

Patterns for payment systems integration

Gary S. D. Farrow

Received (in revised form): 31st January, 2012

Triari Consulting, 30 Clothorn Road, Didsbury, Manchester, M20 6BP, UK.
Tel: +44 (0)161 445 5958; Fax: +44 (0)161 445 5958; e-mail: gary.farrow@triari.co.uk

Gary Farrow is Director of Triari Consulting, provider of systems integration and IT architecture services for the financial sector. A lead architect on many successful, high-value projects, he has undertaken senior architect roles at major banks and financial services firms in the UK. Gary has broad expertise in large-scale systems integration, enterprise service bus architectures and Service Oriented Architecture (SOA), and deep domain specialism in payments systems. He has written many articles on SOA and payments processing, discussing wide-ranging topics such as anti-patterns, the role of data and the optimisation of delivery models. Gary is a member of the IT Architecture Certification Board for the Open Group. His professional qualifications include Fellow IET, Chartered Engineer and Open Group Master Certified Architect. He holds BSc and PhD degrees from Manchester University and is an alumnus of Warwick Business School, UK.

ABSTRACT

Regulatory, industry and competitive drivers dictate that payment initiatives within a bank are many and continuous. However, a common situation is that the supporting IT systems have evolved in an ad hoc manner — there are often limited separation of architectural concerns, many tightly-coupled legacy systems and duplication of processing capability. Systems integration complexity is a limiting factor in determining how quickly IT changes can be achieved. These factors detract from a bank's ability to respond to changes in the business

environment. Selecting the appropriate architectural style for payments processing is therefore an extremely important activity. This paper presents architectural patterns for payment systems integration. These enable the selection of optimal architectural solutions for payments processing. Each pattern is described in terms of a number of generic Architectural Building Blocks¹ and their interactions. The general characteristics of each pattern are compared, and the suitability of each pattern in supporting specific business scenarios is presented. The paper also introduces the concept of the Payments Integration Space, which defines the scale of the systems integration problem. This metric is used to illustrate the role of the patterns in reducing complexity. It is shown that two specific patterns, Payments Hub and Payments Service Bus, are candidates for the target end state. Given significant differences in characteristics of the two patterns, it is concluded that a bank must make an early and overt choice about selecting the most appropriate target end state for payment systems integration.

Keywords: architectural patterns, transaction processing, Payments Hub, Payments service bus, systems integration, TOGAF

INTRODUCTION

Deficiencies in payments systems processing manifest themselves in business terms as:



Journal of Payments Strategy & Systems
Vol. 6, No. 1, 2012, pp. 15–36
© Henry Stewart Publications,
1750–1806

- high business operational overheads associated with correcting system processing errors;
- additional time to effect payments, leading to low customer and business partner satisfaction.

An effective payment processing capability, achieving timely completion of payments and having 'straight through' systems processing capability with low operational intervention is therefore considered vital. Also, regulatory, industry, competitive and cost-reduction drivers dictate that payments initiatives within a bank are many and continuous. Systems must be continually enhanced to support these initiatives and there is extensive integration effort to accommodate acquisitions and divestments. Systems integration complexity is thus a limiting factor in achieving such business changes quickly.

Elegant payments systems processing, having reduced integration complexity and shared system components, can help in enabling the business agility of a bank:

- providing faster speed to market of products and services;
- in meeting legal and regulatory compliance;
- in responding to industry initiatives quickly.

It can also offer many benefits to a bank's customers, notably the ready availability of the latest payment processing features, improved reliability and consistency of service, and improved transparency of the payment life cycle.

However, many barriers to effective payment processing exist. A typical scenario is that the underlying payments systems have developed in an *ad hoc* manner. Many legacy systems continue to exist, bundling scheme-specific transactional processing with account services.

Consequently there is limited separation of architectural concerns. Duplication of payments-related services, such as fraud detection and anti-money laundering, is common, and there is invariably no unified approach to systems integration. All these factors detract from a bank's ability to respond to changes in the competitive environment. Selecting the appropriate architectural approach to payment processing is therefore an important activity.

Architectural patterns provide a specific solution to a known IT problem. Thus, patterns provide an abstract solution template applicable to specific scenarios. In this respect, the architectural patterns help:

- in understanding what existing approaches are deployed, where they deviate from an ideal solution, and hence their limitations;
- illustrate solution options when payments processing is impacted due to a change in the business environment allowing the bank to make an informed choice of architectural solution;
- to provide an ideal target end state to support long-term payments processing planning and incremental improvement to the payments processing landscape.

This paper discusses a number of architectural patterns for payments systems integration.

First, an architectural model for payments processing is presented that forms a framework for the underlying analysis. The patterns are identified and described in terms of the building blocks of this model, supplemented with an additional component representing a systems integration construct.

Each pattern is presented in terms of the relationships between all the collaborating components. A summary of the characteristics of the specific integration component in the given pattern context is also provided.

Finally, the adoption of each of the patterns as the candidate target end state for payments processing is explored. To this extent, the degree to which each facilitates the desired system processing objectives is assessed, focusing on the extent to which the integration complexity is reduced.

PAYMENTS PROCESSING ARCHITECTURAL MODEL

An architectural meta-model is now presented that defines a number of abstract solution components required for payments processing. These coarse-grained solution components are termed Architectural Building Blocks (ABBs).³

The patterns identified are defined in terms of the relationship between these ABBs. In each pattern scenario, the responsibilities of the ABBs remain broadly similar, but the model allows for differences in functional capability depending on how specific payments Business Services are apportioned. The functional capabilities appropriate to payments processing have been previously defined by the author⁴ and these also form part of the architectural model.

The set of payments ABBs is now described.

Payment Gateway

This component represents the system that interfaces to a payments scheme. All communication to and from the scheme is done via this component. A Payment Gateway will process both inbound and outbound payments. Communication to and from the Gateways are typically made using industry data standards.

Channel

This component represents a channel through which a customer interacts with the bank. It may also correspond to the front/back office system used by internal

bank staff to perform payments on behalf of a customer or for the bank itself. A channel system will typically only initiate outbound payments. Communication from the Channel components is usually by using an internal data format unique to the bank, although a derivation of an industry format is also often the basis of this.

Product System

This component represents a core banking system, domiciling the customer accounts that are the remitter and beneficiary of payments.

The remitter and beneficiary accounts would typically each be domiciled in different bank product systems, representing the sending bank and receiving bank. In the case of an 'on us' payment, both remitter and beneficiary accounts are domiciled within the same bank's product systems.

Other circumstances may arise when a particular bank is acting as an intermediary for payments processing for another bank (such as Agency banking or Correspondent banking arrangements). In these circumstances, both remitter and beneficiary accounts are external to the given bank.

Payment Engine

This component performs the activities required to process a payment, these being the discrete set of functions in its value chain within the organisation. The Payment Engine must be invoked:

- upon receiving the payment instruction from the scheme via the Payment Gateway;
- upon initiation of a payment from a Channel or Product System;

The processing applied to each payment is specific to each scheme. In this respect, in this model, a particular Payment Engine processes payments for a given scheme only.

