Albert

POS Integration Guide.

Version 6.0
September 2016

CAN

Contents

1. Document Purpose

This document outlines the main considerations and methods used for Albert POS integration. Key integration methods, considerations and concerns are reviewed, along with a code example and how to handle intent results. For further support, refer to documentation and Forums on **piappbank.com.au** or email **SmartTerminals&Apps@cba.com.au**.

2. Introduction to Albert

Albert is an Android based EFTPOS tablet that allows merchants to securely take a payment, print a receipt, and offer their customers an interactive touch screen experience, on a single device. Albert has the capability to run Android based apps, managed via a secure portal – App bank.

Key device specifications:

- 7” touchscreen
- Integrated thermal printer
- Camera
- Magnetic Swipe, Chip EMV, and NFC reader for card payment

Albert has been developed in conjunction with Aevi.

3. Outline of an Integration

Building your own integration allows you to use your own POS system to control the merchant interaction. The Albert can then act as a customer interaction tool as well as a payment device.

Albert POS Integration
The Albert POS application (POS Client), residing on Albert, must establish a connection to the relevant POS register (POS Terminal).

The POS register needs to be able to send and receive variables from the POS Client. The POS Client receives the variables, and creates an intent to the CommBank Standard Payment app on Albert.

The CommBank Standard Payment app completes the entire transaction where the app takes the customer's card details to process the transaction, prints the merchant and customer receipts and sends the transaction results back to the POS Client.

The POS Client then transmits the details back to the POS.

4. Key Considerations for Albert

- Albert is SEAndroid Enforcing.
- Albert is currently running Android 4.0.4.
- The Merchant Menu Library is required for apps to run on Albert.
- Firewall restrictions are in place that limit inbound network connectivity.
- There are no back, home, and recents buttons seen by the end user.
- The NDK is not supported. C and C++ is not allowed.
- For hybrid apps, contact us at **smartterminals&apps@cba.com.au**

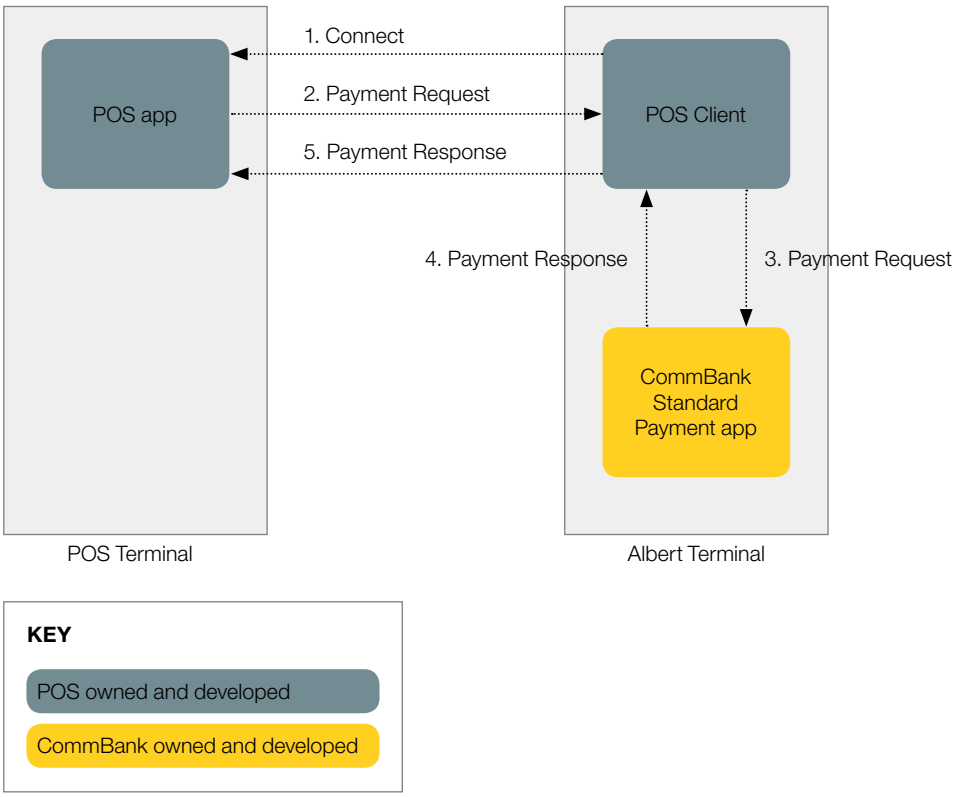
5. Common Pairing Scenarios

5.1 Direct Connection to POS

There are four possible pairing scenarios between a POS and Albert.

