

# PxP

from hyfinity Limited

an evaluation

---

*Author: Peter Abrahams*  
*Published: November 2003*

---

 **Bloor**  
**Research**



# Background

---

During 2003 there has been a lot of progress with web services. Products have appeared, standards have been set and implementations have occurred. Why is there so much interest in web services? Fundamentally because they better reflect the way businesses really work.

Businesses send documents internally between departments and externally to other businesses. The documents can be promises to buy, sell or deliver something or they are requests for information and their responses. In all cases a document, whether it is paper, email or a transaction, is sent to someone who will service it. This is obviously a gross simplification of business but is in essence true.

The documents that flow and the services provided at the end point map directly to the XML messages flowing between web services. This direct mapping enables the business and IT to talk the same language. A business manager will be able to understand the structure of an XML message, the process flows and the definition of a web service.

It is only recently that it has become practical to develop composite applications that map directly to the business requirements. This has come about because of development of new standards and the increased performance of computers and networks. The web services and XML standards ensure that different functions can inter-operate whether they are in-house built, re-purposed legacy applications, bought in applications or external services. The greater performance of computers, and especially the networks, more than compensate for the extra processing required to support these standards.

Bloor Research believes that this reflection of business by IT, in a cost-effective way, means that web services will take off rapidly and that all organisations should be planning to evolve towards web services and XML. With that requirement, both development and integration need to have native support for the new environment and the transition from the legacy state.

hyfinity's products have been built for the promise of XML and web services and provide a novel and productive solution for such projects.



# Fast Facts

---

**Key findings** In the opinion of Bloor Research the following represents the key facts of which prospective users should be aware:

- PxP is hyfinity's flagship product, providing peer-to-peer integration functionality.
- It is based on their Morphyc Architecture, which defines how to build and execute native XML-based application projects.
- A key feature of Morphyc is a forward-chaining logic engine for the processing of all business rules.
- The product strongly supports the concepts of e-business patterns as defined by IBM and provides a set of business patterns with the products.
- hyfinity has based its product development around XML and related standards and this has produced a strong and simple technology in a short time.
- A number of the concepts and constructs in the architecture are different to traditional development and a short learning curve (or more importantly a short un-learning curve) is required to appreciate the products fully. However the benefits of the new approach are worth the effort in terms of developing reliable solutions quickly.
- Besides PxP, hyfinity markets MVC, which implements web user interfaces and is based around the XForms standard. This product is used for the GUI for the PxP development environment as well as the hyfinity web sites.
- PxP is in the business integration space. Its focus is the assembly and construction of composite applications, based on XML messaging. It does not supply adapters to legacy systems, but works with the traditional adapter vendors, for example, iWay. However it does provide extensions to convert XML to SQL and also one to turn well-formed non-XML messages into XML.
- hyfinity has built a series of industry-specific solutions on top of PxP and MVC by providing reusable assets tailored to manufacturing, finance and e-Government.
- The Morphyc architecture defines both a run-time engine (XEngine) and a development environment (XStudio). PxP and MVC are both extensions of these environments.
- The whole of the architecture and the product are based around reuse. Nearly all development is done by copying, cloning and specialising existing assets. This has resulted in very high productivity in the initial client implementations.



**The bottom line** hyfinity is a new company with a fresh approach. They have been able to exploit the emergence of new standards and to develop a unique architecture to provide an easy-to-use, effective development and integration environment. This is a product ideally suited to enterprises that have made an architectural decision to use XML for all internal messaging, especially in the construction of composite applications.

**Vendor Information** hyfinity was formed in 2001. The founders had worked on early mission-critical XML solutions and saw the potential for building a new environment based on native XML and standards and a small kernel.

