Service Component Architecture
Java CAA Specification Version 1.1

SCA-J Issue-127:

Long-Running Request/Response Operations

Proposal

Long-Running Request-Response Operations

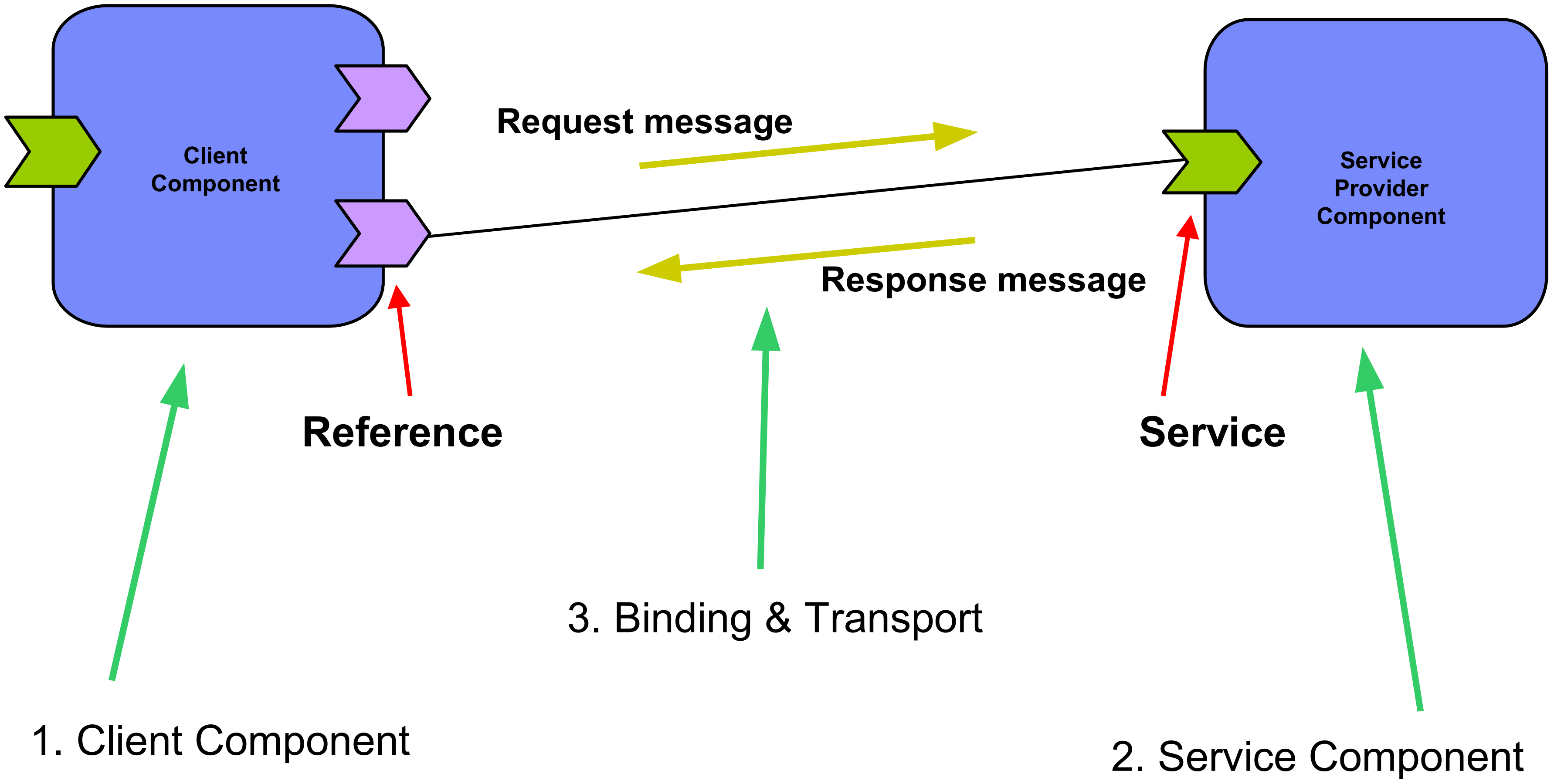
- Assembly-33 issue resolution
  - Definition of "long-running"
  - New intent "asyncInvocation"
  - SCA scenarios
- Reference – JAX-WS 2.0
  - Asynchronous operation mapping for client API

