ISAI

Intelligent Security APPs & Information Cloud Platform

System Integration Manual

iMobile Mind

1 System Integration in a nutshell

The ISAI (short for Intelligent Security APPs & Information) platform, has the capabilities and administrative mechanisms for both mobile device security and push notification services. It allows external Apps or systems to send and receive messages from exclusive channels. This document serves to present in detail the technical information required for the development and integration with the ISAI system.

1.1 System Architecture

ISAI is designed to be a cloud platform. System administrators can, through the main control panel interface of ISAI, setup and administer unique and independent device management platforms and databases for different corporate entities. Each device administration platform has its own exclusive service components and modules to provide services for the Apps on the end-users' mobile devices. The system architecture is designed as illustrated below:

- **ISAI Console**

  System administrators can establish/publish an exclusive device security administration platform for any given corporate entity. Access privileges, functional modules, authorization of user base and adjustments of system options, etc., can all be administered through the main control panel.

- **ISAI for Organization – Device Security Administration Platform**

  It is a device security administration platform for specific corporate entities. Components and modules include:

  - It provides the system administrators a web interface with the capabilities to issue public announcements, manage devices, user accounts, access privileges, track messages and publish software.
  - PNS- Push Notification Service：Responsible for the delivery of push notifications to the corresponding mobile devices with the assistance of APNS/GCM servers.
  - PSS- Push Server Service：Responsible for the delivery and receipt of messages between device administration platform and external devices or systems.
  - ORG DB : Database platform exclusive for this specific corporate entity.

- **ISAI Mobile**

  ISAI constitutes of a PMA and a Push Mobile Agent, for every supported mobile platform, namely Android and iOS. It is responsible for the delivery and receipt of messages and also the execution of the administrative commands issued by the MDM.

  Messages contain channel codes. However, PMA is only responsible for the confirmation of the completion of message delivery and receipt. It will log the receipt of messages in the message database (MSGDB) where PMA resides. A corresponding App will then process the request for

the specified channel. If other Apps needs to send messages, all they need to do is saving the messages in the PMA message database. PMA will automatically detect and forward them to the servers.

- **ISAI Client**

  ISAI supports the Windows operating system. It also constitutes of a surrogate, namely, the MPG (short for Mobile Push Gateway), responsible for the delivery and receipt of messages. However, it does not serve the average end-users, but rather serves the heterogeneous application systems that need to send messages to mobile devices such as BPM, EIP, ERP, etc. MPG provides a communication interface based on database and XML files that significantly reduces the difficulty of integration.

- **Client System**
  Heterogeneous application systems that need to deliver messages to mobile devices, (ie. BPM, EIP, ERP, etc)

- **Mobile App**
  Mobile devices are required to have Apps capable of handling ISAI channel information installed.

1.2 Software development environment

ISAI platform uses industry standard services and interfaces as I/O portals between internal and external networks. Integration of heterogeneous systems at the server end can be accomplished

through accessing system information through Web Services or databases. The integration of end-user and mobile devices is accomplished homogeneously through the communication interface of databases.

With industry standard service interfaces, developers follow the environmental constraints of different platforms. However, they can also pick and choose their own preferred tools/software/programming languages for software development. The following is the recommended software development environment for the ISAI :

1.2.1 Server/Client

Operating System: Windows Server® 2008 R2 and above
Development Tools: Visual Studio .NET®/ Visual C#®/ ASP .NET 4.0
Database: Microsoft© SQL Server® 2008 and above

1.2.2 Android Mobile Devices

Operating System: Android 4.0 and above
Development Tool: Eclipse™ / Java® / Android SDK
Database: SQLite

1.2.3 iPhone Mobile Devices

Operating System: iOS
Development Tool: Xcode® / Interface Builder / Objective-C®
Database: SQLite

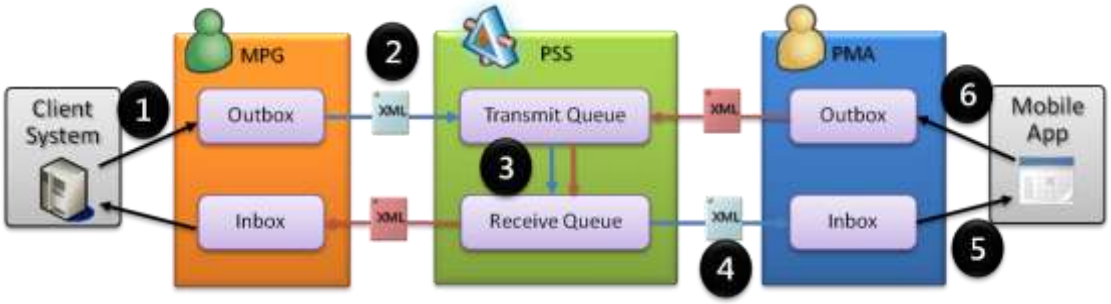
1.3 Network Environment

The administration mechanism needs access to various networks.