They developed the small kernel based on their Morphyc Architecture and built two products on top of the kernel; PxP, a peer-to-peer business integration product and MVC, which implements and extends XForms.

hyfinity is a private limited company funded by its founders and engagement and product income and has made profits in the last two years.

hyfinity web address: [www.hyfinity.com](http://www.hyfinity.com)

# Product description

**Introduction** PxP is hyfinity's main product. It is a peer-to-peer business integration package that allows the integration, and in some cases the development, of composite applications that receive, process and produce XML. It integrates applications by performing a series of operations on XML messages. Examples of operations include transforming, translating, validating and routing of XML messages. If the applications do not produce or accept XML then hyfinity will use third party adapters (such as iWay) to deal with the conversion.

PxP stores all the definitions of the transformations, routings and business processes in XML documents. PxP processes the messages against the definitions by using hyfinity's underlying technology called XEngine. XEngine is the kernel product and is an implementation of their Morphyc Architecture (described later). The definitions are created using their integrated development environment, XStudio. XStudio is also built on top of XEngine.

XStudio has a lot of user interactions through a browser. This user interface is built using hyfinity's other product MVC. MVC is an implementation of XForms and was developed to simplify the development of XStudio. It is now marketed as a separate product for anyone who needs an XForms-based web application.

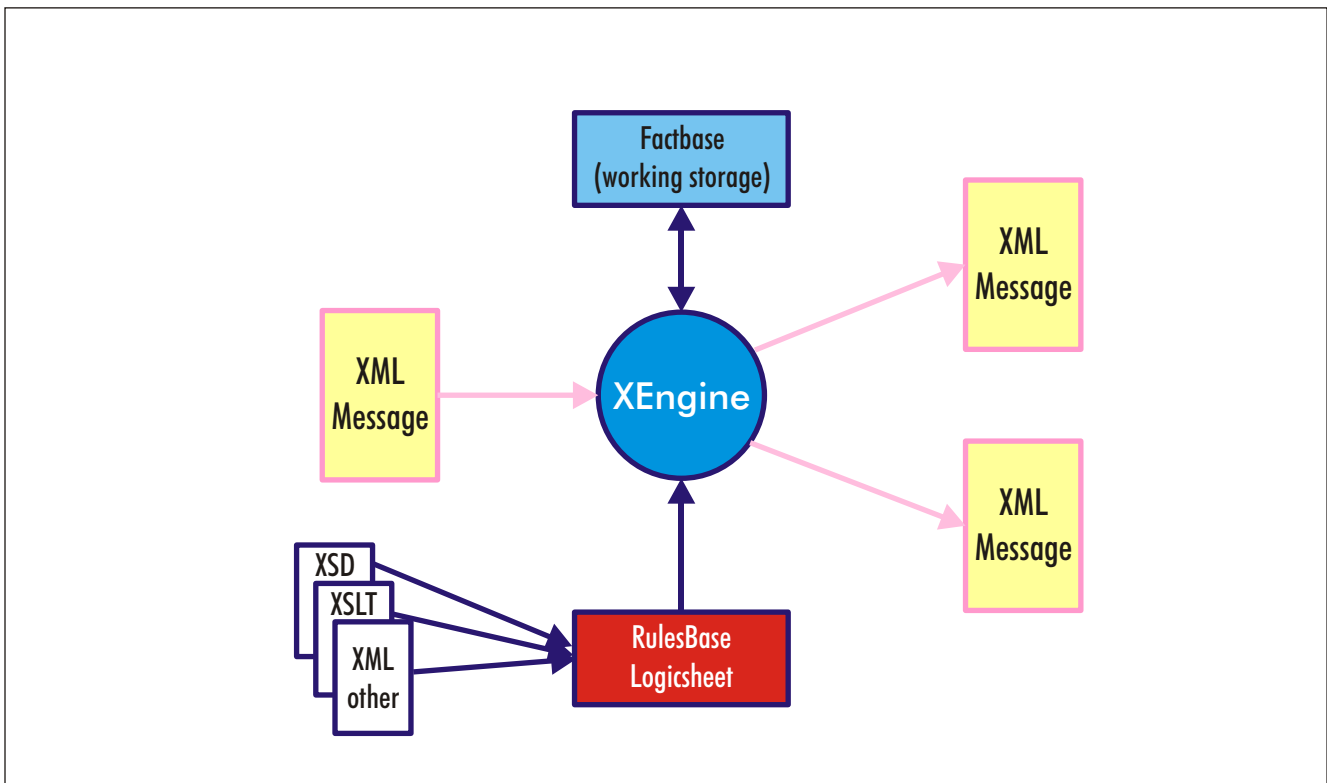


Fig. 1: Architecture of the Morphyc Runtime



All the products and the Morphyc Architecture are closely linked and interdependent. Therefore it would be possible to start the product description in several places. Bloor Research has decided to start from the run-time architecture and then work back to the development environment and the specific products.

