



## Secure Profile Management Software Integration Kit (SPM-SIK)

The information provided in this document is intended for LiteScape internal use only. No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior authorization of Litescape Technologies, Inc.

Litescape Technologies, Inc.

[info@litescape.com](mailto:info@litescape.com)

[www.litescape.com](http://www.litescape.com)

## Table of Contents

<b>Secure Profile Management (SPM) Software Integration Kit (SIK)</b>	<b>1</b>
<b>Table of Contents</b>	<b>2</b>
<b>Table of Figures</b>	<b>4</b>
<b>1 Introduction</b>	<b>5</b>
<b>1.1 Brief Description</b>	<b>5</b>
<b>1.2 Purpose/Scope</b>	<b>5</b>
<b>1.3 Intended Audience</b>	<b>5</b>
<b>1.4 Contact Information</b>	<b>5</b>
<b>1.5 Related Documents and Reference Material</b>	<b>5</b>
<b>2 Functionality</b>	<b>6</b>
<b>2.1 Description</b>	<b>6</b>
<b>2.2 User Requirements</b>	<b>6</b>
<b>3 SPM-SIK Architecture</b>	<b>7</b>
<b>Card Read Scenario</b>	<b>9</b>
<b>3.1 SPM/SPAR Service</b>	<b>9</b>
<b>3.2 SPM Registration Server</b>	<b>9</b>
3.2.1 Message Description - Subscription	9
3.2.2 Message Description - Notification	12
<b>3.3 SPM/SIK SPAR Login Web Services</b>	<b>12</b>
<b>3.4 SPM Bio Server</b>	<b>13</b>
<b>3.5 SPM Card Read Transaction</b>	<b>15</b>
<b>3.6 DBAccess Layer (DAL)</b>	<b>16</b>
3.6.1 DBAccess Class	16
3.6.2 SPMDataAccess Classes	16
<b>3.7 Sample SPM-SIK_info.OCM file</b>	<b>20</b>
<b>3.8 SPM Biometric Simulator</b>	<b>20</b>
3.8.1 OnCast.Configuration XML file	22
3.8.2 SPM Simulator Log File	23
<b>4 Sample ServicePoint Applications</b>	<b>24</b>
<b>4.1 Sample ServicePoint Retail Application</b>	<b>24</b>
4.1.1 OnCastConfiguration.XML	26
4.1.2 ServicePointSession.XML	27
4.1.3 LotteryPrize.XML	27
4.1.4 ServicePointCardSwipe.pay	27
<b>4.2 ServicePoint Time Card Module</b>	<b>28</b>
4.2.1 Time Card Application	29
4.2.2 Time Card Application Integration Interface	30

4.2.3	ServicePoint Time Card Phone User Interface	30
4.2.4	OnCast.Configuration.XML	30
4.2.5	SPM_SIK_pwd_screen.ocm	33
4.2.6	OnCast.SPM.OCM.Mapping.xml	34
4.2.7	TimeCardUser.XML	35
4.2.8	OCM Pay files	35
<b>4.3</b>	<b>Integration with CUAE</b>	<b>37</b>
4.3.1	Time Card Application	37
4.3.2	Retail Application	37
<b>5</b>	<b>APPENDIX A: Glossary</b>	<b>39</b>
<b>6</b>	<b>APPENDIX B: LiteScape SPM Web Services</b>	<b>41</b>
<b>6.1</b>	<b>LS TimeCardService.WSDL</b>	<b>41</b>
6.1.1	LSTimeCardServiceRequest.XML	42
6.1.2	LSTimeCardServiceResponse.XML	45
<b>6.2</b>	<b>LSSparLoginWebServices.WSDL</b>	<b>48</b>
6.2.1	LSSparLoginWebServiceRequest	49
6.2.2	LSSparLoginWebServiceResponse	50
<b>7</b>	<b>Support Inquiry</b>	<b>52</b>

## Table of Figures

Figure 1	Typical use of SPM.....	8
Figure 2	SPM Bio interaction with SPAR .....	14
Figure 3	SPM Card Read Transaction .....	15
Figure 4	SPM Biometric Simulator Flow Diagram .....	21
Figure 5	SPM Biometric Simulator .....	22
Figure 6	Sample ServicePoint Retail (SSR) Application .....	24
Figure 7	Sample ServicePoint Retail Application Flow Diagram .....	25
Figure 8	Time Card Application .....	29
Figure 9	Time Card Application with CUAE integration .....	37
Figure 10	Retail Application with CUAE integration .....	38

## 1 Introduction

### 1.1 Brief Description

LiteScape's Secure Profile Management (SPM) works in conjunction with LiteScape SPAR (Secure Profile Authentication Reader) as part of a secure communications system that authenticates users for activating applications on their telephones.

The SPM Software Integration Kit (SPM-SIK) exposes the authentication portion of the SPM allowing third party application to utilize this functionality to personalize applications for end users after providing multi-factor authentication at the edge of converged networks.

### 1.2 Purpose/Scope

This document describes the designed behavior and architecture of SPM and its interaction with SPAR in relation to third party applications. The document also provides two sample applications built on top of the SPM/OnCast platform offerings.

### 1.3 Intended Audience

This document is intended for software developers who will utilize SPM functionality within their own application.

### 1.4 Contact Information

For development support, please contact [developer-support@litescape.com](mailto:developer-support@litescape.com).

### 1.5 Related Documents and Reference Material

Title	Created by	Version
<i>SFDS SPM(Internal document)</i>	<i>SPM Group</i>	<i>1.0</i>
<i>SPM Administrative guide</i>	<i>SPM Group</i>	<i>1.1</i>
<i>LiteScape SIK 4.0</i>	<i>SDK Group</i>	<i>4.0</i>
<i>LiteScape OnCast Directory Administrative Guide</i>	<i>OnCast Group</i>	<i>4.2</i>

## 2 *Functionality*

### 2.1 *Description*

The LiteScape Secure Profile Management (SPM) solution provides users with secure, personalized, and intuitive access to the following services from IP phones:

- Calls, conferences and broadcasts
- Multi-site extension mobility and device presence management
- Directories (corporate, personal public)
- XML applications
- Personal profile and preferences

The LiteScape SPM solution is comprised of the following two main components:

- Secure Profile Authentication Reader (SPAR) – is a hardware device that is associated with a phone and collects user authentication information and secure SPM-related communications at the endpoint.
- SPM platform – is responsible for collecting information from SPARs, security and authentication, connecting to business application and IP-PBXs.

### 2.2 *User Requirements*

As IP phones become more sophisticated, they may support existing enterprise applications to improve employee collaboration and productivity. Enterprises require the ability to secure access to the advanced services on IP phones while providing a high-level of personalization. Secure Profile Management (SPM) provides multi-factor authentication at the endpoint including biometric, magnetic card, RFID, and username/password. Upon authentication SPM is able to enable various services includes IP-PBX features, custom applications (e.g., time card) or directories that only the logged-in user may access.

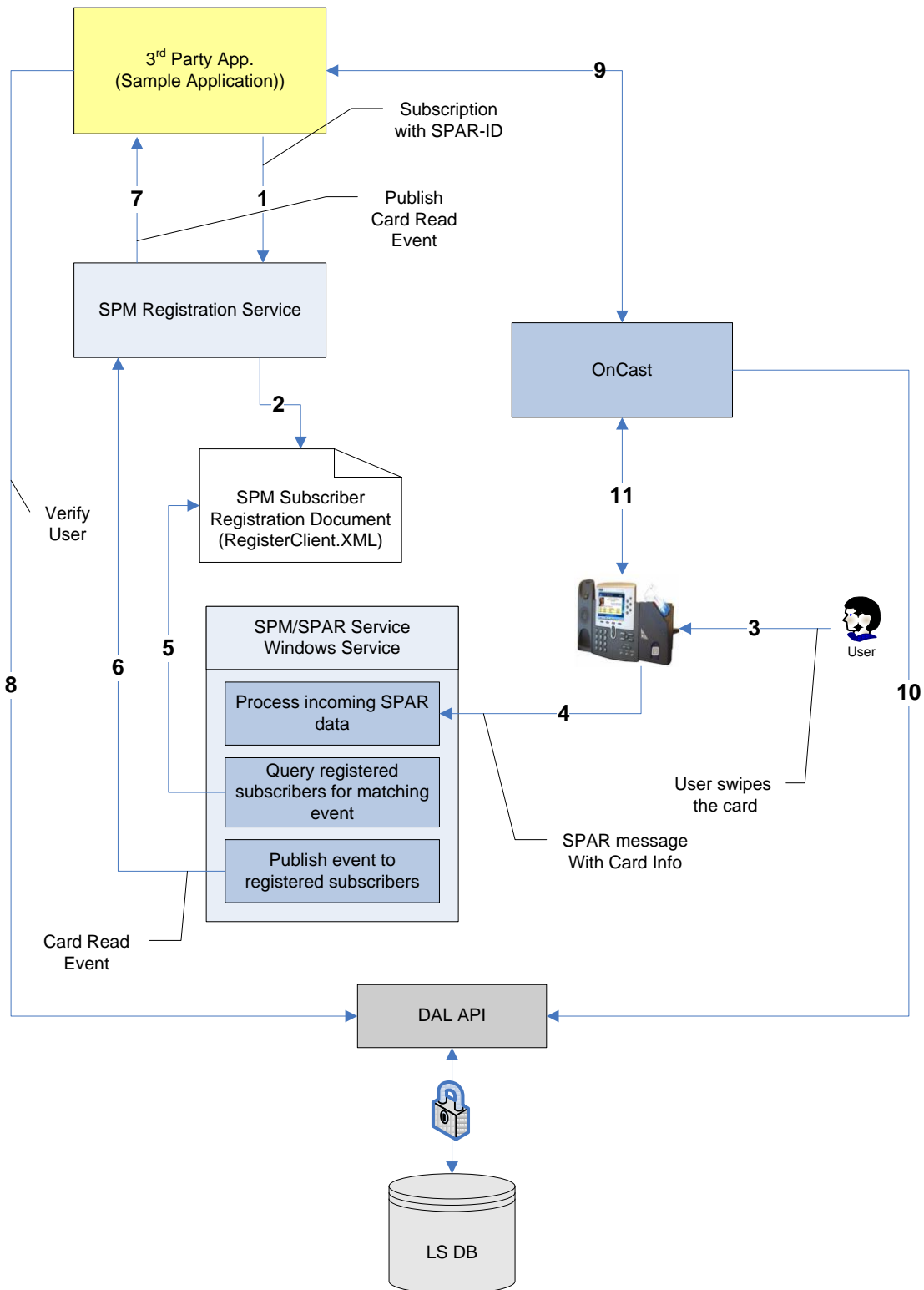
### ***3 SPM-SIK Architecutre***

This section describes the SPM architecture and related components for third party application integration.

A SPAR allows users to identify themselves using a magnetic card, biometric or RFID card. The SPAR encrypts and transmits multi-factor identification information to the LiteScape SPM application. The SPM application will verify the user's information against known identity servers and trigger an action such as transferring the user's profile to an external application, activating a third party application or enabling dialing privileges to the local phone.

SPM communicates with SPAR using various messages and ports (i.e., TCP, UDP). SPM has components that reside in Windows Service Manager, Internet Information Services (IIS), Web Services (login process), OnCast including WebAdmin, LiteScape Database Access Layer (DAL), and SNMP related functionality. The LiteScape database schema currently supports Microsoft SQL Server 2005.

The SPM application is not state-full and relies on the external application for this. The LiteScape database maintains session management in cases where the application requires tracking of login/logout based sessions.



**Figure 1 Typical use of SPM**

### ***Card Read Senario***

1. Third Party application registers itself with SPM Server and subscribe to card read event.
2. SPM Registration service persists subscriber's request in an XML file (RegisterClient.XML)
3. User swipes a card
4. SPM/SPAR service process the incoming data from SPAR card read event
5. SPM/SPAR service queries the subscribes for this specific event
6. SPM/SPAR service sends the event and user info along with subscriber info to SPM Registration Service
7. SPM sends the Card event and user information to subscriber (third party application)
8. Third party application authenticates user information by calling the LS DB Access Layer API (DAL)
9. Third party application calls OnCast with request to broadcast OCM
10. OnCast calls DAL for database interaction
11. OnCast broadcast third party application request to phone and gets the response back to third party application

#### ***3.1 SPM/SPAR Sevice***

SPM/SPAR Service starts as Windows service and runs as a socket server. It publishes SPAR events such as Magnetic Card RFID or biometric events to registered subscribers.

#### ***3.2 SPM Registration Server***

The SPM Registration Server manages the publication of requests to subscribers. Subscriptions include Internal Apps (for example the SPM WebAdmin and the SPM Web application service), SPM Biometric Service and SPM/SPAR Services and external applications (third party solutions).

The SPM Registration Server receives all event subscription requests (biometric, login/logout, and magnetic card/RFID).

Registration data is persistent in an XML file and shared with the SPM SPAR Reader Interface (SRI) components.

For example, to get notifications on activities from a SPAR, a third-party application needs to subscribe to listen for events. The application does this by sending the SPM registration server a subscription request message.