Figure 1
Payments
processing
capabilities (after
Farrow)⁵



A Payment Engine is also referred to in payments processing as a *Transaction Processing System*.

Payment Business Services

Payment Business Services represent separate components each providing a payments functional capability. A number of such capabilities exist and a model for payments processing has been defined previously and is shown in Figure 1.

Payment Business Services must be orchestrated to fulfil the required processing on each inbound or outbound payment. There is not necessarily a single control point for this orchestration, which may be distributed across a number of components. For example:

- Channel components may call validation services to ensure a well formed payments message is constructed.
- A Payment Engine will orchestrate a number of such services to post account entries and to call financial crime services, such as fraud, sanctions and anti-

money laundering checks.

- A separate integration component may call required technical integration services such as routing and transformation.

Payment Processing Spectrum

These capabilities above have been structured previously into an ordered list, termed the *Payments Hub Spectrum*.⁶ Simplistically, middleware capabilities are positioned to the left of the Spectrum with Payments Business Services of increasing functional richness positioned to the right of the Spectrum. Since a 'Hub' is subsequently shown to constitute one very specific payment processing pattern, this paper references this Spectrum more generally as the *Payments Processing Spectrum*, shown in Figure 2.

In the integration patterns defined, these capabilities are shown to be apportioned across the set of ABBs identified. The manner in which these capabilities are apportioned differs for each pattern and is highlighted in each pattern description.

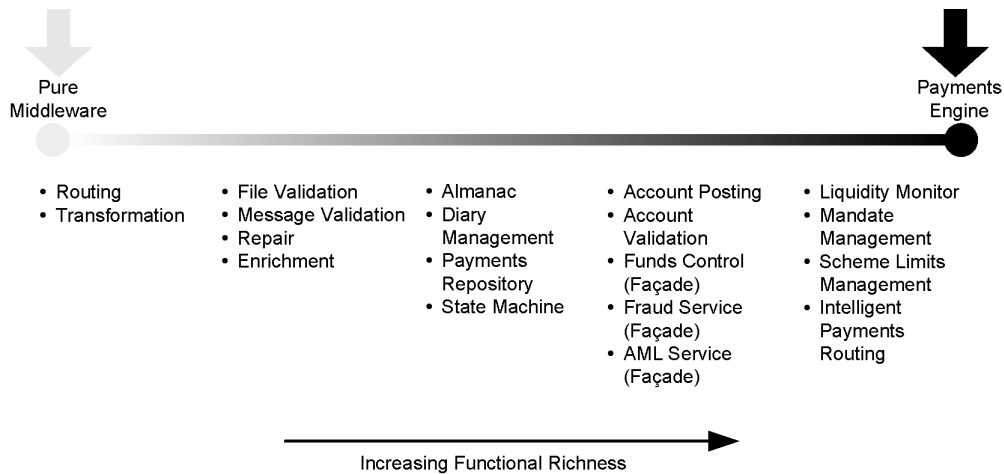


Figure 2
Payments
Processing
Spectrum

Payments Integration Space

The generic *Integration Space* is defined as the set of architectural components and their connections required to support the processes of a given business domain. The Integration Space is defined without middleware components.

Figure 3 shows the Integration Space specifically for the payments business domain, expressed in terms of the Payments Processing Architectural Model described above, the ABBs equating to the architectural components. The shaded area represents an indeterminate number of point-to-point interconnections between the ABBs. External B2B connections are required to support interaction with banks, commercial partners, agency banks, payment schemes and messaging systems. B2C connections are also required with the bank's customers.

A variety of interconnections are required within the bank's organisational boundary. It is these *internal* organisational connections within the Payments Integration Space that are the subject of analysis and to which the patterns presented are applicable.

Interconnections are shown between

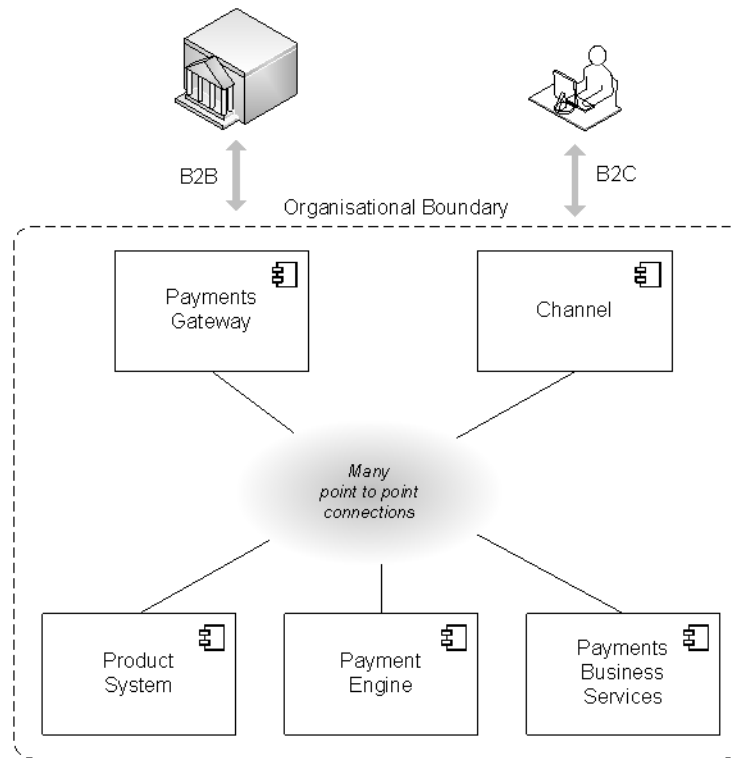
all components. The complexity of the Integration Space is defined by the number of interconnections between the payments system application components.

Table 1 shows the cardinality of each of the inter-connections between the ABB components identified for the Payments domain (m denotes many entities).

The Complexity Table provides a convenient view of the complexity of the integration problem. Many-many connections represent a *hotspot* of complex integration. It is seen that most interconnections between the generic components are many-many. Channels typically connect to Gateways via a Payment Engine and not directly, and so there is no Channel to Payment Gateway connection. Similarly a Channel does not connect to other Channels, nor does a Payment Engine typically connect to other Payment Engines.

The Payment Gateway to Payment Engine connection is shown as 1-1, reflecting the stated model that a Payment Engine is dedicated to one specific scheme and will process payments only for that scheme via its Gateway. In some circum-

Figure 3
Payments
Integration Space



stances, a common messaging infrastructure is used across several schemes and also to communicate with business partners, such as Agencies and Correspondents. The prime example of this is the SWIFT messaging infrastructure. In this scenario, a single Gateway may connect to multiple Payment Engines and the relationship of one Gateway to many Engines (1-m) is also possible.

A further point to note is that the Product System is the only component that connects with itself. This reflects 'on-us' payments made between customers of the same bank.

The Complexity Table must be populated for a given bank to create a specific, quantified, table instance. This then provides a unique view of the Payments Integration Space for that bank and enables a 'heat map' to be deduced, highlighting the areas of greatest integration

complexity. An example is provided later in the analysis.

