Clear2Pay

Meshing Payments and Technology

A White Paper

Open Payment Framework

Clarity in Payments

Table of Contents

Executive Summary

We are in the midst of a period of hectic realignment of payment infrastructures; regulatory change, Eurozone harmonisation, reduced revenues, expansive territorial strategies and, critically, technological advancement. Payment systems are moving away from the monolithic silos, deemed unresponsive, closed and expensive to maintain. A new generation of component based solutions are making the best use of incumbent systems by coupling them into open infrastructures allowing full and close integration of disparate services into a cohesive whole.

The one constant, as ever, is change; whether this originates from regulatory change, technology fashion, bank strategy realignment or customer expectations. The bottom line is that business managers need more agility and adaptability from their underlying infrastructure if they are to maintain or grow their market position. Open component based frameworks, coupled technology developments like Service-Oriented Architecture (SOA) that enable much greater customisation and adaptability of solutions, reduce implementation risks and increase system responsiveness to changing business needs.

A Technological Shift

The payments market is closely monitoring significant technical developments such as the Service-Oriented Architecture standards and their underlying Web Services technology. This development is particularly relevant for large scale enterprises offering a cooperative framework for building large scale IT systems. Service-Oriented Architecture based infrastructure promises an alternative to traditional buy vs. build decisions offering a middle way. Maximising independence, reusability and interoperability, business applications are constructed from services which can be configured to meet changing requirements.

Business Driven Solutions

Embracing the Service-Oriented Architecture and the open platforms has many potential benefits. Existing investment is protected by exposing them to the payments framework as web services. Such re-use of existing systems is augmented by re-use of service resources across the enterprise. Open platforms, by their nature, offer unrivalled service interoperability and scalability through loosely coupled, asynchronous architectures, allowing for easier growth of clients and service scope. Most importantly, open frameworks allow the speedy publication of new services and products without lengthy and costly implementation cycles.

The Response

Clear2Pay offers world class solutions based on the Open Payment Framework (OPF). The Open Payment Framework (OPF) from Clear2Pay is a library of component building blocks from which payments solutions can be derived. The Open Payment Framework is built entirely on a Service-Oriented Architecture (SOA) delivering common, reusable services consisting of a comprehensive data model, choreographed payment business processes and configurable services including parsing, validation, cost based routing, warehousing security, auditing and many more.

Clear2Pay's Core Open Payment Framework ships with a comprehensive Software Development Kit (SDK). The SDK changes the paradigm of a "buy versus build" decision to a "buy and build". Through documented APIs, customisation patterns and a suite of reusable frameworks, the SDK offers our customers the ability to add, change and round out components to meet their unique requirements. Time to market is an essential ingredient of maintaining a competitive edge, so whether it is with a 3rd party Systems Integrator, the Bank's own IT department or Clear2Pay, the SDK provides the Bank with all the necessary tools to implement and round-out a complete payments solution.

Business-Led Payment Strategy

While the much vaunted dream of IT systems created by business people to solve business problems without a technical expert in sight may be too far fetched, this is certainly the direction the technology is leading us. Agility is the strategy maxim whereby financial institutions can adapt to changing payment market circumstances. Can technology underpin this? The future promises IT systems created by business people meeting their specialist needs without prohibitive development delays and costs.

This White Paper aims to review current best practice in payments and the recent technology shifts that aim to deliver true generic, interoperable and cooperative payment solutions.

Next Generation Payment Frameworks

The single most important technical development of recent times is the imminent adoption of the Service-Oriented Architecture (SOA) standards and their underlying Web Services technology. This way of working defines a cooperative framework for building large enterprise IT systems. Business managers can create the IT functionality they need to support their business initiatives without long IT department backlogs for development.

To reiterate, assume the business climate changes and the business processes need to adapt accordingly, the business manager can make the necessary changes to the IT system online at minimal expense and without protracted IT development project. The implications are potentially enormous. As if this was not enough, SOA and related web services forces IT departments to speak business language rather than techno-speak and further forcing technology into step with the business strategy.

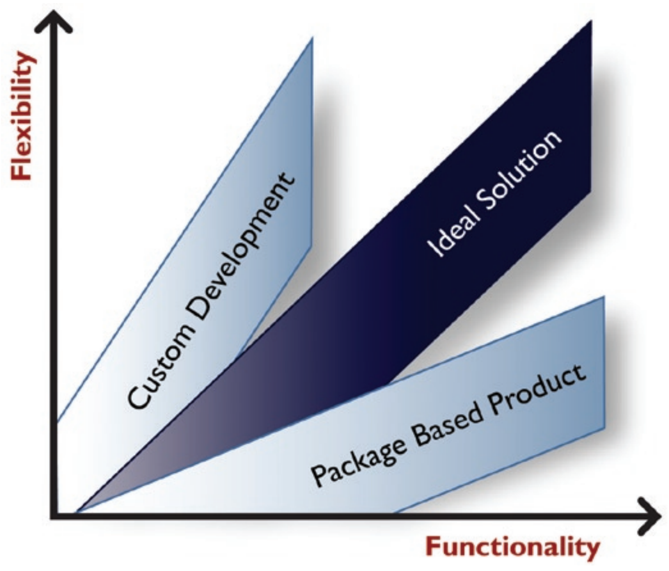
Current State of Play

Financial institutions typically build and maintain their own custom applications to service their niche requirements. If banks do not build their own systems, they are forced to modify their procedures to match a packaged solution’s capability. Either way it is a costly exercise and wrestles control of the process away from those that know it best.

Often, technical capability is duplicated across numerous departments and lines of business and any need for these diverse IT systems to inter-communicate results in a major project with significant cost and time delay.
This working paradigm has automated much of the financial business processes but has inevitably become operationally unworkable and intellectually limiting.

