OERA Service Message Requests and Responses

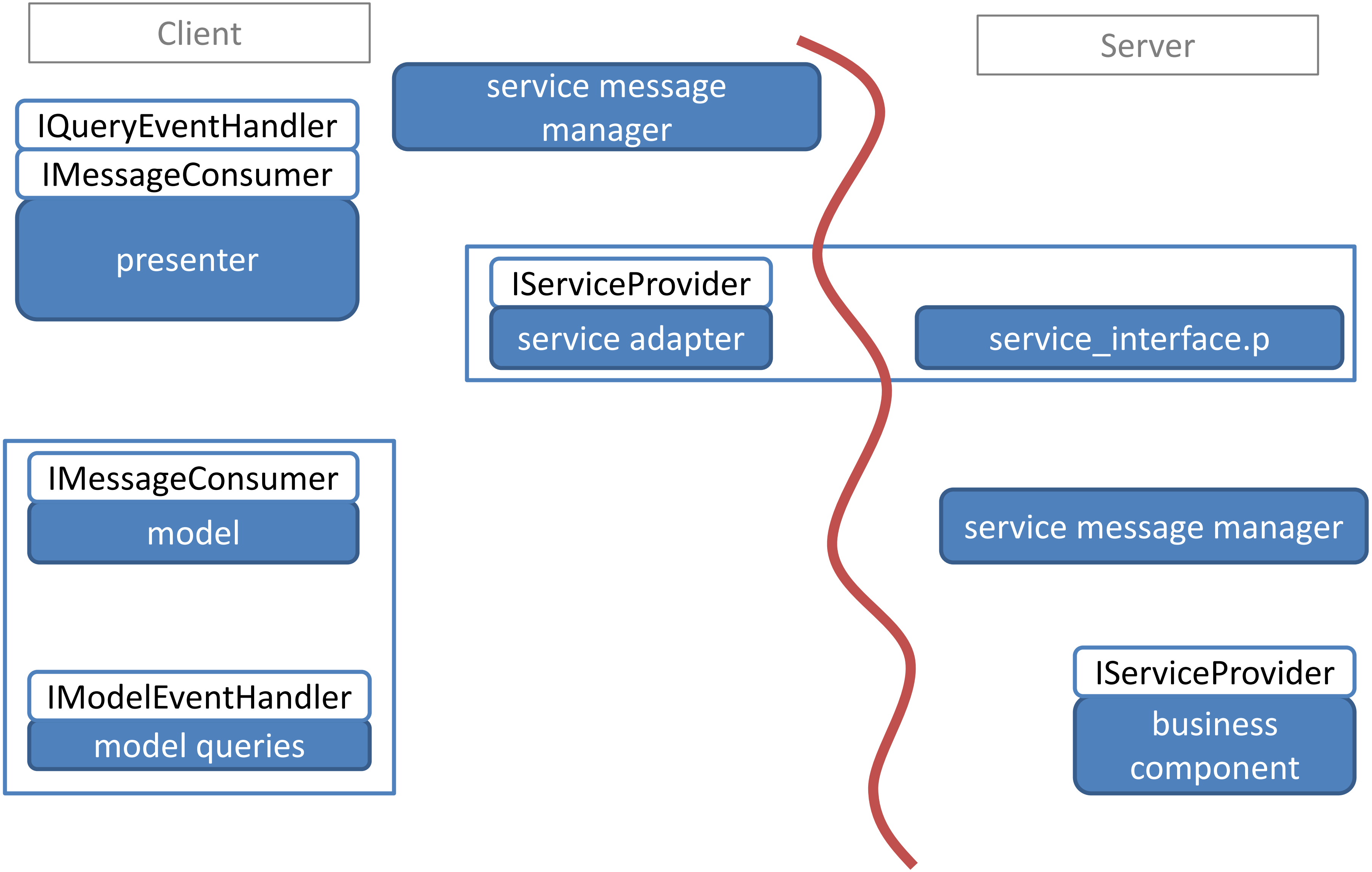
Publish

Run

Interface

service messages, objects

System at rest



Client

Server

IQueryEventHandler

IMessageConsumer

presenter

service message manager

IServiceProvider