PATTERN DESCRIPTIONS

Six architectural patterns for payments integration are now presented:

- (i) Payment Engine Routing Utility;
- (ii) Product System Routing Utility;
- (iii) Front End Payments Hub;
- (iv) Back End Payments Hub;
- (v) Payments Hub;
- (vi) Payments Service Bus.

Each pattern incorporates the ABBs described in the architectural model, supplemented with an additional component representing a systems integration construct.

In each pattern, the relationships, including cardinality, between all the

Table 1: Payments Integration Space Complexity Table

	<i>Channel</i>	<i>Payment Gateway</i>	<i>Product System</i>	<i>Payment Engine</i>
Channel	–	–	m-m	m-m
Payment Gateway	–	–	m-m	1-1
Product System	m-m	m-m	m-m	m-m
Payment Engine	m-m	1-1	m-m	–

Table 2: Pattern attribute descriptions

<i>Attribute</i>	<i>Description</i>
Integration Capability	Defines whether the component has integration capability
Business Capability	Defines whether the component incorporates functionality from one or more of the business capabilities
Process Capability	Defines the extent of the process orchestration capability
State Management	Defines the extent of the state management of the component
Connection Style	Defines the connection style in terms of how and when the component is invoked

components are illustrated. A summary of the characteristics of the integration component in the given pattern context is also provided. These characteristics are qualified as a set of attributes, shown in Table 2.

For each pattern, scenarios where the pattern is considered useful are described. Finally, using the Payments Processing Spectrum, the apportionment of functionality to the components in each pattern is illustrated. This provides a convenient visual perspective to compare the patterns.

Payment Engine Routing Utility

Overview

This pattern represents a solution for integrating a single Payment Gateway into multiple payment engines or Product Systems.

The Payment Gateway is connected to a single Payment Engine Routing Utility (Figure 4). This, in turn, is connected to two or more Payment Engines/Product Systems. The Payment Engines and

Product Systems themselves are assumed to be tightly coupled and are shown together, effectively acting as one entity. Thus the pattern does not address the integration problem between the Payment Engine and Product System components.

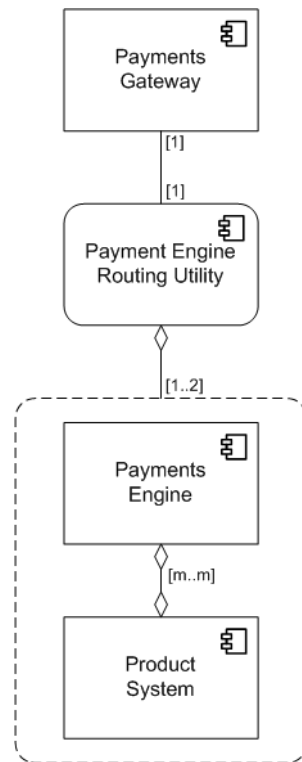
In general there are three categories of functional capability that may be performed by pattern component:

- (i) Integration Services;
- (ii) Business Services;
- (iii) Process Services.

The sole function of the Payment Engine Routing Utility is to interface to the Gateway and route payment instructions to and from the Payment Engines. Thus, of the three categories, the Payment Engine Routing Utility provides only Integration Services. The combination of the Payment Engine and the Product Systems provide the rest of the payments processing capability.

The Payment Engine Routing Utility does not manage any business state. It may,

Figure 4 Payment
Engine Routing
Utility pattern



however, manage technical state relating to interactions with the Gateway.

Spectrum Apportionment

The Payment Engine Routing Utility construct provides pure middleware services and is shown in Figure 5 occupying the left-hand side of the Spectrum.

Capabilities from the Spectrum utilised in this pattern include:

- *Routing*, to achieve the Payment Engine selection;
- *Transformation*, to transform from scheme to respective formats required by each of the Payment Engines;
- *Validation* of inbound messages received from the scheme.

Scenarios

This pattern is useful when introducing a new Payment Engine. Typically the migra-

tion from legacy engine to new engine cannot be achieved in one 'big bang' and a phased approach is adopted, requiring old and new payment engine instances to be concurrently live.

As a specific example of this pattern, a global commercial bank intended to replace its legacy Payment Engine for CHAPS processing with a new product solution, this being Fundtech Global PAYplus. A Routing Utility was introduced to switch payments to one or other of the Payment Engines. This approach de-risked the solution implementation, allowing for a gradual migration of customers to the new Payment Engine.

Similarly it has use in supporting a banking integration. In this situation, a payments scheme Gateway may be consolidated down to one of the original banks' Gateway components. Underlying internal systems processing in the short term may be untouched, requiring routing from the consolidated Gateway to each of the original banks' respective payment engines.

Summary

Payment Engine Routing Utility

Attribute	Value
Integration Capability	Yes
Business Capability	No
Process Capability	No
State Management	No business state, but potential technical transaction state
Connection Style	Single Pass

Product System Routing Utility

Overview

This pattern (Figure 6) represents a solution to integrate a single Payment Engine to multiple Product Systems.

The Payment Gateway is connected to

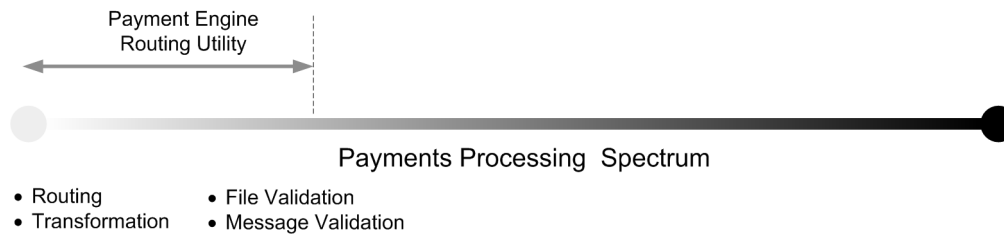


Figure 5 Payment Engine Routing Utility Spectrum Apportionment

a single Payment Engine, dedicated to a specific scheme. The pattern assumes that customer accounts are domiciled across multiple Product Systems. Commonality of processing for all payments relating to one scheme is achieved, but there is an integration problem to route payments to multiple Product Systems that the Routing Utility resolves.

As per the Payment Engine Routing Utility, the Product System Routing Utility provides only Integration Services. The combination of the Payment Engine and the Product Systems provide the rest of the payments processing capability.

The Product System Routing Utility component also does not manage any business state. It may, however, manage technical state relating to interaction with the Product Systems; for example, technical retries and the batching of requests for offline Product Systems.

Spectrum Apportionment

As per the Payment Engine Routing utility, capabilities employed in this pattern are middleware services, and hence the pattern apportions capabilities from the left-hand side of the Spectrum (Figure 7).