2 Message Delivery and Receipt

Push Notification is an important module for the ISAI administration platform and expansion of channel application. This chapter explains in detail the related technical specifications for integration by developers.

2.1 The operating procedure of push notification



The procedure in detail as follows:

1. The client' s system writes the messages to be delivered into the Outbox of MPG. The MPG will then forward the messages to the server automatically. It is important to note that if the messages contain attachments, MPG must be given read/write access privileges of the physical path where the files reside to operate properly.
2. MPG proactively serialize unsent messages in the Outbox in the XML format, encrypt them and forward them to the server. After decryption and restoration, the messages will be entered into the TransmitQueue and ISAI will take over and process them accordingly.
3. The PSS automatically reads messages in the TransmitQueue and distribute the messages into the exclusive ReceiveQueue of each recipient according to the recipient list in the messages.
4. The messages in the ReceiveQueue awaits processing by PMA. They go through the same encryption and decryption procedures and get recorded on the Inbox of the mobile devices

for the Mobile App to process.

5. Mobile Apps will decipher messages and display them according to their own procedures.
6. The procedure for Mobile Apps to send messages follows the same steps as described above. The difference being the recipients are specific MPG accounts. For example, if MPG sends a message to 10 different recipients. The replies from these 10 recipients have the same recipient, namely, the MPG account, thus allowing the MPG to obtain the reply information of these 10 users.

2.2 Message Datasheet Communication Interface

To reduce the complexity of system integration among heterogeneous systems in the aforementioned procedure, the database is used as the main communication interface among them. The corresponding datasheet is as follows:

- Outbox (OutboxAttachment) : responsible for recording the datasheet of the messages delivered.

  When the backend system needs to send message to specific user accounts, all is required is adding a new record to the Outbox datasheet according to the datasheet specifications. MPG will then read all messages yet to be delivered and deliver them accordingly. For the convenience of integration at the client' s end, the information of the attachment recorded on the OutboxAttachment datasheet does not need to be written by the client. All is required is to store the attachment files in the directory accessible by the MPG server. MPG will automatically take appropriate corresponding actions according to the datasheet.

- Inbox (InboxAttachment): responsible for the datasheet that records the received messages and attachments. MPG proactively communicates with the PSS. When there are messages for the MPG account, it automatically download the messages and attachments to the Inbox and InboxAttachment datasheets. Corporate backend system needs to read the Inbox periodically for new messages and change the status to "read" for these messages.

- The two datasheets for TransmitQueue & ReceiveQueue on the server end are not available for heterogeneous systems to access directly.

The definitions of the fields on the aforementioned datasheet used for system integration are listed in detail in the appendix.

2.3 Message Channel

To distinguish between different application scenarios of the messages, for example, processing of document approvals and public announcements use the same message delivery mechanism, but the content and process procedures are different. The system uses the distribution mechanism of the message channel. Different ChannelCodes are defined for different purposes, for example, BPM for document approval messages and BULLTIN for system announcements.

2.4 Message content (Content) and content type (ContentType)

After messages are distributed according to the channel codes. The corresponding Apps can decide how to decipher message

contents according to content type. For example, if the ContentType is MSG then the message content is regular message. If the ContentType is REPLY, then the message content is the reply to certain messages.