##### ***3.2.1 Message Description - Subscription***

<b>Tag</b>	<b>Value</b>	<b>Description</b>
SubscribID	Unique Client ID	This ID is used for subscribe, unsubscribe and send data etc.
SubscribType	SOCKET or HTTP	Set Method in which client want to receive data
SubscribUrl	URL address for HTTP	Required if SubscribType=HTTP
SubscribIP	IP address for socket	Required if SubscribType= SOCKET
SubscribPort	IP port for socket	Required if SubscribType= SOCKET

ReceiveCardDataFormat	dataformat.nonparsed_alphanumeric   dataformat.raw   dataformat.parsed	
SPAR	[A][B][C][D][E][F][G] e.g. to support Password and SMART-CARD default authentication will be AE	Password='A' RFID_CARD='B' BIOMETRIC='C' MAGNETIC_CARD='D' SMART_CARD='E' BAR_CODE_READER='F' JAPAN_RFID_READER='G'
Encrypt	True or False	Not implemented in this release
RegisterStatus	subscrib	Modified on register and unregister request from client
AutoSubscrib	True or False	if=True, The Client is not deleted, on unregister request (RegisterStatus=unsubscrib) from client
ForDeviceIP	IP of SPAR	Optional: Client can subscribe for events from this device IP
UserID	String	Required based on type of events (Bio verification and User Login/logout events)
BiometricAction	Enroll or Verify	Valid Biometric Action
SubscribeTimeout	Numeric value (in minutes)	Optional: Number of minutes elapsed before the subscriber will be auto-unregistered by the SPM if an unregistered request hasn't been previously sent

Each client must have a unique SubscriberID. The SPM Windows service will publish events to all registered subscribers.

**Note** the following for biometrics event registration:

- Registrations are valid for only one verification or enrollment session.
- Enrollment biometric prints are saved in a temporary database table (REFID=TEMP\_TABLE\_ID).
- Verification results are saved in a temporary database table (REFID=TEMP\_TABLE\_ID).
- Once the biometric capture is completed, SPM removes the biometrics event registration.

### 1. Example of Subscription Request – Magnetic Card/RFID

Spar type REFID='B';

Spar type MAG\_CARD='D';

```
<SubscriberID>10.13.0.186</SubscriberID>
<SubscriberType>HTTP</SubscriberType>
<SubscriberUrl>http://10.13.0.81:80/LSSparSPMWeb/LSSparWeb.aspx</SubscriberUrl>
<SubscriberIP> </SubscriberIP>
<SubscriberPort> </SubscriberPort>
<AutoSubscrib>False</AutoSubscrib>
<RegisterStatus>subscrib</RegisterStatus>
<ReceiveCardDataFormat>dataformat.nonparsed_alphanumeric</ReceiveCardDataFormat>
```

```

<SPAR>BD</SPAR>
<Encrypt>False</Encrypt>
<ForDeviceIP></ForDeviceIP>
<SubscribeTimeout>10</SubscribeTimeout>
<Action>subscrib</Action>
<UserID>andygibb.halim</UserID>
</RegisterClient>

```

## 2. Example of Subscription Request – Bio finger print

SPM will initiate connection to SPAR and instruct the device to start capturing the users' print.

Registration Request: BioMetricAction = verify/enroll. If BioMetricAction is empty, it is treated as enroll.

```

<SubscribID>10.13.0.186</SubscribID>
<SubscribType>HTTP</SubscribType>
<SubscribUrl>http://10.13.0.81:80/LSSparSPMWeb/LSSparWeb.aspx</SubscribUrl>
<SubscribIP>
</SubscribIP>
<SubscribPort>
</SubscribPort>
<AutoSubscrib>False</AutoSubscrib>
<RegisterStatus>subscrib</RegisterStatus>
<ReceiveCardDataFormat>dataformat.nonparsed_alphanumeric</ReceiveCardDataFormat>
<SPAR>C</SPAR>
<Encrypt>False</Encrypt>
<ForDeviceIP>10.13.0.186</ForDeviceIP>
<SubscribeTimeout>10</SubscribeTimeout>
<SubscribeTime>4/24/2006 2:41:02 PM</SubscribeTime>
<Action>subscrib</Action>
<UserID>andygibb.halim</UserID>
<BioMetricAction>verify</BioMetricAction>
</RegisterClient>

```

## 3. Example of Registration Request for UserEvent

```

<SubscribID>10.13.0.186</SubscribID>
<SubscribType>HTTP</SubscribType>
<SubscribUrl>http://10.13.0.81:80/LSSparSPMWeb/LSSparWeb.aspx</SubscribUrl>
<SubscribIP> </SubscribIP>
<SubscribPort> </SubscribPort>
<AutoSubscrib>False</AutoSubscrib>
<RegisterStatus>subscrib</RegisterStatus>
<ReceiveCardDataFormat>dataformat.nonparsed_alphanumeric</ReceiveCardDataFormat>
<SPAR>IJK</SPAR>
<Encrypt>False</Encrypt>
<ForDeviceIP></ForDeviceIP>
<SubscribeTimeout>10</SubscribeTimeout>
<SubscribeTime>4/24/2006 2:41:02 PM</SubscribeTime>
<Action>subscrib</Action>
<UserID>andygibb.halim</UserID>
</RegisterClient>

```

## 4. Example of un-subscription Request

```
<RegisterClient>
  <SubscribID>CARD_ENROLL</SubscribID>
  <RegisterStatus>unsubscrib</RegisterStatus>
</RegisterClient>
```

### 3.2.2 Message Description - Notification

Internal (Web Admin) or external (third party) applications who have subscribed to SPM Server will receive the following callback message based on a card or biometrics event:

Tag	Value	Description
SPARType	[A][B][C][D][E][F][G][I][J][K] e.g. to support Password and RFID-CARD default authentication will be AB	Password='A' RFID_CARD= 'B' BIOMETRIC= 'C' MAGNETIC_CARD= 'D' SMART_CARD= 'E' BAR_CODE_READER= 'F' JAPAN_RFID_READER= 'G' USER_BLOCKED='I' USER_LOGGED_IN='J' USER_LOGGED_OUT='K'
SPARIP	IP Address	IP Address of SPAR where the event was triggered from
BioStatus	Status code	Status code during a fingerprint enrollment
RefId	Some reference id	The reference id to the record containing the data in the database

#### 1. Example of Event data response to Subscriber - Magnetic Card/RFID

```
"type=CardEvent&S=CRServer&clientIP=" + mCardDataIn.RemoteIP
    + "&SPARType="+mCardDataIn.SPARType D or B
    + "&SPARIP="+mCardDataIn.RemoteIP
    + "&BioStatus=0"
    + "&REFID="+mREFID
    + "&D=";
```

#### 2. Example of Event data response to Subscriber – Bio finger print

```
ype=BioStatus&S=bio&SPARType=5&clientIP=10.13.0.186&SPARIP=10.13.0.186&BioStatus=18&Event
SeverityLevel=1&REFID=&D=
```

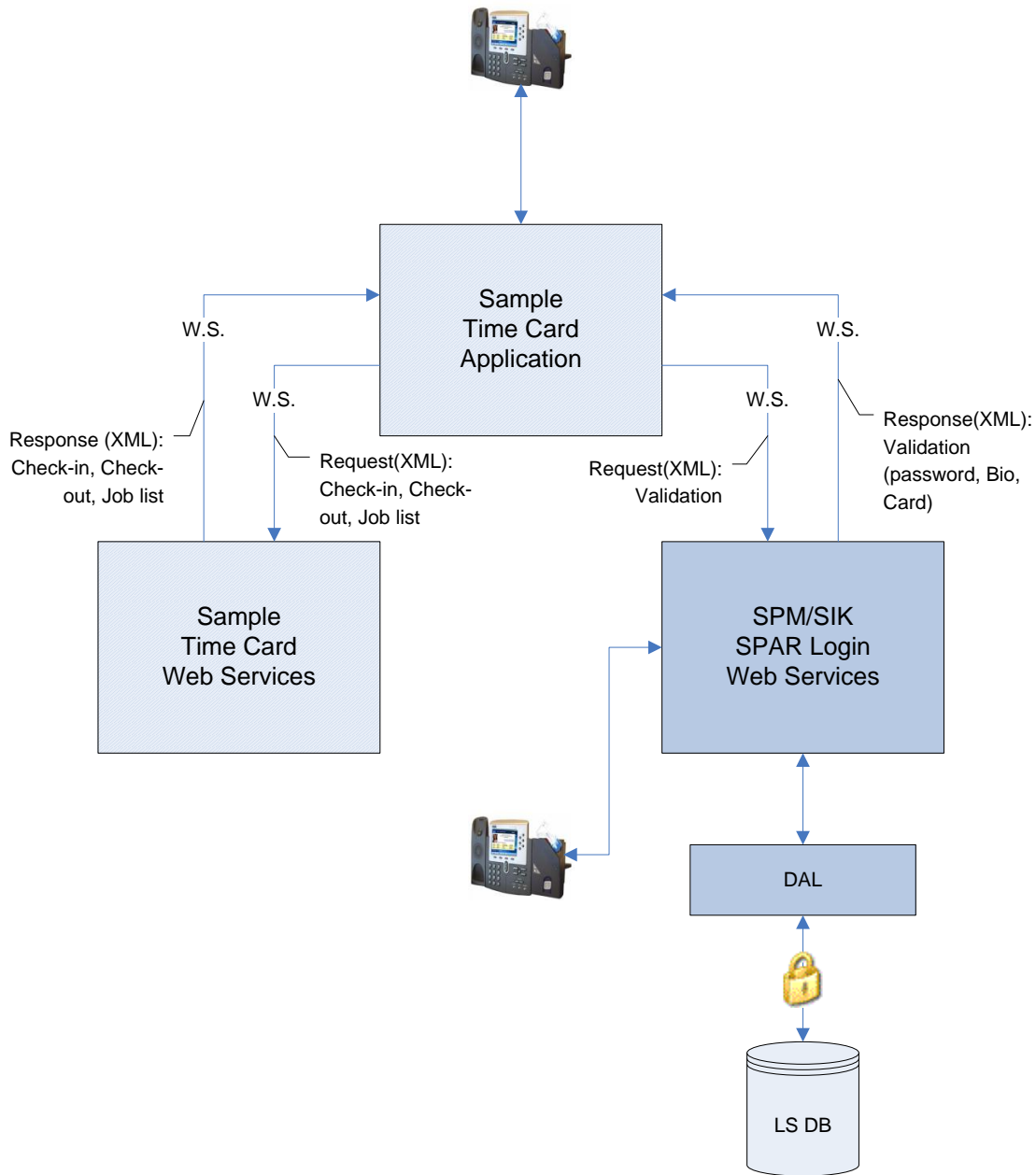
#### 3. Example of Event data response to Subscriber – User Event

```
UserID=UserID&UserEvent=I OR J or K
```

### 3.3 SPM/SIK SPAR Login Web Services

SPAR Login Web Services was designed to allow third party application utilize the user verification functionality in SPM. User information such as Password, Bio, Card (MAG and RFID) can be collected and verified from LiteScape Database via Login Web Services. Validation Request is sent by application

in XML format to the Login Web Services, Response from Web Service will be sent by Session-ID. Once the validation is completed the XML response will be sent to the call back URL specified in the Request XML.



### 3.4 SPM Bio Server

The SPM Bio Server starts as a Windows service and runs as a socket server. It publishes fingerprint captures and matches events to registered subscribers.