Capabilities from the Spectrum utilised in this pattern include:

- *Routing*, to achieve the Product System selection.
- *Transformation*, to transform from scheme to respective formats required by each of the Product Systems

- *Validation* of outbound messages and bulk payment files received from the Product Systems

Scenarios

This pattern is useful in the scenario relating to the introduction of a new payment scheme. In such circumstances, a new Payment Gateway is assumed. The scenario then further assumes that a new Payment Engine is introduced, dedicated to the new scheme. The Routing Utility

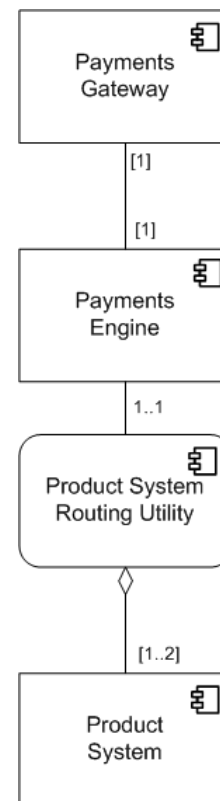


Figure 6 Product System Routing Utility Pattern

Figure 7 Product System Routing Utility Spectrum Apportionment



then provides the necessary integration services to interface the new Payment Engine to the existing Product Systems.

As a specific example of the pattern, when the Faster payments Scheme (FPS) was introduced in the UK, a large retail bank deployed a dedicated solution stack comprising ACI Gateway as the Gateway component and ACI Money Transfer System as the Payment Engine. A Routing Utility was developed to route payments to and from multiple Product Systems using IBM WebSphere Message Broker.

The pattern is also considered relevant when a new Product System is introduced in the bank's IT estate. This may arise:

- As a consequence of core banking platform migration. In such a scenario, account migration is not achieved in one 'big bang' and a phased approach is required, where old and new Product Systems are concurrent. Thus, co-existence of the multiple Product Systems is required as an interim state and the Routing Utility is a transient construct.
- As a consequence of a banking integration when new Product Systems are acquired. This pattern equates to a different integration point for the integrated bank systems, this being the Product Systems rather than the Payment Engine as per the Payment Engine Routing Utility pattern.

Summary

Product System Routing Utility

Attribute	Value
Integration Capability	Yes
Business Capability	No
Process Capability	No
State Management	No business state, but potential technical transaction state only relating to Product System interactions
Connection Style	Single Pass

Front End Payments Hub

Overview

This pattern (Figure 8) constitutes a generalisation of the Payment Engine Routing Utility pattern. It is also similar to the *Front End Landing Zone*, this being a category of Payments Hub suggested by Hayden *et al.*⁷ In this respect, the pattern represents an architectural formalisation of this particular payments integration strategy using the constructs defined in the Payments Processing Architectural Model.

The Front End Landing Zone is shown processing only payment instructions input from the Channels, and the integration from Channels to Hub is shown as being unidirectional. The Front End Payments Hub connects to several Payment Gateways, each relating to a scheme in

which the bank participates. In this respect, the Front End Payments Hub processes both inbound and outbound payments instructions from the Payment Gateway and hence the integration is bi-directional. It also connects to one or more Channels, these being end-user delivery channels. They are typically a source of payment instructions, and hence a source of outbound payments (or on-us) payments only, rather than a target for inbound payments.

The Front End Payments Hub connects to one or more Payment Engines but does not connect to the Product Systems. Thus, as with the Payment Engine Routing Utility pattern, this pattern does not address the integration problem between the Payment Engines and the Product Systems.

Spectrum Apportionment

This pattern construct provides middleware services as per the two Routing Utility patterns. It also provides the opportunity to introduce several payment Business Services (Figure 9). Considering the Capability Model, in addition to the Routing, Transformation and Validation, candidate Business Services include:

- *Diary management*, providing functionality for altering on files that are expected to be received or sent to a scheme on a given day according to the scheme operating times;
- *Almanac*, this being the determination of the Diary events for a given day;
- *Anti-money laundering (AML) and fraud services*, for example, sanctions checking on all inbound payments and event feeds to AML and fraud management system components;
- *Account validation*, to validate the beneficiary account for an inbound payment;
- *Repair*, supporting the correction of payment instructions incorrectly formed from the channels;
- *Enrichment*, supporting enrichment of

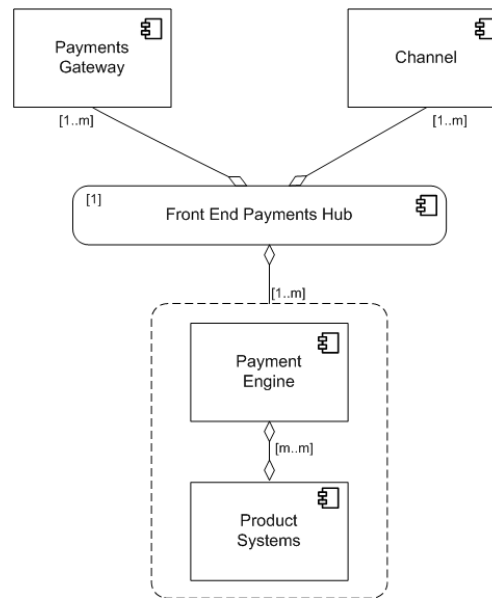


Figure 8 Front End Payments Hub

the payment instructions with bank-specific technical or business data;

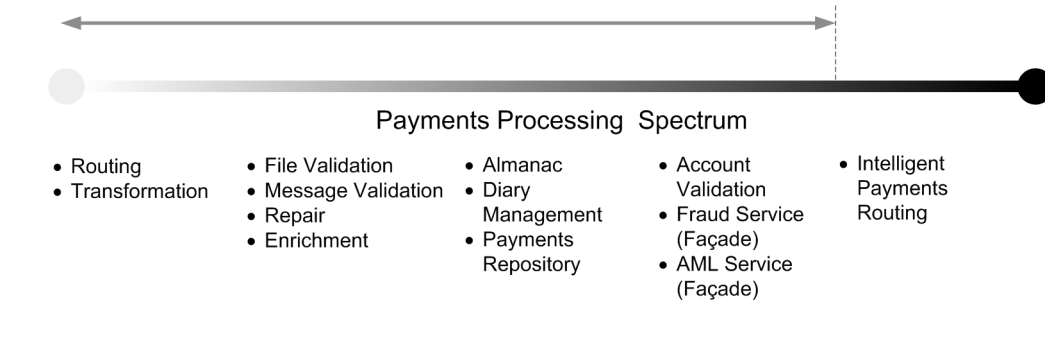
- *Payments Repository*, providing a warehouse of all payments instructions for audit purposes. In this pattern, the Front End Payment Hubs has visibility of all inbound payments from the schemes and all outbound payments originated from the channels. It is therefore a sensible point of control for updating or implementing a centralised Payments Repository;
- *Intelligent payments routing*, enabling scheme selection based on general criteria, for outbound payments originated from the Customer Channels.

Process services may therefore also be required to orchestrate these Business Services.

Scenarios

This pattern addresses the payments channel integration problem across the entire bank. Thus, It is most relevant when the channel integration problem is more complex than the back-end integration prob-

Figure 9 Front
End Hub Spectrum
Apportionment



lem. Circumstances where this may arise are when a bank is a member of many schemes and has many delivery channels but only a limited number of Product Systems. In such circumstances, this pattern is considered a viable solution.