## Morphyc Runtime

The heart of hyfinity is the Morphyc Architecture and this architecture is implemented by XEngine. The basic role of the engine is to consume XML messages and emit other messages. The process required to go from input to output is defined in the Rules Base and will require some working storage known as a Factbase. The engine is a forward-chaining logic engine. For those of you not familiar with this concept the box on the left gives a brief description.

### Forward-Chaining Logic Engine

The processing of a forward-chaining logic engine is defined by a series of 'IF condition THEN action' rules (note there is no ELSE clause).

When the engine consumes a message it looks at its rules base to see if any of the conditions evaluate to true. If so the action in the rule is processed: this action may change some information in working storage.

When the action is complete the engine checks to see if the condition of any other rule evaluates to true; if so, the process is repeated. Rules that previously evaluated false may now be true because of changes in the working storage.

This continues until no rules that have not been triggered evaluate to true. Then one or more messages may be emitted.

Such engines will have mechanisms to decide which order rules should be processed if more than one evaluates as true and also to ensure that a rule does not trigger twice.

As can be imagined, defining processes using a rules base is distinctly different from writing procedural code. However it is a very powerful construction and particularly suitable for business rules as it tends to be the way business people define business rules.

In hyfinity's XEngine the rules are defined by a set of XML statements called a logicsheet. The actions can include, for example, the processing of an XML Schema Definition (XSD) to ensure the validity of the input message or the use of Extensible Style-sheet Language Transformation (XSLT) to transform the input into a different structure or the calling of any other service. As well as the rules the working storage is held as a XML structure.

The fact that message, data and program logic are all held in the same format (XML) harks back to the purity of a von Neumann Machine where all messages, data and program also were held in the same format (in that case binary).

The fact that the Morphyc Architecture uses XML for messages, working storage and rules means that the development of the engine was small and accounts for the power of the product developed by a small team.

However an engine that consumes and emits XML is not sufficient to construct an application. This will require multiple engines interacting with each other by passing XML messages as shown in Figure 2.

This is a peer-to-peer network and the engines may physically reside anywhere. They may be on the same physical machine or anywhere on a network.

Besides the standard XEngine, provided with PXP and MVC, Custom Engines can be developed. These will enable processing that cannot be defined with XEngine and, in particular, connection to non-XML environments. hyfinity provide engines, such as SQLiser for accessing relational data, and XMLiser that connects to messages in non-XML format. Other engines include dispatchers for e-mail, fax and SMS services.

It should be noted that XEngine is not just used for the run-time environment but is also the basis of the development environment, XStudio. XStudio is, in a sense, just another application and therefore it makes sense to develop it on top of XEngine. This has some major benefits:

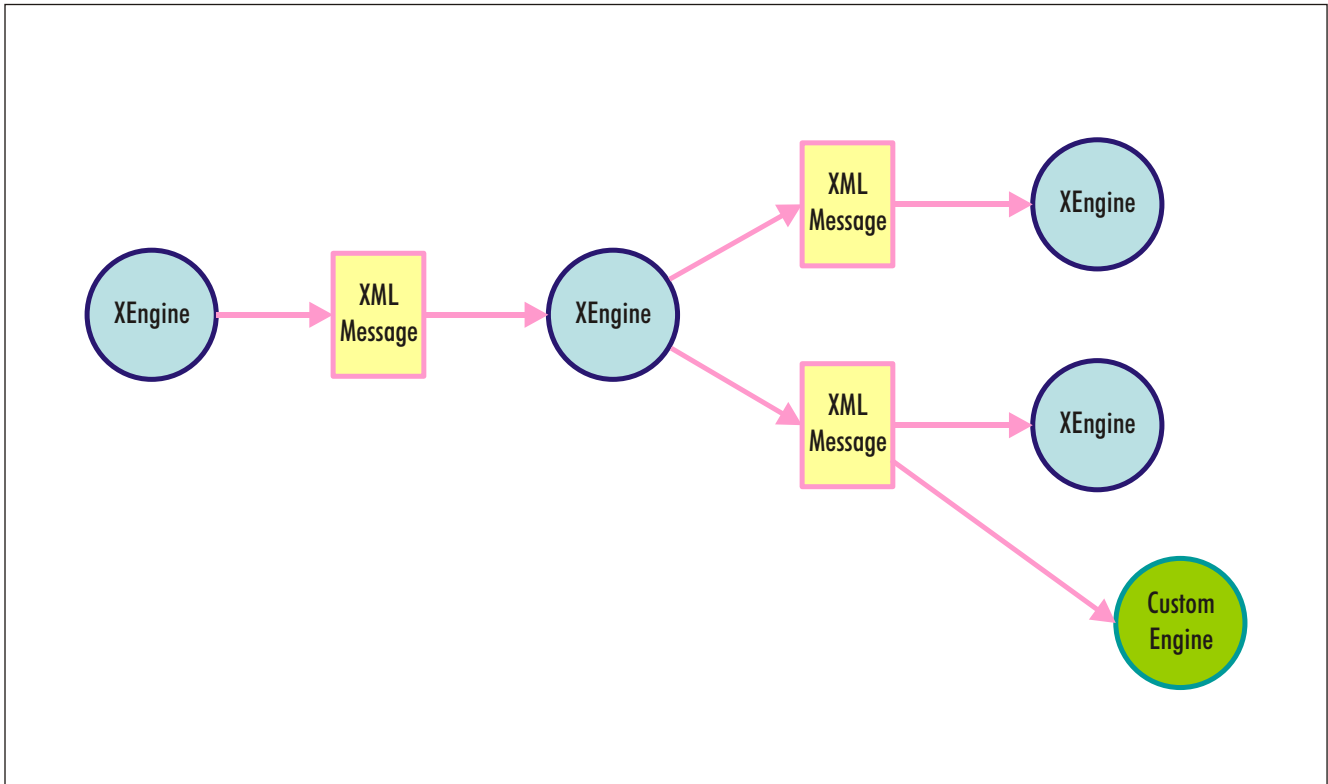


Fig. 2: Multiple engine interaction

- New XStudio functions can be built using XStudio so speeding the development.
- The user interface is used by the developers so it has had an extensive usability test before the first product was released.
- The platform has been thoroughly tested.

Finally, hyfinity’s first two products are also based on the XEngine, PxP (the business integration product) and MVC (the forms product).

### Morphyc Development Environment

Having described the run-time environment it is obvious that there is a requirement for a development environment that can:

- Define new, or use existing, XML resources such as messages, style sheets or schema definitions.
- Define the overall structure of an application project with the interconnectivity between processing nodes.
- Create the logicsheets.
- Test new functions.
- Deploy functions.



All of this is accomplished through the extended development environment XStudio. To understand the structure of XStudio we need to introduce some further constructs that are used to simplify development using XStudio. The constructs are patterns and blueprints

*Patterns* Patterns in Morphyc are based on a concept developed by IBM to assist in the development of e-business applications. It was recognised that e-business applications tended to share similar interaction patterns covering all aspects of e-business solutions. Two example patterns are:

- Exposed\_Business\_Service: that provides a service access point for external organisations. For example a business receiving a purchase-order.
- Managed\_Process: that provides support for long-running transaction conversations that support many business processes. For example the steps required to process a purchase-order.

The patterns define the basic implementation structure for such processes and what has to be done to particularise them.

