



**STATE OF TEXAS  
SPECIAL SUPPLEMENTAL NUTRITION PROGRAM for  
WOMEN, INFANTS, & CHILDREN (WIC)**

**WIC Electronic Benefits Transfer (EBT) /  
Electronic Service Delivery (ESD)**

**WIC Smart Card  
Interoperability Specification**

**Grocer and Clinic  
Electronic Benefit Transfer Systems**

**Version 2.5  
July 11, 2003**

*Published and Maintained by:*

EBT Planning Division  
Texas Department of Health  
1100 West 49<sup>th</sup> Street  
Austin, TX 78756-3199  
(512) 458-7444

[WICEBTDeveloper@tdh.state.tx.us](mailto:WICEBTDeveloper@tdh.state.tx.us)  
<http://www.tdh.state.tx.us/wichd/ebt/ebt1.htm>

# TABLE OF CONTENTS

TABLE OF CONTENTS.....	I
1. PREFACE.....	1
1.1 BACKGROUND.....	1
1.2 SCOPE, LIMITATIONS, AND APPLICABILITY OF THE SPECIFICATION.....	3
1.3 CONFORMING TO THE SPECIFICATION.....	3
3. ARCHITECTURAL MODEL.....	5
3.1 OVERVIEW.....	5
3.2 ARCHITECTURAL CONSIDERATIONS.....	6
3.3 RETAILER APPLICATION.....	7
3.4 WIC SERVICES PROVIDER INTERFACE (WSPI).....	7
3.5 READER DRIVER INTERFACE (RDI).....	7
3.6 CRYPTOGRAPHIC SERVICES INTERFACE (CSI).....	7
3.7 VOC DATA MODEL.....	7
3.8 CARD CAPABILITIES CONTAINER (CCC).....	8
3.9 WIC MESSAGING PROTOCOL (WMP).....	8
3.10 WIC SERVICES MANAGER (WSM).....	8
4. ACCESS CONTROL MODEL.....	9
4.1 OVERVIEW.....	9
5 WIC SERVICES PROVIDER INTERFACE (WSPI).....	10
5.1 OVERVIEW.....	10
5.2 CARD CONTROL FUNCTIONS.....	10
5.2.1 <i>wspiInitiateSession()</i> .....	10
5.2.2 <i>wspiConnectCard()</i> .....	11
5.2.3 <i>wspiGetPAN()</i> .....	11
5.2.4 <i>wspiCardPresent()</i> .....	12
5.2.5 <i>wspiDisconnectCard()</i> .....	12
5.2.6 <i>wspiEndSession()</i> .....	12
5.3 RETAILER DATA FUNCTIONS.....	13
5.3.1 <i>wspiAuthenticateUser()</i> .....	13
5.3.2 <i>wspiReadBalance()</i> .....	13
5.3.3 <i>wspiDebitBalance()</i> .....	14
5.3.4 <i>wspiBlockCard()</i> .....	15
5.3.5 <i>wspiPostStateAuthorizedTransaction()</i> .....	15
5.4 WICITEMSSTRUCTURE.....	17
5.5 WSPI RETURN CODES.....	18
6 READER DRIVER API (RDI).....	20
6.1 OVERVIEW.....	20
6.2 READER DRIVER FUNCTIONS.....	20
6.2.1 <i>rdiConnectReader()</i> .....	20
6.2.2 <i>rdiCardPresent()</i> .....	21
6.2.3 <i>rdiBeginTransaction()</i> .....	21
6.2.4 <i>rdiExchangeAPDU()</i> .....	21
6.2.5 <i>rdiEndTransaction()</i> .....	22
6.2.6 <i>rdiDisconnectReader()</i> .....	23
7 CRYPTOGRAPHIC SERVICES API (CSI).....	24
7.1 OVERVIEW.....	24
7.2 CRYPTOGRAPHIC SERVICES FUNCTIONS.....	24

7.2.1	<i>csiInitiateService()</i> .....	24
7.2.2	<i>csiGetSecurityData()</i> .....	24
7.2.3	<i>csiCheckSecurityData()</i> .....	25
7.2.4	<i>csiEndService()</i> .....	26
8	WIC DATA MODEL .....	27
8.1	OVERVIEW.....	27
8.2	VERIFICATION OF CERTIFICATION DATA CONTAINER.....	27
9	CARD CAPABILITY CONTAINER (CCC) .....	29
9.1	OVERVIEW.....	29
9.2	PROCEDURE FOR ACCESSING THE CCC.....	29
9.2.1	General CCC Retrieval Sequence.....	29
9.2.2	CCC Retrieval Sequence for ISO file system cards.....	29
9.3	CCC STRUCTURE .....	30
9.3.1	CCC Data Layout .....	30
9.3.2	CCC Element Descriptions .....	31
9.3.3	Construction of Capability Tuples .....	31
9.3.4	Example Commands.....	33
9.3.5	Example CCC Contents .....	34
10	GLOSSARY .....	36
	APPENDICES .....	38
	APPENDIX A. WIC MESSAGING PROTOCOL (WMP).....	39
A.1	OVERVIEW.....	39
A.2	GENERIC SERIAL COMMUNICATION .....	39
A.3	SETUP PROCESSING .....	39
A.3.1	ECR Establish Active WSPM Request.....	39
A.3.2	Terminal Validate Active WSPMs Response .....	40
A.4	TRANSACTION PROCESSING .....	40
A.4.1	Get PAN .....	40
A.4.2	Read Balance .....	41
A.4.3	Debit Balance.....	42
A.4.4	Block Card .....	42
A.4.5	End Transaction.....	43
A.4.6	Authenticate User.....	43
A.4.7	Shutdown processing.....	43
A.5	LONGITUDINAL REDUNDANCY CHECK (LRC) CHARACTER .....	44
A.6	MESSAGE FLOW DIAGRAM .....	44
	APPENDIX B. SECURITY POLICY RECOMMENDATION.....	47
	APPENDIX C. WIC SERVICES MANAGER (WSM).....	53
C.1	OVERVIEW.....	53
C.2	WSM FUNCTIONS .....	53
C.2.1	<i>wsmInitiateSession()</i> .....	53
	APPENDIX D. REFERENCE MATERIAL .....	55

# 1. Preface

## 1.1 Background

This specification defines interoperability for clinic and food retailer systems that accept smart cards as the benefit delivery mechanism for participants in the Special Supplemental Nutrition Program for Women, Infants and Children, commonly referred to as the WIC Program. Interoperability is defined as the ability for WIC clinic card readers and food retailer card readers and associated payment and electronic cash register systems to accept and transact benefits from smart cards issued by different States. It does not allow WIC participants to purchase food items in all States, which is the case with Food Stamp EBT interoperability.

The WIC Program is a Federal grant program, not an entitlement program for which funds are set aside so every eligible individual can participate. Each year, Congress authorizes a specific amount of funds for WIC to serve a target population of low-income, nutritionally at risk:

- pregnant women, through pregnancy and up to 6 weeks after birth or after pregnancy ends;
- breastfeeding women, up to an infant's 1<sup>st</sup> birthday;
- non breastfeeding, postpartum women, up to 6 months after the birth of an infant or after pregnancy ends;
- infants, up to the 1<sup>st</sup> birthday; and
- children, up to their 5<sup>th</sup> birthday.

WIC funds are administered by the U.S. Department of Agriculture, Food and Nutrition Service and by 87 WIC authorities, including 50 State health departments, 32 Indian Tribal Organizations, and American Samoa, District of Columbia, Guam, Puerto Rico and the U.S. Virgin Islands. WIC serves over 45% of all infants born in the United States, through more than 2,000 local agencies and 10,000 clinic sites. WIC services are provided through approximately 46,000 authorized retailers. The WIC program provides a variety of services to participants, including supplemental nutritious foods, nutrition education and counseling, and screening and referrals to other health, welfare and social services.

A typical configuration for a smart card system consists of a host computer with one or more smart card readers attached via hardware communications ports. Smart cards can be inserted into the card readers, and software running on the host computer communicates with these cards using a protocol defined by ISO 7816-4 [ISO4]. The ISO standard smart card communications protocol defines Application Protocol Data Units (APDU) that are exchanged between smart cards and host computers. This APDU based interface is referred to as the card edge, and the two terms are used interchangeably.

Client applications have traditionally been designed to communicate with ISO smart cards using the APDU protocol through low-level software drivers that provide an APDU transport mechanism between the client application and a smart card. Smart card families can implement the APDU protocol in a variety of ways, so client applications must have intimate knowledge of the APDU set of the smart cards they are communicating with. This is generally accomplished by programming a client application to work with a specific card, since it would not be practical to design a client application to accommodate the different APDU sets of a large number of smart card families.

The tight coupling between client applications and smart card APDU sets has several drawbacks for the ISO card technology/s use in WIC, both in WIC-authorized retail stores and in local

agencies and clinics. Application programmers must be thoroughly familiar with smart card technology and the complex APDU protocol. If the cards, that an application is hard coded to use, become commercially unavailable, the application must be redesigned to use different cards. Customers also have less freedom to select different smart card products, since their applications will only work with one or a small number of similar cards.

This WIC Smart Card Interoperability Specification (WSC-IS) provides solutions to a number of the interoperability problems associated with smart card technology. For national and regional grocers authorized by more than one state WIC authority, this specification substantially reduces the level of effort (programming) required to integrate WIC functionality within existing store electronic cash register (ECR) systems and in-lane terminals. Grocers authorized to accept WIC in more than one state will be able to 'program once' all core WIC EBT retail system functionality and the card-to-terminal interface specification; the programming required to accept WIC cards issued by another state(s) will be greatly reduced, not individualized programming per additional WIC state. For state WIC authorities, the use of this specification will similarly reduce the level of effort, including programming, required to be able to access Verification of Certification (VOC) data stored on the card of a participant moving from another WIC EBT state. This specification is viewed as a major 'first step' to making offline WIC EBT affordable, helping grocers make the Business Case for introducing and maintaining a new technology within their stores, and assisting states to ensure WIC services go uninterrupted when a WIC participant moves from/to a WIC EBT state.

The original version of the WSC-IS (version 1.0) and subsequent versions through version 2.5 were developed by the Offline WIC EBT Interoperability Specifications workgroup during meetings in Dallas, Texas sponsored by the U.S. Department of Agriculture, Food and Nutrition Services, and hosted by the Texas Department of Health, Special Supplemental Nutrition Program for Women, Infants and Children. MAXIMUS Intelligent Technologies Division, in cooperation with Booz Allen Hamilton, consulting firms supporting TDH and USDA/FNS, respectively, facilitated the development of this specification.

The following list of participants and organizations participated:

Full Name	Organization	Full Name	Organization
Brian Abair	Minyard Food Stores	Ted Mason	Food Marketing Institute
Marc P. Abramson	Citicorp EFS	Robin Williamson McBrearty	Office of Comm. & Public Health-WIC
Dennis Bach	Burger, Carroll, & Associates	Erin McBride	USDA-FNS
Thomas J. Barr	MAXIMUS	Donald J. McGillivray	VeriFone, Inc.
Rich Barsky	Verifone	Ketan Mehta	Booz Allen & Hamilton, Inc.
Stan Bien	MI Dept of Community Health-WIC	Mike Montgomery	Texas Department of Health
Mark Bond	Grocers Supply	Janet Moran	Wyoming Dept of Health-WIC
Bob Bucceri	New England Partners	Edward Oppenheimer	Booz-Allen & Hamilton, Inc.
Arthur W. Burger	Burger, Carroll & Associates	James Peterson	Innovax Corporation
Teresa Byrd	Brookshire Grocery	Jerry Pierce	NM Dept. of Health
John Carroll	Carroll-Kron Consulting, Inc.	Debbie Platts	Albertsons, Inc.
Pat Daniels	USDA-FNS	Norman Plourde	Kincaid Technologies
Mike Delagrance	GemPlus	Art Powell	Fujitsu Transaction Solutions, Inc.
Hank Dembosky	GovConnect	Sondra Ralph	USDA-FNS-SWRO
Kadie Donahoe	Ohio Dept. of Health	Joe Ratcliff	TX Grocery & Convenience Assn.
Jane Duffield	USDA-FNS	Joe Reeder	Cash Register Services
Denise Fedewa	MI Dept of Community Health-WIC	Mike Roberts	Stored Value Systems
Mark Fischer	H.E.B.	Gerald Schoenecker	Hypercom, Inc.
Joe Fogarty	Grocers Supply	Laura Secondo	GovConnect, Inc.
Len Fuller	Cash Register Services, Inc.	Johnny Sena	Stored Value Systems
David Fuller	Cash Register Services, Inc.	Joni Sherry	WY Dept. of Health-WIC Pgm
Jim Garafalo	The Kroger Company	Shenny Sheth	Texas Department of Health
Steve Geist	NCR Corporation	Mark Sloan	Innovax Corporation
Mike Godfrey	Stored Value Systems	Dick Snyder	Wal-Mart
Sid Golden	New Mexico Dept of Health-WIC	Phil Swain	USDA-FNSSWRO
Duane Grabarschick	Brookshire Brothers Ltd.	Penny Tisdale	MAXIMUS
Christina Graff	USDA FNS - NERO	Lori Turner	Wal-Mart
Joe Graves	Texas Department of Health	Roger F. Van Brussel	eMedicalFiles, Inc.
Monte Green	Verifone	Lyn VanRaden	Burger, Carroll, & Associates
Donna Hammond	MAXIMUS	Sunil Vasudevan	GemPlus
Cheryl Hines	Lowe's Pay N Save	Noel Villarreal	TX Dept of Human Services
Chris Hoff	MAXIMUS	Sam von Bose	Wal-Mart Stores, Inc.
Rick Joslin	Wyoming WIC	Lee Waddell	IBM
Won Jun	Giesecke & Devrient	Eric Weber	Biometrics Access Corp.
Dan Kannady	Verifone	Fred Wee	Albertsons
Mike Keck	Kroger Company	Tammy Welch	Wal-Mart Stores, Inc.
William C. Kincaid	Kincaid Technologies	Thomas F. Wenning	National Grocers Association
Julie Kresge	USDA-FNS	Curtis Wiggins	Brookshire Grocery Company
Harry Kron	Carroll-Kron Consulting	Joe Williams	Gulf Coast Retailers Assn.
Ray Krzesniak	Texas Department of Health	Chuck Wilson	Hitachi America
Bradley K. Lee	New England Partners	Mary Alice Winfree	Texas Department of Health
Dan Mandeville	Innovax Corporation	Virginia Wise	IBM-RSS
Jane Marlow	Wal-Mart Stores, Inc.		

