1 Overview

The Service NSW Omni Channel Reference Architecture (OCRA) provides a pattern for the delivery of capability into the Service NSW Ecosystem.

The overall aim of the architecture pattern is to provide omni channel delivery of capability. That is where capability delivered in one program or channel, can be leveraged and provided through alternate channels without the need for major refactoring or rework.

1.1 Constraints

The OCRA works within a number of pre-defined constraints:

- AWS
- PCF
- Apigee

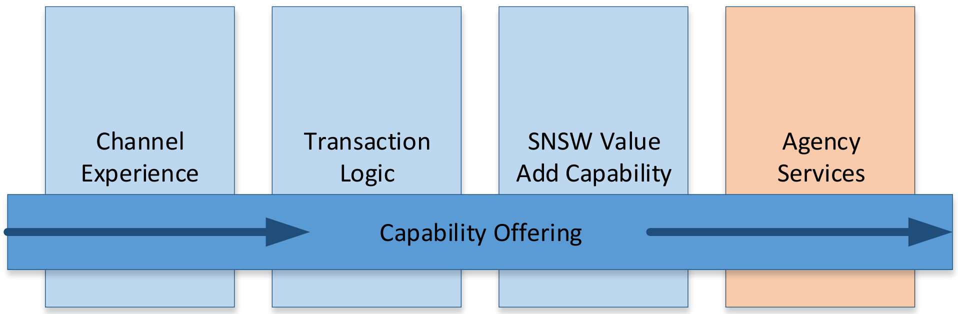
2 High Level Architecture

2.1 Layering

The overall context for OCRA indicates that there is a natural decomposition of the architecture into three distinct layers:

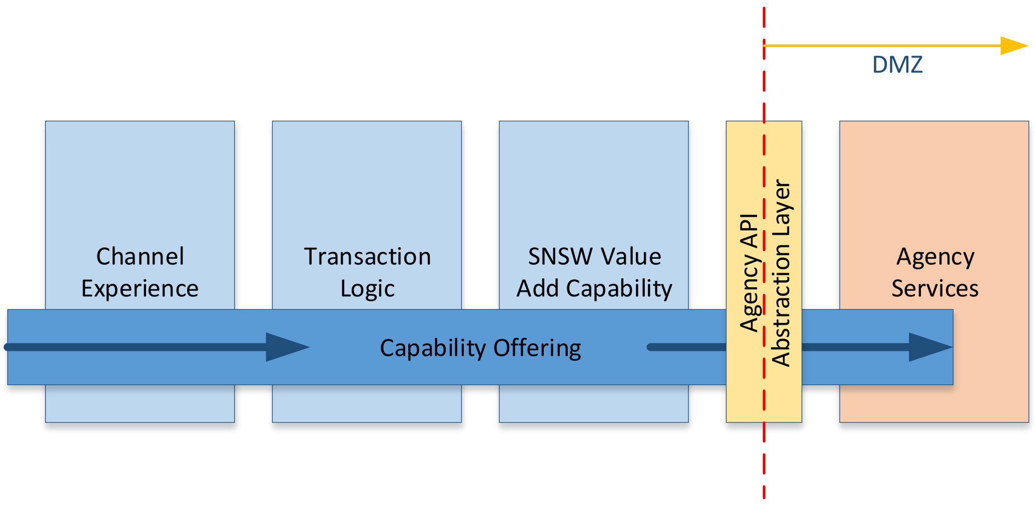
- Channel Delivery
- Transaction Logic
- Value Added Services

And that a transaction is delivered across these layers:



2.1.1 Agency Abstraction Layer

As Service NSW interacts with multiple downstream agencies, there is a need to standardise the interaction between SNSW and any downstream agency. This standardisation is achieved through the use of an API layer. The API layer abstracts the downstream agency and applies any security protocols required to interact with the agency, effectively placing them in the DMZ.