service adapter

service_interface.p

IMessageConsumer

model

IModelEventHandler

model queries

service message manager

IServiceProvider

business component

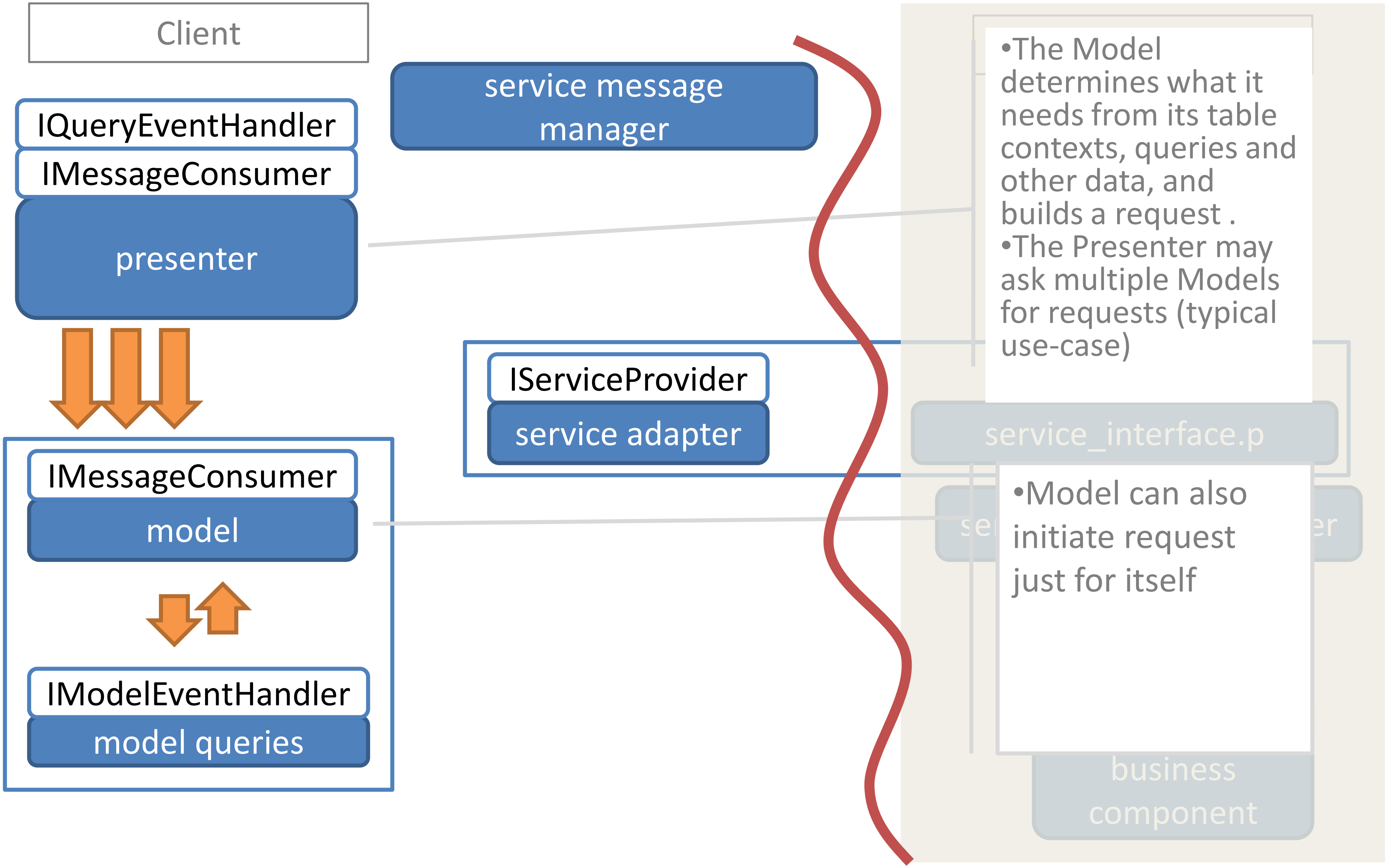
Publish

Run

Interface

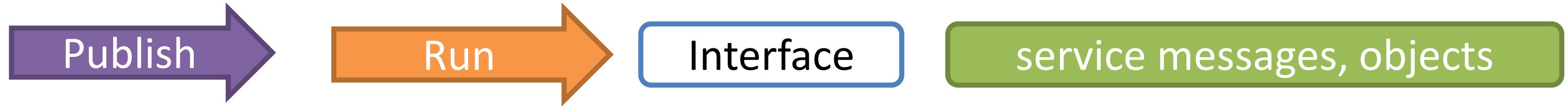