Direct Connection to POS is the recommended approach. The other three options are technically feasible but carry dependencies on variable factors that may be unique to each case and environment.

The easiest way to integrate a POS with Standard Albert running the Standard Payment app.



Once the connection between the POS app and POS Client is open, it can serve multiple requests, in series, until closed.

Components

Component	Description
POS app	The POS app resides on a separate machine and can run any technology stack such as Windows or Linux. Connectivity is assumed to be via WiFi. Connecting via 3G would be harder as the POS would have to allow connections from the internet.
POS Client	The POS Client is an app or service that runs on Albert that may or may not have an interface. This is responsible for sending and receiving requests to and from the POS. It uses the Albert SDK by Wincor to launch the CommBank Standard Payment app. This client would also have to maintain a long running connection/socket/web socket with the POS to allow two way communication (e.g. socket.io). As Albert doesn't allow incoming connections, this connection needs to remain in place and needs to be initiated by Albert.
CommBank Standard Payment app	A black box that handles all the payment processing. It is launched using the SDK and returns a response to the calling app.

Sequences

ID	Sequence	Description
1	Connect	Establishes a connection with the POS (it is assumed this is over manually defined IP). Albert establishes an outgoing long lived connection such as a socket. This is due to Albert being unable to accept incoming connection requests because of firewall restrictions.
2	Payment Request	The POS sends a message to the POS client over the established connection. The format of this message is left to the POS vendor to decide. It can be extended to support any command recognised by the POS Client such as "display advert", "display menu" etc.
3	Payment Request	The POS Client uses the SDK to make a payment request to the CommBank Payment app.
4	Payment Response	Once the payment finishes, a response is sent back to the POS Client.
5	Payment Response	The POS Client sends a message back to the POS over the established connection. The format of this message is left to the POS vendor's discretion.

Pros

- Ease and simplicity of implementation

Cons

- Connection to POS app defined via static IP by default – if a change is required, IP is manually set on Albert POS Client.
  - Addressed in Registry Service example below.

Implementation Considerations

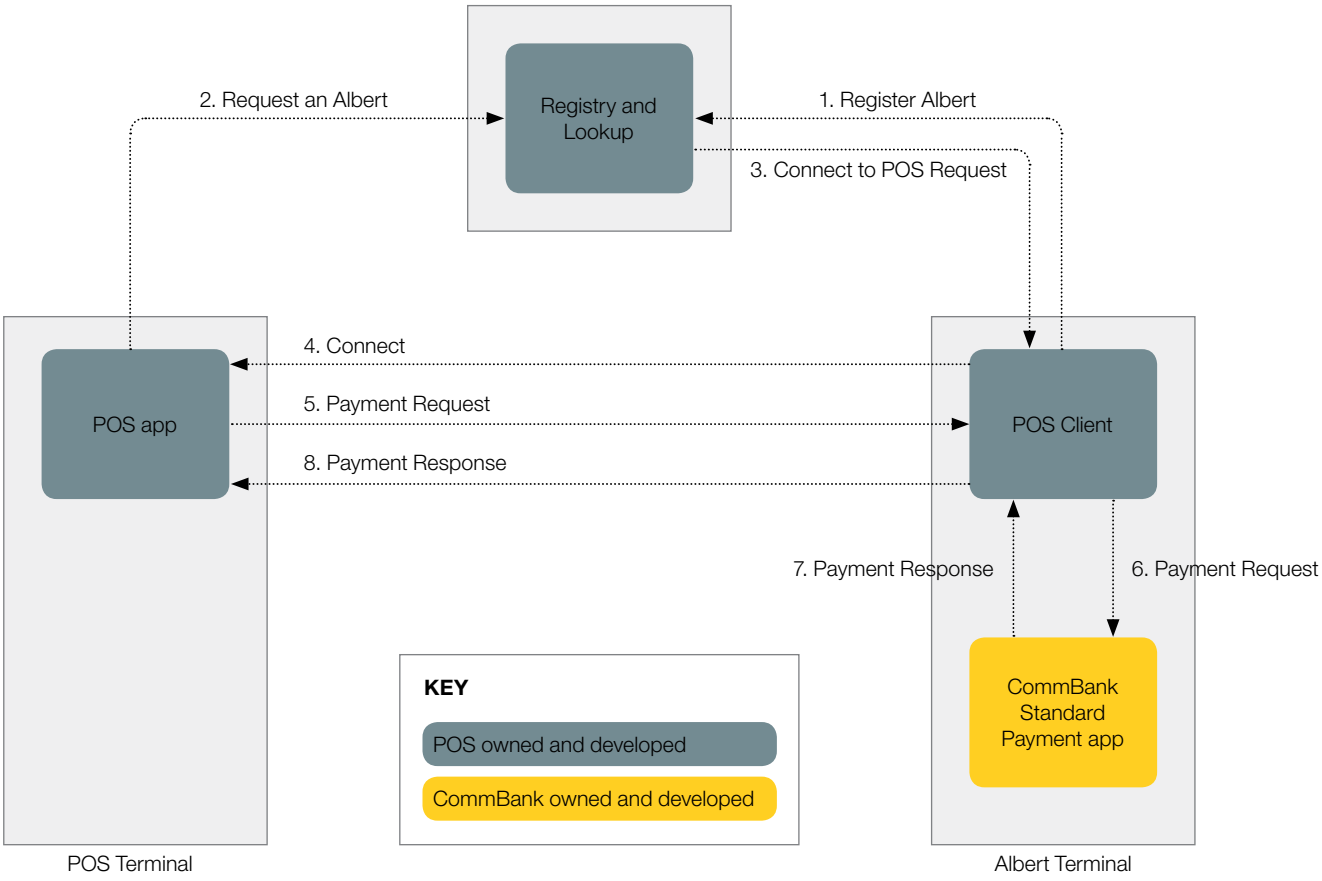
- Securing the connection to the POS.
- Securing traffic between the POS and POS Client.
- Network outages and connection retries.
- Payment app failures e.g. DECLINE returned.
- Timeouts from a payment taking too long.
- A failure causing the POS to never receive a Payment Response.