The Third Way

The overused term “legacy system” was coined to describe out-dated, monolithic, custom made systems that have grown difficult to maintain and yet harder to replace. Packaged applications bring their own issues. Modifying a package to meet specific business and locale needs is expensive, risk heavy and typically results in inadequate installations that leave the financial institution impeded for the long-term.
SOA promises an alternative. Under SOA, business applications are constructed from independent, reusable, interoperable services that can be reconfigured with minimal technical know-how and effort. The reality is that business managers will routinely assemble technology services from reusable components developed by their IT department or from freely available niche services available on the market. It creates a buy-and-build-and-partner model.



Packaged products, while not providing extensive flexibility have the advantage of upgrades and maintenance as part of a normal 3rd party product life-cycle. Custom development, however, offers all the flexibility but without the safety net relying on bespoke maintenance at higher cost and risk. Combining the 2 offering schemes into a component implementation offers extensive customisation with the reassurance of a packaged product offering.

Componentised Business Process Management

Changes in the processes flows have a limited impact on components/systems that execute the process functions. Therefore, it becomes easier to support multiple processes (e.g. per legal entity, product and/or customer). This fits nicely together with other components and/or services of the bank promoting many benefits:

- Re-use of existing components/applications v
  - Reduction of decommissioning
  - Acceleration of implementation time-lines
- Strong emphasis on integration by using industry standards (WSDL/SOAP, BPEL, J2EE, etc.).
- Makes gradual migration easier v
  - Acceleration of implementation time-lines (time-to-market)
  - Reduction of (big-bang) migration risks
  - Increased ROI by bringing quick wins forward in time

The march towards more flexibility through out-sourcing (and therefore, in-sourcing for some bank business models) continues. Such capability enhances such future strategic alignment and offers better monitoring and control of processes.

The outcome is systems that are highly scalable and fault-tolerant through extensive business process management technology and SOA; applications particularly suited for adaptation and customisation without impact on the core product.

An Enabling Technology

What are Service-Oriented Architectures (SOA) and web services? If the hype is to be believed, the march towards this new horizon is unstoppable and the benefits to enterprise agility are boundless. Below we examine the technology in more detail and discuss how combining SOA developments with modern data management techniques offer incremental added value through adaptive services and data.

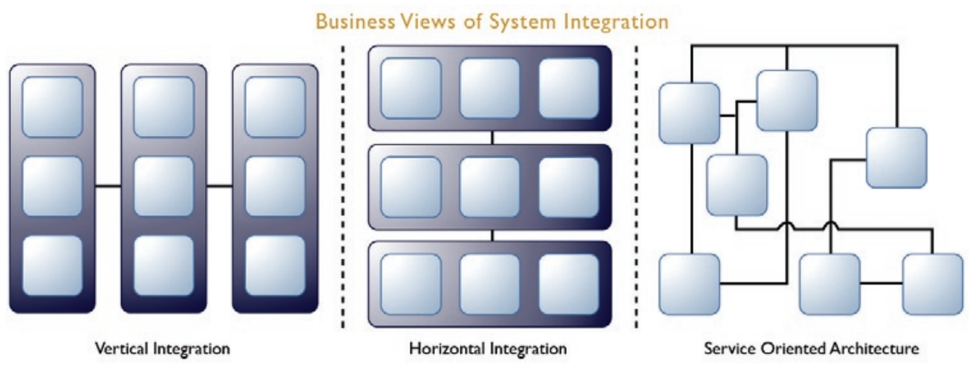
Service-Oriented Architecture and Web Services

Service-Oriented Architecture and web services express a business-driven approach to software architecture that supports integrating the business as a set of linked, repeatable business tasks, or "services". Services are self-contained, reusable software components with well-defined interfaces and are independent from the consuming application; composite SOA applications can invoke services that may, and often will, run on an entirely different infrastructure.

SOA helps build composite applications, which are applications that draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes. SOA helps businesses innovate by ensuring that IT systems can adapt quickly, easily and economically to support rapidly changing business needs. SOA is largely based on a set of web services standards (e.g. SOAP) that have gained broad acceptance over the past several years. These standards have resulted in greater interoperability and avoidance of vendor lock-in. However, one can implement SOA using any service-based technology.

There are three distinct philosophies for each step that business has taken to adapt to requirements integration:

- **Vertical silos of integration** – keeping all applications and systems with similar functionality integrated with each other, but not accounting for applications that may wish to use their core functionality in the future.
- **Horizontal integration** – integration of some but not all similar functionality across vertical systems; for example, using common payments, routing, verification, clearing, etc.
- **The Service-Oriented Architecture** – an environment of ubiquitous service providers and service consumers interoperating with each other in a secure and consistent manner.


Business Views of System Integration
Vertical Integration
Horizontal Integration
Service Oriented Architecture

So, how does SOA compare with traditional architectures? SOA is not contrary to object-oriented (OO) architectures; rather, SOA can be viewed as "macro" OO because both models are based on the same key principles. OO introduced the concept of encapsulation - hiding the implementation details of an object behind a well-defined interface. However, in OO, what an object does (its functionality or behaviour) is intrinsically tied to the data itself. Over time, professional developers recognised the limitations of this approach and envisioned an alternative in which behaviour could be duplicated and evolved outside of data over time.

The Loose Coupling of Services

SOA is a style of enterprise architecture that enables the creation of applications that are built by combining loosely coupled and interoperable services. These services interoperate based on a formal definition (or contract) which is independent from the underlying platform and programming language. The interface definition encapsulates (hides) the language-specific service implementation. A SOA is independent of development technology (such as Java, .NET etc) and is therefore vendor independent. The software components become very reusable because the interface is standards-compliant (e.g. WSDL) and is independent from the underlying implementation. So, for example, a C# (C Sharp) service could be used by a service consumer developed in Java.
SOA can support integration and consolidation activities within complex enterprise systems, but SOA does not specify or provide a methodology or framework for documenting capabilities or services.