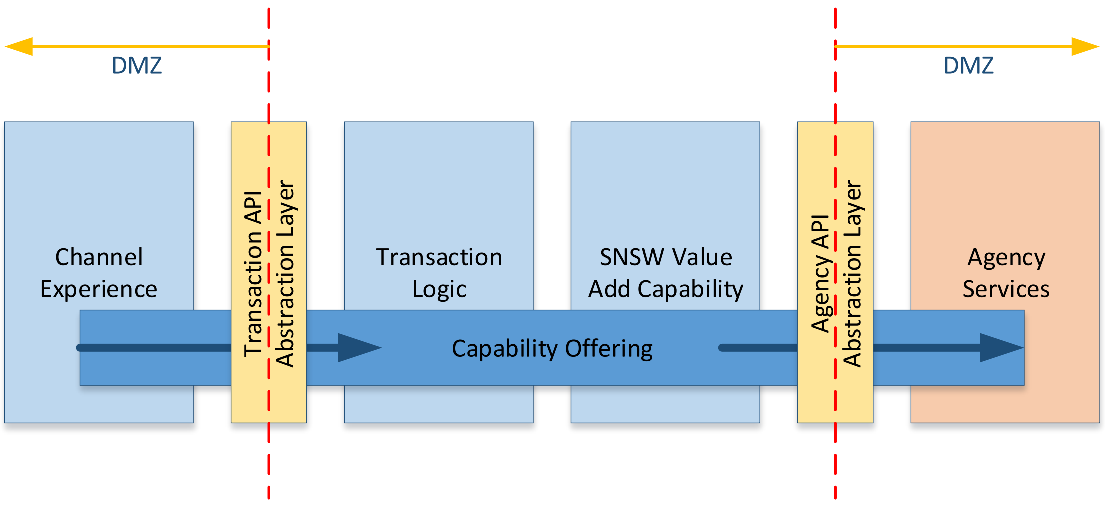


2.1.2 Channel Abstraction Layer

As the Channel Experience often runs outside of the Service NSW domain, on a Customer or External computer, there is a real risk that the Channel Experience is "hacked" and so calls from the Channel Experience should always be treated as 'suspicious'.

Equally there needs to be a level of conformity between different channels accessing the same Transaction Logic.

An Abstraction Layer is required between the Channel Experience and the Transaction Logic. Placing the Channel Experience in the DMZ.

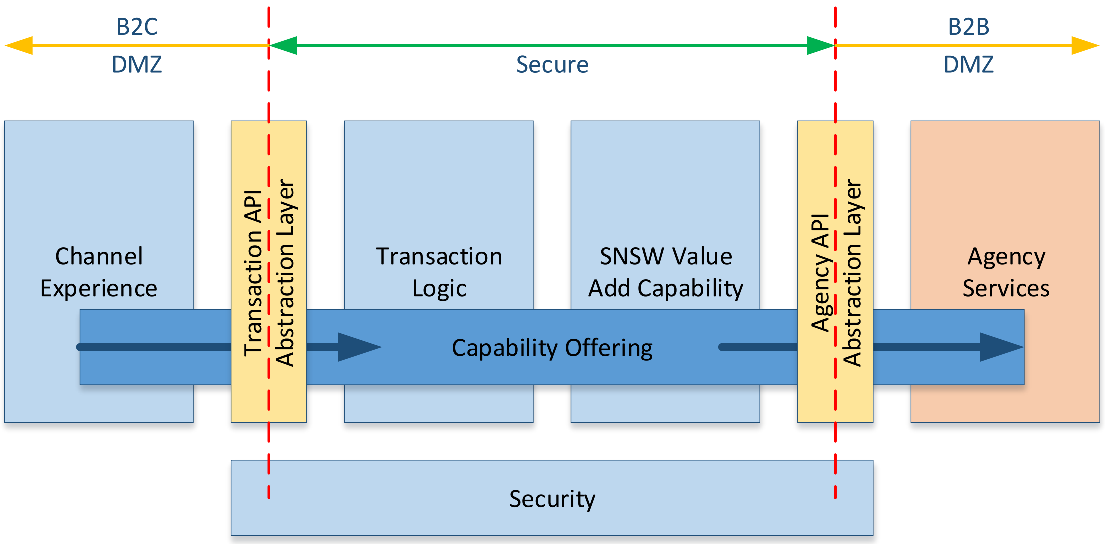

DMZ
DMZ
Channel Experience
Transaction API Abstraction Layer
Transaction Logic
SNSW Value Add Capability
Agency API Abstraction Layer
Agency Services
Capability Offering

2.1.3 Security

Through the use of the Abstraction Layers, the Transaction Logic and Value Added components now reside within a secure environment.