Definition (Long-Running)

- WSDL 1.1 ***request-response operation*** is considered **long-running** if implementation does not guarantee delivery of response within any given time interval

- Clients invoking such request-response operations are strongly discouraged from making assumptions about when the response can be expected
  - i.e. *don't do a synchronous wait*

SCA "Long Running" Service Interaction



Elements of Issue 127 Proposal

- Java CAA spec is concerned with:
  - Client Component model
  - Service Component model

- Not concerned with:
  - Binding & Transport
  - "asyncInvocation" intent drives this
  - concern of *binding implementation*

Outline of this Proposal

- Client model
  - use the JAX-WS async client interface
    - *unmodified*

- Service interface
  - a new async service interface
  - *this is new* - not part of JAX-WS

Long-Running: Client component model

- Client uses JAX-WS async interface for the reference:

```
// synchronous mapping
public interface StockQuote {
   float getPrice(String ticker);
}
```

```
// asynchronous mapping
public interface StockQuote {
   float getPrice(String ticker);
   Response<Float> getPriceAsync(String ticker);
   Future<?> getPriceAsync(String ticker, AsyncHandler<Float>);
}
```

Use either "Async" method when invoking service interface marked with "asyncInvocation"

Client component model: comments

- Same client model for synchronous & for long running services
- Client is "encouraged" to use polling/callback operations for service interface with "asyncInvocation" intent
- ***Binding layer*** takes care of asynchrony
  - based on "asyncInvocation" intent in i/f

Long running: Service component model

- Service uses async method with response handled by ResponseDispatch

```
// synchronous mapping
public interface StockQuote {
   float getPrice(String ticker);
}
```

```
// asynchronous mapping
@Requires("sca:asyncInvocation")
public interface StockQuote {
   void getPriceAsync(String ticker, ResponseDispatch<Float>);
}
```

Service component model: comment

- Use explicit async form of interface
  - marked with "asyncInvocation"

- Invocation passes in ResponseDispatch
  - generated by runtime/binding layer
  - MUST be serializable
  - used to send response message
  - carries full metadata about client

Service component model: comment (2)

- service implementation may call ResponseDispatch *before* or *after* returning from original service method invocation
  - can only call it *once*

Service component model: ResponseDispatch

- Parameterized by Response Bean to hold response message
  - also deals with exception responses

```
public interface ResponseDispatch<T> {
   void sendResponse(T res);
   void sendFault(Throwable e);
   Map<String, Object> getContext();
}
```

Long running: Bindings

- 2 bindings - client side, service side
- React to "asyncInvocation" intent
- Client side:
  - set up separate response message path
  - return to client / send request message
- Service side:
  - introspect service interface for async method
  - set up response message path
  - pass ResponseDispatch on service invocation
  - invoke response path from ResponseDispatch

Example – WSDL Interface

- WSDL 1.1 port type with request-response operation
  (nothing specific here)

```
<portType name="TravelAgencyInterface">
   <operation name="makeReservations">
      <input message="ta:reservationRequest"/>
      <output message="ta:reservationResponse"/>
      <fault name="noFlight" message="ta:noFlightFault"/>
      <fault name="noHotel" message="ta:noHotelFault"/>
   </operation>
</portType>
```

- SCA service with intent (for the service or for an operation)

```
<service name="TravelAgency" requires="sca:asyncInvocation">
   <interface.wsdl portType="ta:TravelAgencyInterface"/>
<service/>
```

Generated Server and Client Interfaces

- Interface used by client

```
@Remotable
public interface TravelAgencyInterface {
   public ReservationResponse makeReservations( ReservationRequest req );
   public Response<ReservationResponse> makeReservationsAsync( ReservationRequest req );
   public Future<?> makeReservationsAsync( ReservationRequest req,
                                           AsyncHandler<ReservationResponse> );
}
```

