retrieves the value of the MxV attribute associated with a product.

Parameters		
Name	***Type***	***Description***
Return Value	String	The value of the MxV attribute for a product.

FUJ00001378

A.12.3 ISmartProduct_MinValue

Retrieves the value of MnV attribute associated with a product.

Parameters		
Name	***Type***	***Description***
Return Value	String	The value of the MnV attribute for a product.

A.12.4 ISmartProduct_MultipleValue

Retrieves the value of MV attribute associated with a product.

Parameters		
Name	***Type***	***Description***
Return Value	String	The value of the MV attribute for a product.

FUJ00001378

A.13 The ISmartApp Interface

This interface is provided to satisfy the need for additional data capture defined by POCL Product Reference Data. This interface is implemented by the application and is not used by SmartMan itself. It provides the interface needed by the CardDLL to supply the application with Additional Info changed several times during a call to a CardDLL method.

The ISmartApp interface also provides the *Card Application* object supplied via the ISmartCard_StartRecovery, ISmartCard_StartReversal and ISmartCard_VerifyCardIntegrity methods.

All methods of this interface must be implemented. If they are not used, a stub must be provided.

A.13.1 ISmartApp_WriteTransactionData

Enables a CardDLL to pass details of the AdditionalInfo attribute grammar to be included in a transaction written on behalf of the CardDLL when the AdditionalInfo changes several times during the execution of an ISmartCard method (in particular, ISmartCard_CardUpdate).

Parameters		
Name	***Type***	***Description***
sAdditionalInfo	String	Identifies the attribute grammar to be included as the AdditionalInfo attribute in the AdditionalData attribute of a transaction written on behalf of the CardDLL.