Access from external Customers or public is governed by the Transaction API Abstraction Layer using B2C security approach (OAuth). The Transaction API Abstraction Layer is required to enforce all access control security, that is both Authentication and Authorisation.

Access from downstream Agencies is governed by the Agency API Abstraction Layer using a B2B security approach (Certificates).



2.2 Component Modelling

Within the layering sits the various components that make up a transaction.

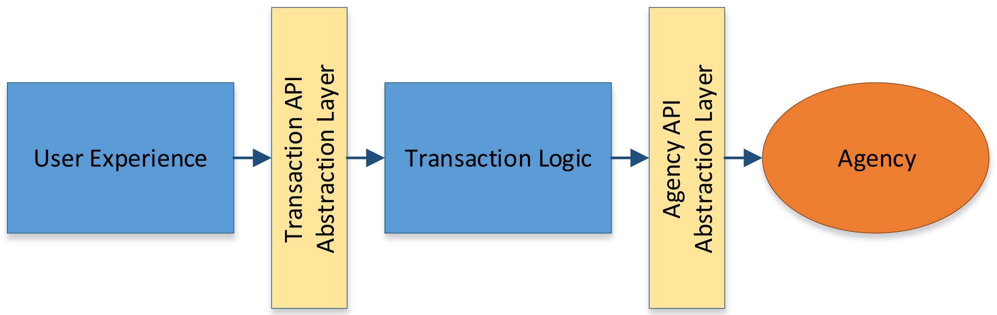
Any transaction will consist of a minimum of two components:

- User Experience
- Transaction Logic

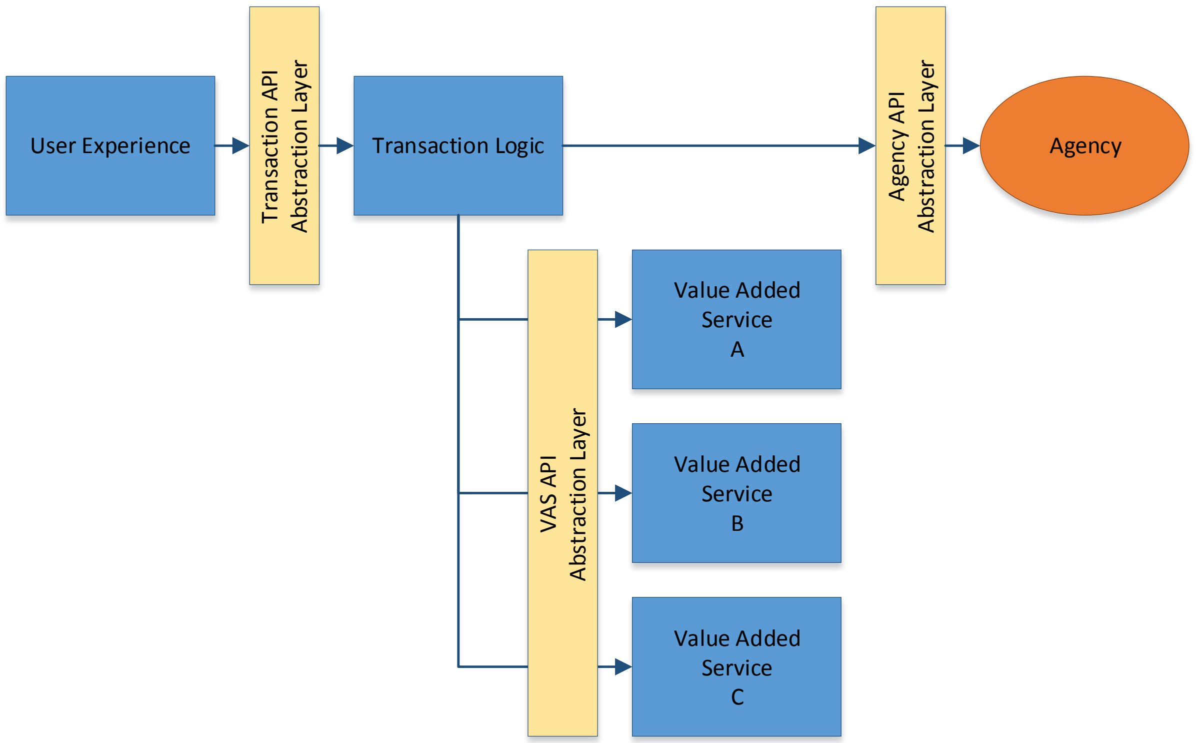
The User Experience will provide the look and feel to the end user for the transaction.