XStudio is delivered with a large selection of these patterns, plus the ability to develop new patterns if they are required or deemed necessary.

hyfinity application projects are constructed by choosing suitable business patterns, configuring them into run-time patterns for the specific application, then connecting them together and filling in the detail.

The use of patterns speeds up development in two ways. It reduces, or eliminates, the effort required in developing base code. It also increases the quality of the project by reusing tried and tested processes.

*Blueprints* Blueprints define the context and framework for applications.

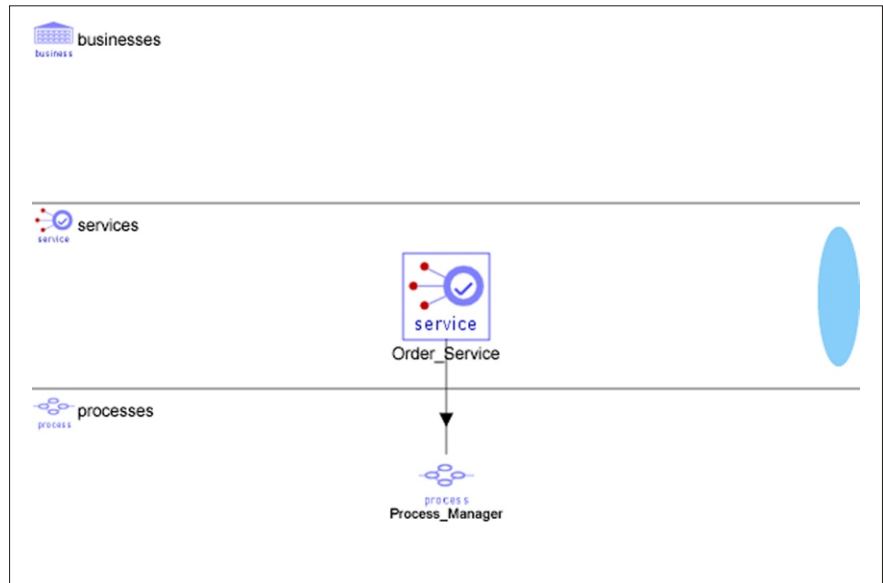
For example, hyfinity's PxP (their peer to peer integration product) is an application and is defined by its blueprint. The blueprint has three parts:

- Pattern Pools: these include all the patterns that could be of interest to this application space.
- Architectural Layers: define the different types of elements that make an application. For PxP these are:
  - » Business Networks—represents networks of businesses interacting, such as a marketplace of buyers and sellers.
  - » Businesses—represents the individual business in the network.
  - » Services—the services offered and exposed by the business.

- » Processes—internal business processes required to provide the services. The processes are not exposed to external businesses.
- » Resources—such as relational databases, message queues, etc.
- Asset Pools: these store messages, schemas, style-sheets, logic sheets and other types of assets.

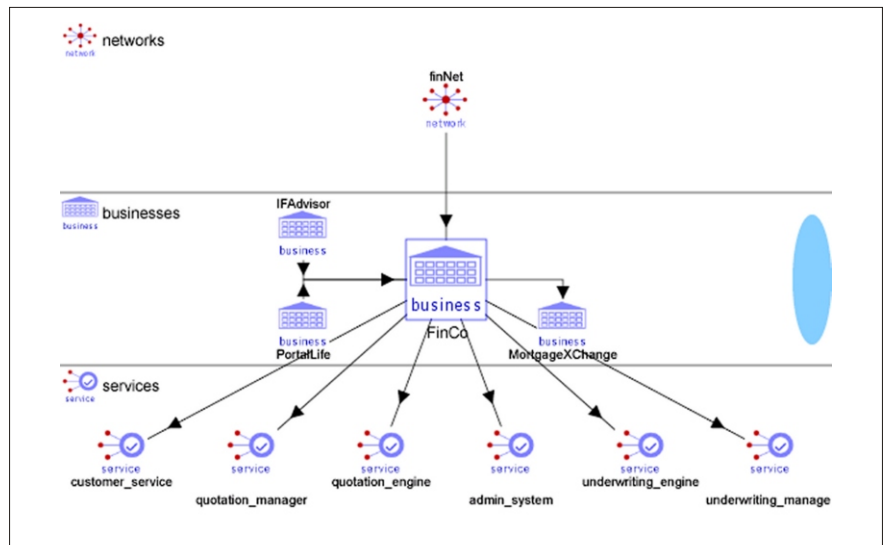
*Defining the Overall Structure*

The structure of the project is defined by choosing business patterns, placing them in the layered architecture and connecting them together. A simple fragment is shown in Figure 3.