5.2 Direct Connection with Registry Service

This gives the connection establishment responsibility to a service, allowing for external management of terminal and POS assignment. A web interface could be developed to manage the "Connection Service" to monitor, audit or control the allocation of terminals to POS systems.

Direct connection using a registry service is more suitable to scenarios where there are multiple locations and multiple terminals that need to be managed off-site at a central location/head office.

- POS Client queries SDK for Merchant and Terminal ID.
- POS Client connects to a registry service, identifies itself, and receives IP address of the POS it should pair with.
- POS Client establishes long lived connection with POS, and proceeds as above.



Components

Component	Description
POS app	The POS app resides on a separate machine and can run any technology stack such as Windows or Linux. Connectivity is assumed to be via WiFi. Connecting via 3G would be harder as the POS would have to allow connections from the internet.
POS Client	The POS Client is an app or service that runs on Albert that may or may not have an interface. This is responsible for sending and receiving requests to and from the POS. It's also responsible for registering itself as an available Payment Terminal with the "Registry and Lookup Service". It uses the Albert SDK by Wincor to launch the CommBank Standard Payment app. This client would also have to maintain a long running connection/socket/web socket with the POS to allow two-way communication (e.g. socket.io). As Albert doesn't allow incoming connections this connection needs to remain in place and needs to be initiated by Albert.
CommBank Standard Payment app	A black box that handles all the payment processing. It is launched using the SDK and returns a response to the calling app.
Registry and Lookup	Provides a service for Alberts to register themselves as an available Payment Terminal as well as a way for a POS to request an Albert. This would allow management of multiple terminals and POS systems via a central point. This would lend itself to a simple web service.

Sequences

ID	Sequence	Description
1	Register Albert	Registers itself as available with (for example): IP address, location, terminal ID, features available.
2	Request an Albert	The POS requests an Albert that may be based on: Location, terminal ID, store, features etc.
3	Connect to POS Request	The service tells Albert it’s been selected to be a Payment Terminal for a POS. It's given the POS's IP address and is told to connect to it.
4	Connect	POS Client opens up a long running connection with the POS to allow two-way communication.
5	Payment Request	The POS sends a message to the POS Client over the established connection. The format of this message is left to the POS vendor to decide. It can be extended to support any command recognised by the POS client such as "display advert", "display menu" etc.
6	Payment Request	The POS Client uses the SDK to make a payment request to the CommBank Payment app.
7	Payment Response	Once the payment finishes, a response is sent back to the POS Client.
8	Payment Response	The POS Client sends a message back to the POS over the established connection. The format of this message is left to the POS vendor’s discretion.