Collaborative Services

Process Definition languages such as BPEL (Business Process Execution Language) and specifications such as WS-Coordination extend the service concept further by providing a method of defining and supporting orchestration of fine grained services into coarser grained business services. This in turn can be incorporated into workflows and business processes implemented in composite applications or portals without any significant impact on performance.

So, SOA is a collection of web services brought together to accomplish business tasks. These tasks may be to check a balance, route a payment based on criteria, check for fraud, etc. In the same way that the early 3rd generation programming languages of the 1980s (Pascal, C, etc.) greatly simplified and sped up IT developments; and object-oriented techniques (Java, COM, etc.) improved and opened up IT development in the 1990s, SOA advocates claim its potential is much greater.

The unprecedented flexibility of SOA is achieved because the services are accessed through standards. Business processes can be altered quickly and software applications can be easily integrated without the need for a lengthy, traditional IT project.

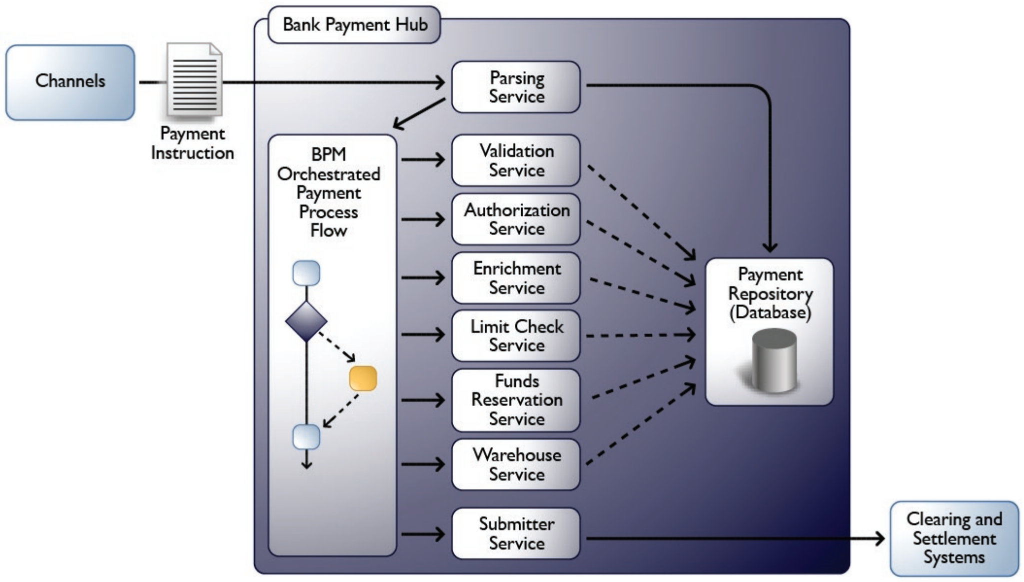
Collaboration is the watchword. Service-Oriented Architectures can interact seamlessly with systems and services outside the organisation. Customers and suppliers are now very much part of the online bigger picture. Add to this the fact that because these composite applications are much smaller and simpler than traditional hardwired applications, they are simpler and much cheaper to maintain. Building such "composite applications" in SOA can be done in languages such as Java or more appropriately, in BPEL with its externalised logic capabilities.

Composite Application Freedom

An underlying principle of SOA is the definition of the business process flow and the process model itself. The process flow defines the order in which the processes are carried out and details the conditions under which they operate. Creating composite applications with BPEL combines the legacy service wrapped in a web service layer with new systems to offer new capabilities.

Each element within the process flows can be manipulated and refined with no impact on the rest of the system. For example, the submitter comes as a selection of predefined options. These can be further customised to meet specific needs and rules without impacting other flows.

Each element within the processes could be external calls to 3rd party elements or established internal applications such as CRM or other back office services. This allows for maximising the return on investment on legacy solutions and also reducing risk by utilising established and proven internal infrastructure.



Asynchronous Flow Services

Such is the sophistication of SOA that the flows can be completely asynchronous. This implies that the process steps and decision points do not necessarily need to be processed serially but can be stored awaiting further actions, maybe manual correction. Once the conditions are satisfied, the process flow can begin again.

The area of customer interaction, whether bank administration, 3rd party or true customer, benefits greatly from this flexibility. The process workflow manages the entire process but asynchronous flows permits modification and pausing within the process depending on the entity and its state. A payment may be ‘parked’ or ‘hydrated’ awaiting a manual intervention. Once continuation criteria are met (the payment is corrected), the payment is ‘dehydrated’ and it continues to the completion of the process flow.

All this is related to the workflow; one can add user defined tasks and include human intervention (repair) as needed.

Business Strategy via Open Frameworks

The payments business is championed by the business strategy. Over the past decade, IT automation has made great strides in efficiency, process excellence and most importantly, in revenue. The watchwords of today focus more on business agility, adaptability and flexibility: the landscape changes rapidly, the winners are those that can adapt to the change the best. While leading IT solutions can facilitate such agility, without a strategic and tactical orientation from the business strategists, the best IT systems just push files around.

To put it another way, as British chess champion Gerald Abrahams once stated, "**Good positions don't win games: good moves do.**" It is the long-game that matters; a solid position today is no guarantee of a secure situation in the future. Business goals need to be reflected at the micro as well as the macro level. Powerful moves to make a small refinement to a payment product or clearing route are as important as overall payment strategy. IT systems that can deliver such suppleness through configuration and dynamic customisation will win in the short and long term.