**Fig. 3: Basic association of patterns**

The Order\_Service is a run-time pattern, based on the Exposed\_Business\_Service business pattern. It is in the service layer so it is clear that it provides a service that will be exposed to the other users. It is connected to the



**Fig. 4: More complex associations**



Process\_Manager that is based on the Managed\_Process business patterns. It is in the process layer because it is not exposed to the outside world.

This layering and structuring very quickly builds up the overall architecture of the application and makes it very easy to navigate around. A more complex example to show the power is shown in Figure 4.

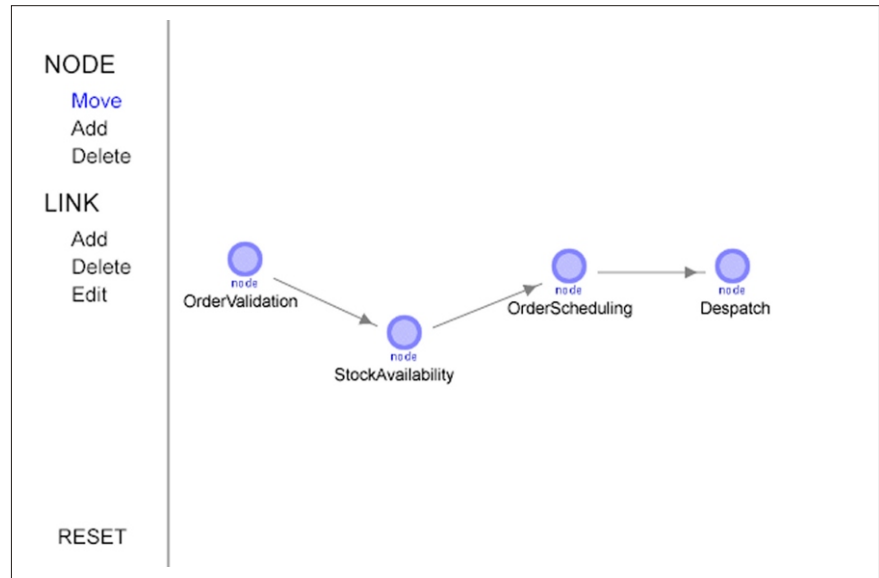


Fig. 5: Internal processing nodes

The run-time patterns can be opened up to show their internal processing nodes; the example in Figure 5 shows an order fulfilment service. Each of the nodes will have various assets associated with it and, most importantly, the logic sheet that defines the rules related to this process.

### *Create the logic sheets*

The logic sheet is attached to a node. Making up the logic sheet is a set of rules. The rules are developed by using the RuleMaker section of XStudio. Figure 6 shows an example of developing a simple rule.

The top just shows the naming and connection to the logic sheet.

The next section shows the condition clauses in graphical form. Only the bottom clause has been developed and the message contains a fragment of XPATH related to order placement. The other two clauses have not yet been developed but will take on a default so that the rule can be tested even while incomplete.

The results section shows that if the condition is true then the order\_placement service will be started.

The same process is continued for each rule related to a node.