Pros

- Allocation of terminal to POS is centrally controlled.

Cons

- Requires a web based Connection Service which is more development and support effort.
- The communication between POS and Albert is direct 1-to-1 and doesn't go via a message broker. This is fine but offers less monitoring or control.

Implementation Considerations

- The Connection Service becoming a single point of failure.
- Securing the connection to the POS.
- Securing traffic between the POS and POS Client.
- Network outages and connection retries.
- Payment app failures e.g. DECLINE returned.
- Timeouts from a payment taking too long.
- A failure causing the POS to never receive a Payment Response.

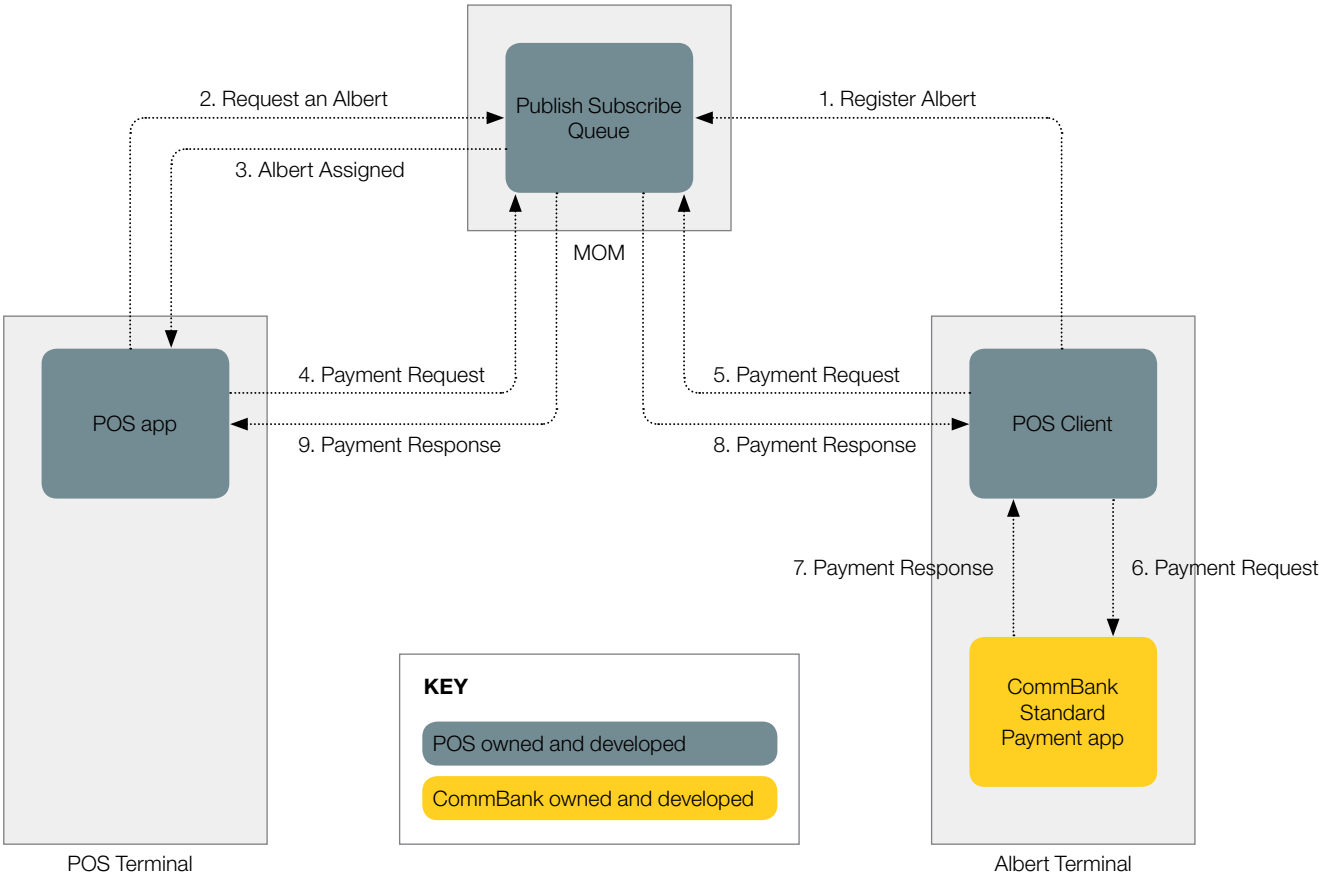
5.3 Web Based Middleware

Similar to the direct connection with registry service approach but taken a step further by completely decoupling Albert and the POS. Albert and the POS are not dependant on each other, increasing solution robustness as they don't have to know of the other entities' existence.