For BPM datasheets, the message content can be defined in the following XML file. It can be used for the inspection of the approval process of certain application forms, from the contents of the application, processes and steps taken to the end results.

```
<?xml version="1.0" encoding="utf-8" ?>
<content contentType="MSG">
    <!-- <params> Used to identify data on this datasheet. Variables in it can be used by Apps for this channel to determine how to process-->
  <params>
    <param name="sheetNo" value="AF-20101010-01"/>
    <param name="sheetType" value="AbsentForm"/>
  </params>
   <!-- <view> Uses to show the definition of the datasheet, marked up as page to paginate-->
  <view>
    <page id="page1" name="ITEM">
      Free Text (HTML or XML)
      Free Text ( could be HTML or XML)
    </page>
    <page id="page2" name="Form">
      Free Text (HTML or XML)
      Free Text ( could be HTML or XML)
    </page>
    <page id="page3" name="Status">
      Free Text (HTML or XML)
      Free Text ( could be HTML or Xml)
    </page>
  </view>
    <!-- <replyInfo> Used to define how user and channel Apps should display and process the replies-->
  <replyInfo>
       <!--isNullable=true means this fields allows the content of reply to be empty-->
    <input id="comment" name="Opinion" isNullable="true" type="TextBox" defaultValue="No opinion"/>
    <!—Decipher and display the types of submission. When user select this option, the user confirms the reply data and sends the reply back -->
    <input id="result" type="Submit" defaultValue=" Approve">
      <option id="btn1" name="approve" value="Approve"/>
      <option id="btn2" name="disapprove" value="Disapprove"/>
      <option id="btn3" name="reject" value="Reject"/>
    </input>
  </replyInfo>
</content>
```

After the XML message content is delivered to the Apps on the mobile devices, the corresponding channel App deciphers the content and display it on the user interface of the mobile devices. After the user signs off on it, the result will then be sent back to MPG through PSE.

```
<?xml version="1.0" encoding="utf-8" ?>
<content contentType="REPLY">
  <!-- <params> Used to identify data on this datasheet. Variables in it can be used by Apps for this channel to determine how to process -->

  <params>
    <param name="sheetNo" value="AF-20101010-01"/>
    <param name="sheetType" value="AbsentForm"/>
  </params>
  <!—The result of the application. It will get recorded the “result” field in the corresponding submitInfo according to <replyInfo> -->

  <submitInfo>
    <result id="result" name="Result" value="Approve"/>
    <result id="comment" name="Opinion" value="No problem"/>
  </submitInfo>
</content>
```

Message Recall mechanism. When the client system wishes to recall previously delivered messages, it needs to obtain the MessageId from the Outbox datasheet. This field is filled by the server after the message was correctly delivered by the Outbox. The client needs to produce another new message and write it to the content field according to this MessageId.

```
<?xml version="1.0" encoding="utf-8" ?>
<content contentType="RECALL">
  <params>
    <param name="RecallMessageId" value=" 02b2ae73-5b88-4cfe-81a8-34f514471355"/>
  </params>
</content>
```

Example of Outbox fields of a recalled message:

Field Name	Data	Notes
Creator	Alvin	
CreateDate	2011/05/10 15:07:00	
TxMessageId	c7cd8812-8c0f-4c5e-a8e6-402aa6f44534	Automatically Generated
AppCode	**SYS**	
SourceDomain	ase	
Sender	pcagent	
ReceiverType	U	
ReceiverList	**pseReceiver**	**Set as the surrogate of recipients on the server end**
BccList		
Subject	[Recall]: 02b2ae73-5b88-4cfe-81a8-34f514471355	Not necessary. For reference only
Content	.....	Fill with aforementioned XML
ContentType	RECALL	
AttachmentList		
AlertFlag	0	
MessageId		
State	10	
TransmitTime	2011/05/10 15:07:00	
StatusTime	2011/05/10 15:07:00	
StatusLog	Status	

The message format above refers to the BPM integration format. In actual practice, adjustments can be made according to needs.