### *Test new function*

Testing is very simple as all messages are XML. To test a new service it is only necessary to create the relevant XML message and send it to the relevant service. XStudio includes a function called XClient. As can be seen in the example

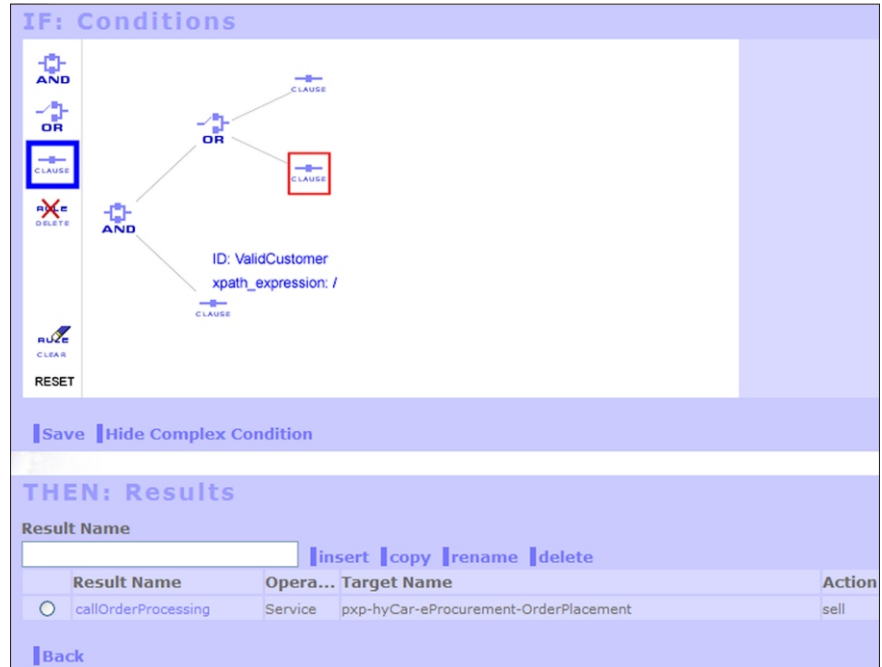


Fig. 6: Developing a simple rule

shown in Figure 7, XClient simplifies unit testing. The name of the service to be tested is entered and the test message is either read from a file or typed in directly. The results are immediately available for viewing in a window below the request window.

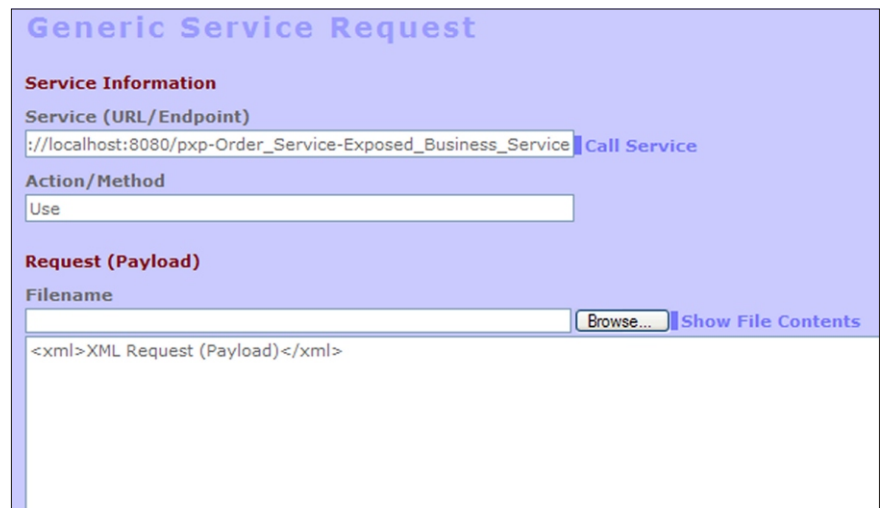


Fig. 7: XClient screenshot

Application projects can be constructed top down or bottom up. With traditional development there has always been the issue of developing harnesses and stubs to allow testing of partially complete systems. This issue does not exist in the Morphyc environment because any pattern or node is always executable, and will return a well-structured piece of XML, so that testing can continue.

*Deploy function*

When a project has been tested it can be deployed into production. This is a matter of moving the required assets (logicsheets, XSD, XSLT, and others) from



the test to the production environment. This deployment is simple because there is no 'compile' process and it is just a matter of moving the relevant source to the new location.