## 2.2 Scope, Limitations, and Applicability of the Specification

The Offline WIC-EBT Interoperability Specification defines two services: First, it defines a common API to access the WIC benefits on a smart card in the retailer environment. This will allow development of retailer applications for nationwide use with minimal modification required to account for interstate differences. Second, it defines the way to transport a common set of Verification of Certification (VOC) data in a secure manner to allow the WIC participant to maintain continuity of benefits in the event they move to another clinic (intra-state or inter-state).

WIC Interoperability is not addressed for card initialization or the manner in which WIC-EBT benefits are arranged on the card itself. It is important to note that neither the movement of security data nor key management are defined within this specification.

## 2.3 Conforming to the Specification

A WIC-EBT smart card is in compliance with this specification if it fulfills the following requirements:

1. The card shall contain a Card Capability Container (CCC) as described in Section 9.
2. The card shall contain Verification of Certification data as described in Section 8.

3. The card shall use the WIC-EBT RID as the first five bytes of all “on card AIDs” for any WIC-EBT related files or applets.

A WIC Service Provider Module (WSPM) is in compliance with this specification if it fulfills the following requirements:

1. The WSPM shall implement all of the functions of the WIC Services Interface as specified in Section 5.
2. The WSPM shall utilize the Reader Driver Module (RDM) as specified in Section 6.
3. The WSPM shall utilize the Cryptographic Services Module (CSM) as described in Section 7.

A WIC Retailer application is in compliance if it uses the WSPI as defined in this specification. In addition, it is suggested that the Retailer Application architecture be as presented in Figure 3.2.

While these requirements are mandatory for compliance, WIC-EBT systems developed before this specification is implemented should be allowed to continue, at USDA/FNS discretion. It is assumed these systems will evolve and eventually comply with this specification.

## **3. Architectural Model**

### **3.1 Overview**

The purpose of the Offline WIC-EBT Interoperability Specification is twofold. First, to provide a way to minimize the amount of work required by the retailers to handle cards from different states. Second, to provide a mechanism for WIC participants to transport Verification of Certification (VOC) data securely across state lines and thus ensure continuity of benefits. The first purpose is fulfilled by a set of defined interfaces that allow the retailer and the state to modularize the code that applies to their area of responsibility. The interfaces defined in this document are the WIC Services Provider Interface (WSPI), the Reader Driver API (RDI), and the Cryptographic Services API (CSI). This document also outlines a WIC Services Manager (WSM) layer that may be used to enable a system to read cards from multiple states and a WIC Messaging Protocol (WMP) that may be used to simplify the interaction between the ECR and the Terminal.

The WSPI specification in Section 4 allows retailers and WIC clinics to communicate with WIC smart cards from their home state as well as provide VOC data interoperability with other states cards at the WIC clinics. The WSPI is the upper edge of the WIC Service Provider Module (WSPM), which is the software created by the individual states to communicate with their smart card. The card layout specification is designed to both accommodate interstate interoperability with regard to reading basic demographic information and allow the states flexibility in designing their own layout with regard to the WIC EBT specific card data. To this end, the card data is divided into two sections, VOC data and EBT Benefit Information (EBT-BI) data.

The VOC data is specified to allow read interoperability between the various states. The VOC data contains WIC participant demographic and local WIC office information as well as participant benefit certification information. All of this data may be read by any state with the submission of the cardholder PIN. Access conditions to control updating this information are controlled by the issuing state and are not covered in this specification.

The EBT-BI data arrangement and security is a matter for each state to specify. None of this data is required to be interoperable between states, although it could be via agreements between states. However, the WSPI-specified methods of controlling the EBT-BI data are modeled after card purse operations, i.e. Read Balance and Debit. The state designed layer may implement these on the card however they wish but they will be treated as purses by the retailer, via the WSPI. Security for this data, both reading and writing, will be contained within the WSPM layer, thus absolving the retailer of any need to know the card keys or other security details.

The RDI specification allows the WSPM to communicate with the smart card regardless of the technology chosen by the retailer by developing the lower edge of the WSPS and the upper edge of the reader driver to the same RDI. The retailer is responsible for developing the RDI software to match whatever hardware choice they have made.

The CSI specification provides a mechanism for the WSPM to obtain secret codes or cryptograms for submission to the card while allowing the retailer flexibility in his key management implementation. The retailer is responsible for developing the CSI software in accordance with the state security guidelines and whatever security infrastructure they are willing to support.

The WIC Messaging Protocol (WMP) is proposed as a simple method for the ECR to conduct WIC transactions with the terminal. It is presented here as a suggestion for those implementations that separate the ECR from the Terminal/CAD package.

The WIC Services Manager (WSM) layer is proposed as an optional layer on top of the WSPI interface that handles switching between each of the WSPMs. The implementation of this layer is left to the retailer.

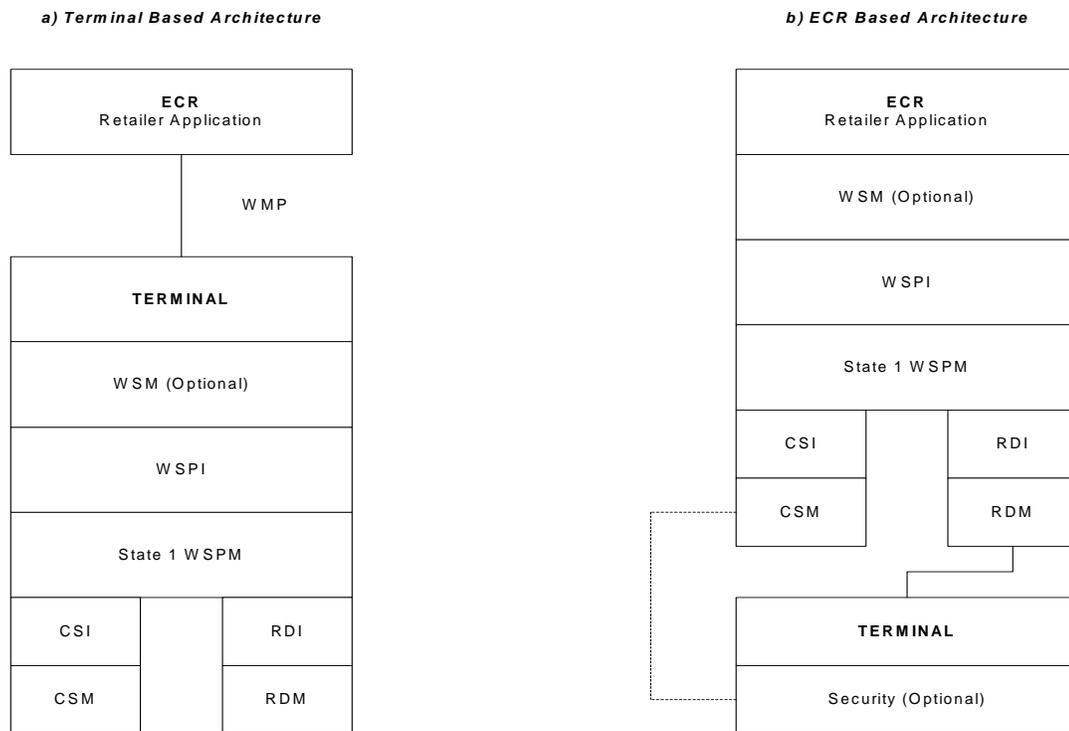
### 3.2 Architectural Considerations

The WIC EBT interoperability specifications defined in this document are independent of the retailer hardware architecture. However, these specifications will impact the ECR and the terminal devices as the actual implementation of the specifications will depend on the retailer environment. Therefore, this document presents two notional, most apparent, architectures for implementing WIC EBT interoperability specifications. The actual implementation may be consistent with, a hybrid of the two, or a deviation from these architectures.

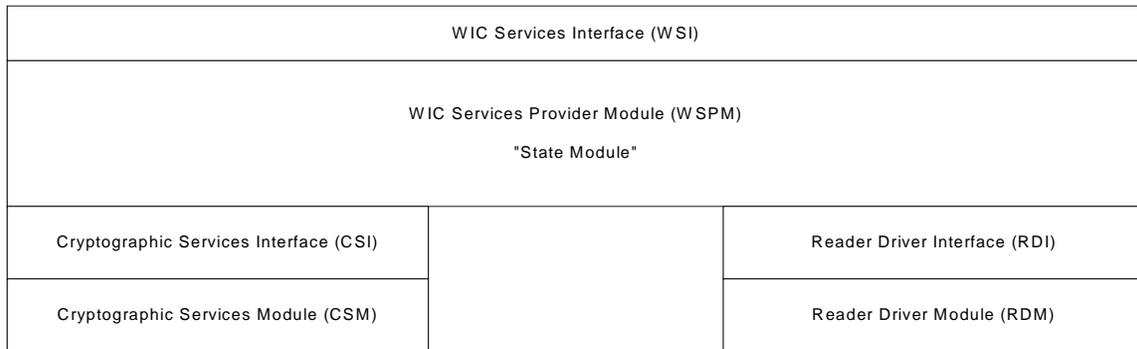
Figure 3.1a) depicts the WSPI implemented on the terminal. In this case, the terminal is equipped with logic to perform necessary card related activities and provide the results to the ECR through a retailer specific messaging protocol. The terminal communicates to the card using WSPI functions and to the ECR using the messaging protocol. This protocol could be implemented in the form of a communication protocol or a scripting language depending on the retailer hardware.

Figure 3.1b) depicts the WSPI implemented on the ECR. In this case, the terminal is a pass through device that receives instructions from the ECR and presents them to the card. The state specific WSPM(s) are loaded and executed on the ECR while the terminal facilitates communication to the card.

The components of the interoperability architecture reside either on the ECR or in the terminal. All the components of the architecture are required except the WMP and WSM. The WMP is applicable if the ECR and terminal communicate via a messaging protocol and the WSM is applicable if the architecture must support multiple states. The following sections further define each component of the architecture.



**Figure 3.1: Examples of Retailer Application Architectures**



**Figure 3.2: The WSPM Layer Close up**

### 3.3 Retailer Application

The retailer application implements functions related to WIC-EBT processes. The WIC-EBT functions, such as scanning items, communicating with the terminal, checking WIC eligible items, hot card lists, maximum prices and shelf prices, printing receipts, displaying messages, selecting transaction and managing user interfaces, are implemented in this component. Depending on the architecture, the entire retailer application may be implemented on the ECR or split between the ECR and the terminal. The retailer application implemented on the terminal provides for the user interface, message display, PIN entry, and transaction selection.

### 3.4 WIC Services Provider Interface (WSPI)

Each version of the WSPM will fully implement the WSPI. The WSPI commands are divided into two groups: (1) Card Control commands and; (2) User Data commands. The Card Control commands are for system initialization and card use activation while the User Data commands are for manipulating the data on the smart card. The functions associated with this interface are described in Section 5. The User Data commands include the WIC VOC functions that access the VOC data in the WIC Data Model, and the WIC EBT functions, which access the EBT data in the WIC Data Model.