What do banks really need from IT solutions to ensure the strategy can be met? The primary goal is the removal of the high cost of introducing and integrating new IT systems and functionality. Big-bang implementations are now, by mutual consent, consigned to history, with the exception of green-field sites or where the incumbent solution is totally out-dated. Paced or trickle releases are more appropriate to ensure a smooth and managed transition gaining the benefits of the established solutions augmented by the value-adding new. The underlying goal is to have an IT infrastructure that truly supports the rapid exploitation of business opportunities, timely response to market environments, or the ability to better comply to regulatory mandates. If this can be achieved while better leveraging prior investment in IT assets, then so much the better. And, if new solutions can be implemented which are, from conception, free from any lock-in to proprietary and vendor specific systems and technologies, then solution openness will deliver the necessary freedom.

The Impetus behind Open Frameworks

Embracing the Service-Oriented Architecture (SOA) and the open platforms and architectures that it underpins offers six primary returns:

- **Investment protection**
  - Exposing existing legacy application code as web services re-uses existing applications
- **Reusability**
  - Publication of services through service registries allows resource sharing across the enterprise
- **Interoperability**
  - Regardless of platform and vendor through a maturing set of Web Service Interoperability (WS-I) standards, etc.
- **Scalability**
  - Replaces tightly coupled application-to-service environments with loosely coupled, asynchronous, course grained architectures, allowing for easier growth of clients and services
- **Flexibility**
  - New business services easily added and extended
- **Cost efficiency**
  - Customised solutions are expensive to build and maintain

Integration through Cohesion

The Service-Oriented Architecture (SOA) model, leveraging the power of web services and service orientated development, has captured the attention of organisations worldwide with its promise of interoperability between legacy systems and new applications. As a result, IT decision-makers are turning to this promising technology to enable their business applications to adapt to customer needs and market changes over time. While agility as the primary benefit of SOA is laudable, measuring agility and putting a cost to it which can be justified is another matter entirely.

The SOA project lifecycle also differs from that of a traditional application. In contrast to a traditional application, building a service-oriented application requires considerably more discipline and upfront focus on the application design. Skilled architects must think through the best approach to solving the business problem, the most beneficial way to partition functionality into services, and the ideal coupling of these services into a working application. This increased focus on the design phase of the project means that it will likely take more time and resources to deliver the first version of a service oriented application. However, the rapid integration and reusability benefits gained by making this initial investment mean that future enhancements to the SOA application can be delivered faster and with less risk and cost.

SOA certainly promises to be the future of IT services by enabling the integration of virtually all IT resources, including isolated data "silos" and previously incompatible legacy, .NET and Java technology applications.

A decade of deregulation has changed the financial services landscape, fuelling globalisation and consolidation, and intensifying competition. Legislation introduced over the past three years has forced regulatory compliance to the top of the agenda for most CIOs. They are also under pressure to manage IT in a way that can help them regain their customers' trust in the wake of the accounting scandals. Cost pressures and decreasing margins are forcing businesses to find new ways to better leverage IT assets, and do more with less.

A Service-Oriented Architecture, when implemented effectively, can enable financial services companies to better leverage and integrate IT assets, while increasing flexibility to respond to business change and opportunity.

Benefit Analysis

The much vaunted SOA benefit of agility is not, however, the end of the story. SOA promises wide ranging advantages in the cost of ownership, increased revenue, time to market and risk reduction.

- Cost of Ownership
  - Easier to outsource and offshore developments
  - Facilitate business process improvement
  - Opportunity to retire and/or consolidate redundant systems
  - Increasingly real-time operations
  - Increased STP opportunity reducing manual intervention
  - Service based reuse of applications across enterprise
  - Greater value from IT resources
  - Shared services creates economies of scale
  - Standardisation of infrastructure platform
  - Reduced support, testing, integration & acquisition costs

- Increased Revenue
    - Improved business design
    - Productising of business services
    - Service based business platform enables collaborative business models
- Time to Market
    - Creation of standalone capabilities
    - Separation of application, process and data layers increases business agility
    - High levels of reuse increases productivity
    - Predictable impact from known dependencies simplifies response to market behaviour
    - Faster response to new channel requirements
    - Fast heterogeneous linking of services speeds up creation
- Reduced Risk
    - Loose coupling and modularity implies higher responsiveness to change
    - High predictability supports strategic business planning
    - Shared services allows regulatory compliance
    - Reuse of standard, trusted services and components
    - Reduced reliance on specific technology

Unifying Architecture

SOA allows businesses to better leverage their investments in technology by promoting wrap and reuse of their existing IT assets. It provides the framework and infrastructure to expose business functionality as loosely coupled, reusable services, thus enabling significant cost savings and faster returns on investment.
Through the use of business process management, SOA makes business applications more agile and improves the ability of the business to respond to changing customer demands and emerging competitive forces.

Interoperability and Vendor Adoption

Standards are what set SOA apart from previous generations of (normally proprietary) integration technology. The SOA standards have been defined for many years but it is only now that they have matured. Encouragingly, virtually every software tool or application vendor, across all markets, is supporting the adoption of the technology. The largest forces in packaged software are employing SOA to deal with their own complexity problems. Their products are getting big and cumbersome and need to be streamlined into manageable component building blocks.

From a vendor perspective, it is advantageous to be able to easily integrate with another vendors' software as this implies an increased functionality proposition of the combined solution.

Legacy Crusader

SOA offers the first real solution to the perennial problem of incumbent legacy systems. Such systems sit comfortably and have an extended lifespan within a SOA. Replacement is positively discouraged. SOA environments are comfortable linking modern web services to mainframes through the use of SOA wrappers or enablers. In fact, wrapping SOA around selected mainframe provisions, such as fraud detection, increases overall reliability and time-to-market.

