

# AsyncContinuation

---

## Asynchronous Continuations

We often find that external interactions with the ESB are required to be synchronous but this does not mean that the internal communication has also to be synchronous. In fact, the best performance and reliability occur when the internal services use asynchronous messaging to communicate between each other.

The example attached to this page demonstrates how an incoming (synchronous) request could be handled internally with asynchronous messaging. It demonstrates two different mechanisms which can be used to compose individual ESB services into logical services.

The first service is composed of Service1 and Service1Continuation whereas the second service is composed of Service2 and Service3.

In this scenario, the first service, having being called by a synchronous ESB client, wishes to call the second service in a synchronous manner. As it does not wish to block threads, or prevent other requests from being processed, it uses a continuation mechanism to split the processing.

The first part of this is handled by Service1, which is a OneWay service responsible for handling the incoming request and preparing an outgoing message for continuation.

Preparing the message for continuation involves specifying the continuation service (Service1Continuation) as the ReplyTo of the outgoing message and also including the original, incoming, ReplyTo as an extension of that EPR. Once the message has been initialised, all that remains is to send it to the second service using an asynchronous mechanism (StaticWireTap in this case). Service1 is now free to process more incoming requests, regardless of how long the next service takes to process the request.

The second service is composed of two ESB services, Service2 and Service 3, and demonstrates how a single, logical, service can be created by chaining services together.

Service 2 is a OneWay service that receives the request from Service1, performs some

processing, and then finally sends the message asynchronously to Service3. Service3, being the final service in this chain, is a RequestResponse service which performs some processing in addition to that performed by Service2 before sending its response to the ReplyTo of the message (in this case the Service1Continuation specified in Service1).

The second part of the first service, Service1Continuation, now receives the response from the second service. This is a OneWay service which continues the processing of the message following the response from the second service. Once it has handled the message it then prepares a response for the original caller. By inspecting the incoming To EPR it can extract the original ReplyTo specified by the external caller and send a response to the client asynchronously.

All communications between the internal services are asynchronous, can take advantage of the retry mechanism, and **do not** cause any threads to block while waiting for a response.

-

---

The attached example will work on the SOA Platform and the current trunk. It does not work on the 4.2.1GA release of the ESB project.

The example can be executed by unpacking into the quickstart directory, changing into the async\_continuation directory, and then executing ant deploy followed by ant runtest