service messages, objects

Presenter makes request

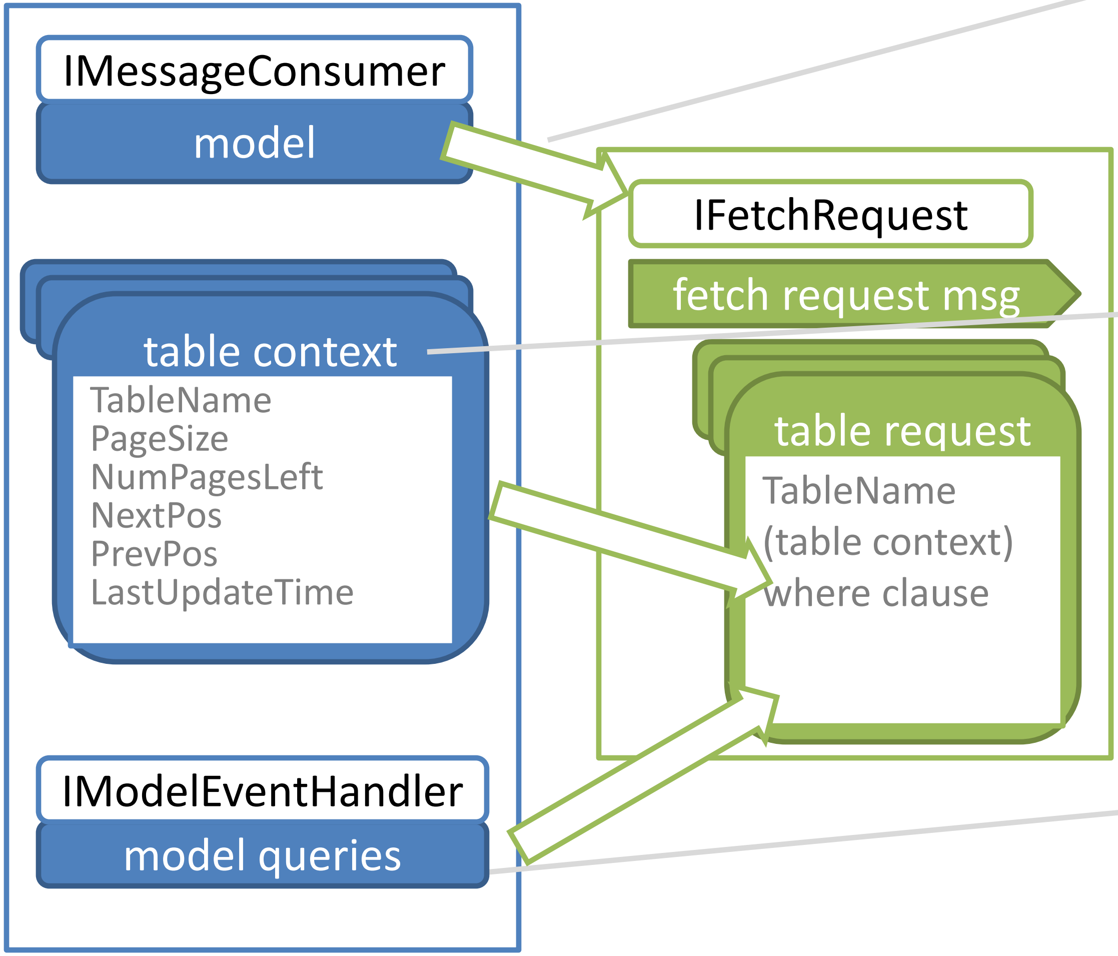


- The Model determines what it needs from its table contexts, queries and other data, and builds a request .
- The Presenter may ask multiple Models for requests (typical use-case)