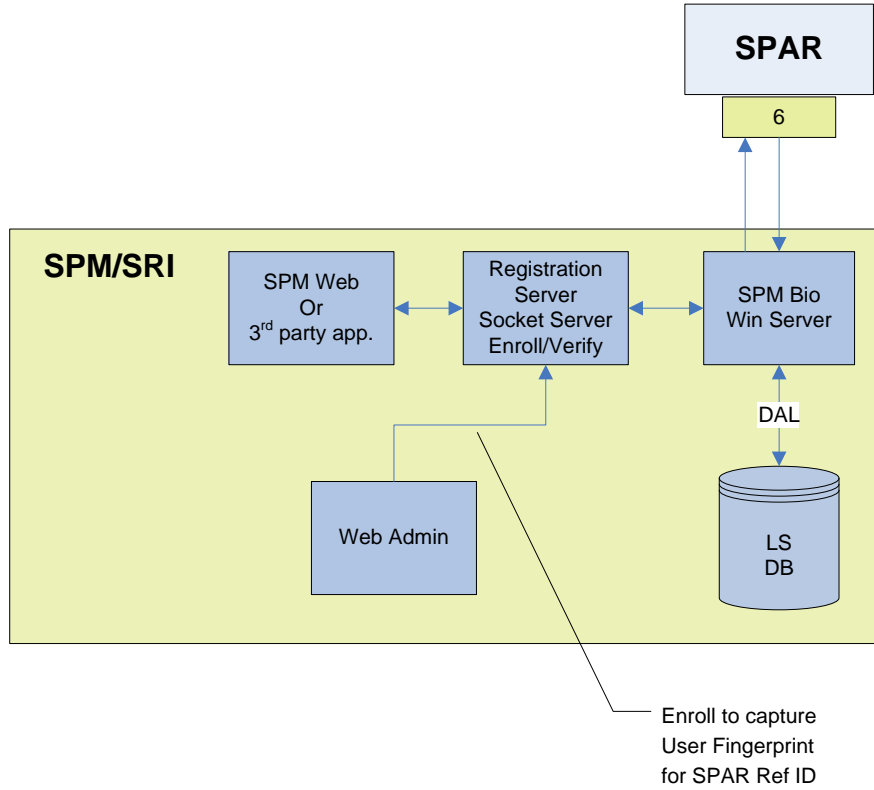


Figure 2: SPM Bio interaction with SPAR

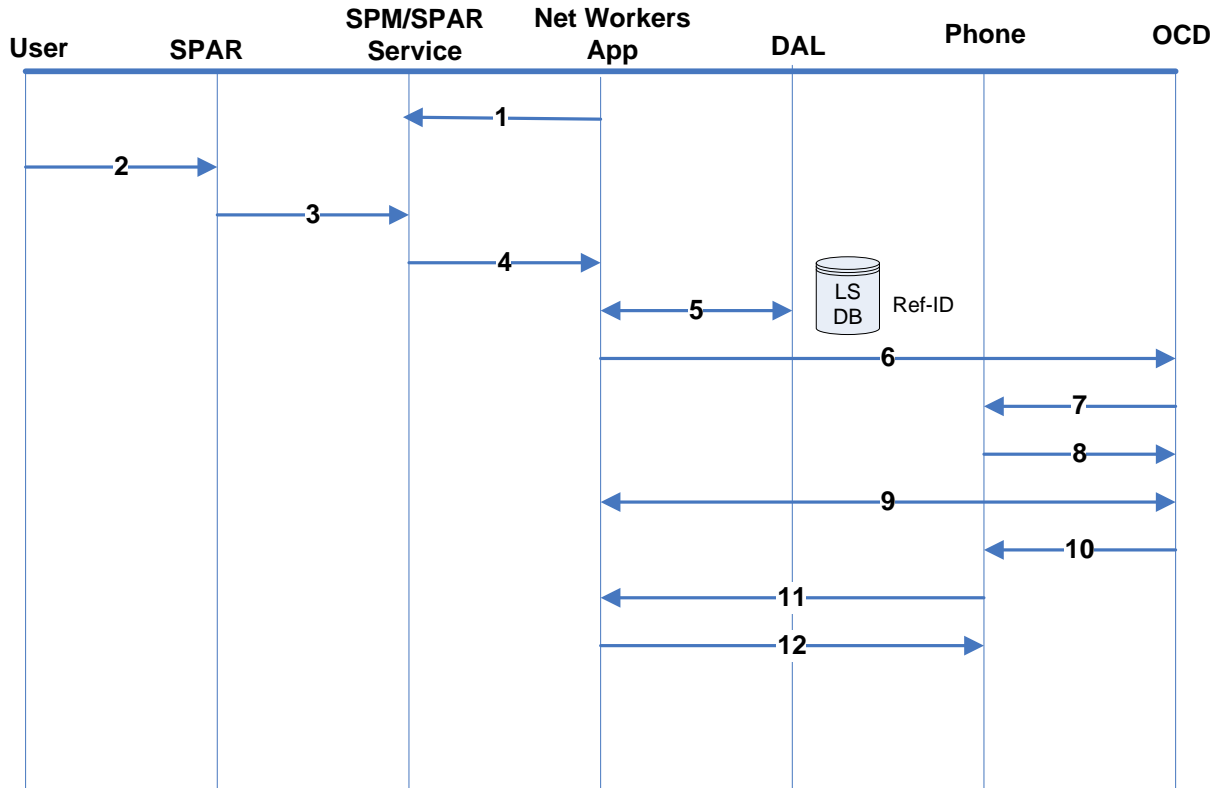


Figure 3 SPM Card Read Transaction

### 3.5 SPM Card Read Transaction

1. External Application subscribes to SPM Registration with SPAR-ID
2. User swipes a card
3. SPAR sends a message with card info to SPM
4. SPM Registration sends a Card Read event to External Application
5. External Application sends a DAL request to DB to identify and verify the user
6. External Application creates an OCM/Send Broadcast request to OnCast
7. OnCast broadcasts to phone
8. Phone sends plays button to OnCast
9. Phone sends the content back to OnCast
10. OnCast acts upon External Application XML command file and sends command to phone
11. Phone UI displays XML command from External Application
12. External Application responds to Phone UI

### 3.6 DBAccess Layer (DAL)

This is the abstraction layer for access to identity management information (such as database or directory access).

The following steps must be completed prior to using any of the Database classes:

- Create an environment variable for your machine named **LITESCAPE\_HOME** and, as the value, put in the path to some directory on your computer (it does not matter where as long as it is a valid path).
- Create a **config** sub-directory under the above directory.
- Copy the SPM supplied **dtb.pr** file to the **config** directory (please note that this file has been encrypted with your machine specific information, which means it will only work on that machine).

The following are the main classes in DAL layer

#### 3.6.1 DBAccess Class

This is the LiteScope main DBAccess class in the DB.Net.LSC.DBAccess namespace.

The `init()` method in the DBAccess class needs to be called only **once**. For example:

```
DB.Net.Lsc.DBAccess.DBAccess.init();
```

#### 3.6.2 SPMDDataAccess Classes

This is the SPM specific classes in the DB.Net.Lsc.DBProxies namespace.

Any of the following convenience methods in the **SPMDDataAccess** class can be called after the initial DBAccess call:

- **public static String getSPMSettingsXml()**  
*It will return you the following XML:*

```
<getSpmGlobalSettings>
  <GlobalSettings>
    <SPMSettings>
      <ShortIdleTimeout>20</ShortIdleTimeout>
      <MediumIdleTimeout>30</MediumIdleTimeout>
      <LongIdleTimeout>40</LongIdleTimeout>
      <MaxInvalidLogins>3</MaxInvalidLogins>
    </SPMSettings>
    <EnrollmentSettings>
      <CardServer ip="10.13.0.81" port="10015"/>
      <BioServer ip="10.13.0.81" port="3000"/>
      <SubscriptionTimeout value="5"/>
      <AuthLogin pwd="LSCITAdmin" usrName="LSCITAdmin"/>
      <AuthenticationFactors>
        <AuthenticationFactor id="PIN" required="Y"/>
        <AuthenticationFactor id="MAG" required="Y"/>
        <AuthenticationFactor id="BIO" required="Y"/>
        <AuthenticationFactor id="RFID" required="N"/>
      </AuthenticationFactors>
    </EnrollmentSettings>
  </GlobalSettings>
</getSpmGlobalSettings>
```

- **public static String getSparKeyByIp(String SPARIP)**  
This method takes the IP address of a SPAR device and returns a String (in Base64) that represents the encryption key for the SPAR device
- **public static String getPhoneSPARProfileXmlFromSPARIP(String SPARIP)**  
This method takes the IP address of a SPAR device and returns the Phone-SPAR profile xml (see below)
- **public static String getPhoneSPARProfileXmlFromDeviceIP(String deviceIP)**  
This method takes the IP address of a phone device and will return you the Phone-SPAR profile xml (see below) based on the SPAR device associated with the phone

The Phone-SPAR XML returned will match the following tags:

```
<CardReader>
  <PhoneMAC>000E839C108A</PhoneMAC>
  <PhoneIP>192.168.50.22</PhoneIP>
  <CardReaderIP>192.168.50.88</CardReaderIP>
  <BioMetricReaderIP>192.168.50.25</BioMetricReaderIP>
  <RFIDMAC_Autold>0020F7FFFFEF</RFIDMAC_Autold>
  <RFIDIP_Autold>192.168.100.90</RFIDIP_Autold>
  <BarCodeReaderIP>192.168.100.90</BarCodeReaderIP>
  <UserExtentionMobilitySupport>true</UserExtentionMobilitySupport>
  <SPMTypes>ABC</SPMTypes>
</CardReader>
```

- **public static String getUserProfileXmlFromCardInfo(String cardNumber, String cardType)**  
This method takes the card number and card type and returns the profile XML of the matching user, if any

The User XML returned will match the following tags:

```
<UserCard>
  <UserID>scmorris</UserID>
  <UserIDLCASE>scmorris</UserIDLCASE>
  <CardID>1401E0070000024DB6E3</CardID>
  <PIN>1401E0070000024DB6E3</PIN>
  <FirstName>Scott</FirstName>
  <LastName>Morrison</LastName>
  <DisplayName></DisplayName>
</UserCard>
```

- **public static String storeUserBioInfo(byte[] data)**  
This method takes the bytes of a fingerprint template and will return an XML, which contains the reference id that should be passed back to any subscribers who's interested in this event

The XML returned will look like the following:

```
<ensureImage>XXX</ensureImage>
```

Where XXX corresponds to the reference Id in an actual call

- **public static byte[] getUserBioInfo(String userId)**  
This method takes the UserId of a User and returns the bytes of the User's fingerprint template.

- **public static String ensureIdentificationDevice(String cardNumber, String cardType)**  
This method takes the card number and type and will return an XML, which contains the reference id that should be passed back to any subscribers who's interested in this event

The XML returned will match the following tag:

```
<ensureIdentDevice>XXX</ensureIdentDevice>
```

Where XXX corresponds to the reference Id in an actual call

- **public static void ensureUserSession(String doc)**

This method stores/updates user's session information.

The parameter should be in the following XML format and this XML is the basis for a user's login/logout DB session/report information:

```
<UserSession>
<TimeCreated>2/23/2006 3:06:09 PM</TimeCreated>
  <UserID>23124</UserID>
  <PhoneIP>10.13.0.79</PhoneIP>
  <SparIP>10.13.0.184</SparIP>
  <FirstName></FirstName>
  <LastName></LastName>
  <SPARTypeCurrent>A</SPARTypeCurrent>
  <SPARTypeCurrentFailedCount>3</SPARTypeCurrentFailedCount>
  <SPARTypesCompleted>BCD</SPARTypesCompleted>
  <isSPARTypesAllCompleted>False</isSPARTypesAllCompleted>
  <StatusFlag>1/2/3</StatusFlag>
--event type 1. Login in process, 2. Logged in 3. Logout in process 4. Logged out
</UserSession>
```

- **public static void deleteUserSession(String userid, String sparIP)**

This method should be used to delete an existing user session based on his/her userID and SPAR IP

- **public static String getUserSession(String userid, String sparIP, String deviceIP)**

This method should be used to retrieve an existing user session based on his/her userID and either a SPAR IP or a phone device IP (cannot be both). To retrieve by phone device IP, just make sure that it is not null. Otherwise, if you want to look up by SPAR IP, just pass in the correct SPAR IP and null for the device IP.

- **public static String getLogKey()**

This method is unrelated to session management, but it can be used to retrieve the encryption key to encrypt data logged at finest level. The XML returned will match the following tag:

```
<getLogKey>uy2Y3qnuoqzNtRqq6fNjQQADjqb/Z+c9</getLogKey>
```

The key is encoded in Base64, so you need to decode it first and get the resulting bytes to be used as the key for the encryption. SPM uses AES as the encryption algorithm with key strength of 256 bits in ECB mode.

- **public static String getSpmGlobalValues()**

This method retrieves global SPAR settings, which includes the two logins and the encryption key. This is useful in the case of handling device reset as we discussed earlier. There is already an existing method in DAL that will return the default SPAR settings.

The XML returned will match the following tags:

```
<query_group>
  <getSpmDeviceAdminInfo>
    <user userName="admin_user_name" pwd="admin_password"/>
  </getSpmDeviceAdminInfo>
  <getSpmDeviceUserInfo>
    <user userName="user2_user_name" pwd="user2_password"/>
  </getSpmDeviceUserInfo>
  <getSpmDeviceKeyStored>000102030405060708090a0b0c0d0e0f10111213141
  5161718191a1b1c1d1e1f</getSpmDeviceKeyStored>
</query_group>
```

**Note:**

The getSpmDeviceAdminInfo tag refers to device login 1.

The getSpmDeviceUserInfo tag refers to device login 2.

- **public static String getSPMSettingsXml()**

The following is the XML that will be returned when calling this method in DAL

The XML returned is:

```
<getSpmGlobalSettings>
  <GlobalSettings>
    <SPMSettings>
      <LockedIdleTimeout unit="hours" value="60"/>
      <LongIdleTimeout unit="hours" value="540"/>
      <LongExtensionPeriod unit="hours" value="120"/>
      <WarningMessageTiming unit="minutes" value="15"/>
      <MaxInvalidLogins value="3"/>
    </SPMSettings>
    <EnrollmentSettings>
      <CardServer ip="10.13.0.81" port="10015"/>
      <BioServer ip="10.13.0.81" port="10015"/>
      <SubscriptionTimeout value="8"/>
      <AuthLogin pwd="LSCITAdmin" usrName="LSCITAdmin"/>
      <AuthenticationFactors>
        <AuthenticationFactor id="PIN" required="Y"/>
        <AuthenticationFactor id="MAG" required="Y"/>
        <AuthenticationFactor id="BIO" required="Y"/>
        <AuthenticationFactor id="RFID" required="Y"/>
      </AuthenticationFactors>
    </EnrollmentSettings>
  </GlobalSettings>
</getSpmGlobalSettings>
```

Note the MaxInvalidLogins element (in bold above), whose value represents the number of invalid login attempts a user can have before the user account associated with the user is locked.

