Joule: Distributed Application Foundations

300 Third Street
Los Altos, CA 94022
ph 415 941 8224
800 54 JOULE
fax 415 941 8225

Agorics Technical Report ADd003.4P

The Joule system is a foundation for building distributed applications. It combines many mechanisms already built and tested in existing products and systems. To encourage widespread acceptance and use of Joule, Agorics expects to release a public license implementation of Joule. Agorics also plans additional Joule development, to support the system as it grows, and to apply the ideas to other platforms.

Introduction

This is the technical manual for the Joule programming language. It is intended to familiarize the reader with the concepts underlying Joule, with Joule syntax, and with the fundamentals of a Joule programming environment. When you have finished reading this book, you should be able to read and create simple Joule programs.

The core of Joule is a new computational model for building distributed systems. Many of the ideas are distilled from existing systems, and could be applied at the language level, at the operating system level, or (as in CORBA) as extensions to existing languages. This manual describes the Joule programming language, a pure realization of these ideas that remains portable across all platforms (where an operating system would not). The Joule language is intended as a foundation for distributed systems, providing support in the language for many of the abstractions needed for network- or multiprocessor-based applications. Heretofore, it has been necessary to "reinvent the wheel" in many instances—to reimplement familiar techniques, tailoring them to the current special case. The goal of Joule is to provide the functionality required for distributed computing, in a straightforward and secure environment.

Chapter 1, *Foundations,* describes the intellectual origins of Joule and outlines the basic ideas on which the language is based.

Chapter 2, *Introductory Examples,* leads the reader through four simple Joule programs—the familiar factorial and compound-interest functions, plus two other servers that demonstrate some of the unique qualities of Joule.

Chapter 3, *Simple Execution Model,* describes the rules that all Joule computations must follow, and is intended to give the reader an intuition of how Joule computations could actually get work done; it is not intended to represent an efficient execution model.

Chapter 4, *Syntax,* presents an informal syntax for Joule. (For a formal syntax, see Appendix B.) Syntactic abstraction—the set of techniques for extending the Joule syntax—is discussed but not specified in this document.

Chapter 5, *Language Definition,* describes the present state of the Joule language design. It describes the computational primitives, along with typical techniques of their use, and provides a description of the syntac-

tic forms built from those primitives to directly support routine programming tasks.

Chapter 6, *Hierarchical Accounts Example,* presents a more complex Joule program, a hierarchical bank account. Hierarchical bank accounts provide a necessary component of *agoric resource management*—the use of market mechanisms to control allocation of computational resources like CPU time and network bandwidth (described more fully in Chapter 9). The program, and the underlying Joule concepts, are explained in detail.

Chapter 7, *Boundary Foundations,* describes the low-level foundations that support boundaries for creation and initiation of new programs in a running system, termination and resource management for existing programs, and access to foreign services. These foundations provide the mechanism on which the policies described in Section 5.10, *Module Programming,* are built.

Chapter 8, *Security*, introduces Joule's security foundations, many of which were drawn from or inspired by KeyKOS, a capability-based operating system, and by public-key security principles.

Chapter 9, *Resource Management,* describes managing resources in Joule. It first describes some underlying principles for resource management abstractions. It then describes facilities for resource encapsulation and transfer, the foundations for resource management. Finally, it describes market-based resource management mechanisms for making resource trade-offs in complex systems.

Chapter 10, *Distribution,* explores the issues affecting distributed systems and describes how Joule deals with them. This chapter describes support for the full spectrum of distribution regimes, from automatic distribution in which processes are automatically spread across multiple processors, to explicit distribution in which the programmer controls or influences the mapping from processes to processors, to untrusting distribution in which the programmer explicitly manages and adapts to trust boundaries and failure properties of the network.

Chapter 11, *Persistence,* describes possible implementations of persistence in Joule. The trade-offs between these implementations remain largely unexplored for Joule, though much of the territory is known for other related systems such as FCP, Actors, and KeyKOS.

Appendix A, *Language Comparison,* reviews other languages and systems relative to the requirements for robust servers and open distributed systems. It also compares the capabilities of Joule with those of its antecedents, Actors and concurrent constraint languages.

Appendix B, *BNF for Joule Syntax,* gives a description of the Joule syntax in Backus-Naur form.

The Energetic Secrets material appears in an appendix because it has not yet been integrated into the rest of the manual.

Appendix C, *Optional Arguments,* presents a proposal for managing optional arguments and "rest" arguments in messages.

Appendix D, *Energetic Secrets,* describes how `SealedEnvelope`s will replace Tuples in the Joule communication model, incorporating public-key semantics into the communication foundations.

Appendix E provides a bibliography of articles and books that influenced the design of the Joule programming language or that present background information on various aspects of the Joule design.

The Joule language is a work in progress, and pieces of this design will change as more experience with the syntax and computational model is gained. This book too is a work in progress; many sections remain unfinished. Some of the unfinished sections require incorporation of already developed techniques (such as the Security sections), others require significant design work (such as Agoric Resource Management).

Many thanks to the people who helped make this document and the technology behind it possible.