- Model can also initiate request just for itself



(detail) Model builds a fetch request



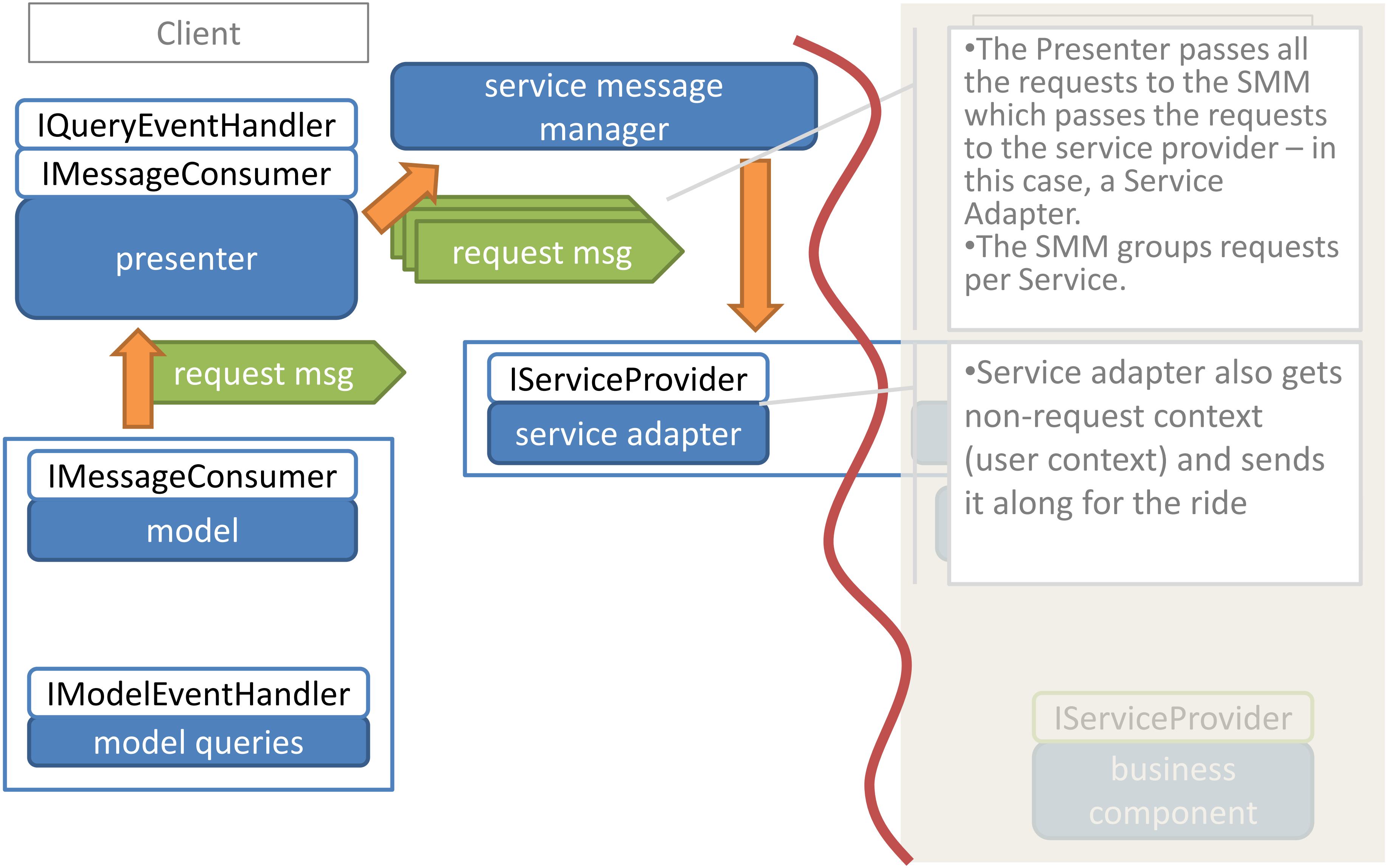
- One request message per Model
- Request message can have multiple table requests

- Each table in the Model has TableContext, which knows what data is in the Model
- TableContexts are updated when response is received
- Useful for paging (aka batching), filtering data
- Used to create the table request, which is fleeting (request lifecycle only)

- Data (eg. off-end) from Model’s queries can also be used to construct table requests (advanced)

Publish Run Interface service messages, objects

SMM bundles & executes requests



- The Presenter passes all the requests to the SMM which passes the requests to the service provider – in this case, a Service Adapter.
- The SMM groups requests per Service.

- Service adapter also gets non-request context (user context) and sends it along for the ride