Equaliser

Possibly the most compelling impact of SOA is how it may alter organisational and governance structures. Typically, IT managers are linked to specific applications and the business unit they support. Because SOA enables solutions that transcend lines of business, IT managers can decouple from the applications they manage and bask in the broader view of the potential they can deliver.

The hope is that IT will design services that enable companies to bring distinctive capabilities, products and services to market quickly.

Business Self-Determination

SOA's new paradigm ensures that the business drives how its IT works, rather than vice versa, leaving it free to focus on its areas of core competence and achieve a higher level of IT-enabled business agility. Furthermore, SOA champions the release of "trapped" ROI from previous IT projects, by better leverage and reuse of existing IT functionality.

In general, SOA offers much greater freedom of choice on procurement of new IT platforms and products: build new solutions by mixing component services made up from legacy applications, various vendor products and some bespoke development resulting in better reuse and leverage of new investment. In conclusion, the new technology offers the improved ability to monitor and audit business processes, and comply with regulations.

The Business Activity Monitor (BAM) is part of the BPEL specifications and enables the tracking of the statuses within composite applications. Applications can be interrogated to uncover the number of payments processed or pending and their detailed statuses.

With the logic externalised, the composite applications can be developed in BPEL. BPEL allows non-technical staff to draw the preferred flow for a particular process and define its path and exceptions without resorting to coding through the use of configuration variables.

Buy versus Build – Migration Strategy

Adopting a Service-Oriented Architecture and organising legacy, new development and 3rd party solutions into an integrated array of web services is changing the payments landscape. The opportunity to slowly migrate towards replacement, component by component, rather than a big-bang implementation is appealing.

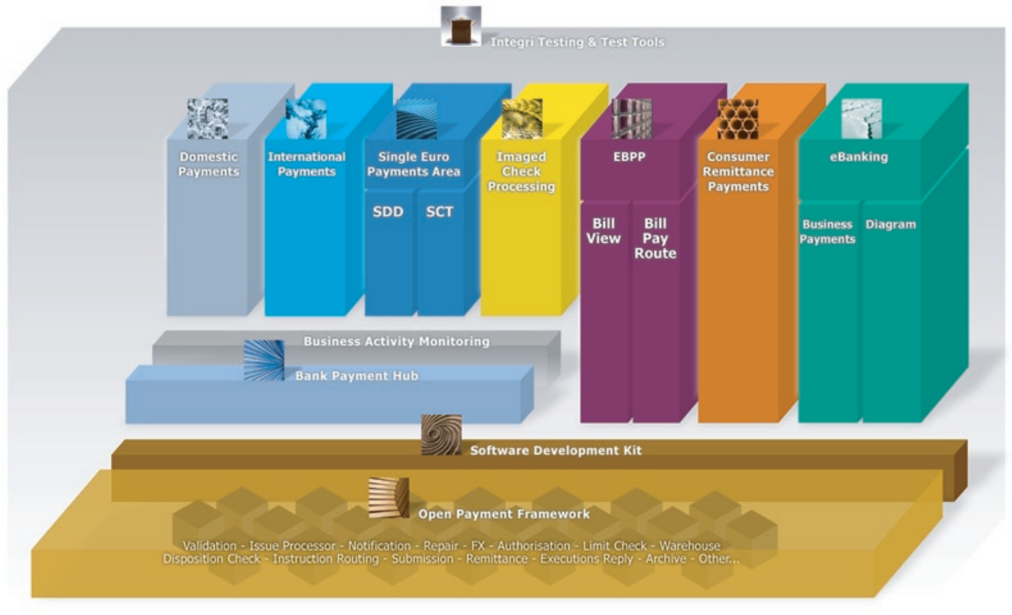
Additionally, once operational, the component blocks of functionality can be easily changed and amended to meet changing needs without any development projects; simple configuration changes can fundamentally alter operation processes. The web service manipulation is further extended by concepts such as Adaptive-SOA (see below) which opens up the data for enhancements, again without any development.

It is no longer about buy, building or partnering, it is a matter of combining all three into the optimal solution taking maximum benefit from established solutions, enhanced by the best of market offerings for niche capability.

The Response

Clear2Pay offers world class solutions based on the Open Payment Framework (OPF), a library of component building blocks from which payments solutions can be derived and is built entirely on a Service-Oriented Architecture (SOA). The Open Payment Framework ships with a comprehensive Software Development Kit (SDK) providing documented APIs, customisation patterns and a suite of reusable frameworks.

From the Open Payment Framework, Clear2Pay has created pre-defined solutions around the Bank Payment Hub

