# Foreword

I started working on the Windows Communication Foundation in 2001 (then known as "Indigo"), when we were a small team—I was perhaps the 20th person to join. In my tenure on the team, I served as lead program manager for storage, manageability, reliable messaging, and queuing. The team had a great vision of facilitating the next generation of Web services by creating a foundation for Web services that could be practically applied to a breadth of distributing computing problems. We wanted to ensure that Web services could be implemented for businesses that enable secure communication—confidentiality, signing, federation—so distributed computing customers could use Web services for communication in the real world. We wanted to make sure that Web services could participate in transactions in the ACID model, ensuring that practical interactions with data-driven systems, or with which "all or nothing" computing is essential. We wanted to ensure that Web services could be written in a way that lossiness of the WAN didn't inhibit development of meaningful distributed applications—where you could actually count on messages getting where you sent them, and in the order you sent them. In retrospect, these "basic plumbing" goals seem almost quaint, but in 2001 we accepted, for the most part, the need to "do it ourselves" when building distributed systems.

We also understood that most computing environments are heterogeneous, with systems from many vendors coexisting with each other, so we wanted to ensure interoperability through great technical web service standards. We made a deep commitment to interoperability, and executed relentlessly on that commitment. WS-Security, WS-AtomicTransactions, WS-ReliableMessaging, WS-Management, WS-Policy, WS-Transfer, WS-Eventing and others were essential for broad interoperability at the "basic plumbing" level—but none existed when we kicked off this project (all were made possible in greater or lesser part by colleagues on the WCF team). Looking back, we might say "Of course we want to work with other systems with broadly accepted composable Web service standards"—but again, in 2001 this was a lofty goal.

We wanted to support a single programming model so developers wouldn't face a new learning curve if they wanted to shift from message-oriented to remote procedural paradigms, or to go from TCP to HTTP or queuing protocols such as MSMQ. In the face of .Net Remoting, ASMX, Sockets, MSMQ, and other programming models, a unified API to accomplish what any of its predecessors could seemed difficult—we were trying to forge one from many. We wanted to support extensibility, so that each new message exchange pattern, protocol, or encryption mechanism wouldn't force yet-another programming paradigm.

For my part as a lead program manager, I helped champion manageability—the idea that anything that should be left to the IT pro (protocol du jour, encryption mechanisms, service addresses, monitoring aspects, and so on), could be. Here again we had lofty goals—we wanted best-of-breed tracing, intrinsic monitoring and control of applications built with WCF, ease of use through great configuration and tracing tools, and integration

with all the Windows management assets through WMI. The goal was simply that applications built with WCF would be intrinsically more manageable with less effort than those built on other frameworks.

And perhaps most ambitiously, we wanted it to be *easy* and *fun* to build great distributed applications for use in the real world. We wanted it to intuitively lead developers to build applications that followed the best practices of distributed systems. As Steve Swartz, one of the greatest champions of "easy and fun" told me, our goal was to build a framework such that "if you set a ball at the top of a hill and let it roll down, it would just naturally stop in a place where you had a well factored Service that helped you avoid all the mistakes distributed systems developers have made in the last 20 years."

So how did we do? Looking at the finished product that shipped as part of .Net 3.0 in Vista and on the Web, I think we actually did pretty darned well. WCF is a unified, extensible framework that does in fact help you build secure, reliable, interoperable, manageable real world distributed applications in a unified framework that is, by gum, actually fun (well, at least for those of us who enjoy programming). It took us six years, but we hit all our key goals. In fact, I like the product so well that my NEW job is focused on building new products for Microsoft to sell that bet entirely on the facilities that WCF provides (and I'm having a blast). This book is on every developer and PM's shelf in the team—it is the "go to" reference for those of us who either provide or consume Web services in the product—which is basically all of us (including several developers and PMs who actually helped build WCF!).

A few words about Craig. Craig and I met while he was serving as the technical evangelist for WCF. His energy and enthusiasm for the product was infectious—he was a great champion of the vision. About 90% of the time, when a question came up about "Can we support that scenario?" Craig would chime in with "Oh yeah, I did that last week—here's the prototype." In his role, he was able to see the "gestalt" of the product in a way that those of us who were heads down on specific features didn't. His candid feedback, technical depth, and enthusiasm were instrumental in making WCF what it is today. I believe his comprehensive knowledge of and enthusiasm for WCF shines through every chapter— I am sure you'll find the book as enjoyable, enlightening, and plain useful as we do here on our team.

—Alex Weinert

Group Program Manager, Microsoft Corporation