7. Boundary Foundations

Modules are built on 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 the Module Programming section are built. By separating policy from mechanism, we enable multiple programmer-level module systems to co-exist.

7.1. Domains

Domains are the foundational primitive for separately-executable pieces of code. They represent the boundaries needed for modules, security, and resource management. This section will describe them in detail.

7.2. Initiation

7.2.1. Necessity of Initiation

Initiation is the ability to start newly generated programs and connect them into an already-running system. This is a fundamental requirement for open systems.

7.2.2. Layers of Initiators

Programs can be initiated at many levels of abstraction. Machine code programs, Joule abstract machine programs, and Joule parse trees are all program representations that could be initiated. Initiators at each of these levels can be built on the initiator for the next level down.

7.3. Export/Import Issues

This section will describe how an initiated process gets properly connected to the rest of the universe of services.

7.4. Debugging Issues

Each separate domain is debugged independently. The typical model of systems that provide general debugger access violate encapsulation in a distributed system with untrusted clients. This section will describe

how debuggers are implemented while maintaining modularity and respecting trust boundaries.

7.5. Interoperability

Domains are the boundaries at which Joule communicates with foreign services (services written in other languages). To Joule, a foreign service looks like an independent Domain with which Joule engages in message communication. The Joule semantics could actually manage entire populations of external programs as if Joule were an operating system.