- Callback interface provided by client

```
@Remotable
public interface MakeReservationsCallbackInterface
       extends AsyncHandler<ReservationResponse> {
   public void handleResponse( Response<ReservationResponse> response );
}
```

- Interface used by Service

```
@Remotable
@AsyncInvocation
public interface TravelAgencyInterface {
   public void makeReservationsAsync( ReservationRequest req,
                                      ResponseDispatch<ReservationResponse> );
}
```

Async Client Implementation

- Handwritten client with callback

```
public class Traveler implements AsyncHandler<ReservationResponse> {

   @Reference
   public TravelAgencyInterface travelAgency;
   private boolean finished = false;

   public void arrangeTrip() {

      ReservationRequest req = new ReservationRequest();
      ...
      travelAgency.makeReservations( req, this );
   }
   public void handleResponse( Response<ReservationResponse> response ) {
      try {
         ReservationResponse resp = response.get(); ...
      }
      catch ( ServiceBusinessException sbe ) {
         Exception e = sbe.getFaultInfo();
         ...
      }
      catch ( ServiceRuntimeException sre ) { ... }
      finally {
         finished = true;
      }
   }

   public boolean isFinished() { return finished; }
}
```

Async Service Implementation (1/2)

- Handwritten service

```
public class TravelAgency implements TravelAgencyInterface {

   private boolean ok = true;
   private boolean noFlightAvailable = false;
   private boolean noHotelAvailable = false;

   // First step of long-running implementation (invoked with callback)
   public void makeReservationsAsync( ReservationRequest req,
                                      ResponseDispatch<ReservationResponse> dispatch ) {
      // Persist callback reference
      cbKey = db.store( dispatch );
      // Do something and trigger next steps of long-running implementation
      // ...typically the Key would be something like an order ID or transaction ID
      // used in the later steps of the process
      ...
      return;
   }

   // Subsequent steps of long-running implementation ...
   //    Reserve flight, reserve hotel, prepare confirmation ...
   //    Perform retry and compensation logic ...
   //    Perform manual intervention if required ...
   //    ...


   (... continued on next page ...)
```

Async Service Implementation (2/2)