### 3.7 *Sample SPM-SIK\_info.OCM file*

```

<?xml version="1.0"?>
<OnCastMessage>
  <Version>0.95</Version>
  <Header>
    <Type>TEXT</Type>
    <Priority>EMERGENCY</Priority>
    <Sender>andygibb.halim</Sender>
    <EmergencyRingTone>Classic2.raw</EmergencyRingTone>
    <NormalRingTone>Classic2.raw</NormalRingTone>
    <Recipients>
      <Recipient>10.13.0.79</Recipient>
    </Recipients>
  </Header>
  <ExitKey>3</ExitKey>
  <Message>
    <Prompt>Powered by LiteScape</Prompt>
    <Subject>SPM</Subject>
    <Body>Welcome to Litescape SPM Application.</Body>
  </Message>
  <Audio>
    <Type>NONE</Type>
    <Streaming>UNICAST</Streaming>
  </Audio>
  <Video>
    <FileName>
    </FileName>
  </Video>
  <Presentation>
    <Count>0</Count>
  </Presentation>
  <Survey />
  <Stocks />
  <RSSFeeds />
  <Workflow>

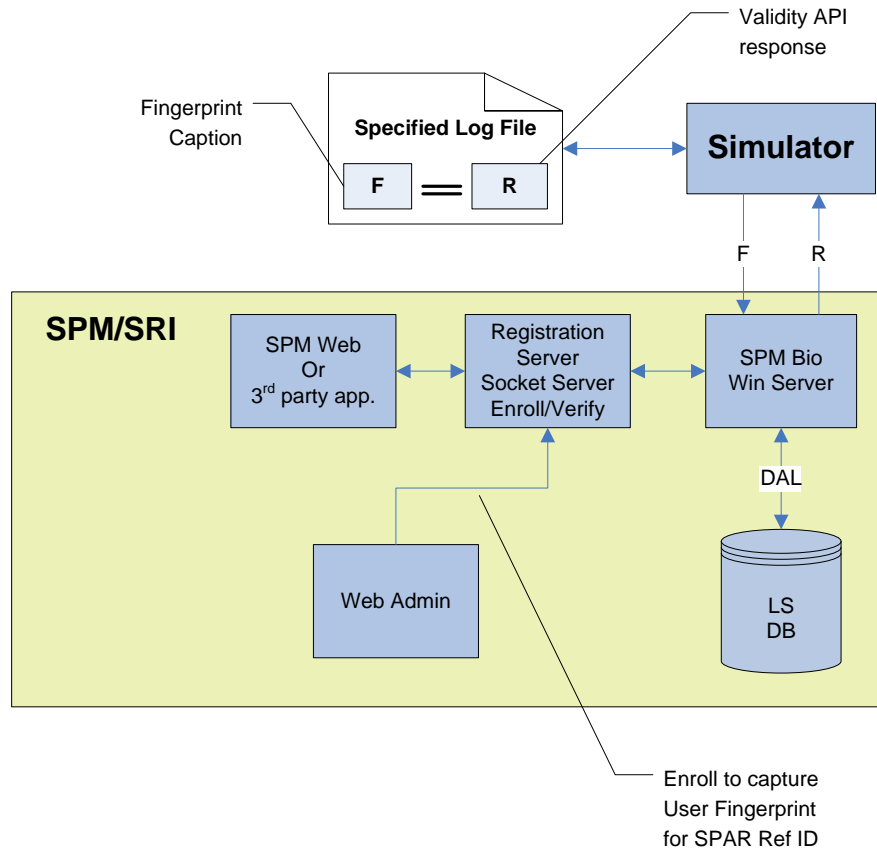
  </Workflow>
</OnCastMessage>

```

### 3.8 *SPM Biometric Simulator*

SPM Biometric Simulator simulates SPAR fingerprint captures by reading a file that has logged pairs of SPAR sensor fingerprints and corresponding Validity API response commands in ASCII format.

The SPAR fingerprint reader data and its response command are captured into a log file and are used to simulate a real-time SPAR for testing or application writing purpose.



**Figure 4 SPM Biometric Simulator Flow Diagram**

The default installation path for SPM Biometric Simulator is:

C:\Program Files\Litescape\LitescapeSparBioSimulationServer\

The SPM Biometric Simulator is invoked by **LitescapeSparBioSimulatorServer.exe** in the above directory path.

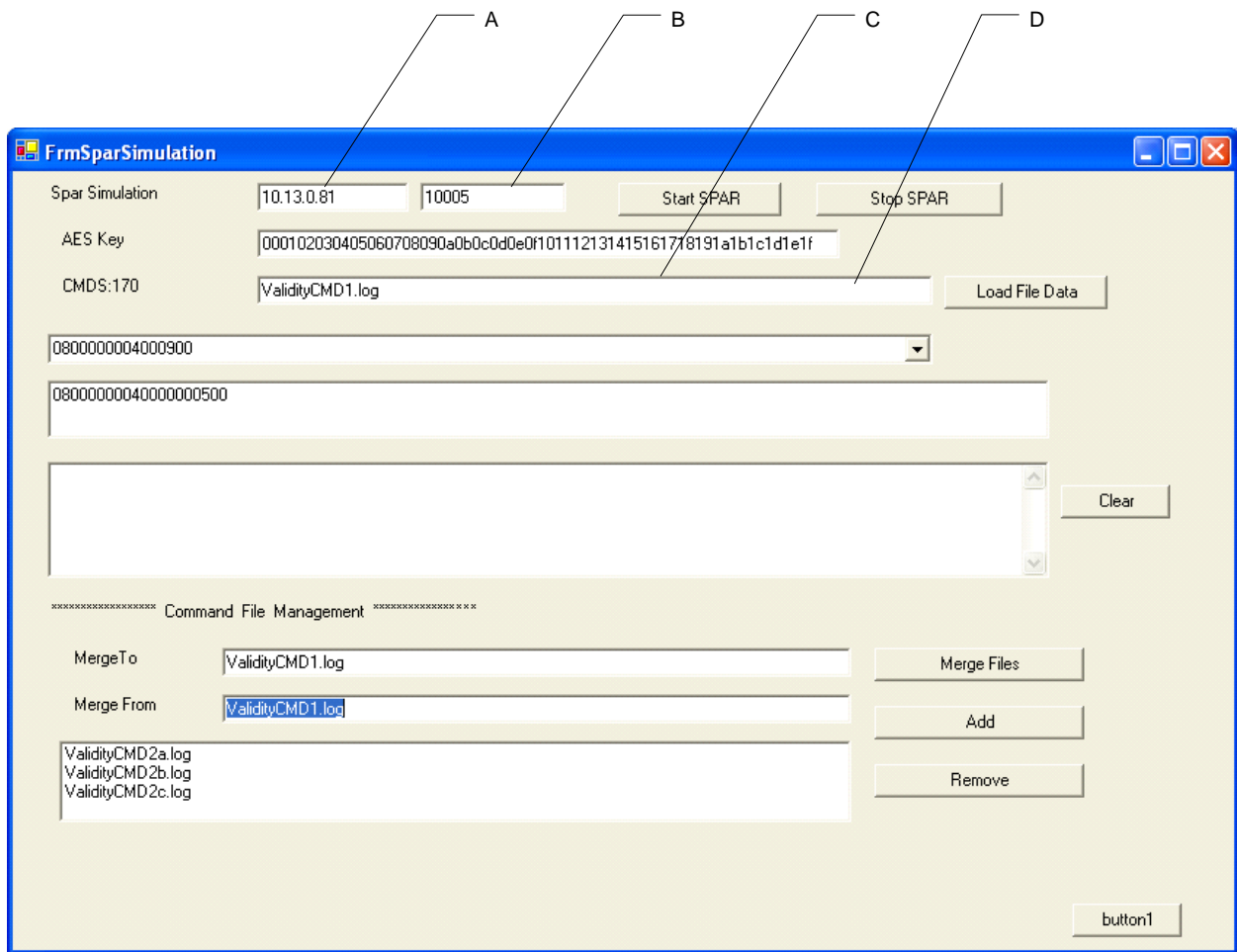
Currently four SPAR fingerprint capture files are available with installation and user can select which log file to run in the C field (ValidityCMD1.log, ValidityCMD2a.log, ValidityCMD2b.log, and ValidityCMD2c.log)

User should not modify these files as they are true SPAR fingerprint capture data and simulator will not work if these files are manually edited.

The SPM Biometric Simulator configuration is shown in Figure 5

Note:

- A – SPAR IP or Configuration IP (OnCast.config)
- B – SPAR Port #
- C – Select Caption File
- D – Select Fingerprint Pair



**Figure 5 SPM Biometric Simulator**

By running SPM Biometric Simulator, SPM server behaves the same as if real-time SPAR was running and capturing fingerprint data.

### 3.8.1 OnCast.Configuration XML file

OnCast.Configuration.XML file includes tags to support the SPM Biometric Simulator.

<spm>

```

<!--
EnableForceBioMetricIPTo: true /false. If true the real SPAR IP address is neglected, and
ForceBioMetricIPTo is used as bio server IP address
ForceBioMetricIPTo: bio simulation server IP address
ForceBioMetricIP_ForIP: this is SPAR IP address
→
    <EnableForceBioMetricIPTo>true</EnableForceBioMetricIPTo>
      <ForceBioMetricIPTo>10.13.0.81</ForceBioMetricIPTo>
        <ForceBioMetricIP_ForIP>10.13.0.186</ForceBioMetricIP_ForIP>
</spm>

```

### 3.8.2 SPM Simulator Log File

Example of first 5 lines of a log file is shown below. Note that each pair format is: Fingerprint Capture = Response command from Validity API in ASCII format.

[BioCommands]

```

vfsCommGetSelectedConfiguration=
0100000004002E00=01000000040000000200
02000000020000=0200000002000000564653313031202D2076657220322E3330000000000000000000
00000000000000000000000000000000
0300000004002800=03000000040000000100
0400000004001100=04000000040000000400

```

## 4 Sample ServicePoint Applications

Third Party Application can utilize SPM authentication functionality via the exposed interface explained in this document.

### 4.1 Sample ServicePoint Retail Application

This Sample ServicePoint Retail (SSR) is a lottery game that involves at least one customer and one cashier station. Customers use pre-registered RFID and/or magnetic card readers to enter the game.

By using SPM functionality and SPAR card reader, SSR authenticates the registered cards. Each card number draws from the lottery to win or lose. SSR then utilizes the OnCast to broadcast and push XML commands and display messages to the IP phones.