Publish

Run

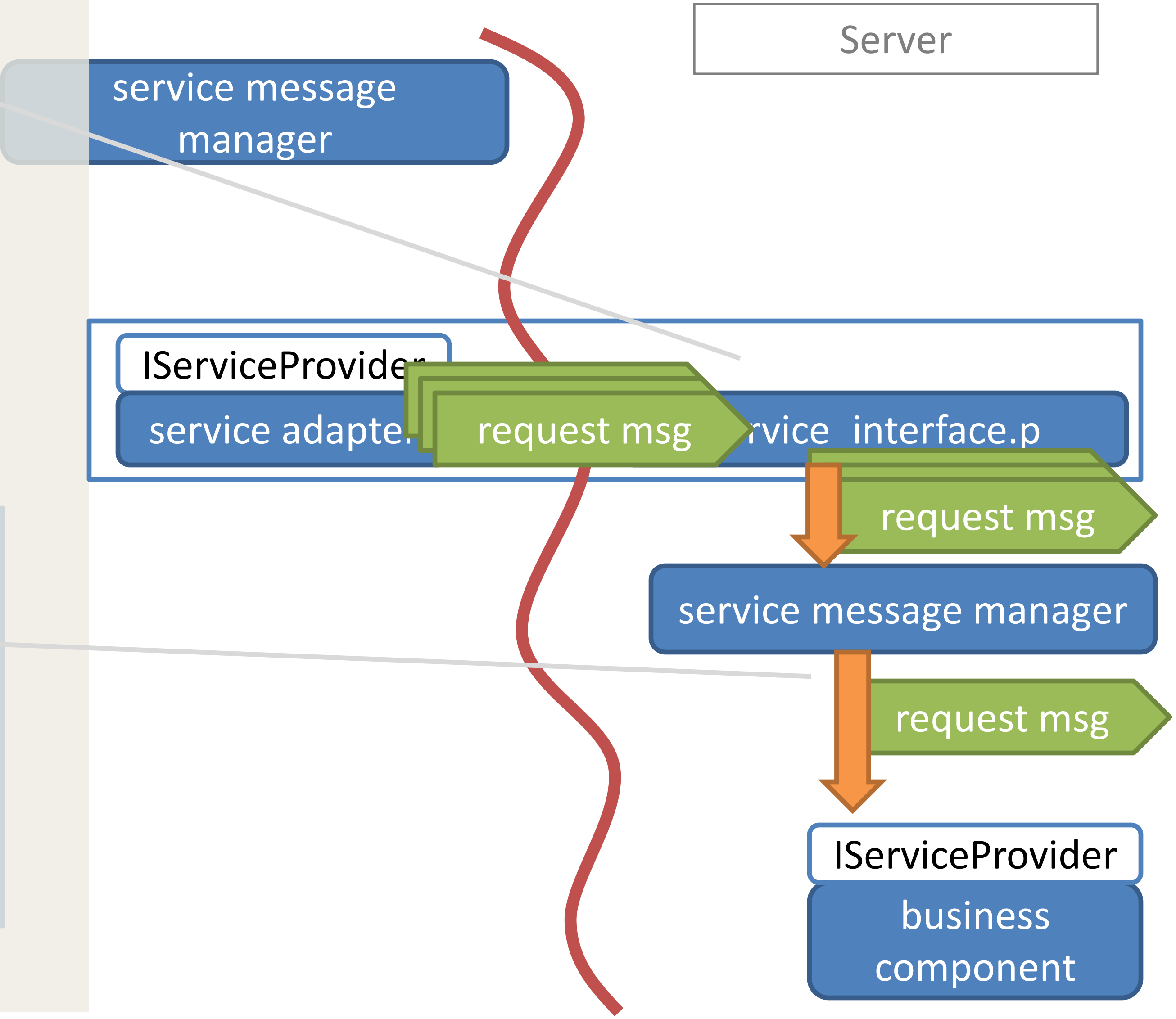
Interface

service messages, objects

Service receives request

- The ServiceAdapter passes the requests across the AppServer
- Each request message is serialized
- All messages are passed as an ABL array of memptrs
- The data is passed as an array of ProDataSets (typically)
- The service interface passes its requests to the server-side SMM.

- While typical service is a Business Entity, we may need to call an external service to complete the request



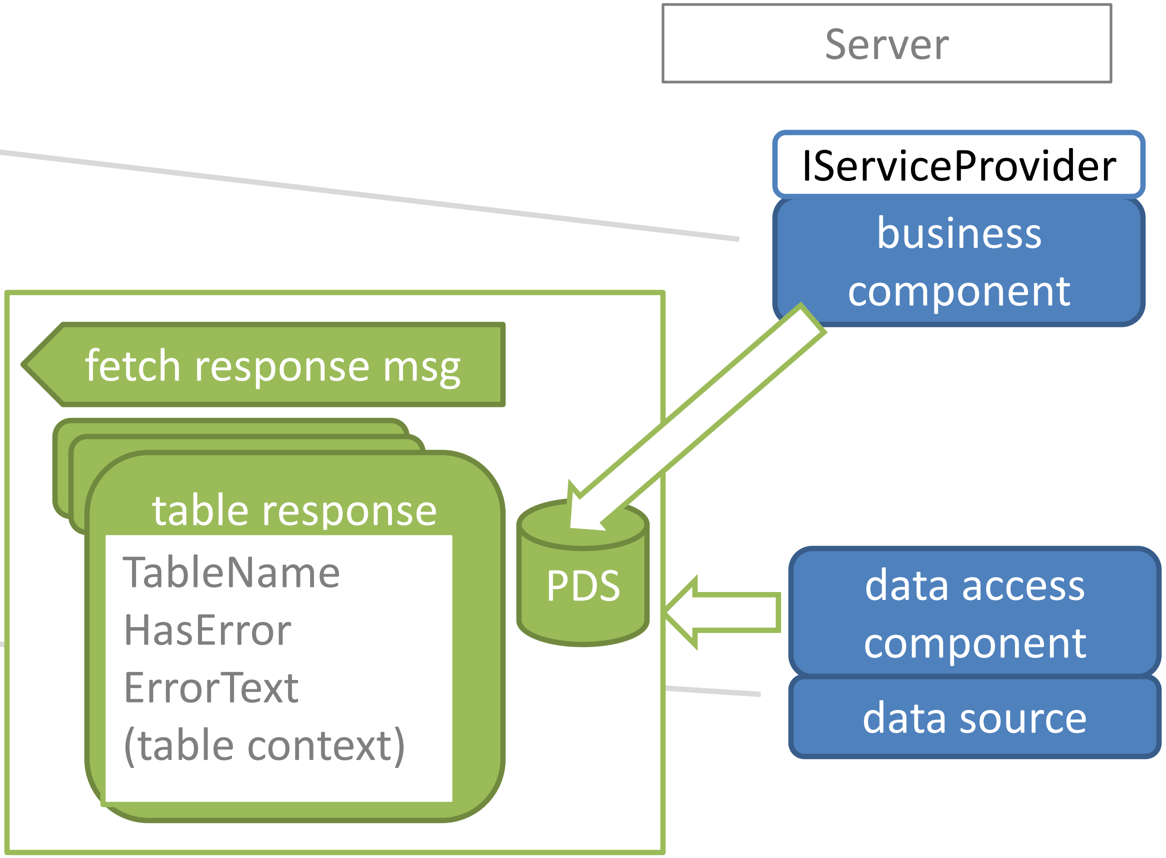
Publish Run Interface service messages, objects