XStudio has a function called XGen that assists with this process. At the simplest it can be pointed at a complete project and told to deploy it in total to a single physical node. It can also be pointed at particular patterns or even individual nodes and deploy just that part. Further, the individual parts can be deployed on a single node or distributed around a network.

*The environment* All of XStudio has been built using itself and in particular it should be noted that all the screens that have been shown above have been developed using MVC. MVC was originally built to speed the development of the GUI interface of XStudio but it is available as a product from hyfinitivity as a very simple way to develop web-based GUIs. As an example of the power of MVC the whole of hyfinitivity's web site has been built using MVC.

*Implementation* The run-time platform only requires a Java Virtual Machine (JVM), an XML Parser, and an XSLT Engine. It ships with an Open Source environment that includes Tomcat (a simple web server), Xerces (an XML parser), Xalan (an XSLT engine) and Sun Microsystems' JVM. If the nodes are to be published as web services then there is a requirement for a web server. Tomcat can be used, but for high-end applications, IBM WebSphere has been tested.

Although there is no requirement for a database, the platform supports SQL Server, DB2 and Tamino, which is an XML database from Software AG. The Platform also supports Linux and testing against Solaris, BEA WebLogic and Oracle is currently underway.

hyfinitivity has also tested their products with a XML hardware accelerator from DataPower. This requires no change to the application definition and provides a considerable boost in performance.



# Summary

---

hyfinitly recognised the future importance of Web Services and XML early. They also recognised the importance of the concept of patterns for simplifying the creation of e-business solutions. They believed that existing application development and run-time tools were not architected for this new environment and did not fit well with it. This understanding led to the development of hyfinitly's Morphyc Architecture, which is based on the new concepts and supports the development of web services.

The architecture defines how all data, messages, process definitions and interface definitions can be held in XML format. This consistent use of XML greatly simplifies the development of new solutions.

Using the architecture hyfinitly have designed, developed and delivered two application environments: PxP and MVC.

MVC creates and runs forms-based applications. It is an implementation of the recently ratified XForms standard. The power and speed of development with MVC can be seen by looking at the development environment for PxP, which was all developed using MVC.

PxP is a Peer-to-Peer web services integration product. The use of the Morphyc Architecture patterns and layers has ensured a well structured, easy to use and productive development environment, supporting a highly efficient distributed run-time capability.

Early users of these products have shown that it can be used to produce industrial strength solutions, which are capable of modification and extension, in a highly effective manner.

hyfinitly has arrived at just the right moment to drive the web services surge that we expect in 2004.



# Glossary

---

Below are some short definitions of technical terms used in this report.

**XML:** Extensible Mark-up Language is a standard for defining information formats made up of mark-up tags that define the data and the data itself.

**XForms:** An XML definition of a form that carefully separates the definition of what a form does and how it looks on a particular device.

**XPATH:** an XML standard for addressing parts of XML documents.

**XSLT:** Extensible Style-sheet Language Transformations is a standard way to describe how to transform (change) the structure of an XML document into an XML document with a different structure.

**XSD:** XML Schema Definition is a recommendation of W3C; it specifies how to formally describe the elements in an XML document. This description can be used to verify that each item of content in a document adheres to the description of the element in which the content is to be placed.

# Copyright & Disclaimer

---

This document is subject to copyright. No part of this publication may be reproduced by any method whatsoever without the prior consent of Bloor Research

Due to the nature of this material, numerous hardware and software products have been mentioned by name. In the majority, if not all, of the cases, these product names are claimed as trademarks by the companies that manufacture the products. It is not Bloor Research's intent to claim these names or trademarks as our own.

Whilst every care has been taken in the preparation of this document to ensure that the information is correct, the publishers cannot accept responsibility for any errors or omissions.



Suite 6, Challenge House, Sherwood Drive,  
Bletchley, Milton Keynes, MK3 6DP, United Kingdom

Tel: +44 (0) 1908 625100 - Fax: +44 (0) 1908 625124  
Web: [www.bloor-research.com](http://www.bloor-research.com) - email: [info@bloor-research.com](mailto:info@bloor-research.com)