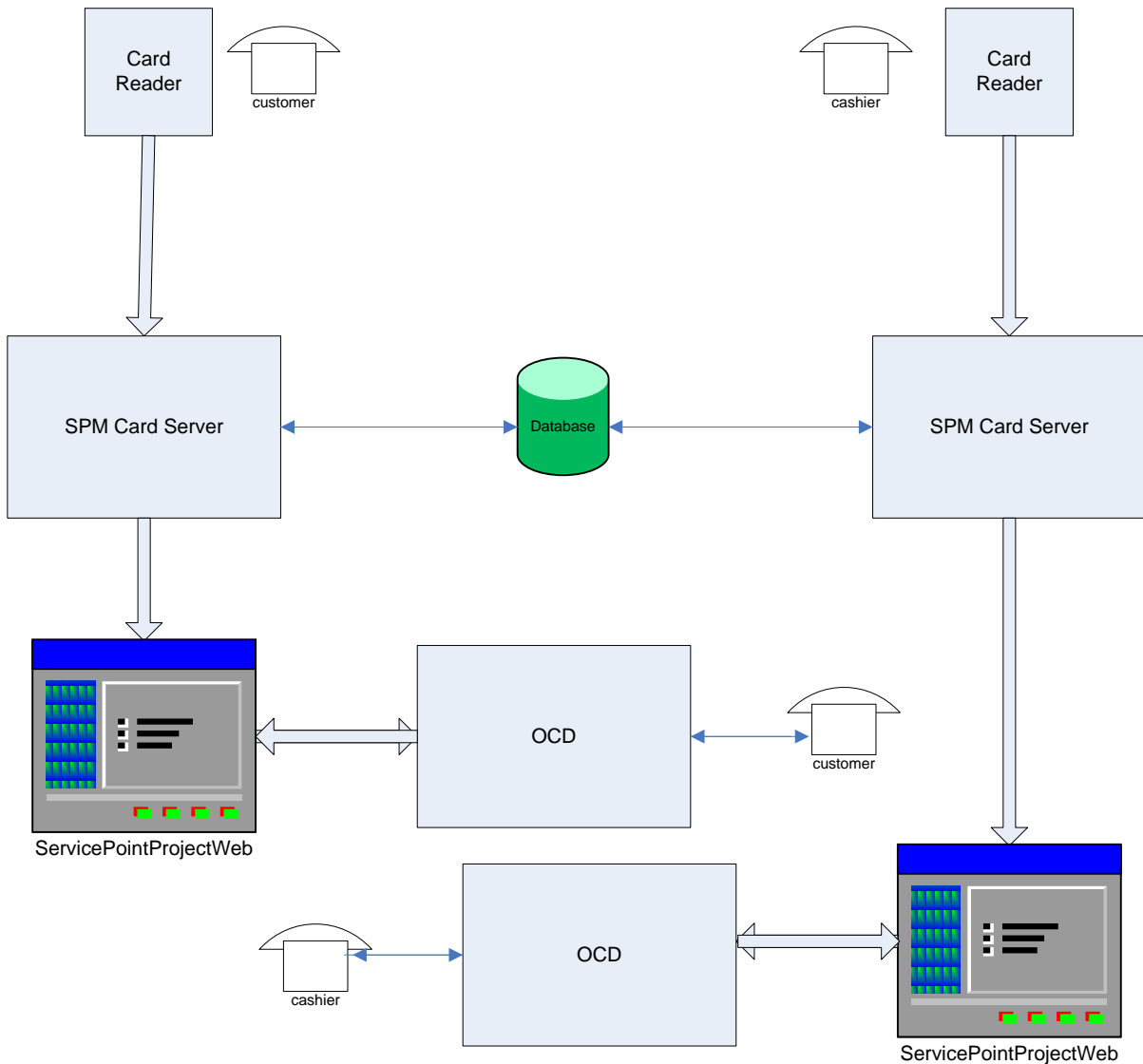
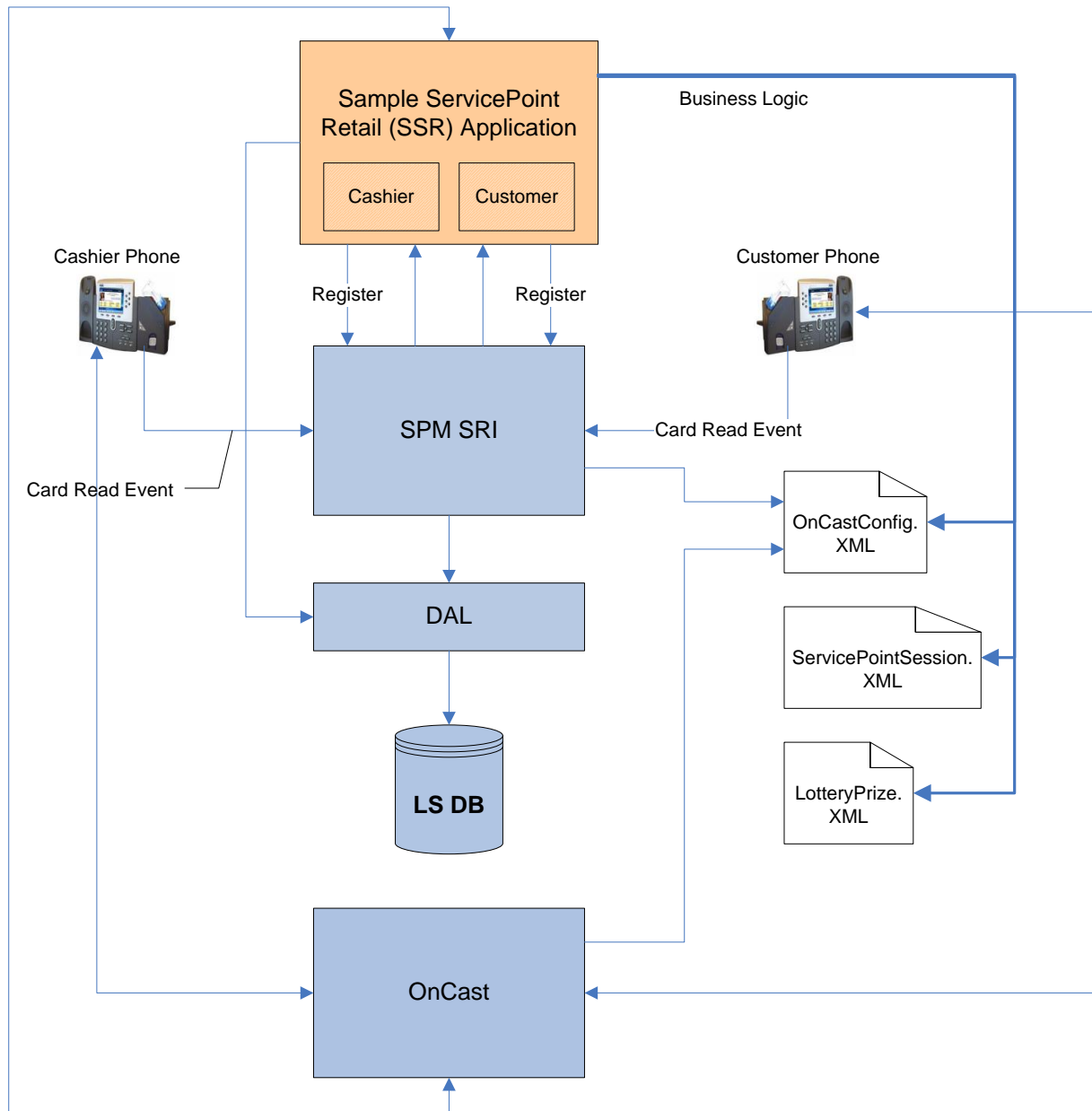


Figure 6 Sample ServicePoint Retail (SSR) Application



**Figure 7 Sample ServicePoint Retail Application Flow Diagram**

The SPM server is the initial contact with SSR. After the first invocation of the application, the control passes to SSR and OnCast (back and forth). OnCast acts between SSR and the IP phone as a content rendering and delivery engine and SSR makes decisions based on business logic, user's choices, and available data.

When the user swipes a card, SPAR sends a request to the SPM where the card is identified. SPM invokes ServicePointProjectWeb, which in turn calls OnCast. Based on the type of station the card was

swiped on (customer or cashier) the appropriate OnCast payload file is invoked which pushes the appropriate OnCast Message (OCM file) to the phone with emergency priority. The OCM is published and presented to the phone that made the original call to the SPM server.

When the user makes a choice on the screen of the phone, the phone sends the request to OnCast which in turn sends it to ServicePointProject application for analysis. Based on the nature of the request the appropriate files are updated, an OCM is invoked and pushed to the phone.

The following XML files implement the business logic and persistence of the session state and data.

#### 4.1.1 *OnCastConfiguration.XML*

OnCastConfiguration.XML must be modified to adopt the application business logic in SPM.

```

- <ServicePoint>
- <LotteryDrawRange>
- <D>
  <NumberStart>1</NumberStart>
  <NumberEnd>60</NumberEnd>
</D>
- <B>
  <NumberStart>1</NumberStart>
  <NumberEnd>60</NumberEnd>
</B>
</LotteryDrawRange>
- <XML_DB>

<ServicePointRetail_Play_Xml>ServicePoint/ServicePointCardSwipe.xml</ServicePointRetail_Play_Xml>

<ServicePointRetail_Collect_Xml>ServicePoint/ServicePointCollect.xml</ServicePointRetail_Collect_Xml>
  <ServicePointRetail_Play_ocm>ServicePointCardSwipe.ocm</ServicePointRetail_Play_ocm>
  <ServicePointRetail_Collect_ocm>ServicePointCollect.ocm</ServicePointRetail_Collect_ocm>

<ServicePointRetail_Error_Xml>ServicePoint/ServicePointError.xml</ServicePointRetail_Error_Xml>
  <ServicePointRetail_Error_ocm>ServicePointError.ocm</ServicePointRetail_Error_ocm>
- <!--
LotteryResult_Xml>LotteryResult.xml</LotteryResult_Xml
-->
<ServicePointSession_Xml>ServicePointSession.xml</ServicePointSession_Xml>
<LotteryPrizes_Xml>LotteryPrizes.xml</LotteryPrizes_Xml>
<BookList_Xml>BookList.xml</BookList_Xml>
</XML_DB>
- <ServicePoint_Messages>
- <ServicePoint_Message>
  <id>1</id>
  <name>IN_COMPLETE_CARD_READ_EVENT</name>
  <status_msg>Either the system got incomplete read from Magnetic/RFID card reader or the card is unrecognized, please try again.</status_msg>
</ServicePoint_Message>
</ServicePoint_Messages>
</ServicePoint>

```

## 4.1.2 *ServicePointSession.XML*

*ServicePointSession.xml* maintains the session data and is updated during the game. *CurrentStage* tag holds User status data. This tag defines user business logic (win or lose) and ultimately is converted to an OCM file to be pushed to the phone.

```
<ServicePointSession>
  <User>
    <UserID>andygibb.halim</UserID>
    <SparIP>10.13.0.186</SparIP>
    <PhoneIP>10.13.0.79</PhoneIP>
    <LastName>Halim</LastName>
    <FirstName>AndyGibb</FirstName>
    <IsWonPrize>True</IsWonPrize>
    <Prize>USB Mug Warmer</Prize>
    <CurrentStage>won</CurrentStage>
    <ImageName>mug.png</ImageName>
    <SPARType>B</SPARType>
  </User>
</ServicePointSession>
```

## 4.1.3 *LotteryPrize.XML*

*LotteryPrizes.xml* is manually created but is updated during the session to hold the session state as well.

As the user wins prizes, the file *LotteryPrizes.xml* is updated to reflect the number of prizes left. At the start of the game, value of *<AvailablePrizes>* is equal to *<TotalPrize>*. In this example 10 prizes have been given away so far.

```
<LotteryPrize>
  <SparType>B</SparType>
  <Name>Vinette Burgundy Ballpoint Pen</Name>
  <ImageName>pen.png</ImageName>
  <TotalPrize>90</TotalPrize>
  <AvailablePrizes>80</AvailablePrizes>
  <LotteryDrawRange_Start>4</LotteryDrawRange_Start>
  <LotteryDrawRange_End>30</LotteryDrawRange_End>
</LotteryPrize>
```

## 4.1.4 *ServicePointCardSwipe.pay*

This OnCast payload document corresponds to the OCM collection:

```
<?xml version="1.0"?>
<OnCastMessage>
  <Version>0.95</Version>
  <Header>
    <Type>IMAGE</Type>
    <Priority>EMERGENCY</Priority>
    <Sender>andygibb.halim</Sender>
    <EmergencyRingTone>Classic2.raw</EmergencyRingTone>
    <NormalRingTone>Classic2.raw</NormalRingTone>
```

```

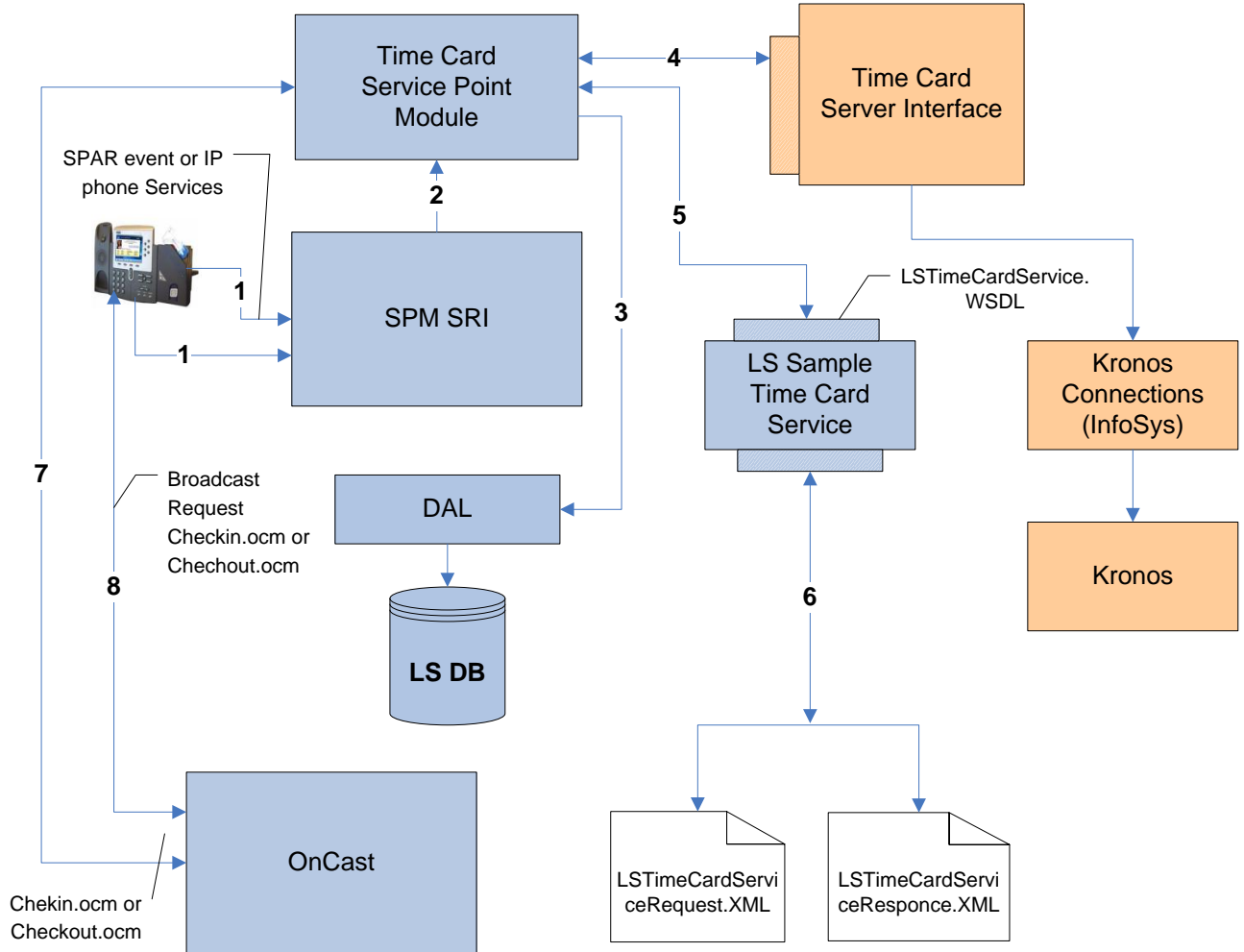
    <Recipients>
      <Recipient>10.13.0.79</Recipient>
    </Recipients>
  </Header>
  <Message>
    <Prompt>Powered by LiteScape</Prompt>
    <Subject>Lottery</Subject>
    <Body>andygibb.halim, Welcome to ServicePoint Application. Press play to win a
prize!</Body>
  </Message>
  <Audio>
    <Type>NONE</Type>
    <Streaming>UNICAST</Streaming>
  </Audio>
  <Video>
    <FileName>
    </FileName>
  </Video>
  <Presentation>
    <Count>0</Count>
  </Presentation>
  <Survey />
  <Stocks />
  <RSSFeeds />
  <Workflow>
    <Key>
      <Key>
        <Caption>Play</Caption>
        <Actions>
          <Action>
            <ID>URLService</ID>
            <Settings>
              <pluginValue>
                <URL>http://10.13.0.81/LSServicePointWeb/LotteryCustomerWeb.aspx?phoneIP=10.13.0.79&a
mp;S=P&SPARIP=10.13.0.186&userid=andygibb.halim&mystage=play&sub
tage=1&SPARType=B</URL>
              </pluginValue>
            </Settings>
          </Action>
        </Actions>
      </Key>
    </Key>
  </Workflow>
</OnCastMessage>

```

## 4.2 ServicePoint Time Card Module

The ServicePoint Time Card Module is a LiteScape component that enables third party implementation of time card activities. The ServicePoint Time Card Module utilizes SPM authorization module to validate user by biometric, magnetic card, RFID or username/password. It then utilizes OnCast to render and deliver the appropriate content using OnCast message files (OCMs) to phone and receive responses. A WSDL based interface defines Check-in, Checkout, Get Status, and Job Change logic. See Appendix B for detail on the Time Card Service WSDL.