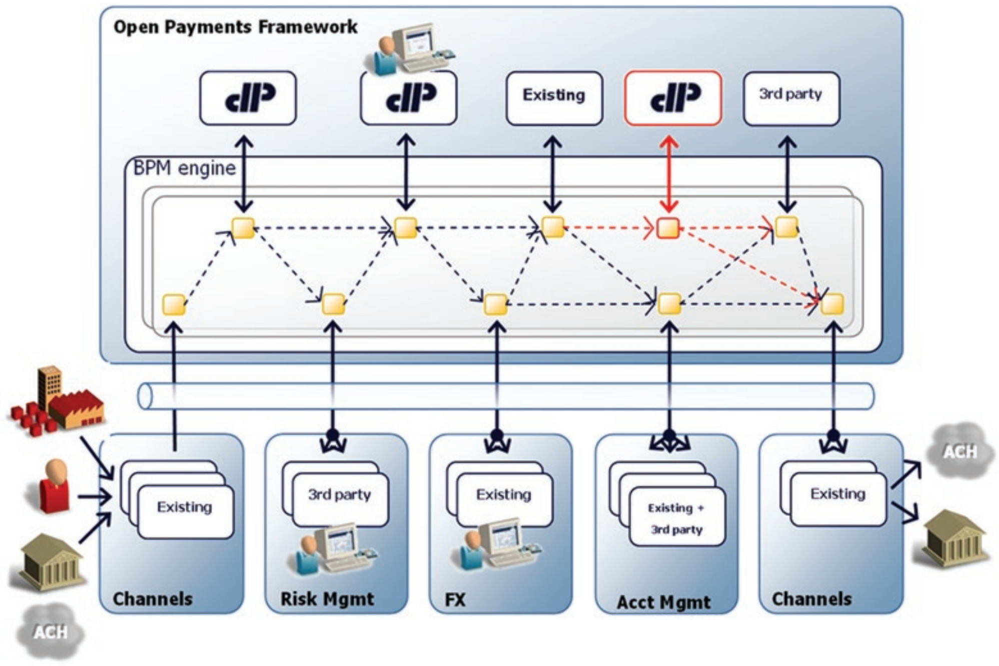

Integri Testing & Test Tools
Domestic Payments
International Payments
Single Euro Payments Area
SDD
SCT
Imaged Check Processing
EBPP
Bill View
Bill Pay Route
Consumer Remittance Payments
eBanking
Business Payments
Diagram
Business Activity Monitoring
Bank Payment Hub
Software Development Kit
Open Payment Framework
Validation - Issue Processor - Notification - Repair - FX - Authorisation - Limit Check - Warehouse
Disposition Check - Instruction Routing - Submission - Remittance - Executions Reply - Archive - Other...

including Domestic Payments, International Payments, SEPA, Remittance, Imaged Check Processing and EBPP, as well as eBanking for retail, small business and corporate payments.

The Open Payment Framework

The Open Payment Framework is built entirely on a Service-Oriented Architecture (SOA) delivering common, reusable services consisting of a comprehensive data model, choreographed payment business processes and configurable services including parsing, validation, cost based routing, warehousing security, auditing and many more.

The OPF increases the payment transparency and visibility across the whole financial supply chain. Capable of operating in mature environments and making the most appropriate use of incumbent internal and 3rd party solutions, the Open Payment Framework maximises the synergy between diverse infrastructures.



Adaptive-SOA – a Grey Box Solution

SOA provides a framework on which applications and services can be shared and extensively reused. But this is not the whole story. Many computer solutions are a black box with data going in at one end and coming out of the other after some due processing. Adaptive-SOA is a concept whereby SOA systems can open their data model to real-time adaptation.

Classic packaged solutions offer little freedom to customise and extend services as opposed to bespoke custom built solutions. **Adaptive-SOA** offers predefined functionality like a package, but this can be easily extended and changed on demand. This represents all the advantages of both a package product (quick implementation, regular upgrades, etc.) and a custom build (flexible and extensible functionality) – but without the disadvantages of either.

The OPF not only provides a set of callable black-box business services: each individual service can be fully customised and extended. This "Adaptive-SOA" approach offers the best of both worlds; the payment process flow can be fine-tuned by invoking a rich set of business services and each service can be adapted to the specific needs of the bank. The OPF allows the adaptation of the underlying data model; the Adaptive-SOA grey box approach enables the customisation teams to incorporate these extensions in the individual services. Meta-data describing the underlying data model is used by the Adaptive-SOA services and can be fully extended.

Adaptive-SOA Services

Rapid re-configuration of IT services is a new concept. Adaptive-SOA takes the SOA concept a step further whereby business managers can change the way a service operates with a greatly reduced involvement from IT department and therefore faster to market. Altering the process steps or conditions, or even adding a new process altogether maximises system agility.

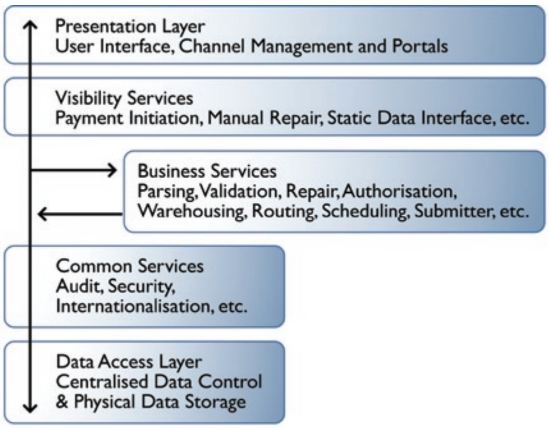
The ability to modify the services and flows quickly is a powerful tool to business. Adaptive-SOA greatly eases the addition of new validation rules to counter legislative changes, extending fraud detection on specific conditions, etc. through intuitive graphic interfaces and configuration parameterisation. This implies that, for example, routing or submitter rules and formats can be modified through the SDK extension points. New back-office systems or services can be added dynamically and new routing destinations, and their applicable rules and formats, are now just configuration elements. These changes will not impact the overall business flow of operations but enable sophisticated fine tuning of the solution with minimal risk, impact and cost.

Adaptive-SOA Data Models

Adaptive-SOA allows major dynamic adaptations in the data model, irrespective of the potential multitude of databases underpinning the solution. In simple terms, this permits new attributes to be added to existing domain objects; e.g. add a new customer field of 'SocialSecurityNumber' to the customer database. An all this is done via configuration parameters.

A Layered Approach

The Open Payment Framework is built on the principle of independent layers and logically grouped services. The visibility services manage customer interactions calling on the business services to move instructions through their life-cycle. Access to the Data Access Layer is totally independent from these services. All services are defined with a contract defining what they can do, what data they can access and what other services they can call. Clear2Pay have defined a standard implementation with pre-built validation rules, routing options, notification properties, etc. which can be customised as required.