### 3.5 Reader Driver Interface (RDI)

As the selection of the card reader hardware is left up to each retailer, the retailer is responsible for developing the reader driver that will work with the hardware. Each retailer's version of the reader driver will implement the RDI. The RDI functions described in Section 6 of this document are the minimum set of functions that are required to communicate with the smart card.

### 3.6 Cryptographic Services Interface (CSI)

While the minimum key security requirements are set by the state, the implementation of those requirements is left up to each retailer. Each retailer is responsible for developing a Cryptographic Services Module for use by the WSPM via the CSI. This will allow the WSPM to have access to keys necessary for the WIC-EBT functions while allowing the retailer flexibility in its implementation. The CSI is described more fully in Section 7.

### 3.7 VOC Data Model

In order to facilitate the transfer of a WIC client from one state to another, a WSC-IS compliant card shall contain the VOC data container in the format as described in Section 8 of the this specification.

### **3.8 Card Capabilities Container (CCC)**

Each WSC-IS compliant card shall have a Card Capabilities Container (CCC) described in Section 9 of this document. To facilitate interstate interoperability for reading VOC data, capability tuples are provided to enable reading of the VOC data.

### **3.9 WIC Messaging Protocol (WMP)**

This optional protocol is modeled upon the messaging layer used in the Texas WIC project. It is designed to remove card related functions from the purview of the ECR. This protocol also assumes an ECR sending messages to and getting messages from a Terminal device, which will handle all card related functions. Examples of how this protocol might be implemented are shown in Appendix A.

### **3.10 WIC Services Manager (WSM)**

This optional layer, if implemented, shall behave as a routing layer that handles the multiple WSPMs loaded on the terminal. This may be accomplished in a number of ways, depending on the software implementation. All of the functions of the WSPI shall be exposed to the application through the upper edge of the WSM in a manner that appears to the application to be identical to interaction with a single WSPM. The functions of the WSM are described in Appendix C of this document.

## **4. Access Control Model**

### **4.1 Overview**

Each file on a smart card is protected by Access Conditions (ACs) that must be satisfied to access the data therein. These ACs are defined during file creation, on a file system card, or during applet development, in the case of an interpretive card. Generally, ACs on a file are set by function, so that the data is protected by protecting the function that will access that data. These ACs can be set anywhere from allowing unlimited access to no access at all with a variety of steps in between the two extremes, as allowed by the card operating system. Because the manner chosen to protect the WIC data is left to the discretion of the card issuer, this specification does not dictate in what way this data will be protected. The VOC data is protected for reading by the card PIN while the write protection is left up to the card issuer.

## 5 WIC Services Provider Interface (WSPI)

### 5.1 Overview

The purpose of the WSPI is to provide the WIC applications a common interface to the WIC smart card. There are no functions allowing applications to submit keys to the smart card, as those operations will be handled at the card layer created by each state. The WSPI functions fall into two categories: Card Control and Retailer functions.

The card control section of the WSPI must support the following functions:

- `wspiInitiateSession()`
- `wspiGetPAN()`
- `wspiCardPresent()`
- `wspiConnectCard()`
- `wspiDisconnectCard()`
- `wspiEndSession()`

The retailer data functions of the WSPI are:

- `wspiAuthenticateUser()`
- `wspiReadBalance()`
- `wspiDebitBalance()`
- `wspiPostStagedTransaction()`
- `wspiBlockCard()`

### 5.2 Card Control Functions

#### 5.2.1 `wspiInitiateSession()`

**Purpose:** To enable runtime authentication of the retailer application to the state-defined module in the manner outlined by the state and to start usage of the WSPM. This function is analogous to a class constructor in C++.

**Prerequisites:** None.

**Prototype:** `unsigned long wspiInitiateSession(  
IN unsigned long ullnitType,  
IN unsigned char* pucnitData,  
IN unsigned long ullnitDataLength,  
IN unsigned char* pucRetailerID,  
IN unsigned char ucProcessingMode)`

<b>Parameters:</b>	<b>ullnitType</b>	State defined flag to indicate authentication method required
	<b>pucnitData</b>	A pointer to a block of data that might be required by the state card layer for system initialization purposes.
	<b>ullnitDataLength</b>	The length of the InitData block, in bytes.
	<b>uiRetailerID</b>	State defined value uniquely identifying the specific retail location within the state Host system
	<b>ucProcessingMode</b>	A flag to indicate the manner in which the WIC

transaction should function within the WSPM

Mode	Description
0	Normal Processing
1	Training
2	Certification
3	Certification/Training

**Return codes:** WSPI\_OK  
WSPI\_ACCESS\_DENIED  
WSPI\_BAD\_AUTH  
WSPI\_BAD\_PARAM  
WSPI\_UNKNOWN\_ERROR

### 5.2.2 wspiConnectCard()

**Purpose:** To power up the card, prepare it for data operations. Once the card is successfully powered up, it is assumed that the card will remain in the reader until the transaction is complete. The date/time passed in with this function is the date and time for the purpose of checking the Hot Card List to determine whether a card found on the Hot Card List will be blocked.

**Prerequisites:** wspiInitiateSession()

**Prototype:** unsigned long wspiConnectCard(  
IN unsigned char\* pucDate)

**Parameters:** **pucDate** Current date and time in YYYYMMDDHHMMSS format, always 14 bytes long

**Return codes:** WSPI\_OK  
WSPI\_CARD\_BLOCKED  
WSPI\_BAD\_PARAM  
WSPI\_CARD\_ABSENT  
WSPI\_TIMEOUT\_ERROR  
WSPI\_UNKNOWN\_ERROR,  
WSPI\_UNKNOWN\_CARD

### 5.2.3 wspiGetPAN()

**Purpose:** To read the PAN and the Blocking Descriptor bytes from the CCC for 1) hotlist checking and selection of the proper WSPM within the WSM and 2) determining if the card is already blocked. This function should be written following the discovery mechanism outlined in Section 9 to ensure proper retrieval of the PAN. For logging purposes, if the PAN is retrievable it should be returned, even in the case of a non-zero return code.

**Prerequisites:** wspiInitiateSession()  
wspiConnectCard()

**Prototype:** unsigned long wspiGetPAN(  
OUT unsigned char\* pucPAN,  
OUT unsigned long ulPANLength)

**Parameters:** **pucPAN** A pointer to the card PAN read in from the CCC.  
**ulPANLength** The length of the PAN, in bytes.

**Return codes:** WSPI\_OK  
WSPI\_ACCESS\_DENIED  
WSPI\_CARD\_REMOVED  
WSPI\_UNKNOWN\_ERROR

#### 5.2.4 **wspiCardPresent()**

**Purpose:** To check the presence of a card inserted in the reader. This function should directly access the reader function 'rdiCardPresent()' and pass the result back up.

**Prerequisites:** wspiInitiateSession()

**Prototype:** unsigned long wspiCardPresent()

**Parameters:** none

**Return codes:** WSPI\_OK  
WSPI\_CARD\_ABSENT  
WSPI\_UNKNOWN\_CARD  
WSPI\_TIMEOUT\_ERROR

#### 5.2.5 **wspiDisconnectCard()**

**Purpose:** To power off the card and clear any data buffers used during the previous transaction.

**Prerequisites:** None.

**Prototype:** unsigned long wspiDisconnectCard(void)

**Parameters:** none

**Return codes:** WSPI\_OK  
WSPI\_CARD\_REMOVED  
WSPI\_TIMEOUT\_ERROR  
WSPI\_UNKNOWN\_ERROR

#### 5.2.6 **wspiEndSession()**

**Purpose:** To clear the processing mode and security data set up by wspiInitiateSession(). This function is analogous to a class destructor in C++.

**Prerequisites:** None.

**Prototype:** unsigned long wspiEndSession( void )

**Parameters:** none

**Return codes:** WSPI\_OK

## WSPI\_UNKNOWN\_ERROR

### 5.3 Retailer Data Functions

#### 5.3.1 wspiAuthenticateUser()

**Purpose:** To allow the retailer application to convey the cardholder authentication information to the card. This information could be Personal Identification Number (PIN), cardholder biometrics, or other data as determined by the state. This function should make use of the CCC information in Section 9 regarding PIN location and specific PIN handling requirements.

**Prerequisites:** wspiInitiateSession()  
wspiConnectCard()  
wspiGetPAN()

**Prototype:** unsigned long wspiAuthenticateUser(  
IN unsigned long ulTypeOfAuthentication,  
IN unsigned char\* pucAuthData,  
IN unsigned long ulAuthDataLength )

<b>Parameters:</b>	<b>ulTypeOfAuthentication</b>	Flag indicating type of user authentication provided, where 0=PIN and other values are reserved for expansion, i.e., biometric.
	<b>pucAuthData</b>	User authentication data, such as PIN or biometric data, as determined by the state.
	<b>ulAuthDataLength</b>	If length of authentication data = 0, then PIN is acquired internally, not passed down by the Retailer Application.

**Return codes:** WSPI\_OK  
WSPI\_BAD\_AUTH  
WSPI\_CARD\_BLOCKED  
WSPI\_CARD\_REMOVED  
WSPI\_PIN\_LOCKED  
WSPI\_UNKNOWN\_ERROR

#### 5.3.2 wspiReadBalance()

**Purpose:** To read the balance of WIC items delineated in the pucWICItems parameter. The standard format for each category is listed in Section 4.4. The date passed into the WSPM via this function is used to determine the current benefit period balance. The prescription data integrity checks are performed and the food package access conditions are fulfilled under this function. This function also implements card authentication processes.

**Prerequisites:** wspiInitiateSession()  
wspiConnectCard()  
wspiGetPAN()  
wspiAuthenticateUser()

**Prototype:** unsigned long wspiReadBalance(

IN unsigned char\* pucDate,  
 OUT unsigned char\* pucExpirationDate,  
 OUT unsigned char\* pucFirstDateToSpendDate,  
 OUT unsigned char\* pucIssuingEntity,  
 OUT unsigned long\* pulNumberWICItems,  
 OUT unsigned char\* pucWICItems)

**Parameters:**

<b>pucDate</b>	Current date and time in YYYYMMDDHHMMSS format, always 14 bytes long.
<b>pucExpirationDate</b>	The Date and time in YYYYMMDDHHMMSS when the benefits expire as stored on the card
<b>pucFirstDateToSpendDate</b>	The first Date and time in YYYYMMDDHHMMSS that benefits are eligible to be used.
<b>pucIssuingEntity</b>	The last WIC Agency to access card (length is always 15 bytes)
<b>ulNumberWICItems</b>	Number of WIC items in pucWICItems string.
<b>pucWICItems</b>	String of 10 byte long WIC items as described in Section 5.4.

**Return codes:** WSPI\_OK  
 WSPI\_ACCESS\_DENIED  
 WSPI\_BAD\_AID  
 WSPI\_BAD\_PARAM  
 WSPI\_CARD\_REMOVED  
 WSPI\_READ\_ERROR  
 WSPI\_CARD\_BLOCKED  
 WSPI\_TIMEOUT\_ERROR  
 WSPI\_UNKNOWN\_ERROR  
 WSPI\_UNKNOWN\_CARD  
 WSPI\_PURSE\_ERROR  
 WSPI\_BENEFITS\_EXPIRED

### 5.3.3 wspiDebitBalance()

**Purpose:** To debit the balance of a collection of WIC items. The date passed into the WSPM via the wspiReadBalance() function is the official Start Transaction Date/Time for use with the subsequent debit transactions. This ensures that the benefits debited are the same benefits read in earlier.

**Prerequisites:** wspiInitiateSession()  
 wspiConnectCard()  
 wspiGetPAN()  
 wspiAuthenticateUser()  
 wspiReadBalance()

**Prototype:** unsigned long wspiDebitBalance(  
 IN unsigned long ulNumberWICItems,  
 IN unsigned char\* pucWICItems,  
 OUT unsigned char\* pucWICTransactionSignature,  
 OUT unsigned long ulWICTransactionSignatureLength)

**Parameters:**

<b>ulNumberWICItems</b>	Number of WIC items in pucWICItems string.
<b>pucWICItems</b>	String of 10 byte long WIC items as described in Section 4.4.
<b>pucWICTransactionSignature</b>	Electronic transaction signature.

