1.1 Publish/subscribe Pattern

1.1.1 Name and Source

Publish/subscribe Pattern
Page 205-222 in the book "Web Service Patterns: Java Edition" [WSP 03]

1.1.2 Also Known As

-

1.1.3 Type

1.1.4 Intent

To send web service notifications to interested clients about generic events that has originated from sources that may not provide web service notification mechanisms themselves.

1.1.5 Problem

The basic problem is similar to the observer pattern, i.e. a web service client wants to get notified about something. The difference is that in this pattern, the real source of the notifications may not itself provide a web service, and the event or topic that the client is interested in can therefore also be of a more generic nature than to be associated with a particular web service business process. For example, a client may want to subscribe to news and might specify topics of interest, e.g. “sports” or “politics”.

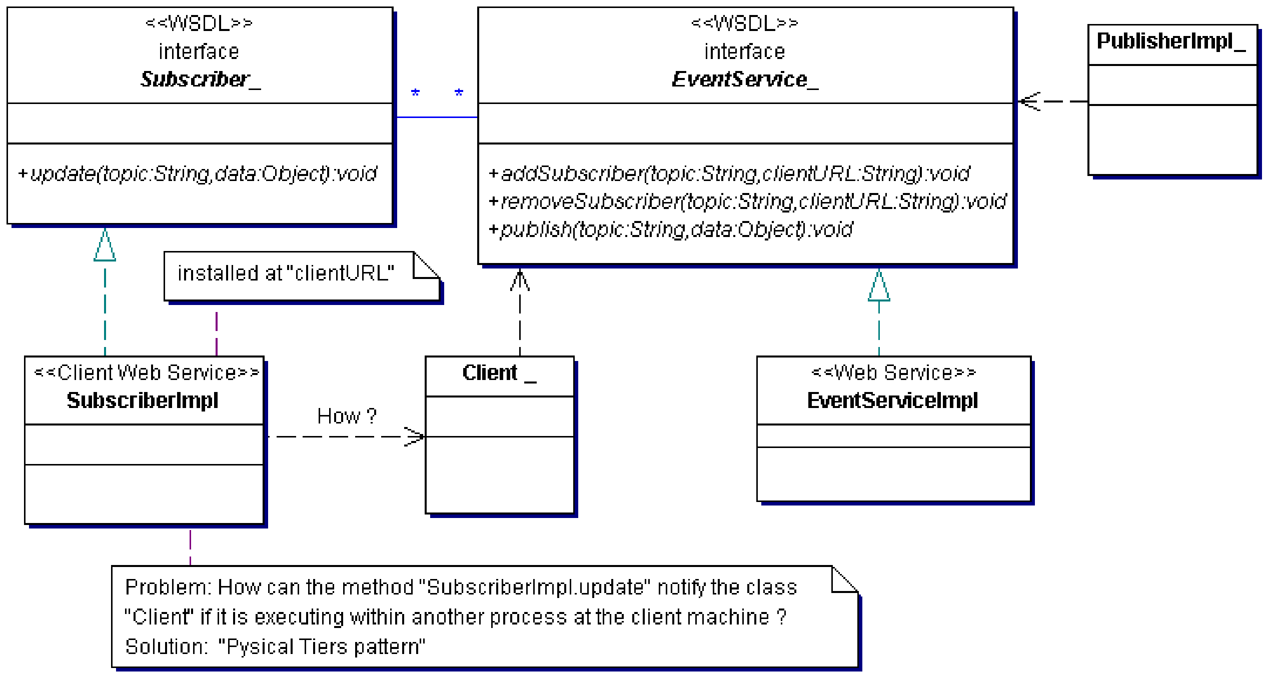
How can a web service client immediately become informed about some generic event or topic when the actual publishing sources of the events do not provide observable web services themselves ?