(detail) service builds a fetch response

Client

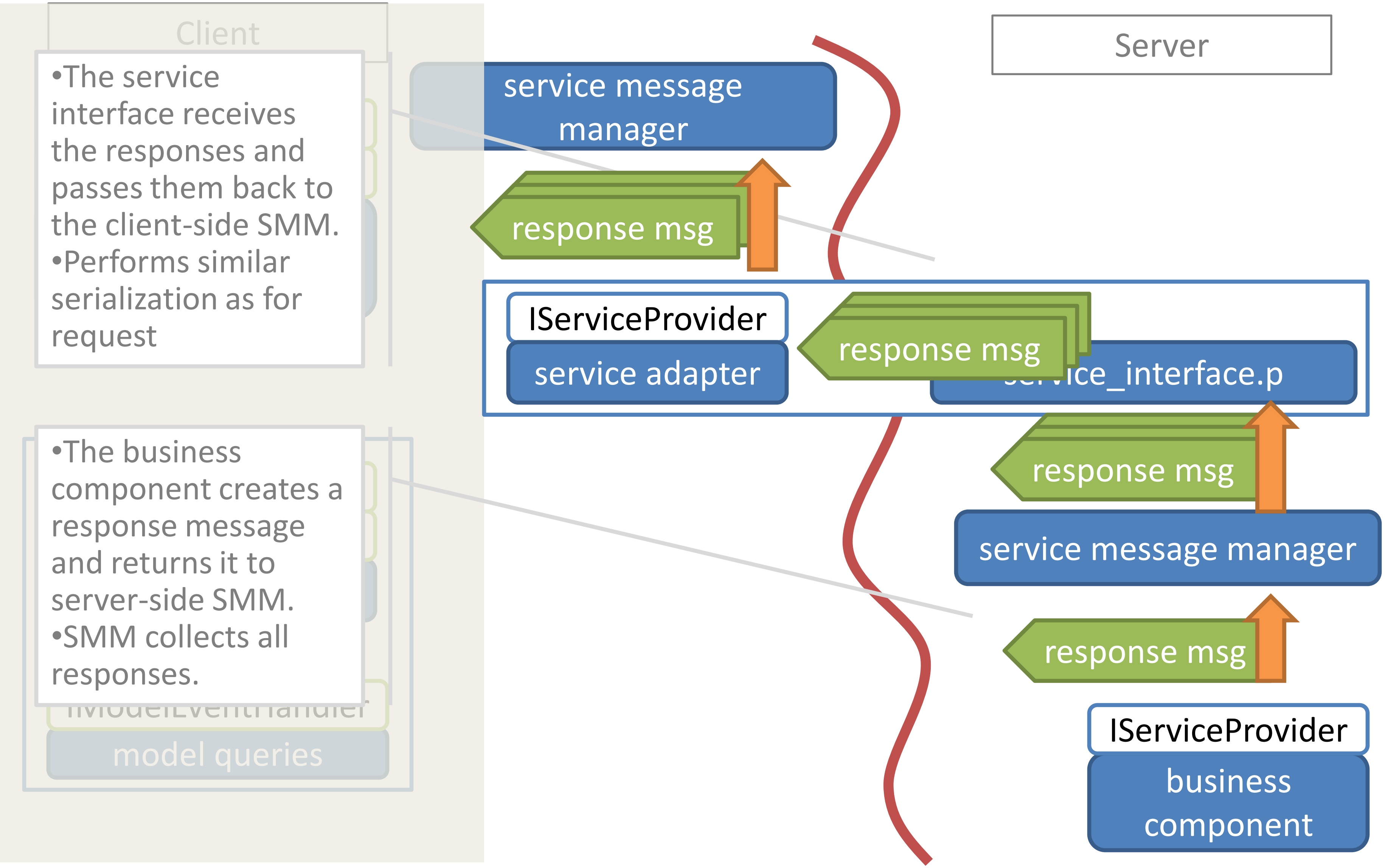
- Response message is created by business component
- Business component pushes the data satisfying the request into the message