Secondly, this pattern has been suggested⁸ as an interim state towards achieving a more complete payments integration solution (ie one that addresses front- and back-end integration). In this case, the Front End Payments Hub supports a strategy for the phased introduction of common payments-processing services, leading to the desired end-state solution. The form of such end-state solutions is defined and discussed in two further candidate patterns: *Payments Hub* and *Payments Service Bus*.

Summary of characteristics

Front End Payments Hub

Attribute	Value
Integration Capability	Yes
Business Capability	Provides some shared services or delegates these to other components
Process Capability	Some orchestration capability required
State Management	Stateful or stateless, depending on the specifics of the Business Services employed
Connection Style	Single Pass

Back End Payments Hub

This pattern (Figure 10) is a generalisation of the Product System Routing Utility pattern. It also broadly equates to *Back End Aggregation*, this also being a further category of Payments Hub suggested by Hayden *et al.*⁹ Again, the pattern represents an architectural formalisation of this payments processing strategy using the defined Architectural Model constructs.

In this pattern, the Back End Payments Hub connects to one or more Product Systems. It may also connect to some Business Services, each providing a service common to all payments processing.

It is seen that the Back End Hub pattern addresses the integration problem of integrating the Payment Engines to the Product Systems. The Back End Payments Hub provides the services to integrate the Product Systems and other components providing Business Services. In this way it exposes Business Services that may be used by several Payment Engines to fulfil the payment process. Thus, in this pattern, the Payment Engine is the main provider of the process services, fulfilling a payments process using Business Services exposed via the Back End Payments Hub.

Spectrum Apportionment

As per the Front End Hub, it may provide a number of Business Services (Figure 11) or delegate these to other components providing shared Business Services.

Candidate Business Services for this pattern, include:

- *Account Posting*, to post to the back end Product Systems;
- *Mandate Management*, providing centralised management of mandates for outbound standing orders, decoupling this functionality from the Product Systems;
- *Intelligent Payments Routing*, enabling scheme selection based on general criteria, for outbound payments. Payment origination for this pattern relates to mandates only;
- *Payments Repository*. In this pattern, the Hub construct has visibility of outbound payments originated by the Product Systems, but not necessarily all inbound payments from the scheme or initiated from the Customer Channels. It is therefore a point of control only for updating a centralised Payments Repository for the outbound payments relating to mandates.

As per the Front End Payments Hub, Process Services may therefore also be required to orchestrate these Business Services, although as is seen, the scope is more limited.

Scenarios

This pattern addresses the back-end payments integration problem across the

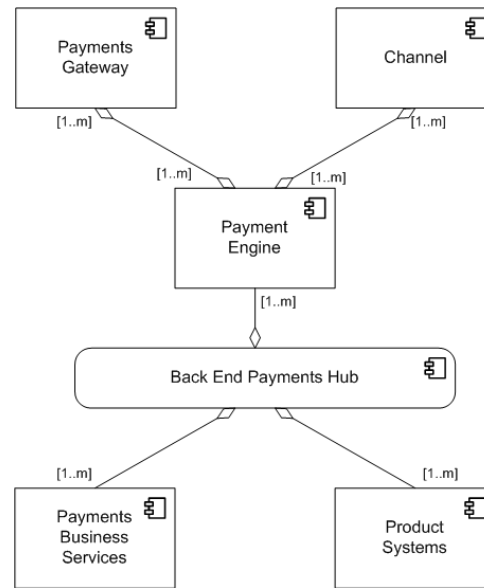


Figure 10 Back End Payments Hub Pattern

entire bank. Thus, this pattern is most relevant when the back-end integration problem is more complex than the channel integration problem. A circumstance where this may arise is when a bank is a member of a limited number of schemes and/or has limited delivery channels, but has several Product Systems. In such circumstances, this pattern is considered an appropriate solution.

Again, this pattern has been suggested as only an interim state towards achieving a more holistic end-state payments integration solution. The Back End Payments Hub, in this case, supports a strategy for

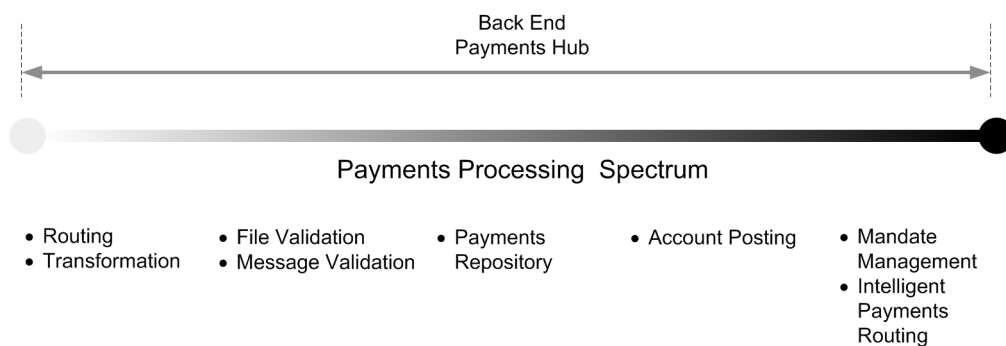


Figure 11 Back End Hub Spectrum Apportionment

the phased introduction of common payments processing services, leading to the desired end-state solution.

Summary of characteristics

Back End Payments Hub

Attribute	Value
Integration Capability	Yes
Business Capability	May provide or delegate some services
Process Capability	Some limited capability may be required
State Management	No business state, but may manage technical state relating to transactions with Product Systems
Connection Style	Single Pass

Payments Hub

Overview

The term 'Payments Hub' is a widely used expression for a generalised solution to payments processing. This pattern (Figure 12) provides a more precise definition of a Payments Hub in terms of the Architectural Model constructs.

In this pattern, the Hub is the central integration point of all the components involved in payments processing. None of the components interact without communication through the Payments Hub. All the necessary integration capability resides in the Payments Hub. Connections between the components require only a single pass through the Payments Hub.

In this respect:

- One or more Customer Channels are connected to the Payments Hub.
- One or more Product Systems are connected to the Payments Hub.
- One or more Payment Gateways are connected to the Payments Hub.

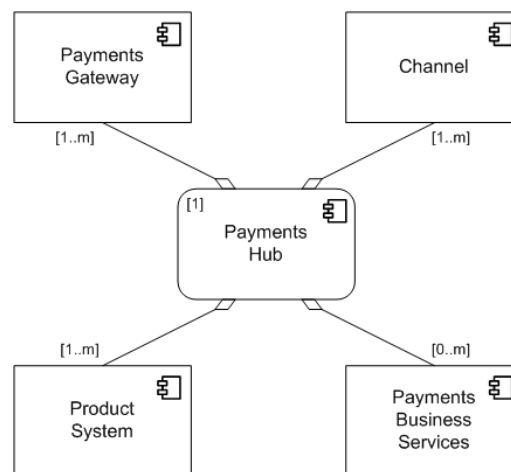


Figure 12 Payments Hub Pattern

- The Payments Hub may either:
 - provide Payments Business Services itself (internal capabilities); or
 - connect to zero or more externally provided Payments Business Services.