The Transaction Logic will provide the interaction between the User Experience and Agency Services.

Interaction between the User Experience, Transaction Logic, and Agency is always via the API Abstraction Layer.


User Experience
Transaction API Abstraction Layer
Transaction Logic
Agency API Abstraction Layer
Agency

Typically the Transaction Logic will enhance the overall transaction through the use of Value Added Services. In this situation the Transaction Logic will orchestrate multiple calls to internal common capability to enhance the overall transaction. Access to Value Added Services is via an internal Abstraction Layer.



2.2.1 User Experience

The User Experience consists of one of more components that provide the visual interface as used by the Customer (either directly such as Web, or indirectly via CSR).

The technology used within the User Experience is unstipulated, so long as it can be deployed as one (or more) components within PCF. While the User Experience technology is unstipulated, as a general guide many programs have settled on building the User Experience in Angular6.

The User Experience may access other external (non SNSW) services directly, such as for Address Validation. The User Experience will only access internal SNSW capability via the Transaction API Abstraction Layer.

The User Experience is expected to be light weigh (throw away), with the majority of the business rules being implemented in the Transaction Logic.

2.2.2 Transaction Logic

The Transaction Logic will typically consist of a single component that provides a limited set of APIs to interact with it.

The Transaction Logic will orchestrate the interaction with the downstream Agency, and also with any Value Added Services.

The Transaction Logic is intended to be stateless, with no internal persistence of data.

Good design practice will necessitate that the Transaction Logic is independent of the User Experience, and has no practical knowledge of which User Experience is calling it. This ensures that any Transaction Logic is channel agnostic, and so can be exposed through multiple channels as needed.

The impact of the channel agnostic design is that where Service NSW is obliged to enforce a business rule (such as 'minimum age') that this rule is implemented in the Transaction Logic so that it can be applied equally across all channels.

2.2.3 API Abstraction Layers

Integration between components is through the use of Abstraction Layers. In alighnment with the principle of "Lightweight Pipes and Smart Endpoints" the abstraction layers provide only pass through capability, and security.

The Abstraction Layers are not intended to provide any level of processing "capability".

2.2.4 Data Persistence

The overall model presented is for Stateless Transactions where data is only held in the Agency.

In practice a more practical approach to data persistence is required. Data is at times held at various points throughout a transaction.

2.2.4.1 User Experience

The User Experience operated within the DMZ, and so should not be considered a trusted component. The User Experience should therefore not persist any data directly, nor call any component or database simply for the purpose of persistence.

Where the User Experience may persist data is within a Customer Session, for the Customers Data that is used to build an end result data object to pass back to the Transaction Logic. This allows the User Experience to build a data model across a number of screens (such as a Wizard) and push this data to the Transaction Logic as a single data packet.

Customer Data should never be persisted outside the Session within the User Experience Layer.

2.2.4.2 Transaction Logic

The Transaction Logic is within the Secure Zone within SNSW, and so can persist data (if necessary). It is not expected that a Transaction Logic component persists data. Where data is persisted it can only be persisted to a Data Base or external repository.