**ulWICTransactionSignatureLength** Length of electronic signature.

**Return codes:** WSPI\_OK  
WSPI\_ACCESS\_DENIED  
WSPI\_BAD\_AUTH  
WSPI\_BAD\_PARAM  
WSPI\_CARD\_REMOVED  
WSPI\_READ\_ERROR  
WSPI\_TIMEOUT\_ERROR  
WSPI\_UNKNOWN\_ERROR  
WSPI\_UNKNOWN\_CARD  
WSPI\_PURSE\_ERROR  
WSPI\_INVALID\_CATEGORY  
WSPI\_INVALID\_SUBCATEGORY  
WSPI\_INVALID\_UNIT  
WSPI\_INSUFFICIENT\_BALANCE  
WSPI\_CARD\_BLOCKED

#### 5.3.4 wspiBlockCard()

**Purpose:** This function would enable the retailer application to block the card if found on the hot list. The policy decision and specific mechanism to block a card will be defined by the state and implemented within the WSPM.

**Prerequisites:** wspiInitiateSession()  
wspiConnectCard()  
wspiGetPAN()

**Prototype:** unsigned long wspiBlockCard(  
IN unsigned long ulReasonCode,  
IN unsigned char\* pucPAN,  
IN unsigned long ulPANLength)

**Parameters:** **ulReasonCode** Code to indicate to the WSPM what type of block is to be applied. The reason codes are supplied by the X9.93 specification  
**pucPAN** PAN value stored on the card  
**ulPANLength** Length of PAN data

**Return codes:** WSPI\_OK  
WSPI\_ACCESS\_DENIED  
WSPI\_BAD\_AID  
WSPI\_CARD\_REMOVED  
WSPI\_TIMEOUT\_ERROR  
WSPI\_UNKNOWN\_ERROR  
WSPI\_UNKNOWN\_CARD

#### 5.3.5 wspiPostStateAuthorizedTransaction()

**Purpose:** An administrative transaction to update the balance of one or more WIC items. Post State Authorized transactions must be applied in the order specified by the state.

**Prerequisites:** wspiInitiateSession()  
wspiConnectCard()  
wspiGetPAN()

wspiReadBalance()  
wspiAuthenticateUser()

**Prototype:** unsigned long wspiPostStateAuthorizedTransaction(  
IN unsigned long ulTransactionType,  
IN unsigned char\* pucBeginDate,  
IN unsigned char\* pucEndDate,  
IN unsigned long ulTransactionSequenceNumber,  
IN unsigned long ulNumberWICItems,  
IN unsigned char\* pucWICItems,  
IN/OUT unsigned char\* pucWICTransactionSignature,  
IN/OUT unsigned long\* pulWICTransactionSignatureLength )

<b>Parameters:</b>	<b>ulTransactionType</b>	Code to indicate to the WSPM what type of a transaction is requested. Reference to X9.93.
	<b>pucBeginDate</b>	Benefits start date and time in YYYYMMDDHHMMSS format, always 14 bytes long
	<b>pucEndDate</b>	Benefits end date and time.
	<b>ulTransactionSequenceNumber</b>	A number used to insure the transactions are applied in the proper order
	<b>ulNumberWICItems</b>	Number of WIC items in pucWICItems string.
	<b>pucWICItems</b>	String of 10 byte long WIC items as described in Section 4.4.
	<b>pucWICTransactionSignature</b>	Electronic transaction signature.
	<b>ulWICTransactionSignatureLength</b>	Length of electronic signature.

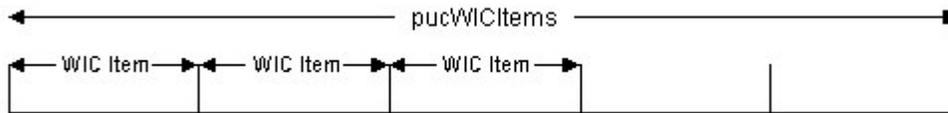
**Return codes:** WSPI\_OK  
WSPI\_ALREADY\_APPLIED  
WSPI\_FUTURE  
WSPI\_NOT\_BENEFIT\_PERIOD\_ON\_CARD  
WSPI\_ACCESS\_DENIED  
WSPI\_BAD\_AUTH  
WSPI\_BAD\_PARAM  
WSPI\_CARD\_REMOVED  
WSPI\_READ\_ERROR  
WSPI\_TIMEOUT\_ERROR  
WSPI\_UNKNOWN\_ERROR  
WSPI\_UNKNOWN\_CARD  
WSPI\_PURSE\_ERROR  
WSPI\_INVALID\_CATEGORY  
WSPI\_INVALID\_SUBCATEGORY  
WSPI\_INVALID\_UNIT  
WSPI\_INSUFFICIENT\_BALANCE  
WSPI\_CARD\_BLOCKED

## 5.4 WICItemsStructure

The `wspiReadBalance` and `wspiDebitBalance` are functions designed to process WIC data and interact with the card so that the WIC data on the card is handled properly. Three pieces of information are necessary to correctly process a WIC transaction for a scanned item: Category number, Sub-category number and the number of units (for credit, debit or balance purposes). The definition of a WIC EBT item is borrowed from and consistent with the ANSI X9.93 specifications on Financial transaction messages – Electronic benefits transfer (EBT) – Part 1: Messages. In order to pass these values down to the card in an efficient manner, the parameters for each item are arranged in an Item structure as follows:



These individual item structures are then concatenated into one buffer of length 10 x N and passed into the `wspi(Read/Debit)Balance` functions. This allows the retailer application to make only one API call for the entire transaction, instead of an API call for each Item. The resulting parameter, `pucWICItems`, is structured as follows:



For example, suppose the benefit food package for the participant included:

Description	Category	Sub-category	Unit of Measure	Quantity
2 gallons of 2% Milk	01	002	Quart	8
1 pound of Cheese	02	000	Ounce	16
2 cans of Orange Juice	04	000	Cans	2

In hexadecimal format, the WIC Item structure read back from the read balance function call will look like

30.31.30.30.32.30.30.38.30.30.30.32.30.30.30.30.31.36.30.30.30.34.30.30.30.30.30.32.30.30

The value of the variable “`ulNumberWICItems`” will be set to 3. This approach is consistent with the X9.93 specification since the complete balance information must be read from the host with one request. Once the balance is read, the retailer application will use this information to check insufficient funds for each item scanned.

The same format is defined for a debit function. For example, if the final purchase was as follows:

Purchase	Category	Sub-category	Unit of Measure	Quantity
1 quart of 2% Milk	01	002	Quart	1
1 can of Orange Juice	02	000	Ounce	1

In hexadecimal format, the WIC Item structure for the `wspiDebitBalance` function will look like:

30.31.30.30.32.30.30.31.30.30.30.32.30.30.30.30.31.30.30

The value of the variable “ulNumberWICItems” will be set to 2.

## 5.5 WSPI Return Codes

The return codes are split into two groups to differentiate between errors associated with hardware errors or mistakes in calling a function and errors associated with the WIC-BI data on the card. The first group of return codes (Table 5.1) is the set of general card or function related errors (found within the WSPI), and the second group is the set of WIC-BI related return codes. The tables identify the Return Codes found within the WSPI, their labels, hexadecimal value, and summary meaning.

**Table 5.1: WSPI Return Codes**

Label	Return Code Hexadecimal Value	Meaning
WSPI_OK	0x0000	Execution completed successfully.
WSPI_ACCESS_DENIED	0x0001	The applicable ACR was not fulfilled.
WSPI_BAD_AID	0x0002	The specified Application Identifiers (AID) does not exist.
WSPI_BAD_AUTH	0x0003	Invalid authentication data.
WSPI_BAD_PARAM	0x0004	One or more of the specified parameters is incorrect.
WSPI_CARD_ABSENT	0x0005	The card associated with the specified card handle is not present.
WSPI_CARD_REMOVED	0x0006	The card associated with the specified card handle has been removed.
WSPI_DELETE_ERROR	0x0007	Error encountered while trying to delete the specified data.
WSPI_INSUFFICIENT_BUFFER	0x0008	The buffer allocated by the calling application is too small.
WSPI_NO_CARDSERVICE	0x0009	The smart card associated with the specified card handle does not provide the requested service.
WSPI_PIN_LOCKED	0x000A	The PIN is locked.
WSPI_READ_ERROR	0x000B	Error encountered while attempting to read the specified data.
WSPI_TIMEOUT_ERROR	0x000C	A connection could not be established with the card before the timeout value expired.
WSPI_TERMINAL_AUTH	0X000D	The card reader has performed a successful Internal Authentication with the card.
WSPI_UNKNOWN_ERROR	0x000E	The requested operation has generated an unspecified error.
WSPI_UNKNOWN_READER	0x000F	The specified reader does not exist.
WSPI_UNKNOWN_CARD	0x0010	The specified card does not exist
WSPI_READER_UNAVAILABLE	0x0011	The specified reader is unavailable
WSPI_READER_BUSY	0x0012	The specified reader is busy
WSPI_UNKNOWN_KEY	0x0013	The specified key number does not exist.

Table 5.2 below identifies the Return Codes that are WIC item related that are found within the WSPI, their labels, hexadecimal value and summary meaning.

**Table 5.2: WSPI WIC Item Related Return Codes**

Label	Return Code Hexadecimal Value	Meaning
WSPI_PURSE_ERROR	0x0080	Error executing purse function. Check Item error for details.
WSPI_INVALID_CATEGORY	0x0081	Category code unknown to this module.
WSPI_INVALID_SUBCATEGORY	0x0082	Subcategory code unknown to this module.
WSPI_INVALID_UNIT	0x0083	Error encountered connected to the unit parameter.
WSPI_INSUFFICIENT_BALANCE	0X0084	Debit attempt will lower the balance below zero.
WSPI_EXCEEDS_MAX_BALANCE	0x0085	Credit attempt exceeds state defined limit on item.
WSPI_CARD_BLOCKED	0x0086	The debit or balance inquiry transactions are blocked on the card.
WSPI_BENEFITS_EXPIRED	0X0087	The WIC benefits on the card have expired.
WSPI_BENEFITS_CONFLICT	0x0088	At least one WIC benefit parameters on the card is out of range.
WSPI_ALREADY_APPLIED	0x0089	The staged transaction has already been posted to the card
WSPI_PERIOD_NOT_ON_CARD	0x008A	The staged benefits are not valid for the current date
-----	0x8000-0x8FFF	Reserved for State use
-----	0x9000-0x9FFF	Reserved for WMP specific use

## 6 Reader Driver API (RDI)

### 6.1 Overview

The purpose of the Reader Driver API is to provide the WSPM with a common interface to a wide variety of readers. The retailers will develop the reader driver based on their hardware requirements and the WSPM will communicate with the card via this interface. The functions defined in this interface are:

- `rdiConnectReader()`
- `rdiCardPresent()`
- `rdiBeginTransaction()`
- `rdiExchangeAPDU()`
- `rdiEndTransaction()`
- `rdiDisconnectReader()`

The expected flow of this driver is for the WSPM to call `rdiConnectReader()` during the operation of `wspiInitiateSession()`. This will ensure that at retailer application startup or during the switching of WSPMs, a reader driver is available to handle the WIC card transactions. The call to `rdiConnectReader()` will also return a reader handle, thus allowing multiple readers, if the retailer requires such functionality. Once the reader driver is connected, the retail application may use `rdiCardPresent()` to set up a card polling function that watches the reader and notifies the module that a card has been inserted. The function `rdiBeginTransaction()` will be used by `wspiConnectCard()` to power up the card and retrieve the ATR. `rdiExchangeAPDU()` will be used by all of the WSPI User Data functions to get data from and send data to the WIC smart card. Once all card interactions are complete, `rdiEndTransaction()` will be called by `wspiDisconnectCard()` to power down the card. Finally, when the application is either closing or switching to another module, `wspiEndSession()` will call `rdiDisconnectReader()` to close out the reader handle.

### 6.2 Reader Driver Functions

#### 6.2.1 `rdiConnectReader()`

**Purpose:** Encompass all of the functionality required to activate the reader and pass back a handle to the reader that will be used by the WIC card. This handle will be used with all subsequent calls to the reader driver and becomes invalid after a call to `rdiDisconnectReader`.

**Prerequisites:** none

**Prototype:** `unsigned long rdiConnectReader(  
OUT unsigned long ulReaderHandle)`

**Parameters:** `ulReaderHandle` a four-byte value that the reader driver will use to refer to the currently active reader.