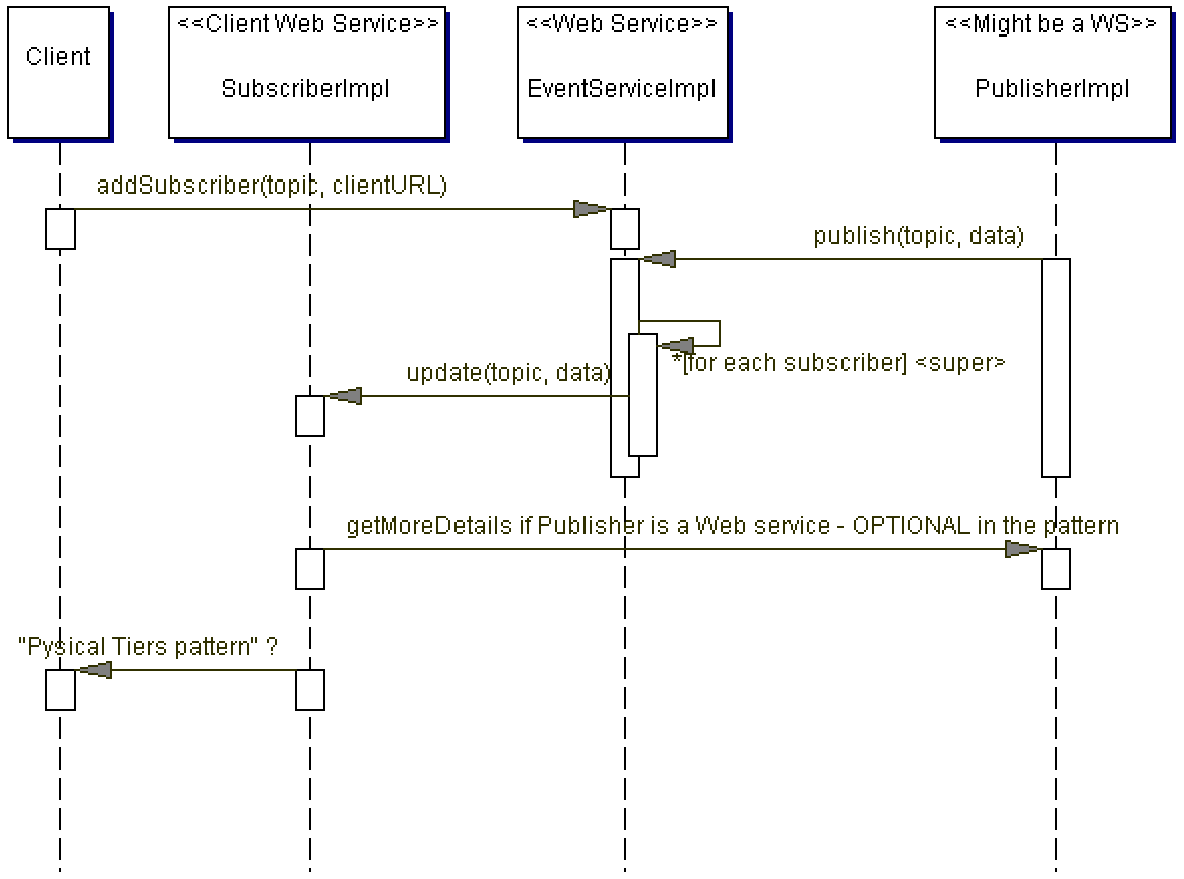
1.1.6 Forces

1.1.7 Solution

The solution assumes that the client itself can provide a web service.

You can use an EventService, which is an intermediate web service dedicated to provide an API for publisher as well as subscribers. When a publisher publishes some data about some particular topic, then the EventService delegates the published data to all subscribers that has subscribed to the topic.


<<WSDL>>
interface
Subscriber_
+update(topic:String,data:Object):void
*
*
<<WSDL>>
interface
EventService_
+addSubscriber(topic:String,clientURL:String):void
+removeSubscriber(topic:String,clientURL:String):void
+publish(topic:String,data:Object):void
PublisherImpl_
installed at "clientURL"
<<Client Web Service>>
SubscriberImpl
How ?
Client _
<<Web Service>>
EventServiceImpl
Problem: How can the method "SubscriberImpl.update" notify the class "Client" if it is executing within another process at the client machine ?
Solution: "Pysical Tiers pattern"


Client
<<Client Web Service>>
SubscriberImpl
<<Web Service>>
EventServiceImpl
<<Might be a WS>>
PublisherImpl
addSubscriber(topic, clientURL)
publish(topic, data)
*[for each subscriber] <super>
update(topic, data)
getMoreDetails if Publisher is a Web service - OPTIONAL in the pattern
"Pysical Tiers pattern" ?

[the occurrence of "<super>" in the diagram is not intentional and will be removed in the final version]

Client – An object that may execute in a stand-alone application, i.e. does not necessarily execute in the same computer process as SubscriberImpl that may be executing in a web service engine.

SubscriberImpl – A web service provided by the client that implements the WSDL interface for subscribers. The URL for this web service is supplied by the client to the EventService as a parameter in the “addSubscriber” method. When the update method is invoked by the EventService this object will have to send the notification back to the client object that wanted the notification and if that client object executes in another computer process within the client machine the “Physical Tiers pattern” may be necessary to use for the interprocess communication. However, the objects might actually execute within the same process, for example if the “Faux implementation pattern” is used. After being notified the Subscriber may want to get more details from the publisher if it provides a web service, which it not always does. The diagram does not illustrate the “SubscriberProxy” that will be generated by the “Subscriber” WSDL and execute at the server communicating with “SubscriberImpl” by sending SOAP messages.

EventServiceImpl – A web service implementing the EventService WSDL interface. It maintains lists of web service URL’s for subscribers that have registered for subscription to particular topics. When a publisher sends a publish message to this EventService it will forward the data to all subscribers as illustrated in the iteration in the sequence diagram. The diagram does not illustrate the “EventServiceProxy” that will be generated by the “EventService” WSDL and execute at the client as well as at the publisher and communicate with “EventServiceImpl” by sending SOAP messages.

PublisherImpl – A publisher of events or topics that sends new messages to the event service with information about the topic as well as the published data itself. Note that this object does not necessarily have to provide a web service itself, but it might do so to provide more information to the client, which is illustrated in the sequence diagram as an optional message.

1.1.8 Consequences

With this pattern, the client can avoid having to subscribe and unsubscribe to the same topics from different web service providers, e.g. news service providers with their own web service implementations.

1.1.9 Related patterns

The “Publish/Subscribe pattern” is similar to the “Observer pattern” which is described in the problem section above.

The pattern typically needs to be combined with the “Physical Tiers Pattern” or the “Faux Implementation pattern” since the client object that originated the request may not necessarily be a web service itself that can listen for incoming SOAP notification messages, nor be a POP server listening for incoming SMTP messages.

Paul B. Monday [WSP 03] describes the pattern as an extension to the Observer pattern and a subset of an Event Channel described in OMG’s “Event Service Specification”.