Beneficial if there are multiple subscribers that can use event information to trigger other actions, log data, provide analytics etc.

It assumes a piece of Message Oriented Middleware (MOM) that provides the message orchestration via queues, publish/subscribe, topics etc.

- POS connects to a middleware service that will handle all transaction communications.
- Albert app queries SDK for Merchant ID and Terminal ID, and connects to middleware service.
- Middleware service passes all communications between POS and Albert.
- Middleware could be set up as a publish/subscribe queue, or a direct message transfer.



Components

Component	Description
POS app	The POS app resides on a separate machine and can run any technology stack such as Windows or Linux. Connectivity is assumed to be via WiFi. Connecting via 3G would be harder as the POS would have to allow connections from the internet.
POS Client	The POS Client is an app or service that runs on Albert that may or may not have an interface. This is responsible for sending and receiving requests to and from the Publish/Subscribe Queue. It's also responsible for registering itself as an available Payment Terminal with the Publish/Subscribe Queue as well as two-way messaging. It uses the Albert SDK by Wincor to launch the CommBank Standard Payment app. This client would also have to maintain a long running connection/socket/web socket with the Publish/Subscribe Queue to allow two-way communication (e.g. socket.io). As Albert doesn't allow incoming connections this connection needs to remain in place and needs to be initiated by Albert.
CommBank Standard Payment app	A black box that handles all the payment processing. It is launched using the SDK and returns a response to the calling app.
Publish/Subscribe Queue	Provides a way for an Albert to register itself as a payment terminal and a POS to request a payment terminal. It also acts as a message broker between the POS and Albert. It's ideally suited to handling multiple POS and multiple Alberts and possibly across multiple merchants. It will also need to provide management of open sockets to Albert and to the POS as well as web services to register the devices. This box could well be broken down further into sub components such as socket server and web service.

Sequences

ID	Sequence	Description
1	Register Albert	Registers itself as available with (for example): IP address, location, terminal ID, features available. It would also need to establish a long running connection with the terminal for two-way messaging.
2	Request an Albert	The POS requests an Albert that may be based on: Location, terminal ID, store, features etc. The POS will establish a long running connection with the service.
3	Albert Assigned	A POS and terminal are assigned that means messages can pass between the two connections.
4	Payment Request	The POS sends a message to the MOM over the established connection. The format of this message is left to the POS vendor to decide. It can be extended to support any command recognised by the POS client such as "display advert", "display menu" etc.
5	Payment Request	The payment request is routed to the correct terminal. At this point the message can be verified, audit logged, augmented and finally routed.
6	Payment Request	The POS Client uses the SDK to make a payment request to the CommBank Payment app.
7	Payment Response	Once the payment finishes, a response is sent back to the POS Client.
8	Payment Response	The POS Client sends a message back to the POS over the established connection to the MOM. The format of this message is left to the POS vendor's discretion.
9	Payment Response	The payment request is routed to the correct POS. At this point the message can be verified, audit logged, augmented and finally routed.

Pros

- Very well suited to a multi-terminal setup and ideal for multi-merchant.
- Gives the ability to monitor and control flow of messages between terminals.