3 Appendix

3.1 Table Schema

Table Name: **Outbox**				
English ID	Data Type	Length	Allow NULL	Details
TxMessageId	nvarchar	50	PK	Defined by user, GUID recommended
AppCode	nvarchar	20		Server defined AppCode, mainly in English
SourceDomain	nvarchar	50		This required field is used to determine the source of the message, so that account information without domainName can be processed.
AgentSender	nvarchar	100		End-User’ s MPG account
Sender	nvarchar	100		Please refer to the UserAccount.UA002 user account field
ReceiverType	char	1		Specifies the recipient type If U = User then the ReceiverList contains a user list If D= Device then the ReceiverList contains a list of device serial numbers.
ReceiverList	nvarchar	max		The recipient list is delimited by semicolon (;). Please refer to the UserAccount.UA002 user account field
BccList	nvarchar	max	✓	BCC List, processed the same way as above
Subject	nvarchar	200		The subject line of the message
Content	nvarchar	max		The content of the message
ContentType	nvarchar	20		Used to distinguish the type of contents Command Message Reply Update Delete Status Recall
AttachmentList	nvarchar	max	✓	The list of attachments to send. Each attachment file should be given a complete name and path. Delimited by semicolon (;)
AlertFlag	bit	1		Should user be notified if new messages arrive (Y: Yes, N: No)
MessageId	nvarchar	50	✓	The serial number of the message generated by the server
State	int			0 = initialize 10 = yet to be uploaded, 11 = uploading attachment, 12 = uploading of attachment is completed, 19 = upload complete, 20 = waiting for server to deliver, 29 = server finished delivery, 97 = timed out, 98 = forfeit, 99 = process failed
TransmitTime	nchar	20	✓	The time of transmission recorded as string in the yyyy-MM-dd HH:mm:ss format
StatusTime	nchar	20	✓	The update time recorded as string in the yyyy-MM-dd HH:mm:ss format
StatusLog	nvarchar	200	✓	The status log of the message processing procedure
Priority	nvarchar	1		E: Emergency, H: High Priority, N: Normal (Default), L: Low Priority

Keywords	nvarchar	200	✓	Keywords for the message can be defined in this field delimited by semicolon (;). The system can use this field to query the message at time of display or search

Table Name: **Inbox**				
English ID	Data type	Length	Allows NULL	Details
RxMessageId	nvarchar	50	PK	Serial number replied by server, usually the GUID
AppCode	nvarchar	20		Server defined AppCode, mainly in English
SourceDomain	nvarchar	50		This required field is used to determine the source of the message. So that account information without domainName can be processed.
AgentSender	nvarchar	100		End-User' s MPG account
Sender	nvarchar	100		The sender' s account, please refer to the UserAccount.UA002 user account field
Owner	nvarchar	max		Message owner can be average user or the account of ChannelAgent
ReceiverType	char	1		Specifies the recipient type If U = User then the ReceiverList contains a user list If D= Device then the ReceiverList contains a list of device serial numbers.
ReceiverList	nvarchar	max		The recipient list is delimited by semicolon (;). Please refer to the UserAccount.UA002 user account field
BccList	nvarchar	max	✓	BCC List, processed the same way as above
Subject	nvarchar	200		The subject line of the message
Content	nvarchar	max		The content of the message
ContentType	nvarchar	20		Used to distinguish the type of contents Command Message Reply Update Delete Status Recall
AttachmentFlag	bit	1		Boolean flag to indicate if the message contains attachment
AlertFlag	bit	1		Should user be notified if new messages arrive (Y: Yes, N: No)
MessageId	nvarchar	50		Serial number replied by server, usually the GUID
State	int			0 = initialize 10 = yet to be uploaded, 11 = uploading attachment, 12 = uploading of attachment is completed, 19 = upload complete, 20 = waiting for server to deliver, 29 = server finished delivery, 97 = timed out, 98 = forfeit, 99 = process failed
TransmitTime	nchar	20	✓	The time of transmission recorded as string in the yyyy-MM-dd

				HH:mm:ss format
ReceiveTime	nchar	20	✓	The time of receipt recorded as string in the yyyy-MM-dd HH:mm:ss format
StatusTime	nchar	20	✓	The update time recorded as string in the yyyy-MM-dd HH:mm:ss format
StatusLog	nvarchar	200	✓	The status log of the message processing procedure
Priority	nvarchar	1		E: Emergency, H: High Priority, N: Normal (Default), L: Low Priority
Keywords	nvarchar	200	✓	Keywords for the message can be defined in this field delimited by semicolon (;). The system can use this field to query the message at time of display or search

Table Name: **InboxAttachment 、 OutboxAttachment**

English ID	Data Type	Length	Allows NULL	Details
AttachmentId	uniqueidentify		PK	Automatically generated serial number
MessageId	nvarchar	50		Please refer to the .PK of Outbox/Inbox
SeqNo	int			To sequence the files
FileName	nvarchar	250		
FileType	nvarchar	10		File Type
FileSize	int			
FileData	varbinary	max	✓	Store files in this field in the database
SourceUri	nvarchar	10	✓	To indicate the location of the file (Inbox=>remote, Outbox=>local)
DownloadFlag	int			0 : Default, indicates the attachment will not be pre-downloaded to the mobile device, but only downloaded when user lick on the file link. 1 : The mobile device will download the attachment together with the message. It works with webpages with graphic links that need to be loaded offline.