- Each table in business entity has data access component, which provides a table response
- TableContext here is same object as that stored in model



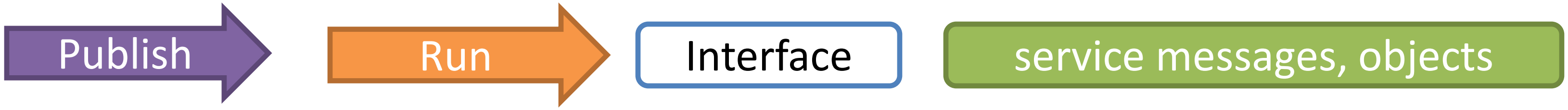
Publish Run Interface service messages, objects

Service sends response

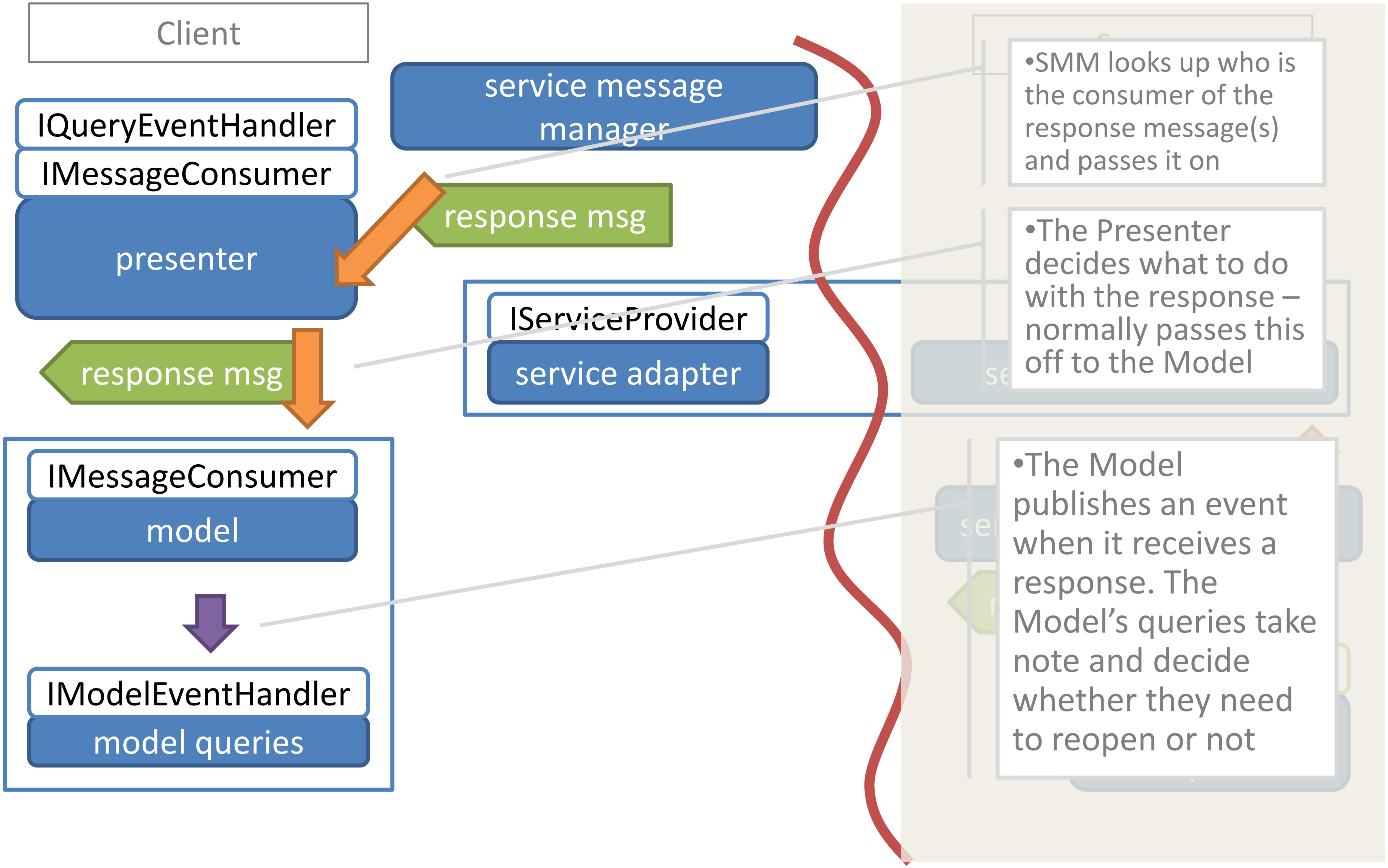


- The service interface receives the responses and passes them back to the client-side SMM.
- Performs similar serialization as for request

- The business component creates a response message and returns it to server-side SMM.
- SMM collects all responses.