Cons

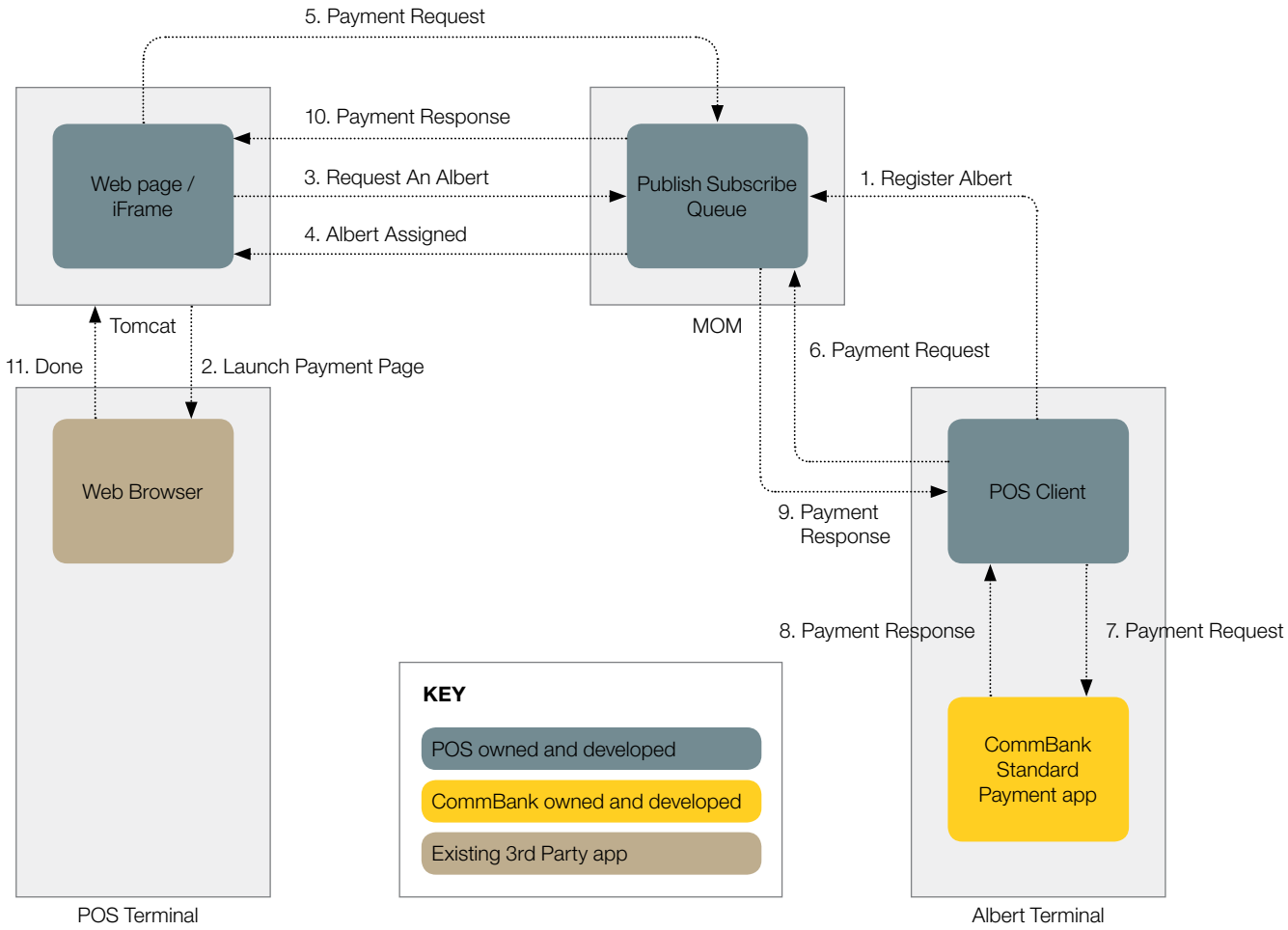
- Complex setup and support.
- Single point of failure.

Implementation Considerations

- The MOM becomes a single point of failure.
- Securing the connections and traffic.
- Network outages and connection retries.
- Payment app failures e.g. DECLINE returned.
- Timeouts from a payment taking too long.
- A failure causing the POS to never receive a Payment Response.

5.4 Web Based POS

An example of a web based POS not running on Albert. The Publish/Subscribe service could be replaced with something simpler such as a simple pass through of messages between two open connections - the web page connection and the Albert connection. The web page connection can be achieved using something like WebSockets, Long Polling etc. The Albert connection could be a simple TCP/IP socket. The MOM box then becomes a facilitator or broker rather than the POS and the Albert Terminal connecting directly. If there are multiple POS connections and multiple Alberts it makes sense to be a complete MOM solution. POS connects to a middleware service that will handle all transaction communications.