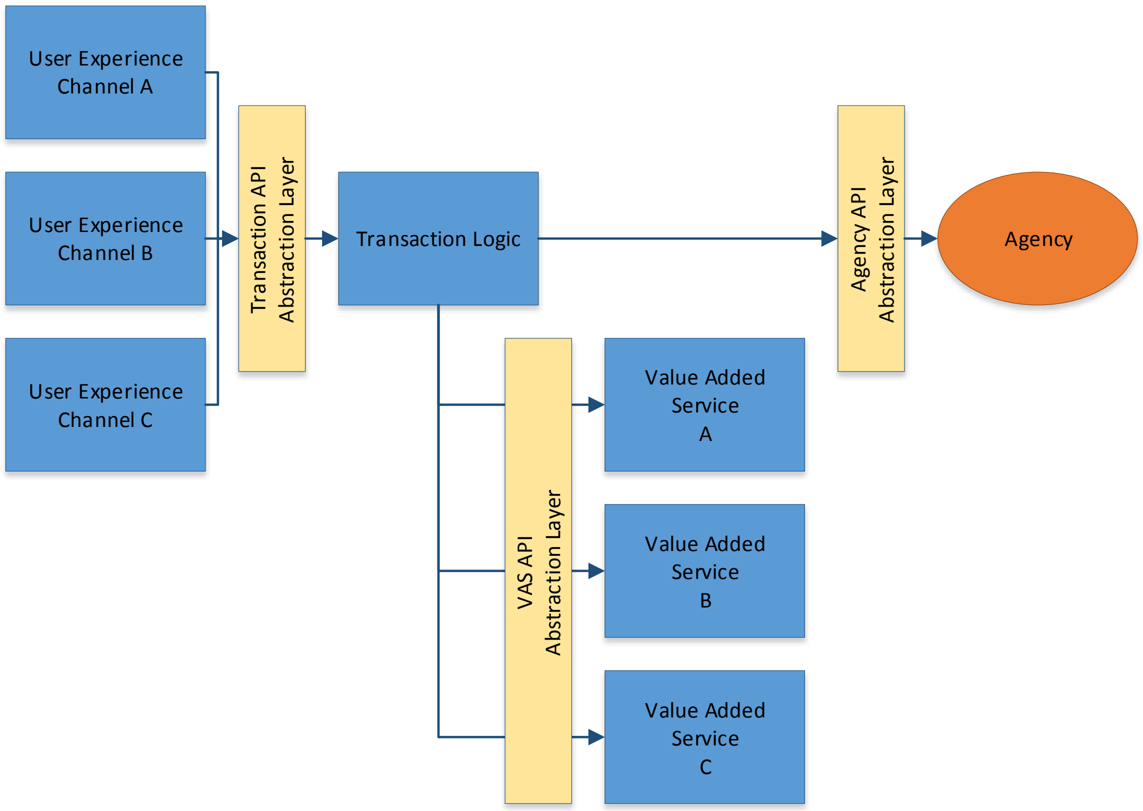
2.2.4.3 Value Added Service

The Value Added Service will, depending on capability, persist data (such as a Logging Value Added Service). Where data is persisted it can only be persisted to a Data Base or external repository.

2.3 Multi Channel Delivery

Multi channel delivery of a transaction capability is achieved through the delivery of multiple User Experience components, where each User Experience component represents a different channel of use.

Note the Multi Channel design allows a single transaction to be delivered equally across multiple channels. It does not cater from channel switching, this occurs in the Omni Channel Delivery.