A user may invoke the time card using either magnetic card, RFID, username/password, or biometrics. The Time Card Application (TCA) must have already registered with the SPM registration server. The SPM authorization server will verify the user's identity. The TCA will call the GetStatus WSDL, calls Checkin WSDL, and sends the Checkin.OCM to the phone via OnCast. The second time the user invokes uses the SPAR, the GetStatus WSDL recognizes that the user is already checked in and therefore the Checkout WSDL is called and Checkout.OCM will be sent to phone.



**Figure 8: Time Card Application**

### 4.2.1 Time Card Application

1. SPAR event caused by magnetic card, RFID, username/password, or biometrics.
2. SPM authenticate the user (user profile is already entered into LS database).
3. SPM sends proper event to Time Card module
4. Time Card module interfaces to third party application via LSTimeCardService WSDL
5. Or Time Card module interfaces to LS Sample Time Card Service via LSTimeCardService WSDL

6. Business logic is read from XML documents
7. Time Card module calls OnCast functionality depending on the business logic (creating OCM file, broadcast, etc.)
8. OnCast Broadcasts OCM file to phone and sends response back to Time Card Module

## 4.2.2 Time Card Application Integration Interface

The ServicePoint Time Card Module integration relies on a web services based architecture. Refer to Appendix B for a WSDL definition of the web services that need to be implemented by the third-party TCA.

## 4.2.3 ServicePoint Time Card Phone User Interface

The following OnCast Messages (OCM files) are used to provide the user with a phone interface for check-in, checkout and change job functions. These OCM files are customizable using OnCast and are located at: [Data\oncast\BroadcastTemplates\TimeCard\\*.ocm](#)

**CheckIn.ocm** – provides a form on the phone for the user to check-in with the third party application or change their job code.

**CheckOut.ocm** – provides a form on the phone for the user to check-out with the third party application or change their job code.

## 4.2.4 OnCast.Configuration.XML

The following modifications to OnCast.Configuration.XML are required to allow the application flow to work. The OnCast.Configuration.XML is located at: [Data\oncast\AppData\TimeCard\\*.xml](#)

Go to:

```
<CardEventSubscrib>
  <RegisterClient>
```

EDIT

SUBSCRIB URL SECTION:

```
<SubscribUrl>http://10.13.0.81/LSTimeCardWeb/LSTimeCardWeb.aspx</SubscribUr
l>
```

Add the Timecard xml segment below to the root level document (<OnCastConfiguration> and find and replace the tag TIMECARDSERVER with the IP address of your SIK deployed web-service. If everything is running on the same server, this would be the IP address of the server being used.

```
<TimeCard>
  <Authenticate>
    <EncryptionMethod>plain</EncryptionMethod>
    <UserID>TimeCardUser</UserID>
    <Password>TimeCardUserPwd</Password>
  </Authenticate>
  <Image>
    <ImageURLPath>LSTimeCardWeb/image/</ImageURLPath>
    <CheckInImage>TimeCard_CheckIn.png</CheckInImage>
    <CheckOutImage>TimeCard_CheckOut.png</CheckOutImage>
  </Image>
  <Buttons>
    <Button>
      <ButtonID>login</ButtonID>
```

```

        <ButtonDisplay>Login</ButtonDisplay>
    </Button>
    <Button>
        <ButtonID>logout</ButtonID>
        <ButtonDisplay>Logout</ButtonDisplay>
    </Button>
    <Button>
        <ButtonID>jobssel</ButtonID>
        <ButtonDisplay>JobSel</ButtonDisplay>
    </Button>
    <Button>
        <ButtonID>status</ButtonID>
        <ButtonDisplay>Status</ButtonDisplay>
    </Button>

    <Button>
        <ButtonID>sparlogin</ButtonID>
        <ButtonDisplay>SPARLogin</ButtonDisplay>
    </Button>

</Buttons>

<WebServiceURLs>
    <WANBroadcast /> <!-- If TimeServer is deployed separately, this parameter has to be
populated with the address of the corresponding OnCast Directory Server Broadcast Web-
Service -->
    <TimeCardWebService>
        http://localhost:80/LSTimeCardWeb/LSTimeCardService.asmx
    </TimeCardWebService>
    <TimeCardWebASPX>
        http://TIMECARDSERVER/LSTimeCardWeb/LSTimeCardWeb.aspx
    </TimeCardWebASPX>
    <SparLoginWebService>
        http://localhost/LSTimeCardWeb/LSSparLoginWebService.asmx
    </SparLoginWebService>
    <SparLoginWebAspx>
        http://TIMECARDSERVER/LSTimeCardWeb/ValidateUser.aspx
    </SparLoginWebAspx>
</WebServiceURLs>

<XML_DB>
    !-- Time Card ocm -->
    <TimeCard_CheckIn_ocm>TimeCard\TimeCard_CheckIn.ocm</TimeCard_CheckIn_ocm>
    <TimeCard_CheckIn_Xml>TimeCard\TimeCard_CheckIn.xml</TimeCard_CheckIn_Xml>
    <TimeCard_Error_ocm>TimeCard\TimeCard_Error.ocm</TimeCard_Error_ocm>
    <TimeCard_Error_Xml>TimeCard\TimeCard_Error.xml</TimeCard_Error_Xml>

    !-- Spar Login ocm -->
    <SparLogin_FristScreen_ocm>TimeCard\SparLogin_FristScreen.ocm</SparLogin_FristScreen_ocm>
    <SparLogin_FristScreen_Xml>TimeCard\SparLogin_FristScreen.xml</SparLogin_FristScreen_Xml>
    <SparLogin_Error_ocm>TimeCard\TimeCard_Error.ocm</SparLogin_Error_ocm>
    <SparLogin_Error_Xml>TimeCard\TimeCard_Error.xml</SparLogin_Error_Xml>
    !-- Xml -->
    <TimeCardUser_Xml>TimeCard\TimeCardUser.xml</TimeCardUser_Xml>
    <TimeCardSession_Xml>TimeCard\TimeCardSession.xml</TimeCardSession_Xml>
    <TimeCardLogs_Xml>TimeCard\TimeCardLogs.xml</TimeCardLogs_Xml>
    <TimeCardJobCode_Xml>TimeCard\TimeCardJobCode.xml</TimeCardJobCode_Xml>

```

```

    <TimeCardWebSession_Xml>TimeCard\TimeCardWebSession.xml</TimeCardWebSession_Xml>
    <SparLoginUser_Xml>TimeCard\TimeCardUser.xml</SparLoginUser_Xml>
  </XML_DB>
  <SPAR>
    <PasswordSPAR>true</PasswordSPAR>
    <MagRFIDCardSPAR>true</MagRFIDCardSPAR>
    <BioMetricSPAR>true</BioMetricSPAR>
  </SPAR>
</ValidationClientWrapper>
  <WrapperAppMapping>
    <!--
      LS_APP_NAME: do not change LS_APP_NAME tag value
      AppID: use lower case AppID
    -->
    <LS_APP_NAME>card_event_wrapper_aspx_impl</LS_APP_NAME>
    <AppID>called_from_card_event</AppID>
  </WrapperAppMapping>
  <WrapperAppMapping>
    <LS_APP_NAME>phone_service_wrapper_aspx_impl</LS_APP_NAME>
    <AppID>called_from_phone</AppID>
  </WrapperAppMapping>
</ValidationClientWrapper>
  <ValidationClientAppUrlMapping>
    <!--
      AppID: always use lower case for AppID value.
      LS_APP_NAME: map AppID to LS_APP_NAME tag in ValidationClientWrapper.
    -->
    <App>
      <AppID>default_app</AppID>

    <CallbackUrl>http://10.13.0.81/LSTimeCardWeb/LSTimeCardWeb.aspx</CallbackUrl>
    <SPARTypesSupported>AC</SPARTypesSupported>
    <SPARLoginSequence>AC</SPARLoginSequence>
  </App>
  <App>
    <AppID>called_from_card_event</AppID>

    <CallbackUrl>http://10.13.0.81/LSTimeCardWeb/LSTimeCardWeb.aspx</CallbackUrl>
    <SPARTypesSupported>AC</SPARTypesSupported>
    <SPARLoginSequence>AC</SPARLoginSequence>
  </App>
  <App>
    <AppID>called_from_phone</AppID>

    <CallbackUrl>http://10.13.0.81/LSTimeCardWeb/LSTimeCardWeb.aspx</CallbackUrl>
    <SPARTypesSupported>BAC</SPARTypesSupported>
    <SPARLoginSequence>BAC</SPARLoginSequence>
  </App>
</ValidationClientAppUrlMapping>

  <SPARBioMetricSimulation>
    <SPARMapping>
      <SPAR_IP>10.13.0.187</SPAR_IP>
      <BioSimulationServerIP>10.13.0.81</BioSimulationServerIP>
    </SPARMapping>
    <SPARMapping>
      <SPAR_IP>10.13.0.187</SPAR_IP>
      <BioSimulationServerIP>10.13.0.81</BioSimulationServerIP>
    </SPARMapping>
  </SPARBioMetricSimulation>

```

- Add and configure new tags to <SPM></SPM>  
<!--

Replacing OLD tags : ForceBioMetricIPTo, ForceBioMetricIP\_ForIP  
with New tags : RedirectBioMetricIPMapping

if EnableForceBioMetricIPTo ==true than use RedirectBioMetricIPMapping tag.

Each SPAR\_IP can be mapped to only one BioMetricIP  
And vice versa i.e. Each BioMetricIP can be mapped to only one SPAR\_IP.

```
-->
<RedirectBioMetricIPMapping>
  <RedirectBioMetricIP>
    <SPAR_IP>10.13.0.187</SPAR_IP>
    <BioMetricIP>10.13.0.81</BioMetricIP>
  </RedirectBioMetricIP>
</RedirectBioMetricIPMapping>
```

#### 4.2.5 SPM\_SIK\_pwd\_screen.ocm

```
<?xml version="1.0"?>
<OnCastMessage>
  <Version>0.95</Version>
  <Header>
    <Type>INPUT</Type>
    <Priority>EMERGENCY</Priority>
    <Sender>andygibb.halim</Sender>
    <EmergencyRingTone>Classic2.raw</EmergencyRingTone>
    <NormalRingTone>Classic2.raw</NormalRingTone>
    <Recipients>
      <Recipient>10.13.0.79</Recipient>
    </Recipients>
  </Header>
  <ExitKey>3</ExitKey>

  <Message>
    <Prompt>Powered by LiteScape</Prompt>
    <Subject>Time Card</Subject>
    <Body>Welcome to Litescape Time Card Application.</Body>
  </Message>
  <Audio>
    <Type>NONE</Type>
    <Streaming>UNICAST</Streaming>
  </Audio>
  <Video>
    <FileName>
  </FileName>
  </Video>
  <Presentation>
    <Count>0</Count>
  </Presentation>
  <Survey />
  <Stocks />
  <RSSFeeds />
```

```

<Forms>
  <URL></URL>
  <InputItems>
    <DisplayName>UserID</DisplayName>
    <QueryStringParam>U</QueryStringParam>
    <DefaultValue></DefaultValue>
    <InputFlag></InputFlag>
  </InputItems>
  <InputItems>
    <DisplayName>Password</DisplayName>
    <QueryStringParam>P</QueryStringParam>
    <DefaultValue></DefaultValue>
    <InputFlag></InputFlag>
  </InputItems>
</Forms>

<Workflow>
</Workflow>
</OnCastMessage>

```

#### 4.2.6 *OnCast.SPM.OCM.Mapping.xml*

This file contains the broadcast information such as location of templates, menus, and forms.

```

<!--
SPM_SIK
-->
= <OCM>
  <OCM_ID>SPM_SIK_INFO</OCM_ID>

  <OCM_FILE_PATH>BroadcastTemplates\spm_sik\SPM_SIK_info.ocm</OCM_FILE_PATH>
</OCM>
= <OCM>
  <OCM_ID>SPM_SIK_USER_FORM</OCM_ID>

  <OCM_FILE_PATH>BroadcastTemplates\spm_sik\SPM_SIK_pwd_screen.ocm</OCM_FILE_PATH>
</OCM>
= <!--
      <OCM>
        <OCM_ID>SPM_SIK_PROFILE_SELECTION_MENU</OCM_ID>

        <OCM_FILE_PATH>BroadcastTemplates\spm\SPM_SIK_profileSel_screen.ocm</OCM_FILE_PATH>
      </OCM>
      <OCM>
        <OCM_ID>CLUSTER_SELECTION_MENU</OCM_ID>

        <OCM_FILE_PATH>BroadcastTemplates\spm\SPM_SIK_clusterSel_screen.ocm</OCM_FILE_PATH>
      </OCM>
-->
- <!--

```

```

Time_Card
-->
- <OCM>
  <OCM_ID>TIME_CARD_INFO</OCM_ID>

  <OCM_FILE_PATH>BroadcastTemplates\timecard\TIME_CARD_info.ocm</OCM_
  FILE_PATH>
  </OCM>
- <OCM>
  <OCM_ID>TIME_CARD_USER_FORM</OCM_ID>

  <OCM_FILE_PATH>BroadcastTemplates\timecard\TIME_CARD_pwd_screen.oc
  m</OCM_FILE_PATH>
  </OCM>
  - <!--
    <OCM>
      <OCM_ID>SPM_SIK_PROFILE_SELECTION_MENU</OCM_ID>

      <OCM_FILE_PATH>BroadcastTemplates\spm\SPM_SIK_profileSel_screen.ocm</O
  CM_FILE_PATH>
      </OCM>
      <OCM>
        <OCM_ID>CLUSTER_SELECTION_MENU</OCM_ID>

        <OCM_FILE_PATH>BroadcastTemplates\spm\SPM_SIK_clusterSel_screen.ocm</O
  CM_FILE_PATH>
        </OCM>

  -->

```

#### 4.2.7 TimeCardUser.XML

Open Application Data\oncast\AppData\TimeCard\TimeCardUser.xml

Add one <User></User> section for all users you'd like to have access to the TimeCard application and the allowed job codes they can select from when using the LiteScape TimeCard ServicePoint module (the job codes are defined in the TimeCardJobCode.xml document and can be edited as well).