- Handwritten service

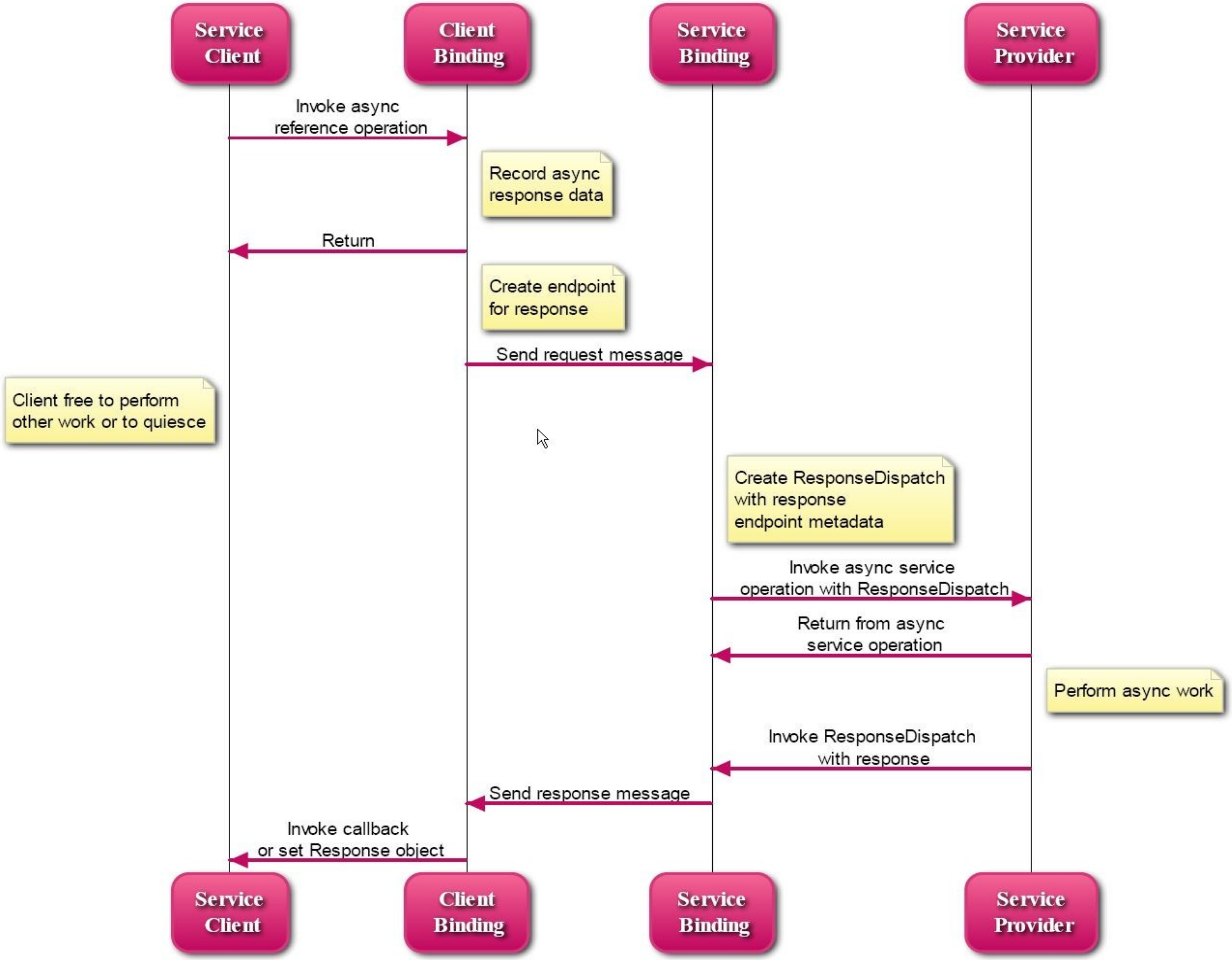
```
(... continuation from previous page ...)

// Last step of long-running implementation (eg kicked off by response to async request)
private void sendResponseToRequester() {

   // Retrieve callback reference from DB
   ResponseDispatch<ReservationResponse> dispatch = db.retrieve( cbKey );
   ReservationReponse response = new ReservationsResponse();

   if (ok){
      ConfirmationData cd = new ConfirmationData();
      response.setConfirmation(cd) ;
      dispatch.sendResponse( response );
   }
   else if (noFlightAvailable){
      NoFlightFault no_flight_fault = new NoFlightFault();
      ServiceBusinessException sbe = new ServiceBusinessException(no_flight_fault);
      dispatch.sendFault(sbe);
   }
   else if (noHotelAvailable){
      NoHotelFault no_hotel_fault = new NoHotelFault();
      ServiceBusinessException sbe = new ServiceBusinessException(no_hotel_fault);
      dispatch.sendFault(sbe);
   }
   else {
      Exception internal_error = null;
      ServiceRuntimeException sre = new ServiceRuntimeException(internal_error);
      dispatch.sendFault(sre);
   }

   return;
}
} // end class TravelAgency
```


Service Client
Client Binding
Service Binding
Service Provider
Invoke async reference operation
Record async response data
Return
Create endpoint for response
Send request message
Client free to perform other work or to quiesce
Create ResponseDispatch with response endpoint metadata
Invoke async service operation with ResponseDispatch
Return from async service operation
Perform async work
Invoke ResponseDispatch with response
Send response message
Invoke callback or set Response object
Service Client
Client Binding
Service Binding
Service Provider

JAX-WS 2.0 – Background

- Asynchronous operation mapping
  - **javax.xml.ws.AsyncHandler**
    - A generic interface that clients implement to receive results in an asynchronous **callback**
  - **javax.xml.ws.Response**
    - A generic interface that is used to group the results of a method invocation with the response context
    - Response provides asynchronous result **polling** capabilities