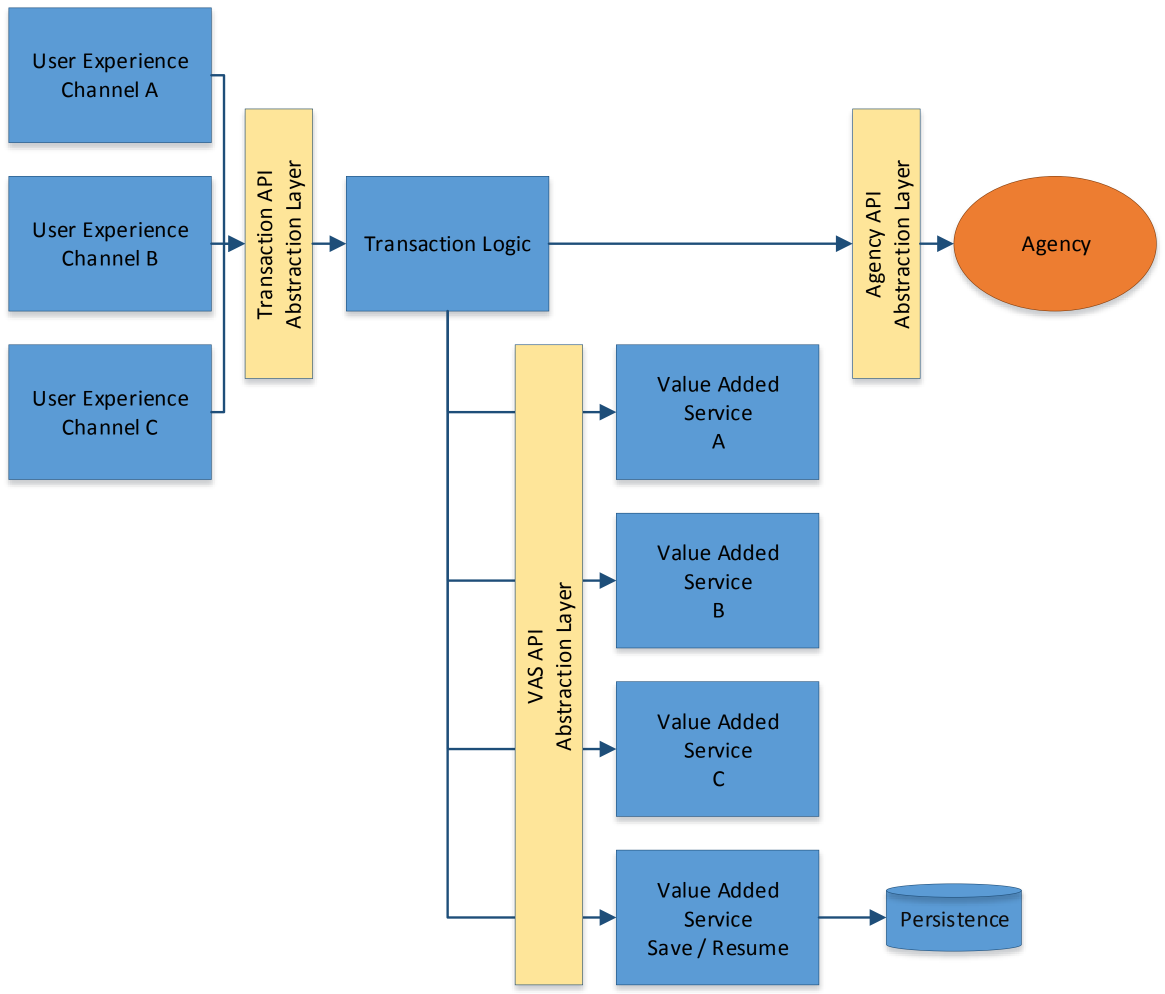

User Experience Channel A
User Experience Channel B
User Experience Channel C
Transaction API Abstraction Layer
Transaction Logic
Agency API Abstraction Layer
Agency
VAS API Abstraction Layer
Value Added Service A
Value Added Service B
Value Added Service C

2.4 Omni Channel Delivery

Omni Channel Delivery is achieved by allowing the Customer to switch channels part way through a transaction. To achieve this the transaction is required to save (and later reload) its state part way through a transaction, thus allowing the Customer to save state in one channel and reload state in another.

As the Omni Channel experience is desired across all transactional capability, the Save/Restore function is a prime candidate for a Value Added Service. Thus this component will allow any transaction to save its state and reload it at a future time.

Note the exact mechanism to key the data to allow it to be reloaded (ie what is used to identify the data) will vary between transactions. The transaction itself will need to know how to key the data.


User Experience Channel A
User Experience Channel B
User Experience Channel C
Transaction API Abstraction Layer
Transaction Logic
Agency API Abstraction Layer
Agency
VAS API Abstraction Layer
Value Added Service A
Value Added Service B
Value Added Service C
Value Added Service Save / Resume
Persistence

3 Summary

Service NSW offers Transactions across multiple channels.