Each user specified here should have a corresponding entry in the SPM application (needs to be 'enrolled' into SPM for password, biometrics, RFID and magnetic card validation):

```

<User>
  <UserID>andygibb.halim</UserID>
  <LastName>Halim</LastName>
  <FirstName>AndyGibb</FirstName>
  <Department>qa</Department>
  <JobCode>job_1</JobCode>
  <JobCodes>job_1,job_2,job_3,job_4,job_5,job_6</JobCodes>
</User>

```

#### 4.2.8 OCM Pay files

The following is an example of SPM-SIK OCM Pay file.

```

<?xml version="1.0"?>
<OnCastMessage>
  <Version>0.95</Version>
  <Header>
    <Type>INPUT</Type>
    <Priority>EMERGENCY</Priority>
    <Sender>andygibb.halim</Sender>
    <EmergencyRingTone>Classic2.raw</EmergencyRingTone>
    <NormalRingTone>Classic2.raw</NormalRingTone>
    <Recipients>
      <Recipient>10.13.0.79</Recipient>
    </Recipients>
  </Header>
  <ExitKey>3</ExitKey>

  <Message>
    <Prompt>Powered by LiteScape</Prompt>
    <Subject>Time Card</Subject>
    <Body>Welcome to Litescape Time Card Application.</Body>
  </Message>
  <Audio>
    <Type>NONE</Type>
    <Streaming>UNICAST</Streaming>
  </Audio>
  <Video>
    <FileName>
  </FileName>
  </Video>
  <Presentation>
    <Count>0</Count>
  </Presentation>
  <Survey />
  <Stocks />
  <RSSFeeds />

  <Forms>
    <URL></URL>
    <InputItems>
      <DisplayName>UserID</DisplayName>
      <QueryStringParam>U</QueryStringParam>
      <DefaultValue></DefaultValue>
      <InputFlag></InputFlag>
    </InputItems>
    <InputItems>
      <DisplayName>Password</DisplayName>
      <QueryStringParam>P</QueryStringParam>
      <DefaultValue></DefaultValue>
      <InputFlag></InputFlag>
    </InputItems>
  </Forms>

  <Workflow>
</Workflow>
</OnCastMessage>

```

## 4.3 Integration with CUAE

### 4.3.1 Time Card Application

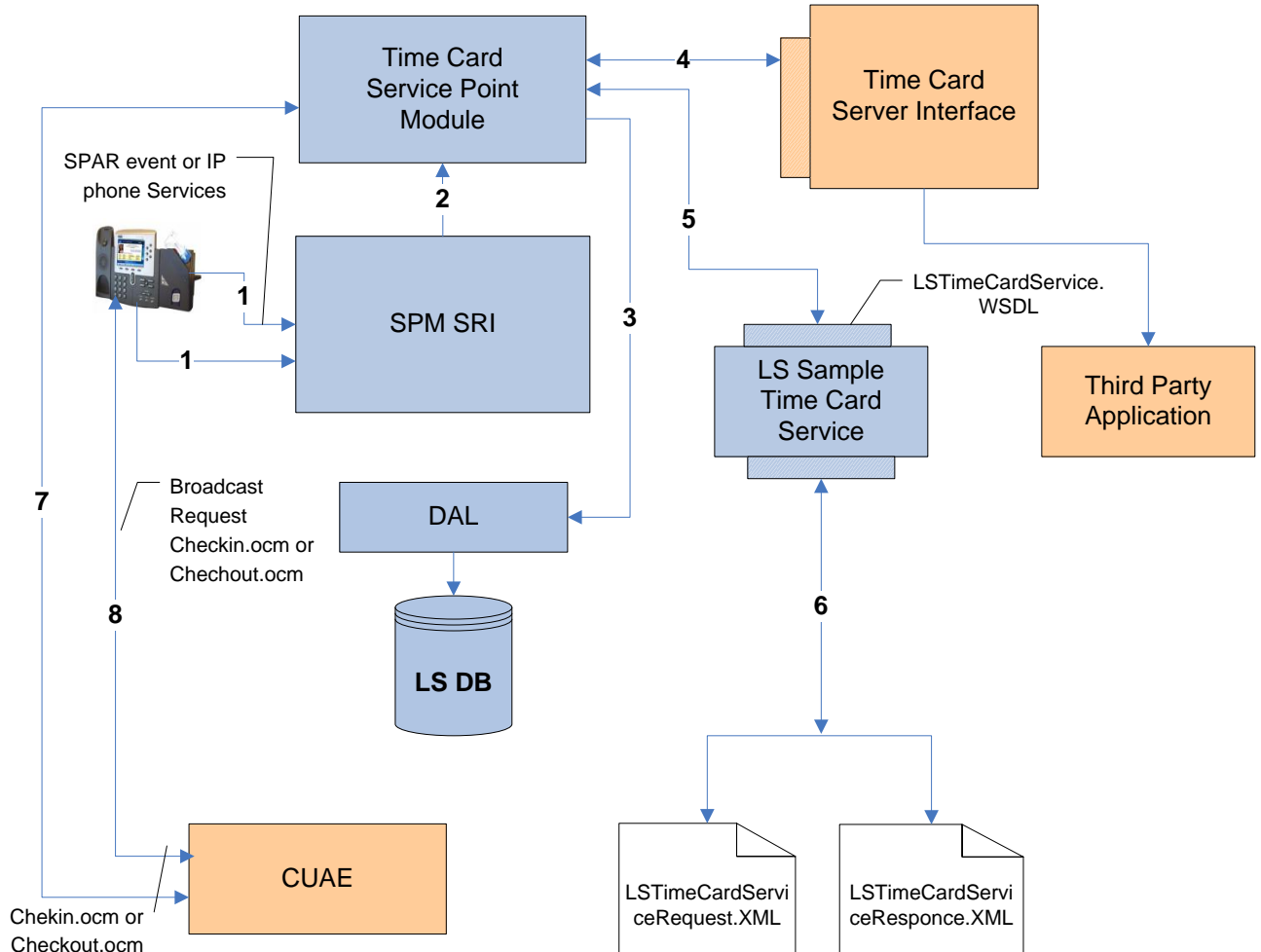


Figure 9 Time Card Application with CUAE integration

### 4.3.2 Retail Application

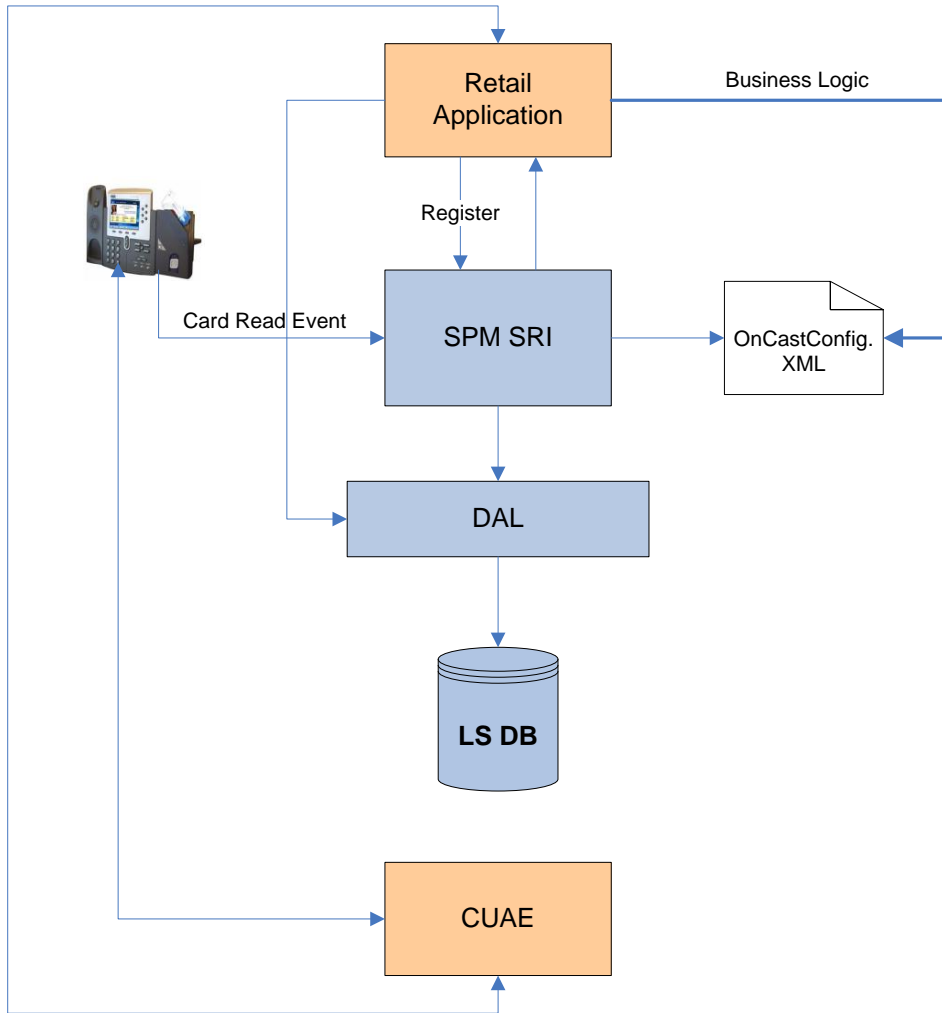


Figure 10 Retail Application with CUAE integration

## 5 APPENDIX A: Glossary

SPM	LiteScape™ proprietary Secure Profile Manager, which provides the authentication and other services for the IP Phones
SPAR	Secure Profile Authentication Reader is the reader located next to the telephone which can capture the input from the magnetic card, biometrics, smartcard or RFID and send the data in an encrypted format to the SPM. SPAR also provides Proxy Encryption between SPM and IP Phone.
SRI	SPAR Reader Interface
Proxy Encryption	The procedure where SPAR will perform encrypt/decrypt for the associated IP Phone
MAP	LiteScape™ Multimodal Application Platform
CCM	Cisco Call Manager
AES	The Advanced Encryption Standard which is replacing DES and 3DES as the standard for encryption as defined by the NIST. AES is an iterative symmetric –key block cipher that can use keys of 128, 192 and 256 bits in length
DES	The Data Encryption Standard has been the main encryption/decryption standard used since about 1977. DES used symmetric cipher with a 56 bit key. DES has been replaced by 3DES also known as “Triple DES” where the encryption cipher is run three times
IIS	Internet Information Services
UDP	User Datagram Protocol  Offers only a minimal transport service -- non-guaranteed datagram delivery -- and gives applications direct access to the datagram service of the IP layer. UDP is used by applications that do not require the level of service of TCP or that wish to use communications services (e.g., multicast or broadcast delivery) not available from TCP.  UDP is almost a null protocol; the only services it provides over IP are check summing of data and multiplexing by port number. Therefore, an application program running over UDP must deal directly with end-to-end communication problems that a connection-oriented protocol would have handled -- e.g., retransmission for reliable delivery, packetizing and reassembly, flow control, congestion avoidance, etc., when these are required. The fairly complex coupling between IP and TCP will be mirrored in the coupling between UDP and many applications using UDP
TCP	Transmission Control Protocol - Responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received
IP	Internet Protocol - Responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world
SSL	Secure Socket Layer
DAL	Database Access Layer (LiteScape)



## 6 APPENDIX B: LiteScope SPM Web Services

LiteScope SPM had defined the following Web Service Definition Language (WSDL) for the following functionality: Check in, Check out, Get Status, and Job Change.

