

# Record and Reuse using Publish/Subscribe Messaging

John Ibbotson  
IBM UK Ltd

Hursley Park  
Winchester, UK

+44 (0) 1962 815188

john\_ibbotson@uk.ibm.com

David De Roure

Intelligent, Agents, Multimedia Group

School of Electronics and Computer Science  
University of Southampton, UK

+44 (0) 23 8059 2418

dder@ecs.soton.ac.uk

## ABSTRACT

The Publish/Subscribe messaging model is widely used within distributed enterprise systems as a scalable delivery mechanism for business events. More recently, implementations have been developed for more compact messages such as those used for instrumentation applications. In this paper, we develop a proposal for the publication of events that may be linked to multimedia data objects stored in a distributed repository. Events and objects are linked via reference information. We show how this messaging model, together with a distributed object store might meet the requirements for the Equator project record and reuse infrastructure.

## Keywords

Keywords are your own designated keywords.

## 1. INTRODUCTION

Message based processing is a technology widely deployed in enterprise IT infrastructures. Messages represent business events such as stock prices, purchase requests, hiring approvals and delivery notices. These events trigger processing activities that in turn go to make up sets of business processes. Events flow sequentially and are ordered in time. They are usually generated in real time, but batch processing and other time window requirements mean that they may be stored and replayed at some time in the future. Auditing and regulatory requirements also mean that the messages representing actions and events within an enterprise have to be retained for future inspection.

The Equator project has similar requirements. In its case, the events being captured include multimedia information together with rich metadata associated with the event. The multimedia information may be of many different forms and be less structured than business information.

The publish/subscribe messaging model is widely used within enterprise IT infrastructures for the distribution of messages between processing applications. Its flexibility allows distributed processing, scalability and resilience together with anonymity for the processing clients. That is, a producer of messages does not need to know where its messages are ultimately consumed. Similarly, message consumers do not need to know where the messages are produced.

This paper describes the basic publish/subscribe messaging model for topic based subscriptions. It then describes the extended model which includes processing within the broker and content based subscriptions. Finally it proposes an architecture for the storage and retrieval of multimedia data of the kind generated

within Equator using publish/subscribe messaging as a distribution mechanism. We also comment on the applicability of Grid notification services for implementing the same architecture.

## 2. PUBLISH/SUBSCRIBE MESSAGING

This section introduces the publish/subscribe messaging model and its extension into content based processing with mediation brokers.

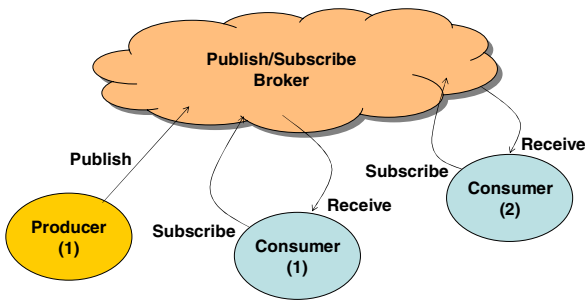
### 2.1 Producers, Consumers and Brokers

In complex messaging infrastructures such as trading room floors, there is a requirement to separate the applications that produce messages from applications that consume them. There must be no interdependence between producers and consumers that will add to the complexity of managing such an environment. The flexibility requires that both message producers and consumers can be created and destroyed at will; their presence being managed through a distributed broker whose role is to forward published messages to relevant subscribers.

Message producers publish messages that consist of two parts. The first part consists of the message data which may be of any structure (XML, name/value pairs, binary encoded etc). The second part is a topic string that is associated with the contents of the message. Topics usually form part of a hierarchical namespace that is common to all producers and consumers. For example, for publishing stock price information a topic that identifies the market, segment and company might be used. Such a topic space might be represented by the string "NYSE/Industrial/BigSteelCo".

Message Consumers register an interest in published messages by subscribing to topics within the topic space. For example an application interested in receiving all messages related to Industrial stocks on the New York Stock Exchange (NYSE) would subscribe to the topic string "NYSE/Industrial/\*". Interest in all NYSE stock prices would result in a subscription to "NYSE/\*" where the \* wildcard has its usual meaning.