Components

Component	Description
Web Browser	Viewing window for web based POS.
Web page/iFrame	Remote web page that controls the Albert Terminal.
CommBank Standard Payment app	A black box that handles all the payment processing. It is launched using the SDK and returns a response to the calling app.
Publish/Subscribe Queue	Provides a way for an Albert to register itself as a payment terminal and web page to request a payment terminal. It also acts as a message broker between the web POS and Albert. It's ideally suited to handling multiple POS and multiple Alberts and possibly across multiple merchants. It will also need to provide management of open sockets to Albert and to the POS as well as web services to register the devices. This box could well be broken down further into sub components such as socket server and web service.
POS Client	The POS Client is an app or service that runs on Albert that may or may not have an interface. This is responsible for sending and receiving requests to and from the Publish/Subscribe Queue. It's also responsible for registering itself as an available Payment Terminal with the Publish/Subscribe Queue as well as two-way messaging. It uses the Albert SDK by Wincor to launch the CommBank Standard Payment app. This client would also have to maintain a long running connection/socket/web socket with the Publish/Subscribe Queue to allow two way communication (e.g. socket.io). As Albert doesn't allow incoming connections, this connection needs to remain in place and needs to be initiated by Albert.

Sequences

ID	Sequence	Description
1	Register Albert	Registers itself as available with (for example): IP address, location, terminal ID, features available. It would also need to establish a long running connection with the terminal for two-way messaging.
2	Launch Payment Page	Request a payment by launching a web page or iFrame.
3	Request an Albert	The POS requests an Albert that may be based on: Location, terminal ID, store, features etc. The web page will establish a long running connection with the service using technology such as Long Polling, WebSockets and/or a combination of webhooks.
4	Albert Assigned	Pairing complete.
5	Payment Request	The web page sends a message to the MOM over the established connection. The format of this message is left to the POS vendor to decide. It can be extended to support any command recognised by the POS client such as "display advert", "display menu" etc.
6	Payment Request	The payment request is routed to the correct terminal. At this point the message can be verified, audit logged, augmented and finally routed.
7	Payment Request	The POS Client uses the SDK to make a payment request to the CommBank Payment app.
8	Payment Response	Once the payment finishes, a response is sent back to the POS Client.
9	Payment Response	The POS Client sends a message back to the POS over the established connection to the MOM. The format of this message is left to the POS vendor's discretion.
10	Payment Response	The payment response is routed to the correct web page. This could be done using a long running connection or webhooks. At this point the message can be verified, audit logged, augmented and finally routed.
11	Done	Web page indicates payment made. The iFrame closes and order/payment is complete.

Pros

- A web based POS can talk to an Albert.

Cons

- Complicated.
- Many moving parts.

Implementation Considerations

- There are multiple points of failure.
- Securing the connections and traffic.
- Network outages and connection retries.
- Payment app failures.
- Timeouts from a payment taking too long.
- A failure causing the POS to never receive a Payment Response.

6. Points to Consider

6.1 Security

See attached document for more detail and specific technical implementations of the points below.

Secured Communications
Merchants using the terminal or the application may be using unsecured WiFi to connect to the internet. Consider the nature of the communication sent and received by your app. Unless there is a specific reason, all communications should be secured e.g. HTTPS, SSL, SFTP.

Always authenticate remote services
Enforce webservice authentication prevents unauthorised access to merchant information. When in control of both the webservice and application, developers should look to enforce webservice authentication and authorisation to help prevent unauthorised access.

Certificate pinning
It is worth considering if you need to configure the expected certificate within the application code. If apps are local, or connecting only to public services, then this is not required. In all other scenarios, especially if the app sends sensitive customer information, then this may be necessary.

Logging guidance
Debug logging must be removed for production release, as this may be turned on by attackers once the app has been compromised, to gain access to sensitive merchant information. Removing debug and verbose logging may be performed as part of the "Application Obfuscation and Protection" with tools such as ProGuard.

At a minimum, debug and verbose logging code (i.e. Log.d() and Log.v()) must be removed for production deployments.

6.2 Other things to keep in mind

Receipt printing settings
Whilst you are able to access the printer as a third party app, the entire payment process flow may not be interrupted. You are able to use the printer before and after the Payment app has been called, but any merchant or customer receipts will be controlled entirely by the Payment app.

Merchant and Terminal IDs
You are able to access these via the SDK. Refer to SDK documents for more information.

Storing configuration settings
Consider the impact of a terminal hardware swap out. Your app configuration settings, data, and associated information, if stored solely on the terminal, may be lost in the event of a swap out.

UX considerations
The design and UX of your app should take into consideration the entire customer base of the merchant. This may include considering less technologically-savvy, vision or hearing impaired, or elderly customers.

Pay at table
For certain scenarios, such as pay at table, the terminal will initiate the transaction, rather than the POS.

Store and Forward (SAF)
When Albert can't connect to the host, a 'SAF' transaction may be initiated. In these instances the STAN for the transaction will be '0'.