### 6.1 LS TimeCardService.WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://tempuri.org/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://tempuri.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org/">
      <s:element name="TimeCardService">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="XmlDocumentInput"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="TimeCardServiceResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
name="TimeCardServiceResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="TimeCardServiceSoapIn">
    <wsdl:part name="parameters" element="tns:TimeCardService" />
  </wsdl:message>
  <wsdl:message name="TimeCardServiceSoapOut">
    <wsdl:part name="parameters" element="tns:TimeCardServiceResponse" />
  </wsdl:message>
  <wsdl:portType name="LSTimeCardServiceSoap">
    <wsdl:operation name="TimeCardService">
      <wsdl:input message="tns:TimeCardServiceSoapIn" />
      <wsdl:output message="tns:TimeCardServiceSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

```

    <wsdl:binding name="LSTimeCardServiceSoap"
type="tns:LSTimeCardServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
    <wsdl:operation name="TimeCardService">
    <soap:operation soapAction="http://tempuri.org/TimeCardService"
style="document" />
    <wsdl:input>

        <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="LSTimeCardService">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/" />

    <wsdl:port name="LSTimeCardServiceSoap"
binding="tns:LSTimeCardServiceSoap">
    <soap:address
location="http://localhost/LSSparSPMWeb/LSTimeCardService.asmx" />
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

## 6.1.1 LSTimeCardServiceRequest.XML

```
= <!--
```

```
    XmlDocumentInput : this is xml request to LSTimeCardService
```

```
-->
```

```
= <!--
```

```

LSTimeCardService requests can contain multiple child tags of type
LSWebServiceCall.
LSWebServiceCall: each tag will invoke one LSTimeCardService method.
The service currently supports the following methods:
checkin, checkout, getstatus, changejob and getjobCodes.
LSWebServiceCall:
Attribute MethodName: checkin, checkout, getstatus, changejob and
getjobCodes.
This MethodName will be returned in the response document.
Attribute MethodCallID: Optional attribute to specify unique request id for
the each method called.
This ID is returned in the response to help identify the calling method.
Params: Specify the corresponding method parameters.
Param: Multiple tags. Method invocation param.

```

```
-->
```

```
= <LSTimeCardService>
```

```
= <LSAuthenticate>
```

```
  = <!--
```

```
      Params:
      Param UserID: String, matching service account in Time-card
      application server or identity management system
      Param Password: String, encrypted. Encryption algorithm based
      on OnCast/SPM security settings
```

```
-->
```

```
<Param name="UserID" />
```

```
<Param name="Password" />
```

```
</LSAuthenticate>
```

```
= <LSWebServiceCall MethodName="checkin" MethodCallID="1">
```

```
  = <!--
```

```
      Params:
      Param UserID: String, matching registered/enrolled user info
      in SPM identity management system
      Param Optional JobCode = Encoded String, identifying Job code.
      If empty will use users default job code.
      Param Optional Time: String, time in UTC. Optional; If
      omitted, TimeCard server will apply current system time
```

```
-->
```

```
= <Params>
```

```
<Param name="UserID" />
```

```
<Param name="JobCode" />
```

```
<Param name="Time" />
```

```
</Params>
```

```
</LSWebServiceCall>
```

```
= <LSWebServiceCall MethodName="checkout" MethodCallID="2">
```

```
  = <!--
```

```
      Params:
      Param UserID: String, matching registered/enrolled user info
      in SPM identity management system
      Param Optional Time: String, time in UTC. Optional; If
      omitted, TimeCard server will apply current system time
```

```
-->
```

```
= <Params>
```

```
<Param name="UserID" />
```

```
<Param name="Time" />
```

```

</Params>
</LSWebServiceCall>
= <LSWebServiceCall MethodName="getstatus" MethodCallID="3">
  = <!--

          Params:
          UserID: String, matching registered/enrolled user info in SPM
identity management system

-->
= <Params>
<Param name="UserID" />
</Params>
</LSWebServiceCall>
= <LSWebServiceCall MethodName="changejob" MethodCallID="4">
  = <!--

          Params:
          Param UserID: String, matching registered/enrolled user info
in SPM identity management system
          Param JobCode: Encoded String, identifying Job code. String,
time in UTC. Optional; If omitted, TimeCard server will apply current
system time
          Param Optional Time: String, time in UTC. Optional; If
omitted, TimeCard server will apply current system time

-->
= <Params>
<Param name="UserID" />
<Param name="JobCode" />
<Param name="Time" />
</Params>
</LSWebServiceCall>
= <LSWebServiceCall MethodName="getjobCodes" MethodCallID="5">
  = <!--

          Params:
          Param UserID: Optional String, matching registered/enrolled
user info in SPM identity management system

-->
= <Params>

```

```
<Param name="UserID" />
</Params>
</LSWebServiceCall>
</LSTimeCardService>
```

## 6.1.2 LSTimeCardServiceResponse.XML

```
= <!--
```

LSTimeCardService response contains multiple child tags based on the original request received.

Params: Method response params.

Param: multiple tags. Method response param.

```
-->
```

```
= <LSTimeCardService>
```

```
= <LSAuthenticate>
```

```
= <!--
```

Param Result: Number or Strings indicating invocation result: 0 = OKAY, 1 = SESSIONTIMEOUT, 2 = CONNECTIONFAILURE, 3 = AUTHENTICATION FAILURE, 255 = APP\_ERROR

Param ErrorNumber/ErrorDescription: Only in case of Error. Number / String indicating error during invocation.

```
-->
```

```
<Param name="Result" />
```

```
<Param name="ErrorNumber" />
```

```
<Param name="ErrorDescription" />
```

```
</LSAuthenticate>
```

```
= <LSWebServiceCall MethodName="checkin" MethodCallID="1">
```

```
= <!--
```

Params:

Param Result: Number or Strings indicating invocation result: 0 = OKAY, 1 = SESSIONTIMEOUT, 2 = CONNECTIONFAILURE, 255 = APP\_ERROR

Param ErrorNumber/ErrorDescription: Only in case of Error. Number / String indicating error during invocation.

Param jobCode: Encoded String, identifying Job code.

```
-->
```

```
= <Params>
```

```
<Param name="Result" />
```

```
<Param name="ErrorNumber" />
```

```
<Param name="ErrorDescription" />
```

```
<Param name="jobCode" />
```

```
</Params>
```

</LSWebServiceCall>

= <LSWebServiceCall MethodName="checkout" MethodCallID="2">

= <!--

Params:

Param Result: Number or Strings indicating invocation result: 0 = OKAY, 1 = SESSIONTIMEOUT, 2 = CONNECTIONFAILURE, 255 = APP\_ERROR

Param ErrorNumber/ErrorMessage: Only in case of Error. Number / String indicating error during invocation.

Param jobCode: Encoded String, identifying Job code.

-->

= <Params>

<Param name="Result" />

<Param name="ErrorNumber" />

<Param name="ErrorDescription" />

</Params>

<Param name="jobCode" />

</LSWebServiceCall>

= <LSWebServiceCall MethodName="getstatus" MethodCallID="3">

= <!--

Param Result: Number or Strings indicating invocation result: 0 = OKAY, 1 = SESSIONTIMEOUT, 2 = CONNECTIONFAILURE, 255 = APP\_ERROR

Param ErrorNumber/ErrorMessage: Only in case of Error. Number / String indicating error during invocation.

Param Status: Integer indicating status: 0 = Checked-In, 1 = Checked-Out

Param Time: UTC time of check-in/check-out, empty if user has not checked in yet or does not exist

Param jobCode: Encoded String, identifying Job code, empty if user has not checked in yet or doesn't exist

-->

= <Params>

<Param name="Result" />

<Param name="ErrorNumber" />

<Param name="ErrorDescription" />

<Param name="status" />

<Param name="Time" />

<Param name="jobCode" />

</Params>

</LSWebServiceCall>

= <LSWebServiceCall MethodName="changejob" MethodCallID="4">

= <!--

Params:

Param Result: Number or Strings indicating invocation result: 0 = OKAY, 1 = SESSIONTIMEOUT, 2 = CONNECTIONFAILURE, 255 = APP\_ERROR

Param ErrorNumber/ErrorDescription: Only in case of Error. Number / String indicating error during invocation.

Param jobCode: Encoded String, identifying Job code.

-->

= <Params>

<Param name="Result" />

<Param name="ErrorNumber" />

<Param name="ErrorDescription" />

<Param name="jobCode" />

</Params>

</LSWebServiceCall>

= <LSWebServiceCall MethodName="getjobCodes" MethodCallID="8">

= <!--

Params:

Param Result: Number or Strings indicating invocation result: 0 = OKAY, 1 = SESSIONTIMEOUT, 2 = CONNECTIONFAILURE, 255 = APP\_ERROR

Param ErrorNumber/ErrorDescription: Only in case of Error. Number / String indicating error during invocation.

Param jobCodeData: String Xml listing multiple Jobs (Job ID, short description and long Job Description)

<jobs>

<job>

<Param name="JobCode"/>

<Param name="shortDescription" />

<Param name="JobDescription" />

</job>

<job>

<Param name="JobCode" />

<Param name="shortDescription" />

<Param name="JobDescription" />

</job>

</jobs>

-->

= <Params>

<Param name="Result" />

<Param name="ErrorNumber" />

<Param name="ErrorDescription" />

<Param name="jobCodeData" />

</Params>

</LSWebServiceCall>

</LSTimeCardService>

= <!--

```
ErrorNumber : ErrorDescription
10001 : User already checked-in.
10002 : User not checked-in.
10003 : Method not supported.
10004 : General Error.
10005 : Invalid Request Xml. Authenticate block missing.
10006 : Authenticate failed.
10007 : User does not exist.
```

-->

## 6.2 LSSparLoginWebServices.WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://tempuri.org/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://tempuri.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org/">
      <s:element name="SparLoginService">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="XmlDocumentInput"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="SparLoginServiceResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
name="SparLoginServiceResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="SparLoginServiceSoapIn">
    <wsdl:part name="parameters" element="tns:SparLoginService" />
  </wsdl:message>
  <wsdl:message name="SparLoginServiceSoapOut">
    <wsdl:part name="parameters" element="tns:SparLoginServiceResponse" />
  </wsdl:message>
  <wsdl:portType name="LSSparLoginWebServiceSoap">
    <wsdl:operation name="SparLoginService">
```

```

        <wsdl:input message="tns:SparLoginServiceSoapIn" />
        <wsdl:output message="tns:SparLoginServiceSoapOut" />
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="LSSparLoginWebServiceSoap"
type="tns:LSSparLoginWebServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
    <wsdl:operation name="SparLoginService">
        <soap:operation soapAction="http://tempuri.org/SparLoginService"
style="document" />
        <wsdl:input>

            <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="LSSparLoginWebService">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/" />

    <wsdl:port name="LSSparLoginWebServiceSoap"
binding="tns:LSSparLoginWebServiceSoap">
        <soap:address
location="http://localhost/LSTimeCardWeb/LSSparLoginWebService.asmx" />
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

## 6.2.1 LSSparLoginWebServiceRequest

```

= <!--
SparLoginWebService Request:
SparIP: (required) the SPAR IP to use.
PhoneIP: (required) the phone IP to use.
CallBackUrl: (required) Call Back URL is used to communicate final result
of validation.
SPARTypesSupported: (required) Supported SPAR types to be used for
validation/login.
e.g. BAC
SPARLoginSequence:(required) SPAR types sequence to be used for
validation/login.
UserID: (optional)
                UserID should be provided, if Magnetic or RFID
                are not part of SPARTypesSupported.
FirstName: (optional) First Name.
LastName: (optional) Last Name.
-->
= <!--

```

SPAR Types:

//follows are all SPAR Types defined. Note not all may be applicable/relevant here.

```
PASSWORD= 'A';
RFID_CARD= 'B';
BIOMETRIC = 'C';
MAGNETIC_CARD= 'D';
SMART_CARD= 'E';
BAR_CODE_READER= 'F';
RFID_AUTOID= 'G';
SPAR_REBOOT_EVENT= 'H';
//user Events
USER_BLOCKED='I';
USER_LOGGED_IN='J';
USER_LOGGED_OUT='K';
//exten
USER_LOG_IN_MULTIPLE_PROFILES_CHECK = 'L';
USER_LOG_IN_PROFILE_SELECTED = 'M';
//stages of login/enroll/error
NOT_STARTED= '0';
NONE= '0';
DONE= '1';
USER_EXTENTION_MOBILITY = '2';
USER_INFO='3';
QUIT='4';
BIO_STATUS='5';
UNKNOWN='6';
FARWORD_APPS='7';
BIO_FAILED='8';
INVALID_PASS='9';
```

-->

= <SparLoginWebService>

<UserID />

<SparIP>10.13.0.187</SparIP>

<PhoneIP>10.13.0.79</PhoneIP>

<FirstName />

<LastName />

<CallBackUrl>http://10.13.0.81:80/LSTimeCardWeb/LSTimeCardWeb.aspx</CallBackUrl>

<SPARTypesSupported>BAC</SPARTypesSupported>

<SPARLoginSequence>BAC</SPARLoginSequence>

</SparLoginWebService>

## 6.2.2 LSSparLoginWebServiceResponse

= <!--

SparLoginWebService Response:

SessionID: (Returned from web service) SparLoginWebService creates a unique sessionID.

-->

```
= <LSSparLoginWebService>  
<SessionID>3347421471</SessionID>  
</LSSparLoginWebService>
```

## 7 *Support Inquiry*

For support questions about products from LiteScape Technologies, please contact us using the following information:

LiteScape Technologies, Inc.

[info@litescape.com](mailto:info@litescape.com)

[www.litescape.com](http://www.litescape.com)