Client-side SMM processes responses



- SMM looks up who is the consumer of the response message(s) and passes it on

- The Presenter decides what to do with the response – normally passes this off to the Model

- The Model publishes an event when it receives a response. The Model's queries take note and decide whether they need to reopen or not

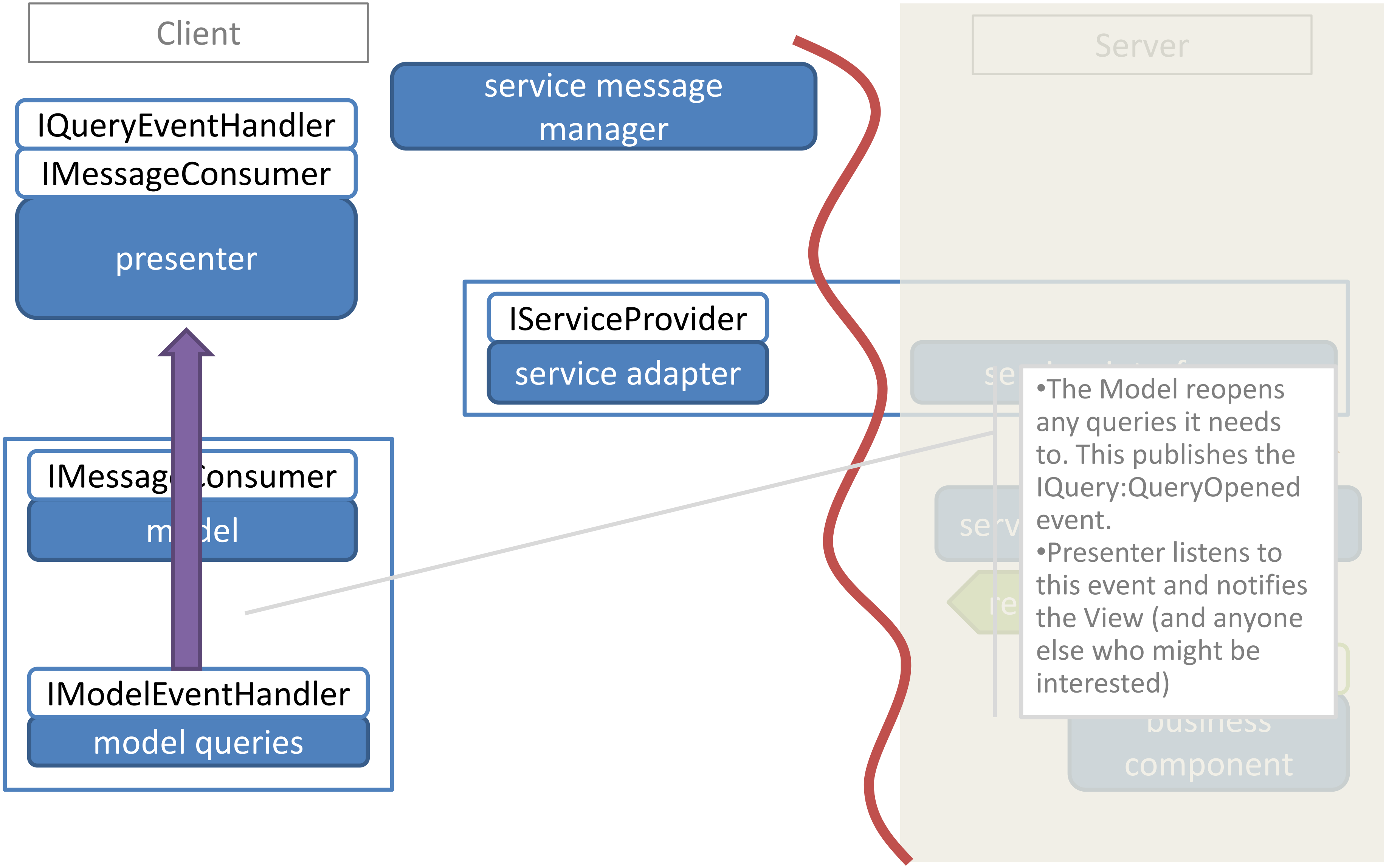
Publish

Run

Interface

service messages, objects

Model queries reopen & notify



•The Model reopens any queries it needs to. This publishes the IQuery:QueryOpened event.

•Presenter listens to this event and notifies the View (and anyone else who might be interested)