7. Passing the Transaction

Variables

To initiate payment transactions, only amount needs to be passed. Setting the currency is optional as the Payment app will take the default currency set. No other parameters are required to initiate the payment transaction.

(Used with SDK 2.1.0)

- `AMOUNT`: The amount of the transaction.
- `STAN`: Unique identifier to identify each online transaction.
- `CURRENCY`: The currency in which the transaction was performed.
- `TX_RESPONSE_CODE`: The response code returned for the transaction.

Calling the Payment app

```
public void onPaymentButtonClick(View view) {
// Construct a new payment for $20.00.
PaymentRequest payment = new PaymentRequest(new BigDecimal("20.00"));
payment.setCurrency(Currency.getInstance("AUD"));
// Launch the Payment app.
startActivityForResult(payment.createIntent(), 0); }
```

Parsing Activity Result - Intents

```
@Override
protected void onActivityResult (int requestCode, int resultCode, Intent data) {
if (resultCode != Activity.RESULT_OK) {
   return;
   }
TransactionResult transactionResult = TransactionResult.fromIntent(data);
   // Check whether the transaction was successful
   switch (transactionResult.getTransactionStatus()) {
   case Approved:
      // Getting the response code from PA
      TransactionReferences transactionReferences =
      transactionResult.getTransactionReferences();
      // for demonstration purposes only,
      // please be careful when logging financial information!
      for (TransactionReferenceCode code : transactionReferences.getTransactionCodes())
      { Log.i(TAG, code.getName() + " : " + code.getValue()));
      }
      break;
   case Cancelled:
      // Transaction cancelled
      break;


   case Declined:
      // Transaction Declined
      break;
   case Error:
      // Error
      break;
   case Timeout:
      // Transaction Timeout
      break;
   }
}
```

8. Test and Submit on App bank

8.1 Testing

Please endeavour to test your device on an Albert as soon as possible. Contact us when you think you are at this stage. You are able to test using the Payment Simulator available within the SDK.

8.2 App Submission

Account Upgrade
In order to submit apps, your developer account needs to be upgraded.

We check that the bank account associated with app billing belongs to the correct organisation or entity, preventing fraud and identity theft.

A CommBank bank account is required to protect your account against fraud and identity theft. This will be used for billing, and revenue generated by your app will be deposited into this account. If you don't have one already, contact us at **SmartTerminals&Apps@cba.com.au** and a free account will be set up.

Submit an app
Once your account has been upgraded, you can submit your app. You'll need to have the following ready at a minimum:

- App launcher icon
- At least two screenshots of your app
- Source code, ensuring that the code for any private libraries is enclosed, as a single .zip file
- Disaster recovery plan

The process will take approximately four weeks, and undergoes business, technical, and security review.

App Updates
Should your app require any updates, navigate to the 'My Apps' > 'Manage' page. For the required app, click the "Resubmit" icon, and fill out and submit the form. You'll need to resubmit the updated source code, ensuring that the code for any private libraries is enclosed, as a single .zip file.

The process will take approximately four weeks, and undergoes business, technical, and security review.

Please contact us at **SmartTerminals&Apps@cba.com.au** or on the Forums if you have any further questions.

9. Billing

For private applications (those not publicly available on the Pi App bank store), CommBank charges $100 per merchant, per month. There is also a charge for the review of the app as part of the approval process.

More details can be found on the App bank site. See **Terms and Conditions**.

10. App Support

As the POS developer, you are responsible for supporting:

- The pairing process between Albert and the target POS.
- Communications to facilitate the pairing process.
- The installation of the POS app on Albert.

Support calls to CommBank will be referred on to the POS developer, proving the fault out of the CommBank systems.

Support obligations of any person registering as a developer in App bank as outlined in the **Terms and Conditions** on Pi App Bank as at 1 June 2015

5. ***Support Obligations***. *If an App is accepted for publication, You are responsible for providing technical support for App users. As part of the Development & Publication Process, You must satisfy CommBank that you have an appropriate support structure in place for any App that is submitted for publication. This will require that You have a relevant support model enabling you to directly service the Merchants' needs and requirements, enabling a successful adoption and optimised use of your App, including as a minimum:*
    - ***a.*** *adequate numbers of suitably skilled resources to ensure users of Your App receive prompt technical assistance when required;*
    - ***b.*** *an 8am to 6pm AEST helpdesk with a 24 x 7 email support/logging service;*
    - ***c.*** *the capability to provide support to users in accordance with suitable response times.*