Supporting Legacy Clients

At this point we hope we've demonstrated ASP.NET Web API as an excellent platform on which to implement REST-based services. However, what if you found yourself in a situation where you had to support existing "legacy" (SOAP-based) clients? Would that automatically eliminate ASP.NET Web API as a technology choice? Fortunately, the answer is no. And that leads us to the subject of the current chapter; namely, how to simulteneously support legacy clients and extend client reach using ASP.NET Web API.

# The Situation

Recently, we were given the job of developing a REST-based Web API for an existing system. There were a couple of interesting requirements for this project:

1. The new REST-based API and the legacy SOAP-based API must be packaged together in a single application.
2. Existing customers must be able to benefit from the new features of the new application without affecting any external integration points.

An architectural overview of this is depicted in the following figures. Note that though the system didn't have anything to do with task-management, we will use our task-management service for illustration:



Figure 8-1. Architectural Overview - Current State

In the current state (Figure 8-1), we see the legacy service integrating with various clients via the SOAP messaging protocol. The target architecture (Figure 8-2), however, shows the task-management service supporting existing clients and new clients by offering a choice of messaging protocols and formats.



Figure 8-2. Architectural Overview - Target State

We'll begin this chapter by demonstrating how to support various message formats (i.e., XML and JSON) for our REST-based API. Then we will demonstrate how to support the existing SOAP-based clients.

# Content Negotiation

# SOAP