All orchestration of Business Services is undertaken by the Payments Hub. In this pattern, the Payments Hub contains the aggregate set of Process Services capabilities required to support all scheme processing. The Payments Hub therefore subsumes the functionality of the Payment Engine (transactional processing) components, and this is not required in this pattern. Some commonality of processing steps may be achieved across schemes but, more significantly, reusable shared services are employed to undertake the specific tasks in the processing chain.

The placement of the orchestration capability within the Hub infers business state management, as the Hub must accommodate and manage a variety of business exception conditions that typically occur in processing a payment.

Spectrum Apportionment

In terms of the categories of service provided by the Hub, it provides all of:

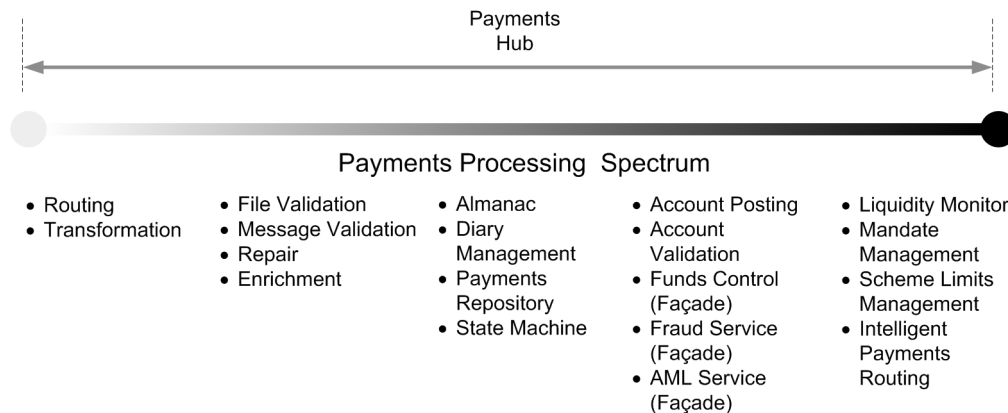


Figure 13
Payments Hub
Pattern Spectrum
Apportionment

- Integration Services;
- Business Services;
- Process Services.

Thus, in this pattern, the Payments Hub construct may employ capabilities from the entire Spectrum. Some of the capabilities may be provided by the Payments Hub itself, or some or all of the capabilities may be delegated, but always called from the Hub. Unlike the Front or Back End Payments Hub, this construct has visibility of all payments from all schemes and channels. This enables the opportunity for additional Business Services to be provided or orchestrated from this component (Figure 13), specifically:

- *Liquidity Management*, monitoring the overall liquidity position of the bank with respect to the schemes;
- *Scheme Limits Management*, providing the opportunity for alerting and pro-active management should the scheme limits be approached or if individual payments exceed the limits;
- *Payments Repository*, now encompassing all payments;
- *Transformation*, providing a more general transformation service of payment messages into a specific canonical data

format appropriate to the bank. Achieving this allows for harmonisation of payments processes, simplification and reuse;¹⁰

- *State Machine*, this being necessary to support the stateful processing performed by the Payments Hub.

Scenarios

This pattern provides a generalised solution to payments integration within a bank. It is applicable when there is a complex integration problem with respect to the channels and the back end Product Systems — simplistically many Gateways and Customer Channels — and many Product Systems.

The pattern provides a single solution to scheme-specific transactional processing. Thus, it is useful in achieving architectural simplification as it consolidates all Payment Engines into one component.

As an example of the use of this pattern, a UK retail bank embarked on a programme of core banking modernisation, introducing Infosys Finacle as the new Product System. The bank was a full member of all UK payment schemes. Channel integration complexity was high, but there were a limited number of Product Systems. However, to support a

strategy of gradual migration of payment services to the new banking platform, a Payments Hub solution was conceived and implemented using technologies from Clear2Pay.

Summary of characteristics

Payments Hub

Attribute	Value
Integration Capability	Yes
Business Capability	Yes
Process Capability	Yes
State Management	Stateful with respect to business state and also technical state
Connection Style	Single Pass

Payments Service Bus

Description

The Payments Service Bus (PSB) is also a widely used term for a generalised solution to payment processing within the bank (Figure 14). The differences between this and the Payments Hub are how clearly articulated in terms of the Architectural Model.

Unlike the Payments Hub, this pattern is premised on the existence of a number of separable Payment Engines, typically one per scheme. All integration to and from the Payment Engines is provided by the Payments Service Bus. In this way, a common utility is provided for all payments integration within the bank. The Payments Service Bus can be considered as a specialism of an Enterprise Service Bus, but specific to payments integration.

The PSB connects to all Consumer Channels and Gateways. It also connects to multiple Product Systems and to shared Business Services. No payment instruction passes between the payments system components without mediation by the PSB.

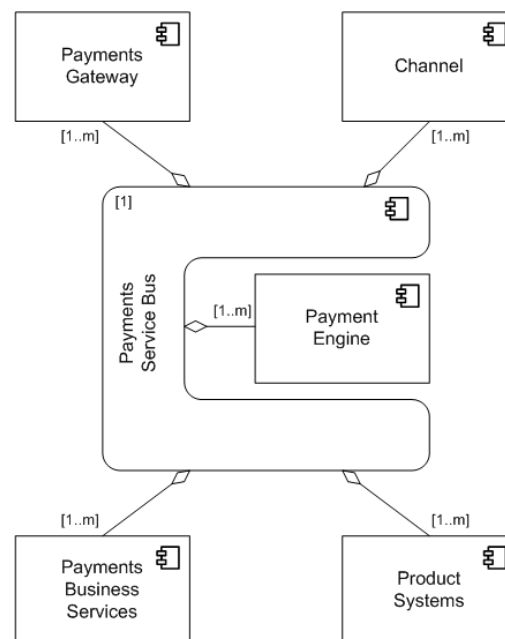


Figure 14 Payments Service Bus Pattern

Business Service orchestration is provided solely by the Payment Engines, each of which may be tailored to provide the specific processing demanded by a scheme, optionally complemented by the banks own value-add services.

Processing of an inbound payment instruction, from receipt at a Gateway to posting to an account in a Product System, requires several passes through the PSB:

- collection from the Gateway with routing and onward transmission to the correct Payment Engine;
- calls to a number of the Business Services, orchestrated by the Payment Engine, but mediated by the PSB;
- selection of the recipient ledger system and associated account posting to the correct Product System.