Subscriptions are managed by a broker. Message consumers subscribe to the broker which keeps a list of consumers together with their registered subscriptions. Consumers may register for many different subscriptions. When a message producer publishes a message, it does so to the broker rather than directly to the consumers. The broker then matches the published subscription to the set of registered subscriptions and forwards the published message to the appropriate consumers.



**Figure 1 Publish/Subscriber Infrastructure**

Figure 1 illustrates a Broker with a single producer and two consumers. In a distributed environment, physical brokers may exist on many processors, sharing their subscription information through close network coupling. This close coupling between the physical brokers results in a single logical distributed broker.

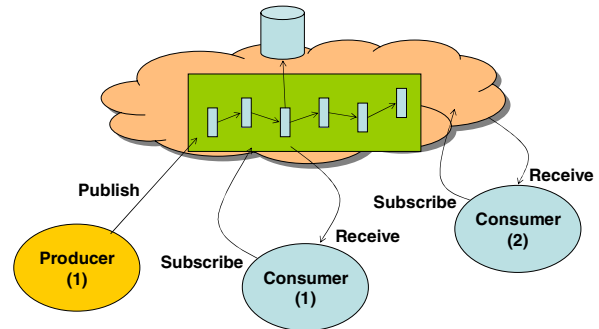
Note that the broker may provide different qualities of service (QoS). Examples of QoS may include message persistence to support broker failure recovery, security services such as producer/consumer authentication and queuing support for consumers that are not permanently connected.

## 2.2 Mediation Brokers

The basic broker architecture described in the previous section simply acts as a forwarder of messages from a producer to a set of consumers. All processing of the message contents is handled by the producing and/or consuming applications. More recent developments include the addition of processing pipelines to the broker – these are called message flows. The additional processing capabilities provided within the broker allow more comprehensive subscription capabilities to be implemented. Content based subscriptions extend the topic based subscription mechanism. In content based subscriptions, consumers can add terms to their subscriptions based on elements within the message content. To extend the previous example, a consumer may wish to receive messages about “NYSE/Industrial/BigSteelCo”, but only when the stock price is greater than 75. For this additional term to be accepted, the broker must have access to the message contents schema so that it can extract the value of the “stock\_price” element to check that a subscription is satisfied.

Figure 2 illustrates a broker including a message flow. An example scenario that would use such an infrastructure might be for processing stock price messages within a stockbroker. Messages received from a trading floor consisting of company and price are published and received by the broker. Within the message flow, the company element of the message is extracted and a “join” implemented between the company element and a set of customers held within a client database. This can identify which customers hold stock in that company. For each of the customers, a computation step in the message flow computes the new value of the holding given the new stock price. This results in

a new message that is sent to the consumers representing each of the customers that subscribe to the service.



**Figure 2 Mediation broker with message flow**

## 3. EVENT RECORDING AND REUSE

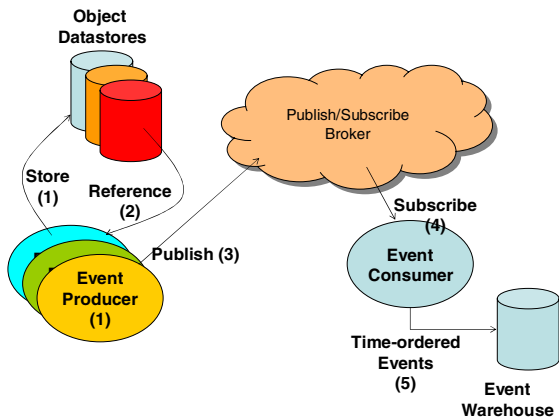
We now discuss how the publish/subscribe messaging model may be used to implement an event recording and reuse architecture.

### 3.1 Recording Events

We view events within Equator as a time-ordered set with associated data and metadata. The recording and reuse of an event stream is analogous to the use of a tape recorder for recording sequential data. Our proposed model supports this analogy with the difference that data associated with an event is represented by reference rather than by value. The notion of persistent content being addressable by reference in this manner is well established in the hypertext research community, based on Nelson’s “permascroll” storage philosophy [1], though generally for textual content.