**Return codes:** `WSPI_OK`  
`WSPI_UNKNOWN_ERROR`  
`WSPI_UNKNOWN_READER`  
`WSPI_READER_UNAVAILABLE`  
`WSPI_READER_BUSY`

### 6.2.2 rdiCardPresent()

**Purpose:** To check the currently connected reader for an inserted card.

**Prerequisites:** rdiConnectReader()

**Prototype:** unsigned long rdiCardPresent (  
IN unsigned long ulReaderHandle)

**Parameters:** **ulReaderHandle** a four-byte value returned from rdiConnectReader indicating which reader to check.

**Return codes:** WSPI\_OK  
WSPI\_BAD\_PARAM  
WSPI\_CARD\_ABSENT  
WSPI\_TIMEOUT\_ERROR  
WSPI\_UNKNOWN\_ERROR  
WSPI\_UNKNOWN\_READER  
WSPI\_UNKNOWN\_CARD  
WSPI\_READER\_UNAVAILABLE  
WSPI\_READER\_BUSY

### 6.2.3 rdiBeginTransaction()

**Purpose:** To power up the card in the current reader and return the ATR.

**Prerequisites:** rdiConnectReader()

**Prototype:** unsigned long rdiBeginTransaction (  
IN unsigned long ulReaderHandle,  
OUT unsigned char\* pucCardATR,  
OUT unsigned long ulCardATRLength )

**Parameters:** **ulReaderHandle** a four-byte value returned from rdiConnectReader indicating which reader to power up.  
**pucCardATR** a string of bytes returned by the card when powered up.  
**ulCardATRLength** length of ATR string.

**Return codes:** WSPI\_OK  
WSPI\_BAD\_PARAM  
WSPI\_TIMEOUT\_ERROR  
WSPI\_UNKNOWN\_ERROR  
WSPI\_UNKNOWN\_READER  
WSPI\_READER\_UNAVAILABLE  
WSPI\_READER\_BUSY

### 6.2.4 rdiExchangeAPDU()

**Purpose:** To send in an APDU string and return the card response.

**Prerequisites:** rdiConnectReader()  
rdiBeginTransaction()

**Prototype:** unsigned long rdiBeginTransaction (
   
     IN unsigned long ulReaderHandle,
   
     IN byte bCLA,
   
     IN byte bINS,
   
     IN byte bP1,
   
     IN byte bP2,
   
     IN unsigned char\* pucData
   
     IN unsigned long ulDataLength,
   
     OUT unsigned char\* pucResponseData,
   
     OUT unsigned long ulResponseLength,
   
     OUT byte SW1,
   
     OUT byte SW2 )

**Parameters:**

<b>ulReaderHandle</b>	a four-byte value returned from rdiConnectReader indicating which reader to use to send the APDU.
<b>bCLA</b>	the class byte of the APDU.
<b>bINS</b>	the instruction byte of the APDU.
<b>bP1</b>	the P1 byte of the APDU.
<b>bP2</b>	the P2 byte of the APDU.
<b>pucData</b>	data string sent in to the card.
<b>ulDataLength</b>	length of data string sent in to the card.
<b>pucResponseData</b>	data string sent out from the card in response to the incoming APDU.
<b>ulResponseLength</b>	length of response data.
<b>bSW1</b>	SW1 byte of card error value.
<b>bSW2</b>	SW2 byte of card error value.

**Return codes:** WSPI\_OK  
 WSPI\_BAD\_PARAM  
 WSPI\_CARD\_REMOVED  
 WSPI\_TIMEOUT\_ERROR  
 WSPI\_UNKNOWN\_ERROR  
 WSPI\_UNKNOWN\_READER  
 WSPI\_READER\_UNAVAILABLE  
 WSPI\_READER\_BUSY

### 6.2.5 rdiEndTransaction()

**Purpose:** To power down the card in the current reader.

**Prerequisites:** none

**Prototype:** unsigned long rdiEndTransaction (
   
     IN unsigned long ulReaderHandle)

**Parameters:**

<b>ulReaderHandle</b>	a four-byte value returned from rdiConnectReader indicating which reader to power down.
-----------------------	---

**Return codes:** WSPI\_OK  
 WSPI\_BAD\_PARAM  
 WSPI\_CARD\_REMOVED  
 WSPI\_TIMEOUT\_ERROR  
 WSPI\_UNKNOWN\_ERROR  
 WSPI\_UNKNOWN\_READER

WSPI\_READER\_UNAVAILABLE  
WSPI\_READER\_BUSY

### 6.2.6 rdiDisconnectReader()

**Purpose:** To invalidate the reader handle when switching WSPMs or stopping application.

**Prerequisites:** none

**Prototype:** unsigned long rdiDisconnectReader (  
IN unsigned long ulReaderHandle)

**Parameters:** **ulReaderHandle** a four-byte value returned from rdiConnectReader indicating which reader to disconnect.

**Return codes:** WSPI\_OK  
WSPI\_BAD\_PARAM  
WSPI\_TIMEOUT\_ERROR  
WSPI\_UNKNOWN\_ERROR  
WSPI\_UNKNOWN\_READER  
WSPI\_READER\_UNAVAILABLE  
WSPI\_READER\_BUSY



**Parameters:**

- ulSecurityType** Flag to indicate the type of authentication data requested.  
0 = Secret Code  
1 = Cryptogram  
2 = Public Key
- ulKeyNumber** The key number of the security data requested.
- pucRandomData** Random data to be used in calculating the cryptogram.
- ulRandomDataLength** Length of random data passed into CSM
- pucSecurityData** Data returned from the CSM to be used to authenticate the card
- ulSecurityDataLength** Length of security data returned from CSM.

**Return codes:** WSPI\_OK  
WSPI\_ACCESS\_DENIED  
WSPI\_BAD\_PARAM  
WSPI\_UNKNOWN\_ERROR  
WSPI\_UNKNOWN\_KEY

### 7.2.3 csiCheckSecurityData()

**Purpose:** To authenticate security data passed to the CSM from the WSPM.

**Prerequisites:** wspiInitiateSession()  
csiInitiateService()

**Prototype:** unsigned long csiCheckSecurityData (  
IN unsigned long ulSecurityType,  
IN unsigned long ulKeyNumber,  
IN unsigned char\* pucRandomData,  
IN unsigned long ulRandomDataLength,  
IN unsigned char\* pucSecurityData,  
IN unsigned long ulSecurityDataLength)

**Parameters:**

- ulSecurityType** Flag to indicate the type of authentication data to be authenticated.  
0 = Secret Code  
1 = Cryptogram  
2 = Public Key
- ulKeyNumber** The key number of the security data.
- pucRandomData** Random data to be used in calculating the cryptogram.
- ulRandomDataLength** Length of random data passed into CSM
- pucSecurityData** Data passed into the CSM to be used to authenticate the key value.
- ulSecurityDataLength** Length of security data passed to the CSM.

**Return codes:** WSPI\_OK  
WSPI\_ACCESS\_DENIED  
WSPI\_BAD\_AUTH  
WSPI\_BAD\_PARAM  
WSPI\_UNKNOWN\_ERROR  
WSPI\_UNKNOWN\_KEY

#### 7.2.4 **csiEndService()**

**Purpose:** To clear out security data from any buffers generated and break the connection to any hardware security tokens currently connected.

**Prerequisites:** `wspilnitiateService()`

**Prototype:** `unsigned long csiEndService (void)`

**Parameters:** **none**

**Return codes:** `WSPI_OK`  
`WSPI_UNKNOWN_ERROR`

## 8 WIC Data Model

### 8.1 Overview

The WIC Data Model comprises Verification of Certification (VOC) data, which has to be readable by all WIC states and the EBT Benefit Information (EBT-BI) data, which is specific for each state's card design.

VOC data read interoperability is achieved by the use of a VOC data container that is defined across all WSPI compatible smart cards. To support this, a WSC-IS interoperable card will need to have a Card Capability Container (CCC), similar to that defined in the GSC-IS v2.0 specification, located in an EF at the MF level and described in Section 9 of this document. The VOC data container stores data pertaining to a theoretically unlimited number of authorized WIC participants. This is accomplished by the use of a compound TLV scheme whereby each WIC participant record is preceded by the 'Participant Record Indicator' tag and length of the participant record that follows. On VM cards, the AID of the VOC Data container will be (WIC RID || C100). On ISO file system cards, the VOC Data container will reside at the MF level with an EFID of C100. The VOC data shall be read-protected by a key authentication mechanism to be designed by USDA. The write-protection of this data is not specified in this document.

The EBT-BI data is specified on a state-by-state basis, and is not defined in this document. This data is not interoperable at the WSPI level, however, the EBT-BI specific functions, `wicReadBalance()` and `wicDebitBalance()` are available to all applications through the state-specific WSPM.

### 8.2 Verification of Certification Data Container

Table 8.1: VOC Data Container

VOC Data Container		EF C100	Key Authentication
Data Element (TLV)	Tag	Type	Max. Bytes
Length of Container Contents	-----	Numeric	2
Date of Income Eligibility Last Determined (YYYYMMDD)	0x10	Fixed String	8
Certifying Local Agency Name	0x11	Variable String	30
Local Agency Address 1	0x12	Variable String	30
Local Agency Address 2	0x13	Variable String	30
Local Agency City	0x14	Variable String	20
Local Agency State	0x15	Fixed String	2
Local Agency Zip	0x16	Variable String	9
Local Agency Phone	0x17	Variable String	10
Local Agency Official	0x18	Variable String	30
Participant Record Indicator	0xA0	Compound Tag	1
Participant Record Length	--	Numeric	2
Participant ID	0xB0	Variable String	16
Participant First Name	0xB1	Variable String	15

VOC Data Container		EF C100	Key Authentication
Data Element (TLV)	Tag	Type	Max. Bytes
Participant Last Name	0xB2	Variable String	20
Participant Middle Initial	0xB3	Fixed String	1
Participant Risk 1	0xB1	Variable String	3
Participant Certification Date (YYYYMMDD)	0xB6	Fixed String	8
Participant Certification Expiration (YYYYMMDD)	0xB7	Fixed String	8
...	...	...	...
Error Detection Code	0xFE	LRC	1

## 9 Card Capability Container (CCC)

### 9.1 Overview

The concept of a CCC is taken from the GSC-IS specification. The CCC is meant to provide a means for mitigating the differences between the smart card vendors APDU set implementations. The CCC should be accessed inside the function `wspiGetPAN()` to determine which WSPM to access and so that the PAN may be retrieved for hotlist checking. Note that for the standard functions listed below, VM cards must implement the APDUs in the manner shown but the only Select APDU required is Select File by AID.

### 9.2 Procedure for Accessing the CCC

The procedure for accessing the CCC differs between VM cards and ISO file system cards. The VM cards access all of their containers via the container AID. ISO file system cards only allow AIDs for the DFs and not for the EFs so identifying a data container by AID on an ISO file system card is impossible. Therefore, two different procedures must be taken into account.

#### 9.2.1 General CCC Retrieval Sequence

The WSPM will attempt to select the CCC using a standard AID first, under the assumption that the card inserted is a VM card.

CLA	INS	P1	P2	P3	AID
00	A4	04	00	07	RID    DB01

If this fails, the WSPM then assumes the card is a file system card and follows the file system procedure. If the file system procedure fails, the WSPM will assume the card is not a valid WIC card.

#### 9.2.2 CCC Retrieval Sequence for ISO file system cards

In this sequence, a series of APDUs are sent to the card until the CCC is successfully read. The sequence starts, after the card is powered up, with an attempt to select the MF. While most freshly powered up cards default to the MF level, this step is done to determine the proper Class byte to be used for this card. Once the MF is selected successfully, an attempt is made to select the CCC using the class byte determined above. Once the CCC selection is successful, a series of Read Binary commands are performed to retrieve the data from the card. The data is then parsed to obtain the required information about the card capabilities.

Here are the steps outlined:

1. Loop over sending Select File APDU to select the MF using the test class byte:

CLA	INS	P1	P2	P3	FIDH	FIDL
TEST CLA	A4	00	00	02	3F	00

The default TEST CLA values are: 0x00, 0x80, 0x90, 0xA0, 0xC0, 0xF0, 0xBC, 0x01, and 0xB0-0xCF.

2. If the status bytes returned are 0x6E00 (Class not supported), continue the loop. If the status bytes returned are 0x9000 or 0x61XX, the APDU is correct, so the CLA byte should be stored for use in future commands.
3. If the loop completes and the returned status bytes are not 0x6E00, 0x9000 or 0x61XX, set P2 = 0x0C and repeat the loop. If this step does not work, the card is not a WIC-EBT compliant card.
4. Once the CLA has been determined, attempt to select the CCC EF using:

CLA	INS	P1	P2	P3	FIDH	FIDL
Determined CLA	A4	00	P2	02	0xDB	0x01

5. If the status bytes returned are other than 0x9000 or 0x61XX, set P1 = 0x02 and try again. If the second attempt to select the file fails, the card is not a WIC-EBT compliant card.
6. Once the WIC CCC has been selected, the contents are read via a Read Binary command in the following manner:

CLA	INS	P1	P2	P3
Determined CLA	B0	Off/H	Off/L	L

Because the first two bytes of the CCC contain the length of the CCC contents (see below), it is strongly suggested that the WSPM read in the first two bytes in the first read attempt and use that value to determine the length of the second read attempt.

### 9.3 CCC Structure

The CCC shall reside at the MF or Master Directory level (0x3F00) and is designated by a standard AID (WIC-RID||DB01) on a VM card and an EFID of 0xDB01 on an ISO file system card.