The PSB itself need only provide stateless 'one shot' services. It is the Payment

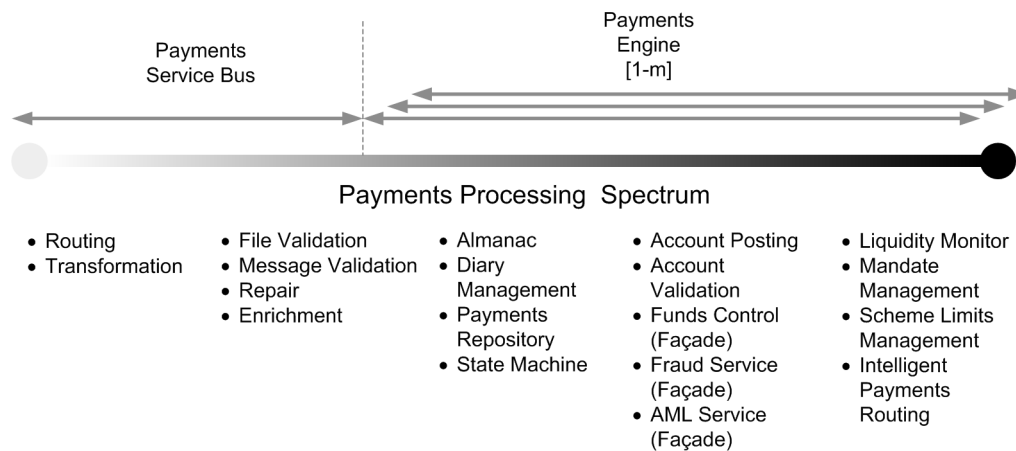


Figure 15
Payments Service
Bus Spectrum
Apportionment

Engines that manage any payment business state if it is necessary to do so.

Spectrum Apportionment

In this pattern, the Bus construct provides only the integration services and hence occupies the left-hand portion of the Spectrum. Multiple Payment Engines provide scheme-specific process services and implement (or orchestrate) some or all the Business Services, and hence are shown occupying the middle and high end of the Spectrum. As per the Payments Hub pattern, the construct has visibility of all payments and so Business Services across the entire Spectrum are applicable in this pattern (Figure 15).

Scenarios

As with the Payments Hub pattern, the PSB also provides a generalised solution to payments integration across the bank, solving the problems of front- and back-end integration. This pattern does not require the replacement of legacy Payment Engines; rather it recognises and complements the scenario of multiple Payment Engines.

The pattern is therefore useful in banks where there exist multiple, diverse, deployed instances of Payment Engines.

Given the knowledge capital invested in the existing Payment Engines, likely complexity of legacy implementations (implying huge effort to replicate and replace) this pattern supports a strategy of their continued use. Thus, the PSB solution allows for their retention and focuses on addressing the integration problem between the Payment Engines and the other payment solution components. In comparison with the Payments Hub pattern it therefore avoids significant effort in Payment Engine replacement.

This solution pattern was selected by a large UK retail bank as a solution to the payment integration problem. The bank was a full member of all UK and European schemes with dedicated Payment Engines and having several major Product Systems. The bank took a view, as highlighted above, that consolidation of its diverse range of Payment Engines into a single solution across the bank was a massive undertaking, not feasible in a short to medium time frame. Rather a strategy of leveraging existing Payment Engines, and a policy of gradual, targeted, replacement was employed. The Payments Service Bus pattern was identified to support this strategy. Technology selection and implementation is ongoing within the bank.

Summary of characteristics

Payments Service Bus

Attribute	Value
Integration Capability	Yes
Business Capability	Delegated
Process Capability	No
State Management	Stateless
Connection Style	Many passes

TARGET END STATE SELECTION

Considerations

The target end state represents the ideal long-term vision for payments systems processing.

Recall the dual objectives of effective and elegant system processing. These objectives will be met to a large extent by reducing the integration complexity. All the patterns described do provide a reduction in this complexity and therefore support these objectives to some degree. This now poses an interesting question: of the patterns identified, is there a single pattern that represents the preferred end state for payments processing, or are alternative target end states viable?

It is evident that to achieve integration simplification, and target end state must address the holistic integration problem covering both front-end and back-end integration.

Hayden *et al.*¹¹ present the Consolidated Transaction Hub as the single preferred end state that achieves this. In terms of the patterns described here, this aligns most closely with the Payments Hub pattern. In their context, both the Front End Payments Hub and Back End Payments Hub patterns are useful only as introduction strategies for achieving this end state.

However, the pattern analysis presented here reveals that there are in fact two key

patterns that address the holistic payments integration problem, covering both front- and back-end integration. These are the *Payments Hub* and the *Payments Service Bus*. Therefore, *both* these represent viable candidate end states.

To help assess the suitability of these two patterns as the preferred target end state and achieve a comparison, their effectiveness in reducing integration complexity is assessed using the Integration Space Complexity Table defined in Table 1.

End state suitability assessment

For the purposes of comparison, a specific Integration Space Complexity Table is illustrated using the following parameters, typifying a medium retail bank with full UK scheme membership and a multi-channel delivery strategy. In this context, it is assumed the bank offers the following channels:

- (i) branch;
- (ii) internet banking;
- (iii) telephone banking;
- (iv) postal services with associated back office systems;
- (v) a business partner channel via a dedicated system.

Six schemes are assumed, equating to country-specific schemes. For the UK, these are BACS, Clearings, CHAPS, Faster Payments, cards, and also International payments. The example assumes that a shared Payment Gateway is used to support two schemes (eg CHAPS and International payments, both processing SWIFT messages). The numbers are therefore arbitrary but relate to realistic circumstances.

Schemes	6
Consumer Channels:	5
Payment Gateways:	5
Product Systems:	3
Payment Engines:	6

For simplicity, it is assumed that all Channels:

- support payments types for all schemes; and
- a connection to all Product Systems is required for account updates.

In practice all payment types may not be offered from all Channels and some account updates would be controlled by the Payment Engine. Such factors would lower the integration complexity in reality. This derives a specific table below (Table 3).

Note that the Integration Space does not account for any integration to components providing Business Services. These are excluded on the basis that:

- There is duplication of functions specific to each transaction processing scenario bundled into the application components specific to the scenario. Hence, there are no connections to external services and hence no integration problem *per se*.
- It is unlikely that many common services exist pre-end state, and hence there are no many-to-many connections to resolve.
- If there are some shared services, then connection point is difficult to generalise in terms of the model presented.

The net effect of excluding these is recognition that the overall complexity is increased *in all cases*, so excluding the

Business Services does not detract from the analysis given.

The original Payments Integration Space relating to this example is shown graphically in Figure 16a.

For the Payments Integration Space, an overall complexity coefficient is introduced, defined as the total sum of the integrations. In the example defined:

Example Integration Space Coefficient: 175

The effect on the reduction in the Integration Space complexity is now illustrated for the two candidate end-states patterns. By applying them to the example problem space, the connectivity is re-evaluated to give the revised Integration Space.

Payments Hub Pattern Coefficient: 21

Payments Service Bus Pattern Coefficient: 33

The Integration Space for each of the patterns is shown visually in Figures 16b and 16c.