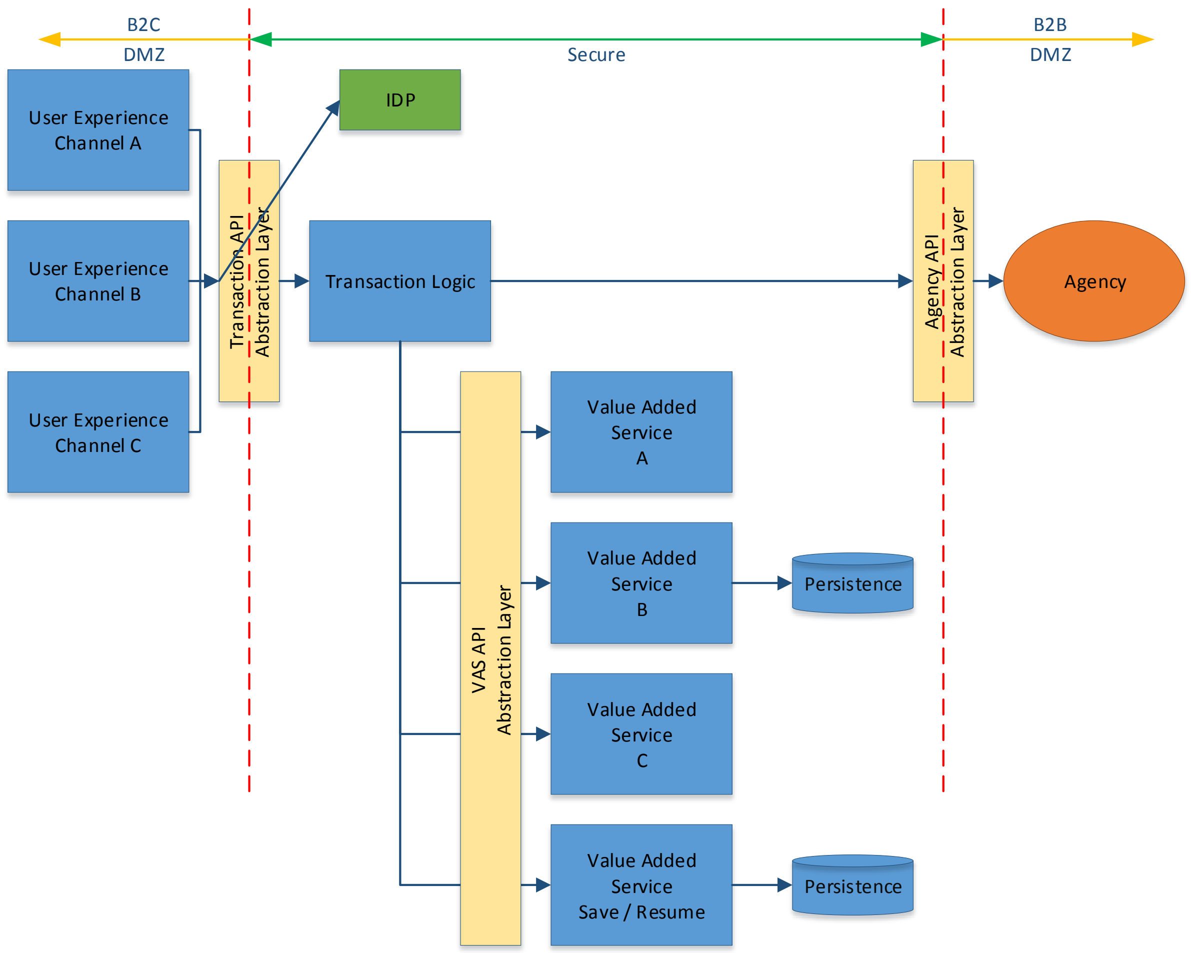
Each transaction is decomposed into a minimum of two primary components, User Experience and Transaction Logic.

Integration between components is via an Abstraction Layer.

Security is applied at the Abstraction Layer via the IDP.

Each Transaction can leverage existing Value Added Services to extend their capability.

The solution is intended to be light weight and stateless.


B2C
DMZ
Secure
B2B
DMZ
IDP
User Experience Channel A
User Experience Channel B
User Experience Channel C
Transaction API Abstraction Layer
Transaction Logic
Agency API Abstraction Layer
Agency
VAS API Abstraction Layer
Value Added Service A
Value Added Service B
Value Added Service C
Value Added Service Save / Resume
Persistence
Persistence