#### 9.3.1 CCC Data Layout

Table 9.1: CCC Contents

Card Capabilities Container		EF DB01	Always
Data Element (TLV)	Tag	Type	Length
CCC Length	-----	Numeric	2
PAN Number	0xF0	Variable String	19
Card Version Number	0xF1	Numeric	1
Capability Container Version Number	0xF2	Numeric	1
Capability Grammar Version Number	0xF3	Numeric	1
Capability Tuples	0xF4	Variable	2X
Error Detection Code	0xFE	LRC	1

### 9.3.2 CCC Element Descriptions

The Primary Account Number (PAN) is assigned by the WIC clinic at card issuance. This number will be read off the card and used by the WSPM to identify the card's state of origin. The Block Descriptor is a WSPM specific value to indicate the card is blocked and in what manner it is blocked. A non-zero value indicates the card is blocked and the value can only be properly interpreted by the WSPM specific to the card. In the case of a state with multiple card versions, the Card Version Number is used by the WSPM to determine the proper card handling logic. The Capability Container and Grammar Version Number will be used to allow for modifications to the layout in future version of the card. The Capability Tuples are described in the next section.

### 9.3.3 Construction of Capability Tuples

The purpose of the Capability Tuples is to provide a mechanism to allow the use of a large set of file system smart cards with different APDU implementations in the same manner to access the VOC data. These tuples need only be defined in the CCC if the card's APDUs vary from the accepted set of APDUs required to access the VOC data. The standard APDU set is as follows:

**Table 9.2: Standard APDU set**

Standard APDU Set								
FC	Card Function	CLA	INS	P1	P2	P3	Data	Notes
1	Select File by FID	00	A4	00	00	02	FID	
2	Select Child DF	00	A4	01	00	02	FID	
3	Select EF under DF	00	A4	02	00	02	EFID	
4	Select File by AID	00	A4	04	00	L	AID	
5	Verify CHV	00	20	00	01	L(08)	CHV	PIN = key 1
6	Read Binary	00	B0	Off/H	Off/L	L	--	
7	Get Response	00	C0	00	00	L	Data to retrieve	

Each two-byte tuple comprises a C-byte and a V-byte as shown in Table 9.3, which describes one piece of an APDU for a particular command. Therefore, in some cases, proper description of the APDU modifications may require multiple two-byte tuples for the same command. The capability tuples are constructed using the following bitmaps:

**Table 9.3: Tuple Byte Descriptions**

C – Code Byte								V – Value/Descriptor Byte							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0 = Const		Parameter				Function Code		If C bit 7 = 0, then V contains a constant value							
1 = Desc								If C bit 7 = 1, then V contains a Descriptor code							

The C-byte of the tuple is the Code byte. It identifies the particular command and parameter that is being defined. The V-byte is the Value Byte, which provides either the value to be used for the

parameter or a descriptor code that represents the definition of the parameter, i.e. what the parameter is in the APDU. The Parameter and function codes are defined in Table 9.4.

**Table 9.4: Parameter and Function Codes**

Parameter Codes		Function Codes	
0x00	DATA	0x00	Reserved
0x01	CLA	0x01	Select File by FID
0x02	INS	0x02	Select Child DF
0x03	P1	0x03	Select EF under DF
0x04	P2	0x04	Select File by AID
0x05	P3	0x05	Verify CHV
0x06	Prefix	0x06	Read Binary
0x07	Suffix	0x07	Get Response

Parameter codes 0x06 and 0x07 represent prefix and suffix commands, respectively. These parameter codes indicate that another command is required either before or after the specified function code. For example, some card types require a **Get Response** after a **Read Binary** but do not return a 0x61XX code to the **Read Binary** command. In this case, a tuple must be added to indicate this requirement.

The descriptor codes are used to add processing information for data values or parameters. Parameters can be described by one descriptor code while data values may be described by a series of descriptor codes.

**Table 9.5: Descriptor Codes**

Descriptor Codes			
0x00 – 0x0F	Execute Function Code	0x11	Challenge
0x12	Algorithm Identifier	0x15	Length
0x16	MSB of Offset	0x17	LSB of Offset
0x18	Key Level	0x19	Key Identifier
0x1A	CHV Level	0x1B	CHV Identifier
0x1C	AID	0x1D	EF
0x1E	System ID	0x1F	RFU
0x21	2 Byte FID	0x22	Short FID
0x23	File Name	0x2F	8 Byte Random Number



Select Child DF 12 C0 32 00

The third permutation is similar to the second in that the CLA byte and the P1 byte require modification. The Parameter 01 (CLA) of Function 03 (Select EF under current DF) should be a Constant (bit 7 of C-byte = 0) value of C0. The Parameter 03 (P1) of Function 03 (Select EF under current DF) should be a Constant (bit 7 of C-byte = 0) value of 00. Therefore, the CT for this command is:

Select EF under current DF 13 C0 33 00

The fourth permutation of **Select File** is not supported on this card type. In this last case, the Parameter 02 (INS) of Function 04 (Select File by AID) has a Descriptor code (bit 7 of C-byte = 1) of 'Command not available' (FE). Therefore, the CT for this command is:

Select File by AID A4 FE

Therefore, the CTs for all of the **Select File** commands are:

Select File by FID	11 C0
Select Child DF	12 C0 32 00
Select EF under current DF	13 C0 33 00
Select File by AID	A4 FE

#### 9.3.4.2 Verify CHV

Likewise, for the **Verify CHV** command, the APDU specified is:

Verify CHV 00 20 00 01 L(08).

For the Cryptoflex card, the only changes required are the CLA byte and, possibly, the P2 byte, if the CHV is not 01. For a card that's CHV is 02 instead of 01, the CT would be:

Verify CHV 15 C0 45 02

#### 9.3.4.3 Read Binary

Lastly, for the **Read Binary** command, the APDU specified is:

Read Binary 00 B0 Off/H Off/L L

For the Cryptoflex card, only the CLA byte needs to change so the CT would be:

Read Binary 16 C0

#### 9.3.5 Example CCC Contents

Putting all of this information together, the CCC on a Cryptoflex card would look like the following:

Tag	Len	Val	
		35	// Length of subsequent data
F0	13	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	// PAN
F1	01	01	// State Card Version Number

F2	01	01		// Capability Container Version Number
F3	01	01		// Capability Grammar Version Number
F4	12			// Capability Tuples:
		11	C0	// Select By FID
		12	C0 32 00	// Select Child DF
		13	C0 33 00	// Select EF under DF
		A4	FE	// Select by AID: Unavailable
		15	C0	// Verify CHV (PIN = 01)
		16	C0	// Read Binary
		17	C0	// Get Response
FE	01	XX		// ErrorDetectionCode (LRC computed per card)

## 10 Glossary

<b>AC</b>	Access Condition. A security mechanism that must be satisfied before access to a protected card function is granted.
<b>ACK</b>	Acknowledge byte. Used in communication protocols by a receiver to indicate the message has been received from the sender.
<b>AID</b>	Application Identifier. A string of bytes that identifies an applet on an interpretive card or a dedicated file on a file system card.
<b>ANSI</b>	American National Standards Institute. ANSI
<b>APDU</b>	Application Protocol Data Unit. A string of bytes containing instructions, and possibly data, to be sent to the smart card for processing.
<b>API</b>	Application Programming Interface.
<b>ATR</b>	Answer To Reset. Data returned from the smart card when power is applied to the chip.
<b>BI</b>	Benefit Information.
<b>CAD</b>	Card Acceptance Device. The device used to interface with the smart card at the point of retail transaction. (See "Terminal")
<b>CCC</b>	Card Capabilities Container.
<b>CEI</b>	Card Edge Interface.
<b>CHV</b>	Card Holder Verification. Usually a PIN but may be a biometric.
<b>CLA</b>	The Class byte of an APDU as defined in ISO7816-4
<b>CSI</b>	Cryptographic Services API.
<b>CSN</b>	Card Serial Number
<b>CT</b>	Capability Tuple. A two-byte descriptor used to define a single deviation of a given card operating system command from the standard commands defined in this document.
<b>DF</b>	Dedicated File. A specific type of file only found on file system cards and may contain a set of elementary files. Analogous to a directory or folder in a computer operating system.
<b>EBT</b>	Electronic Benefits Transfer.
<b>EBT-BI</b>	EBT Benefit Information.
<b>ECR</b>	Electronic Cash Register - a point-of-sale device that is able to host variety of peripheral equipment such as a display, keypad, bar-code scanner, terminal, printer, PIN pad, check reader, etc., to support retail transactions.
<b>EF</b>	Elementary File. A specific type of file on a file system card that contains the cardholder data stored on the card.
<b>ETX</b>	End of text.
<b>FIDH</b>	MSB of the File Identifier
<b>FIDL</b>	LSB of the File Identifier
<b>INS</b>	The Instruction byte of an APDU as defined in ISO7816-4

<b>LOUD</b>	Length of Useful Data.
<b>LRC</b>	Longitudinal Redundancy Check
<b>MF</b>	Master File. The uppermost, or root, directory on a file system card.
<b>NAK</b>	
<b>P1</b>	The P1 byte of an APDU as defined in ISO7816-4
<b>P2</b>	The P2 byte of an APDU as defined in ISO7816-4
<b>P3</b>	The P3 byte of an APDU as defined in ISO7816-4
<b>PAN</b>	Primary Account Number.
<b>PIN</b>	Personal Identification Number.
<b>POS</b>	Point-of-Sale - a compilation of hardware devices and software applications generally co-located in retail environment and placed closest to the point of transaction. The POS processes information related to retail transactions.
<b>RID</b>	Registered Application Provider Identification
<b>RDI</b>	Reader Driver API.
<b>TBD</b>	To Be Determined.
<b>Terminal</b>	Device used to interface with the smart card at the point of retail transaction. The terminal incorporates the card acceptance device (CAD) and also includes other components and interfaces such as a keypad and display to facilitate communications with the attendant, cardholder, and the terminal's hosting device. (See CAD)
<b>TLV</b>	Tag length value.
<b>TUPLE</b>	A fixed size collection of elements or a data object containing other objects as elements.
<b>VM</b>	Virtual Machine. VM cards are interpretive cards. Example: JavaCard and Multos.
<b>WMI</b>	WIC Messaging Interface.
<b>WMP</b>	WIC Messaging Protocol.
<b>WSPI</b>	WIC Services Provider Interface.
<b>WSM</b>	WIC Services Manager.
<b>WSPM</b>	WIC Service Provider Module. The software implementation of the WSPI in accordance with the state supplied card specifications.
<b>WSPS</b>	WIC Service Provider Software.

## APPENDICES

## Appendix A. WIC Messaging Protocol (WMP)

### A.1 Overview

The WMP removes the burden of card handling from the ECR and places it on the attached Terminal or CAD<sup>1</sup>. The WMP potentially standardizes ECR software nationwide for the retailer community. A retailer authorized to transact offline WIC EBT transactions for multiple states will elect a WMP architecture, which has the effect of making the Terminal vendors responsible for handling the WSPMs utilized by the Retailer Application.

This appendix defines the communication functions and messages to be passed between the ECR and the Terminal for WIC transactions. These functions/messages are defined as being generic to ECR or Terminal manufacturer. For clarity, however, they are presented as they exist in the context of a representative IBM EFT reference implementation.

### A.2 Generic Serial Communication

In order to provide a message set that does not conflict with existing communication between ECRs and attached peripherals, the format for all WIC messages is

<STX>\_nnxxxx...xx<ETX><LRC>

where

- 1) <STX> is a start of text character, 0x02,
- 2) the underscore character, “\_”, identifies this message as a WIC message,
- 3) the “nn” is a two ASCII message digit identifier,
- 4) the “xxxx...xx” represents the data specific to the request or the response,
- 5) <ETX> is the end of text character, 0x03, and
- 6) <LRC> is a single character longitudinal redundancy check character.

In all cases, “nn” is an even number for messages sent from the ECR to the Terminal and an odd number for messages sent from the Terminal to the ECR. All messages sent in either direction are error checked by validation of the LRC. If the LRC is calculated to be correct, an <ACK> response of 0x06, will be returned to the sender. If the LRC is calculated to be incorrect, a <NAK> response of 0x15, will be returned to the sender.

To be compatible to this message format, “control” characters will not be included in the message stream, since this may confuse the packet delimiters. Thus, characters hex 0 thru hex 5 are not allowed except as a computed LRC character.

The proposed message format represents a “least common denominator” for compatibility with legacy message transport protocols by using ASCII characters and no control characters. Other message formats supporting pure binary data, similar to ISO 8583 or standard NCR pinpad messages, are also permissible. However, the content and functions described in this document should be used as appropriate guidelines.