Events within Equator may be captured by one or more message producers; each event having its own unique identifier and timestamp. Example events might be a probe reading taken by a school student in a science experiment, a PowerPoint slide transition within a meeting, the joining of a delegate to a phone teleconference, a digital photograph or sequence of video or audio taken at a particular place and time.

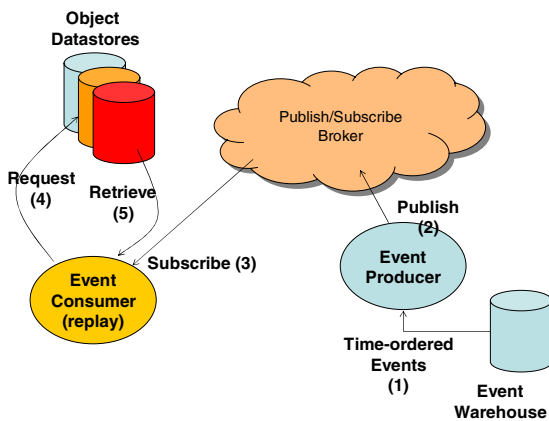
Figure 3 illustrates the event recording process. A set of producers capture events. Associated with an event is some optional data which can be stored in an appropriate Object Datastore (1). The object is stored and a reference is returned to the producer (2). Note that the mechanism used to return the reference might itself be via a published message. Having received the stored data reference, the event producer can now assemble an event message and publish it (3). The topic used for publication might reflect an event stream identifier (common to all events), and a producer identifier.



**Figure 3 Recording events and associated data objects**

To record an event stream, an event consumer subscribes to all published messages for that stream (4). Note that there may be multiple event consumers subscribing to different event streams. In the case of a recording, the consumer acts as a logger, storing the received events in some Event Warehouse (5). The content of the Event Warehouse consists of the sequential set of events that constitute the event stream. Note that the Event Warehouse does not include any object data; that is stored separately in the set of Object Datastores. This architecture extends to multiple event warehouses according to the requirements of the application,

### 3.2 Reusing Events



**Figure 4 Reusing events**

When reusing an event stream, the stored events may be replayed from the Event Warehouse. This is illustrated in Figure 4. The stored events are retrieved from the Event Warehouse (1) as a sequential stream. Each event is then published by an event producer (2) using some well-known topic. An application used to inspect or process the replayed event stream is an event consumer that subscribes to the well-known topic (3).

If an event has some associated multimedia data, the data may be requested from its object datastore using the reference included in the event (4). The retrieved data object is returned to the replay application (5) for further inspection and/or manipulation.

Note that most publish/subscribe broker implementations use a queuing mechanism to deliver messages to a consumer. Using this mechanism, the consumer is able to request the next message once it has completed processing the current message. This simplifies the implementation of the message consumer.

## 4. IMPLEMENTATION WITHIN GRID

The recent publication of the Web Services Notification [2] specification proposes a mechanism for Web Services to notify (publish) other services when certain events occur. This can be directly, or via a third party broker service. The proposals described in this paper rely on customized software as an implementation. In the future, adoption of WS-Notification within the Grid community will allow a publish/subscribe architecture of the type described here to be implemented using standard Grid interfaces and architectural components.

## 5. CONCLUSIONS

In this paper we have presented the messaging infrastructure used in business applications and contrasted it with the needs of multimedia record and reuse in the context of Equator. We have proposed an architecture based on messages containing references to objects - the events effectively carry metadata. Topic- and content-based subscription enables consumers to select relevant events during replay.

## 6. REFERENCES

- [1] Nelson, T. "Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning, and Deep Re-Use", ACM Computing Surveys 31(4), December 1999.
- [2] Graham, S., et al. "Web Services Notification (WS-Notification)." <http://www-106.ibm.com/developerworks/library/ws-resource/ws-notification.pdf>