Summary

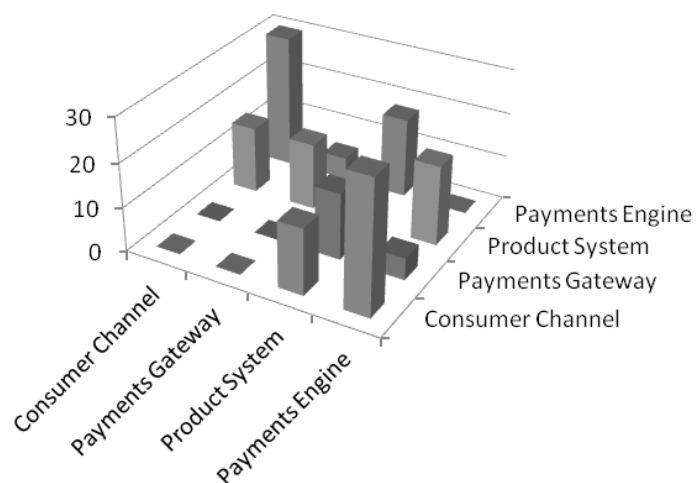
Both Payments Hub and Payments Service Bus are seen to provide a significant reduction in Integration Space complexity. However, they represent quite different solutions to the payments integration problem. The decision on choice of pattern depends on several factors:

- the specific circumstances of the bank, accounting for the uniqueness of each

Table 3: Integration Space Complexity Table

	<i>Consumer Channel</i>	<i>Payment Gateway</i>	<i>Product System</i>	<i>Payment Engine</i>
Consumer Channel	–	–	15	30
Payment Gateway	–	–	15	5
Product System	15	15	9	18
Payment Engine	30	5	18	–

16a Original



System 1	System 2	Connections
Consumer Channel	Payments Gateway	5
Consumer Channel	Payments Hub	2
Consumer Channel	Product System	0
Consumer Channel	Payments Engine	0
Payments Gateway	Payments Hub	15
Payments Gateway	Product System	10
Payments Gateway	Payments Engine	5
Payments Hub	Payments Engine	10
Product System	Payments Engine	5

Layer	Payments Engine	Product System	Payments Integration Bus	Payments Gateway	Consumer Channel
Consumer Channel	1	1	1	1	1
Payments Gateway	1	1	1	1	1
Payments Integration Bus	1	1	1	1	1
Product System	1	1	1	1	1
Payments Engine	1	1	1	1	1

Table 4: Summary of advantages/disadvantages of Patterns as a Target End-State

	<i>Advantages</i>	<i>Disadvantages</i>
Payments Service Bus	• Complements existing legacy Payment Engines; (does not require replacement of them). • Supports multiple technologies for Payment Engine implementations • Enables 'out of the box' Payment Engine functionality to be leveraged from vendors products • Pure integration rather than replacement design and build effort	• Residual integration space remains more complex than that of the Payments Hub, albeit marginal • Difficult in practice to achieve pure integration capability as some business decisioning (eg intelligent payments routing) must take place outside of the Payment Engines • Resulting IT estate is more complex with multiple product vendors relating to each Payment Engine
Payments Hub	• Least complex integration space, albeit marginally • Suits legacy replacement • Single Architectural Building Block for all payments scheme processing, simplifying IT estate and supplier management • Single technology implementation and simplified systems management • Is an enabling architecture for custom processing supporting uniqueness of payments processing in each bank	• More development effort to reach end-state than the Bus as it requires replacement and consolidation of each legacy Payment Engine. • Single technology choice may not be optimal for all scheme processing

- bank and the general approach to payments processing within the business;
- the scale of the bank's payments processing requirements;
 - the complexity of the existing IT architecture for payments processing, comprising the heritage Payment Engines and Product Systems.

A summary of the advantages and disadvantages of each of the two candidate patterns is presented in Table 4.

CONCLUSION

Payments integration complexity is a major barrier to enabling the business agility of a bank. When the integration problem is addressed, desirable architectural goals are achieved, which can improve business agility:

- separation of architectural concerns, with components optimised for one purpose;
- elimination of the tight coupling of systems;
- provision of the foundation for shared payment services to be introduced.

A number of patterns for payment systems integration have been presented and their processing characteristics defined. The patterns are shown to relate to specific apportionments of the Payments Processing Spectrum, which itself provides a useful visual tool for comparing them.

IT scenarios to which the patterns are relevant have also been highlighted and discussed.

Certain patterns are seen to address the front-end (scheme and channel) integration problem, while others address the back-end (Product System) integration

problem. The relevance of employing specific payment Business Services in each of the pattern contexts has been demonstrated.

Of the six patterns identified, only Payments Hub and Payments Service Bus patterns offer a holistic solution to this payments integration problem, addressing both the front-end and back-end integration problems. For this reason, the Payments Hub and Payments Service Bus patterns are presented as prime candidates for the target end state for payment systems processing. The architectural difference between a Payments Hub solution and a Payments Service Bus solution has been qualified using the respective patterns. Further, the impact of the patterns on the overall integration complexity has been quantified in terms of the reduction in Integration Space complexity.

There are considered to be significant implications in choosing one or other of the end state patterns. In particular, the technology selection for the respective integration constructs is influenced heavily by the characteristics identified in the patterns. Depending on choice of end-state pattern, long-term commitment to retain or decommission specific legacy technologies may be inferred, and alignment to overall IT strategy must be assessed and validated. Consideration must also be given to how development of a payments processing system is undertaken within the bank; how the development teams are structured; and the long-term skills required.

In conclusion, given the significant implications of pattern selection, a bank must take an early and overt decision as to the desired target end state for payments processing. Once this decision has been reached, business events on the planning horizon can be assessed in terms of their suitability as triggers for payments process-

ing modernisation. Such planning activity is anything but trivial and must account for many factors, not least the demanding time constraints imposed by regulatory deadlines, which tend to dictate non-strategic solutions. An iterative and incremental transition strategy can help accommodate such factors and a roadmap should be determined on this basis.

ACKNOWLEDGMENTS

Thanks are due to Greg Brougham for his extensive review comments and suggestions for improvements to the architecture model.

REFERENCES

- (1) The Open Group Architecture Framework (2009) TOGAF 9, ISBN 978-90-8753-230-7, available at: <http://www.opengroup.org/togaf> (accessed 2nd March, 2012).
- (2) Farrow, G. (2011) 'The Payments Hub spectrum; A model for the design of Payments Hubs', *Journal of Payments Strategy & Systems*, Vol. 5, No. 1, pp. 52-72.
- (3) TOGAF 9, ref. 1 above.
- (4) Farrow, ref. 2 above.
- (5) *Ibid.*
- (6) *Ibid.*
- (7) Hayden, R. Akash, L, Ledford, S. and Nunn, C. (2010) 'Payments Hubs: Redefining the industry's infrastructure', *McKinsey Quarterly*, June, available at: http://www.mckinsey.com/clientservice/Financial_Services/Knowledge_Highlights/Recent_Reports/Payments.aspx (accessed 16th February, 2012).
- (8) *Ibid.*
- (9) *Ibid.*
- (10) Farrow, G. (2011) 'The Canonical Data Zone: Issues in data selection for Service Oriented Architectures', The Open Group Conference, 9th-13th May, Westminster, London.
- (11) Hayden *et al.*, ref. 7 above.