### A.3 Setup Processing

The setup processing message sequence will be executed when communications with the Terminal are established or reset. The WSPMs will most likely all be downloaded into flash memory.

#### A.3.1 ECR Establish Active WSPM Request

---

<sup>1</sup> For our purposes here, the word “Terminal” is used to refer to a CAD as well.

The ECR establishes communication with the Terminal by specifying the WSPM to be accessed. The number and specific WSPMs for which communication will be established is determined by the ECR. (See Section 5.2.1 `wspInitiateSession`)

`<STX>_00yyyymmddhhmmssmmppnaabbcc...<ETX><LRC>`

yyyymmdd	current date
hhmmss	current time
mmm	three ASCII digits defining the maximum number of bytes in the messages sent to or read from the Terminal (buffer size);
pp	processing mode (see 4.2.1 for listing of modes)
nn	two ASCII digits defining the number of valid states supported by this ECR/store;
aa, bb, cc	two ASCII characters defining the states (standard postal code for states), the number of entries equal to nn

For example, if Texarkana supports three states, e.g. Texas, Arkansas, and Louisiana, the message would look like:

`<STX>_00200306240210355120003TXARLA<ETX><LRC>`

### A.3.2 Terminal Validate Active WSPMs Response

The Terminal must confirm to the ECR that the respective **WSPMs** are valid.

`<STX>_01EEEEannaabbcc...rr<ETX><LRC>`

EEEE	four ASCII digit error codes converted from WSPI errors.
nn	two ASCII digits defining the number of states for which activation was requested (number of states in message)
aa, bb, cc	two ASCII digits/characters indicating the activation of the <b>WSPMs</b> ,
rr	two ASCII digits indicating the version number of the WSPM or "FF", which means the <b>WSPM</b> was not found.

`<STX>_0101nnaabbcc...nn<ETX><LRC>`

"01" one or more **WSPMs** valid.

For example, if one of the supported states were not activated, i.e., Arkansas, but the other WSPMs were version 1, the response message would be:

`<STX>_010004TXARLA01FF01<ETX><LRC>`

## A.4 Transaction Processing

### A.4.1 Get PAN

Upon receiving the PAN request the Terminal will send the PAN to the ECR for validation against the hot list.

#### A.4.1.1 ECR Get PAN request

	<code>&lt;STX&gt;_10yyyymmddhhmmss&lt;ETX&gt;&lt;LRC&gt;</code>
yyyymmdd	current date
hhmmss	current time

#### A.4.1.2 Terminal Get PAN Response

If card is not already present, the Terminal will display a message to insert a card; when the card is present, the card will be opened and the PAN and the Block byte are retrieved from the CCC. The response will be sent to the ECR.

<STX>\_11EEEESSllp...pa...a<ETX><LRC>

EEEE	four ASCII digit error codes converted from WSPI errors.
SS	two ASCII characters denoting the state for which the card is valid
ll	two ASCII digits defining the length of the PAN
ppp	p...pp PAN – “ll” ASCII digits
a...a	15 ASCII characters denoting the Benefits Issuing Entity which most recently updated the card.

The Benefits Issuing Entity / agency was added as this is required to be logged by the ECR in the claims file.

#### A.4.2 Read Balance

ECR has verified the PAN is not on the hot list and requests the balance of WIC items.

##### A.4.2.1 ECR Read Balance Request

<STX>\_20SSyyyymmddhhmmss<ETX><LRC>

SS	two ASCII characters denoting the state for which the card is valid
yyyymmdd	current date
hhmmss	current time

The Terminal uses `wspiReadBalance` to obtain card balance information.

##### A.4.2.2 Terminal Read Balance Response

<STX>\_21EEEESScyyyymmddyyyymmddssmmnniiii...ii <ETX><LRC>

EEEE	four ASCII digit error codes converted from WSPI errors.
SS	two ASCII characters denoting the state for which the card is valid.
c	one ASCII digit = “0” means last message of items, = “1” means more messages to follow
yyyymmdd	benefits expiration/last date to spend
yyyymmdd	benefits first date to spend
ss	two ASCII digits > “00” reflecting the sequential block number
mm	two ASCII digits reflecting the total number of items in the balance (repeated on each block).
nn	two ASCII digits reflecting the number of items in the current block;
iiii...ii	ASCII string of 10 byte entries of WIC items

Expiration date is added for the ECR to print this on the Current Balance receipt. The first date to spend is included within the claim file.

### A.4.3 Debit Balance

After items have been scanned and WIC tender indicated, the ECR requests the valid WIC items be debited from the balance on the card.

#### A.4.3.1 ECR Debit Balance Request

<STX>\_30SScscsmmnniiii...ii<ETX><LRC>

SS	two ASCII characters denoting the state for which the card is valid
c	one ASCII digit = "0" means last message of items, = "1" means more messages to follow
ss	two ASCII digits > "00" reflecting the sequential block number
mm	two ASCII digits reflecting the total number of items in the balance(repeated on each block);
nn	two ASCII digits reflecting the number of items in the current block;
iiii...ii	ASCII string of 10 byte entries of WIC items Only items to be debited.

#### A.4.3.2 Terminal Debit Balance Response

<STX>\_31EEEESSlltttt..tttt<ETX><LRC>

EEEE	four ASCII digit error codes converted from WSPI errors.
SS	two ASCII characters denoting the state for which the card is valid
ll	the length of the transaction signature
tttt..tt	electronic transaction signature

The ECR must log the transaction signature as part of the claim file entries for the transaction. The WSPI wspiDebitBalance returns the transaction signature and it's length.

### A.4.4 Block Card

If the PAN is found in the "hot file" for the corresponding state, the ECR will request the card be blocked for the reason specified in the "hot file". The ECR will pass in the date and reason code referenced in the "hot file" so the state module can decide whether the card should be blocked.

#### A.4.4.1 ECR Block Card Request

<STX>\_40SSyyyymmddhhmmssrr<ETX><LRC>

SS	two ASCII characters denoting the state for which the card is valid
yyyymmdd	date of card block from hot card list
hhmmss	time of card block from hot card list
rr	two ASCII characters which are the reason code from the "hotfile"

#### A.4.4.2 Terminal Block Card Response

<STX>\_41EEEESS<ETX><LRC>

EEEE	four ASCII digit error codes converted from WSPI errors.
------	--

#### A.4.5 End Transaction

The ECR informs the Terminal that the WIC part of the transaction is complete and the card can be removed.

##### A.4.5.1 ECR End Transaction Request

<STX>\_50SSyyymmddhhmmss<ETX><LRC>

SS	two ASCII characters denoting the state for which the card is valid
yyymmdd	transaction end date
hhmmss	transaction end time

##### A.4.5.2 Terminal End Transaction Response

<STX>\_51EEEESS<ETX><LRC>

EEEE	four ASCII digit error codes converted from WSPI errors.
SS	two ASCII characters denoting the state for which the card is valid

The ECR must inform the Terminal the WIC portion of the transaction is completed and the Terminal should prompt for card removal.

#### A.4.6 Authenticate User

The ECR informs the Terminal that a PIN request must be made to the user at this point.

##### A.4.6.1 ECR Authenticate User Request

<STX>\_60SSAA<ETX><LRC>

SS	two ASCII characters denoting the state for which the card is valid
aa	two ASCII characters denoting the number of PIN attempts allowed by the terminal.

##### A.4.6.2 Terminal Authenticate User Response

<STX>\_61EEEErr<ETX><LRC>

EEEE	four ASCII digit error codes converted from WSPI errors.
rr	two ASCII characters denoting the result of the authentication. "00" = PIN entry successful "FF" = PIN entry unsuccessful but card not locked "LL" = PIN locked at some point during these PIN entries "PP" = PIN was locked prior to current attempts.

Note: The software making the calls to `wspiAuthenticateUser()` must keep track of the return codes from each attempt in order to differentiate between a card that locked during this set of PIN entries and one that was already locked before the recent attempt was made.

#### A.4.7 Shutdown processing

The ECR informs the Terminal that it may close down all WSPMs currently loaded.

##### A.4.7.1 ECR Deactivate WSPMs Request



	ECR Terminal		CAD Device Functionality	Functionality Notes
			Response _61	
If user is not authenticated or the card is on the hotlist, block the card				
Block card	BlockCard _40	→		
			<b>wspiBlockCard</b>	The WSPM will determine if the Block request should be executed
		←	Blocking Status _41	
If user is authenticated and/or the block request failed, then continue				
Request Balance	Item Balance Request _20	→		
			<b>wspiReadBalance</b>	Read Prescription
		←	Item Balance Response (until all data is sent) _21	Send Prescription Data
Determine list of eligible WIC items				
Request Debit	Item Debit Request _30 (until all data sent)	→		
			<b>wspiDebitBalance</b>	Update Prescription
		←	Item Debit Response _31	
	End Transaction request _50	→		
			<b>wspiDisconnectCard</b>	Power-down card
				Issue Remove Card message
		←	EndTransaction response _51	

	ECR Terminal		CAD Device Functionality	Functionality Notes
Sent only when the ECR is closed out	Deactivate WSPMs _70	→		
			<b>wspiEndSession</b>	Deactivates all WSPMs
		←	Deactivate WSPMs response _71	

## Appendix B. Security Policy Recommendation

### B.1 Overview

The Security Policy Group of the Offline WIC EBT Interoperability Specification effort convened to develop security recommendations for adoption by the USDA. The group agreed upon a set of high-level requirements that must be met for each WIC-EBT transaction. These requirements will be met via the interaction of a set of WIC entities. The entities involved in a transaction are the state WIC host ('State'<sup>2</sup>), the store controller system ('Store'), the WIC clinic ('Clinic'), the store terminal ('Terminal'), the cardholder ('User') and the card itself. Each entity is responsible for some aspect of system security directly related to its own interest in the system. Key management requirements and specific security implementations are not outlined in this recommendation.

### B.2 Risks

There were several risk scenarios considered to be significant enough to be addressed by this document. The risk scenarios considered were:

#### B.2.1 The counterfeit card is presented to a legitimate retailer.

This scenario would arise via a non-state organization that creates a card that mimics the appearance and behavior of the state card. The risk associated with this scenario is the effective acquisition of WIC benefits by an invalid participant and the defrauding of the party responsible for WIC reimbursement.

#### B.2.2 A legitimate retailer<sup>3</sup> posts a fraudulent transaction.

This scenario would arise if a legitimate retailer attempts to post a fraudulent transaction to the WIC host for the purpose of financial gain. The risk associated with this scenario is the defrauding of the party responsible for WIC reimbursement.

#### B.2.3 A legitimate card is presented to a rogue application for debiting purposes.

This scenario would arise if an unauthorized party creates a program to debit the WIC benefits on a card for whatever reason. The risk associated with this scenario is the loss of WIC benefits by a legitimate WIC participant.

#### B.2.4 A legitimate card is presented to a rogue application for crediting purposes.

This scenario would arise if an unauthorized party creates a program to credit the WIC benefits on a card for whatever reason. This scenario may also arise from a legitimate retailer who attempts to credit the WIC benefits on a card, aside from the crediting possibly done within the WSPM. The risk associated with this scenario is the defrauding of the party responsible for WIC reimbursement.

#### B.2.5 A stolen legitimate card is presented.

This scenario would arise from the loss of possession of the card by the WIC participant. The risk associated with this scenario is the loss of WIC benefits by a legitimate WIC participant.

---

<sup>2</sup> The term "State" in this document refers to US States, Territories, Commonwealths, and Indian Tribal Organizations (ITO)

<sup>3</sup> A "legitimate retailer" for our purposes here is a retailer who currently has a contract with the state to handle WIC transactions.

### **B.2.6 A Cardholder lends his card to others.**

This scenario would arise from the use of the card by someone other than the WIC participant but with the WIC participant's knowledge and consent. This risk associated with this scenario is the potential loss of WIC benefits by a legitimate WIC participant.

### **B.2.7 A WIC clinic operator knowingly issues fraudulent cards.**

This scenario would arise from an unscrupulous WIC clinic operator issuing fraudulent cards to family and friends or for sale to others. This scenario presents the greatest risk to the WIC system because no on-card protection is effective against this attack. The risk associated with this scenario is the defrauding of the party responsible for WIC reimbursement.

### **B.2.8 A legitimate credit transaction is double posted**

This scenario would arise from the posting of a State-authorized transaction more than once to a legitimate WIC card. The risk associated with this scenario is the defrauding of the party responsible for WIC reimbursement.