Presentation Layer
User Interface, Channel Management and Portals
Visibility Services
Payment Initiation, Manual Repair, Static Data Interface, etc.
Business Services
Parsing, Validation, Repair, Authorisation, Warehousing, Routing, Scheduling, Submitter, etc.
Common Services
Audit, Security, Internationalisation, etc.
Data Access Layer
Centralised Data Control & Physical Data Storage

Data Access Layer

The Data Access Layer (DAL) is a highly performant extensible persistency layer that provides a common set of persistence services for accessing and manipulating elements of payment system domain models. The DAL maintains concepts surrounding static data (banks, customers, agreements, etc.), payment data (interchanges, instructions, transactions, etc.), permissions and user roles, and auditing and history information.

Within the DAL, new data entities can be created, updated, deleted and queried on search criteria. The flexibility extends to supporting batch file mass-updates. The DAL provides common set of persistence services for accessing and manipulating elements of the domain model. This implies flexible configurable control of static data (Banks, Customers, Agreements, etc.), payment data (Interchanges, Instructions, and Transactions), permissions and user roles, auditing and history information.

The SDK provides scenarios to add new attributes to existing domain objects using either "Dynamic Attributes" or "Polymorphic behaviour". Additionally, new domain objects can be added using DAO handler classes.

Common Services

The Open Payment Framework provides an extensive array of common services available to all service layers within the framework. These include controlling user access to functions and data according to pre-defined user group membership. Roles can be defined within the system providing various levels of data and functional access for that role. This extends to authorisation levels within the framework, all flexibly configurable and open to integration with internal legacy solutions (e.g. CRM, etc.).

Open Standards

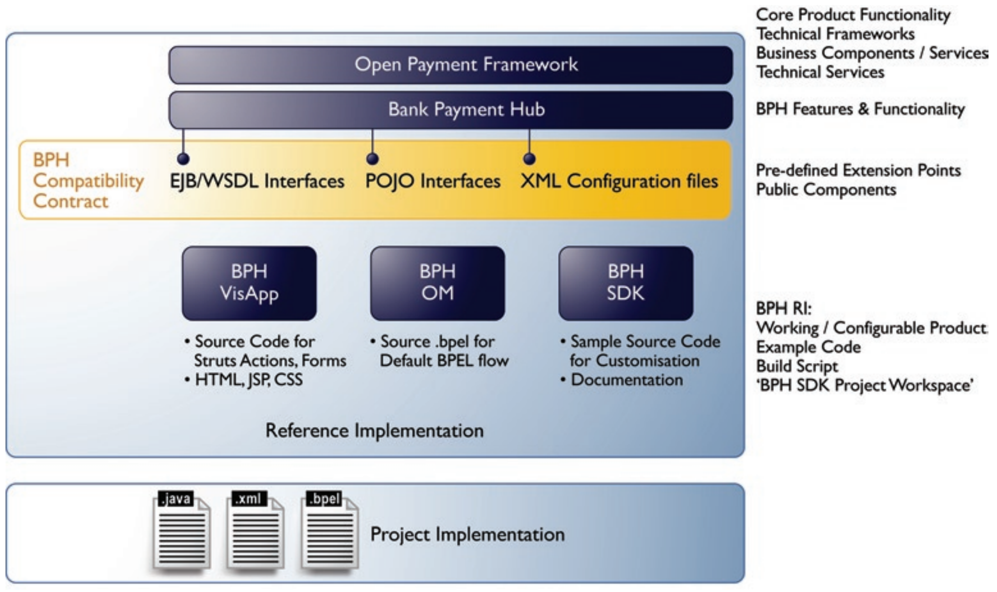
The Open Payment Framework is based on industry standard best practices and standard frameworks. The architecture leverages the following technologies:

- J2EE 1.4
- Struts and Tiles 1.2.7
- Hibernate 3.1.2
- JAAS 1.4
- SOAP/XML/XSLT
- BPEL 1.1

The reference platform is Linux/WebSphere/Oracle and utilises WebSphere Process Server 6.0 - IBM's Business Process Manager (BPM) tool for BPEL flow management. Ports can be made to AIX and Solaris, DB2. The robust Service-Oriented Architecture provides for extensibility throughout the application architecture.

The SDK – Customisation through Extension Points

The Open Payment Framework is delivered as a software development kit (SDK) through documented APIs, customisation patterns and a suite of reusable frameworks. The guiding design principles are a set of different customisation scenarios and use cases (e.g. Write New Validation Rules or Customise the Submission Service). The SDK provides set of guidelines and documentation concerning the predefined extension points and public API's. These include the EJB/WSDL interfaces, POJO interfaces and access to the XML Configuration files and BPEL flows. The SDK comes with code samples and best practice guidelines on how to customise the implemented product.


Core Product Functionality
Technical Frameworks
Business Components / Services
Technical Services
Open Payment Framework
Bank Payment Hub
BPH Features & Functionality
BPH Compatibility Contract
EJB/WSDL Interfaces
POJO Interfaces
XML Configuration files
Pre-defined Extension Points
Public Components
BPH VisApp
• Source Code for Struts Actions, Forms
• HTML, JSP, CSS
BPH OM
• Source .bpel for Default BPEL flow
BPH SDK
• Sample Source Code for Customisation
• Documentation
BPH RI:
Working / Configurable Product
Example Code
Build Script
'BPH SDK Project Workspace'
Reference Implementation
.java
.xml
.bpel
Project Implementation

SDK Deliverables

The SDK delivers all the necessary artefacts to customise and build the OPF-derived payments solution. These include the .jar files, .config files and any 3rd Party .jar files deemed appropriate. This also includes the SDK Project Workspace to assist development based on well-defined project structure and the SDK InfoCenter holding all technical docs, samples, config files, javadoc, database schema, best practices, etc. The SDK fully supports backwards compatibility of custom and core libraries.