## **B.3 Security Requirements**

After reviewing the risks scenarios, the Security Policy Group developed general security requirements:

- Separate credit and debit keys, limiting the retailer authorized in-lane EBT transactions to 'read balance' and 'debit balance' only.
- Separate credit and debit keys, limiting credit key distribution to state-controlled clinic, local agency and state office environments only.
- Authentication<sup>4</sup> of cardholder
- Authentication of card or benefits on the card
- Authentication of the terminal
- Authentication of the transaction
- State defined maximum balance on a given category/subcategory pair
- Tracking card issuance by WIC clinic operators
- Card validation of state authorized transactions<sup>5</sup>

### **B.3.1 Separate credit and debit keys, limiting the retailer authorized in-lane EBT transactions to 'read balance' and 'debit balance' only.**

It is recommended that security keys required for credit and debit be distributed so as to limit store in-lane EBT transactions to 'read balance' and 'debit balance' only. Current off-line WIC systems deploy file system card technology in a closed system design allowing both credit and debit transactions be performed in-lane at a WIC authorized store site. Separation of security keys will enable states to limit retailer authorized EBT transactions performed in-lane at store sites to 'read balance' and 'debit' transactions only, thereby increasing overall system security and avoidance of unnecessary risk and loss resulting from unauthorized crediting of a valid WIC EBT card. State authorized transactions are permitted subject to specific design considerations and risk assessment.

---

<sup>4</sup> In smart card terminology, authentication involves the use of a shared secret between two parties where one party passes a cryptogram generated with that secret to the other party (internal or external authentication) or both pass it to each other (mutual authentication) to achieve access. Verification involves the presentation of the secret itself, e.g. a secret code, PIN or biometric template, to the smart card for validation. In this document, the term 'authentication' is meant to allow either process, unless otherwise specified.

<sup>5</sup> Applicable to states that elect to implement in-lane posting of state authorized transactions.

### **B.3.2 Separate credit and debit keys, limiting credit key distribution to the state-controlled clinic, local agency and state office environments only.**

It is recommended that security keys required for credit and debit be distributed so as to allow credit of value to a valid EBT card only at state-controlled clinic, local agency and state office environments. State-controlled environments include state owned, deployed, operated and maintained hardware and software, as well as third party owned, deployed, operated and maintained hardware and WIC application software required to perform WIC management information systems data processing and storage. The separation of security keys and the limiting of WIC EBT benefit credits to state-controlled environments will increase overall system security and avoid unnecessary risk and loss that could result from the fraudulent crediting of benefits to a WIC EBT card.

### **B.3.3 Authentication<sup>6</sup> of cardholder**

It is recommended that the identity of a cardholder be authenticated prior to the successful initiation of a WIC EBT transaction, in the Store at a state-controlled clinic, local agency and state office location. Authentication of the cardholder may be accomplished by the presentation of a PIN, biometric or other means, as proposed by the State and approved by the USDA/FNS. Authentication of the cardholder serves to reduce the risk of loss resulting from fraudulent use of a valid card.

### **B.3.4 Authentication of card or benefits on the card**

It is recommended that the benefits on a valid WIC EBT card, or the card itself, be authenticated prior to the successful completion of a WIC EBT transaction initiated in the Store state-controlled clinic, local agency and state office environments. Authentication of the benefits on the card or the card itself will be achieved by the exchange of security keys or codes stored on the card and present within the card acceptance device software, as defined in the WSPM. Authentication of the benefits on the card or the card itself will serve to reduce the risk of loss resulting from the use of a counterfeit card or fraudulently issued benefits placed on a valid EBT card.

### **B.3.5 Authentication of the terminal**

It is recommended that the Store be responsible for authenticating all in-store terminals used in the processing of off-line WIC EBT transactions. In-store authentication of installed card acceptor devices or terminals used in off-line WIC EBT transaction processing will be a function of the retailer application. Authentication of Store terminals will serve to reduce the risk of loss that may result from the use of unauthorized terminals to initiate and process fraudulent off-line WIC EBT transactions.

### **B.3.6 Authentication of the transaction**

It is recommended that the State authenticate and verify integrity of each transaction presented by a Store within its WIC claim file. Authentication of the transaction will involve a process that verifies the date, time and location of a transaction as well as a transaction signature. Authentication of the transaction serves to limit the risk of loss that may result from a fraudulent transaction being presented to the State within a claim file for processing and settlement.

---

<sup>6</sup> In smart card terminology, authentication involves the use of a shared secret between two parties where one party passes a cryptogram generated with that secret to the other party (internal or external authentication) or both pass it to each other (mutual authentication) to achieve access. Verification involves the presentation of the secret itself (e.g. a secret code, PIN or biometric template) to the smart card for validation. In this document, the term 'authentication' is meant to allow either process, unless otherwise specified.

### **B.3.7 State defined maximum balance on a given category/subcategory pair**

It is recommended that the State define a maximum balance for any given benefit category/subcategory pair. The definition of a maximum balance will be the responsibility of the State, presumably through existing or modified written agreements between the State and the Store for the processing of off-line WIC EBT transactions. The definition of a maximum balance for any given benefit category/subcategory pair will serve to limit the risk of loss that may result from a valid card issued at a state-controlled clinic, local agency or state office site being credited with a significantly larger volume of benefits or an incorrect type(s) of benefits than authorized for one or more specified benefit periods.

### **B.3.8 Tracking card issuance by WIC clinic operators**

It is recommended that the State record and track for each WIC EBT card issued the authorized WIC clinic system operator (i.e., clinic, local agency and/or state office personnel) who issued it. The tracking of card issuance by WIC system operators will be accomplished through events recorded by the WIC management information or related systems. The tracking of card issuances will serve to limit the risk of loss that may result from an issuance of a card and benefits to a non-eligible person.

### **B.3.9 Card validation of state authorized transactions**

It is recommended that States electing to implement posting of state authorized transactions in-lane do so in a manner that does not allow duplicate postings, e.g. the use of transaction sequence numbers when posting state authorized updates to a legitimate WIC EBT card.

## **B.4 Entity Responsibility**

### **B.4.1 State**

The responsibilities of the State include 1) authorizing a Store to process WIC transactions, after which the Store is a trusted party; 2) authenticating each WIC transaction presented to it by the Store, using the transaction signature; 3) maintaining the State Uniform Product Code (UPC) table and Hot Card List, making both available to the Store in a manner agreed to by both the State and Store; 4) maintaining a secure systems environment that ensures only an authorized clinic operator has issued a valid card to an eligible participant or proxy with WIC package benefits accessible for a specified benefit period(s); 5) providing through the card a transaction signature to be returned by the Store for use in authenticating each WIC EBT transaction; and 6) protecting debit and credit keys and codes, enabling balance inquiry and debit transactions only to occur in the Store and both credit and debit transactions to occur at a State-authorized clinic, local agency and state office site, thereby limiting the security required for debit transactions but imposing a very high level of security over credit transactions. The state or state-controlled clinic, local agency or state office site is responsible for recording and tracking card issuances by system user/operator.

### **B.4.2 Store**

The responsibilities of the Store include 1) authenticating the cardholder, the card and the operator; 2) collecting and forwarding the transaction signature to the State; 3) downloading the State Hot Card List and maintaining the store-level State UPC table to ensure only WIC approved food items are redeemed by an authorized cardholder, with a valid card; 4) ensuring security of the WIC application, including privacy of security keys and codes, card acceptor device level Cardholder verification through PIN, biometric or other means in a manner specified by the State, and WIC transaction processing performed only by authorized Store personnel (Operator), and at a states discretion, 5) posting of state authorized transactions.

## **B.5 Minimum Requirement Guidelines**

- Authentication of the cardholder via 4-digit (minimum) PIN, biometric or other means.
- Separation of credit and debit security codes and keys with no credit capability in-lane at the store
- Separation of credit and debit security codes, but allowing both credit and debit capability at state-controlled clinic, local agency and clinic sites
- Access/download of Hot Card List and State WIC UPC Database at least once every 48 hours
- Authentication of the card at the card level or the benefit level
- Transaction authentication via a transaction signature
- Tracking of card issuance by WIC clinic operator

## **B.6 Liability**

Federal regulations assign liability for WIC program administration and management to the State, and within the regulations, FNS officials interpret Federal, State and stakeholder roles, responsibilities and rights related to program quality, performance and integrity. In the off-line WIC EBT systems operating today, FNS has allowed a State to assign some or all of its primary responsibilities and accompanying liabilities to third parties. The most commonly functions outsourced today include 1) assignment of participant enrollment, eligibility determination, nutrition education and benefit issuance functions to a public or county health district, local agency and/or clinic; 2) delivery of WIC-authorized foods by WIC-authorized stores (i.e., pharmacies and/or small independent, regional, national or international chain stores; 3) store compliance monitoring, investigations and education; 4) stand-beside system design, development and deployment (i.e., hardware, software, site readiness and installation, help-desk, maintenance); 5) EBT card issuance, card management and replacement; and 6) EBT transaction processing, financial settlement and reconciliation reporting.

In addition to assigned or delegated liability, FNS and States must consider the costs of maintaining current with WIC EBT systems design and technologies, including the normal replacement upgrades to hardware and software and the scheduled replacement/upgrade of card technology. EBT systems must remain current in technology and design. The operational, financial and other impacts of technology and system upgrades should be carefully identified and assessed to determine whether which technical, operational or organizational alternative is the most feasible for WIC and key WIC stakeholders, particularly WIC authorized stores.

The USDA/FNS should continue to adopt policies promoting cost-effective implementations of off-line WIC EBT systems. Future EBT States should be required to adopt these guidelines unless a compelling argument is made for a waiver of system design or technology requirement that results in a significant, direct cost-savings or avoidance to the USDA/FNS. States with systems developed before these guidelines are implemented should be allowed to continue 'as is' at the discretion of USDA/FNS, provided they plan to adopt these guidelines through future system upgrades and enhancements.

Finally, as smart card technology advances and as applications using smart cards in government and industry mature, States will need to consider changes and/or upgrades to WIC EBT system design. Technology upgrades should be thoroughly defined, with fiscal, operational and organizational impacts identified and quantified for each key stakeholder group. To ensure the probability of success, it is recommended any change to WIC EBT system design being contemplated by the State be shared well in advance with key stakeholder groups, particularly Stores. The grocer community has indicated any change(s) to an offline WIC EBT system's requirements or to in-store WIC EBT system design or operations after deployment, must be

straightforward, not complex; cost neutral to the extent possible; high return (cost savings/avoidance); and scheduled well in advance of the proposed implementation date.

## Appendix C. WIC Services Manager (WSM)

### C.1 Overview

The purpose of this optional layer is to provide the retailer POS applications or the WIC Messaging Protocol a common interface to the potentially multiple WSPS modules loaded on a terminal. If the retailer decides to implement this layer, this layer shall perform three primary functions: Initialization of all of the WSPMs currently accessible underneath this layer, pass-through functions for accessing the WIC-EBT information from the smart card via the WSPS modules and retrieval of the PAN from the CCC for checking against a hotlist.

The WSM shall support the functions of the WSPI in the same manner with one modification. Unlike the similar function at the WSPI level, the call to `wsmInitiateSession()` takes only one parameter. The function list is as follows:

- `wsmInitiateSession()`
- `wsmGetPAN()`
- `wsmCardPresent()`
- `wsmConnectCard()`
- `wsmAuthenticateUser()`
- `wsmReadBalance()`
- `wsmDebitBalance()`
- `wsmBlockCard()`
- `wsmDisconnectCard()`
- `wsmPostStagedTransaction()`
- `wsmEndSession()`

The expected usage of this layer is first a call to `wsmInitiateSession()` is made when the POS application starts. This function will take whatever steps are necessary to initialize at least one WSPM so that the cards inserted may be read.

### C.2 WSM functions

The majority of the functions listed above behave as pass-through functions down to the WSPI layer. The WSM shall be responsible for directing the function call to the proper WSPS for the card presently inserted. All of the functions of the WSPI are repeated in the WSM with the following exception:

#### C.2.1 `wsmInitiateSession()`

**Purpose:** To enable initialization of all WSPMs currently loaded on the platform.

**Prerequisites:** none

**Prototype:** unsigned long `wsmInitiateSession`(  
IN unsigned char `ucProcessingMode`)

**Parameters:** **`ucProcessingMode`** A flag to indicate the manner in which the WIC transaction should function within the WSPM

<b>Mode</b>	<b>Description</b>
0	Normal Processing
1	Training
2	Certification
3	Certification/Training

**Return codes:** WSPI\_OK  
WSPI\_ACCESS\_DENIED  
WSPI\_BAD\_AUTH  
WSPI\_BAD\_PARAM  
WSPI\_UNKNOWN\_ERROR

## Appendix D. Reference Material

Throughout this document, references are made to a variety of external reference materials. The following list is intended to provide the sources/location of this material.

Reference	Section of Reference	Reference Location
ANSI		
GSC-IS	Version 2.0	
ISO	ISO-7816-4 (ISO-4)	
WIC SC-IS	WIC Smart Card Interoperability Specification	
X9.93		