Extending with the SDK

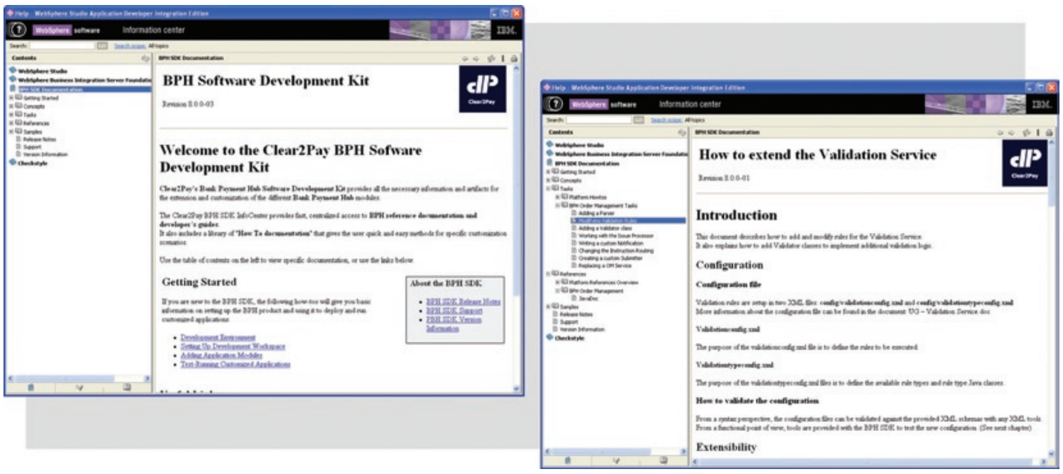
The SDK enables the customisation of the payment flow, modification of the behaviour of individual business services and the extension of the underlying data model. One of the many possible customisation scenarios is to add additional data fields to the payment repository. The SDK guides the user through the customisation of the parser to capture new fields, customise validation rules, change submission to include new fields and customise JSP to show fields.

Another scenario would be the ability to alter the routing and clearing rules based on configurable criteria. It is possible to instantly alter business process rules and controls without having to resort to development delays. Similarly, submitter changes require only alterations to some Java code from the standard interface – no reimplementation is needed.

Similar flexibility is offered throughout the solution wrestling control from development partners back into the hands of the in-house business experts.

SDK InfoCenter

The SDK InfoCenter is an integrated "help system" documenting the Open Payment Framework product and SDK scenarios. The content is logically grouped so developers can readily find the necessary information (Getting Started, Concepts, Tasks, References and Samples). The target audience is the project teams and other users who need to customise the standard behaviour of the OPF-derived payment solution.
The SDK InfoCenter contains reference documents on architecture, technical design documents on Business Services and the data model. It contains numerous examples on how to use, configure and extend services including "public" interfaces and classes. All configuration files are described along with the full database schema.



BPH Software Development Kit

Revision 2.0.0-03

Welcome to the Clear2Pay BPH Sofware Development Kit

Clear2Pay's **Bank Payment Hub Software Development Kit** provides all the necessary information and artifacts for the extension and customization of the different **Bank Payment Hub** modules.

The Clear2Pay BPH SDK InfoCenter provides fast, centralized access to **BPH reference documentation and developer's guides**.
It also includes a library of **"How To documentation"** that gives the user quick and easy methods for specific customization scenarios.

Use the table of contents on the left to view specific documentation, or use the links below.

Getting Started

If you are new to the BPH SDK, the following how-tos will give you basic information on setting up the BPH product and using it to deploy and run customized applications:

- Development Environment
- Setting Up Development Workspace
- Adding Application Modules
- Test-Running Customized Applications

About the BPH SDK

- BPH SDK Release Notes
- BPH SDK Support
- PBH SDK Version Information

How to extend the Validation Service

Revision 2.0.0-01

Introduction

This document describes how to add and modify rules for the Validation Service.
It also explains how to add Validator classes to implement additional validation logic.

Configuration

Configuration file

Validation rules are setup in two XML files: **config/validationconfig.xml** and **config/validationtypeconfig.xml**
More information about the configuration file can be found in the document: UG – Validation Service.doc

Validationconfig.xml

The purpose of the validationconfig.xml file is to define the rules to be executed.

Validationtypeconfig.xml

The purpose of the validationtypeconfig.xml files is to define the available rule types and rule type Java classes.

How to validate the configuration

From a syntax perspective, the configuration files can be validated against the provided XML schemas with any XML tools.
From a functional point of view, tools are provided with the BPH SDK to test the new configuration. (See next chapter)

Extensibility

Kick-start Development with the SDK Workspace

The SDK Workspace provides a complete development environment for IBM WebSphere Integration Developer. This offers accelerated customisation to developers providing all necessary scripts and example projects. It defines where the developers work and splits the artefacts and libraries into core (product) and custom (project) areas. The platform ships with the necessary build scripts and best practices regarding managing the class paths and handling the manifest files. Importantly, custom code is not connected to the core product code to ensure backwards compatibility of the product. New versions can be released without impacting any custom changes made on the previous version.

Future Perspective

Service-Oriented Architecture is based on the use of distributed objects and components and is the next evolutionary step in computing environments. SOA does not have a standardised, reference model yet; however, implementations share the main concepts of services, service descriptions, advertising and discovery, the specification of an associated data model, and the use of a service contract. A SOA may also implement optional concepts that include a service consumer, a service client, acceptance of the service contract and invoking the service.

There are many business drivers affecting the development of a standardised SOA reference model. Once this is achieved, SOA will likely be part of the solution for many business and world issues.

Clear2Pay NV/SA

Schaliënhoevedreef 20A
B-2800 Mechelen
P: +32 15 79 52 00
F: +32 15 79 52 01
E: info@clear2pay.com
w: www.clear2pay.com