R6RS Status Report

A brief history of Scheme
Guiding principles
Language overview
Design process
Public review
Brief History

1975 First report on Scheme (Sussman/Steele)

1978 Revised report (Steele/Sussman)

1985 R2RS (ed. Clinger)

1986 R3RS (ed. Rees & Clinger)

1991 R4RS (ed. Clinger & Rees)

1998 R5RS (ed. Kelsey, Clinger, & Rees)
Brief History: R6RS

1998 Scheme workshop: R6RS discussion
• records, Unicode, exceptions, other items
• nothing formally settled
• SRFI process proposed

2002 Scheme workshop: Strategy committee
   Alan Bawden, William Clinger, Kent Dybvig,
   Matthew Flatt, Richard Kelsey,
   Manuel Serrano, Michael Sperber

2003 Scheme workshop: Charter ratified
Brief History: R6RS

Steering committee:
   Bawden, Guy L. Steele, Mitchel Wand

Editors committee:
   Marc Feeley (Editor in chief), Clinger, Dybvig, Flatt, Kelsey, Manuel Serrano, Sperber

Kelsey resigned April 2005
Replaced by Anton van Straaten

Feeley and Serrano resigned January 2006
Committee left at five editors
Sperber appointed project editor, Dybvig chair
Guiding Principles

Like R5RS:

- Power from simplicity, composability
- Procedural, syntactic abstraction
- S-expression representations
- Control via procedures and continuations
- Proper tail calls, garbage collection
- Useful for Education and language research
Guiding Principles

Additionally:

- Distribute portable programs, libraries
- Unique types, fully general macros
- Run-time type and bounds checking (with option to declare undesirable)
- Allow efficient code from portable source
Guiding Principles

Tests for inclusion:

- Provide building blocks
- Include commonly used user-level features
- Exclude other features
Guiding Principles

Backward compatibility:

Maintain backward compatibility . . .

. . . without compromising principles, viability

More Scheme programs in future than past
Language Overview

(Portable R6RS) Scheme Programs:

Executable scripts

Libraries used by scripts and other libraries

Expressions used via `eval`

No top-level variables, definitions
(library library-name
  (export export-spec ...)
  (import import-spec ...)
  library-body)

Example:

(library frob
  (export make-frob frob-print)
  (import r6rs)
  (define make-frob —)
  (define frob-print —))
Scripts

#!/usr/bin/env scheme-script
(import import-spec ...)
script-body

Example:

#!/usr/bin/env scheme-script
(import r6rs)
(display "hello world\n")
Syntactic Changes

Identifiers are Case Sensitive

Identifiers may start with \( \rightarrow \)

Matched \([\text{brackets}]\) may replace \((\text{parens})\)

Block \#\| ignore this \# comments

Datum \#; (ignore this) comments

\#!r5rs and other \#!id

\#!r6rs may precede standard R6RS syntax

other \#!id must precede nonstandard syntax
Standard Libraries

Base library:
   (r6rs base)

Additional libraries:
   (r6rs unicode)
   (r6rs bytes)
   (r6rs lists)

Composite library:
   (r6rs)
Base Library

Corresponds roughly to R5RS
I/O and file operations removed
some list, string operations removed
R5RS environments removed
some R5RS optional features removed
declarations added
error and violation procedures added
full numeric tower required
Unicode Library

Characters and strings hold Unicode data

Library (r6rs unicode) provides

- category operators
- normalization operators
- comparison operators
- case-mapping operators
Bytes Library

Bytes objects replace strings for binary data

Library (r6rs bytes) provides

- bytes object creation operators
- 8-, 16-, 32-, 64-bit accessors/setters
- IEEE float accessors/setters
- various other operators
Lists Library

Library (r6rs lists) provides

a few R5RS list operators (memq, assq, etc.)
fold operators
find and partition operators
predicate-mapping operators
Records Libraries

The record libraries allow

- new record type creation
- constructor, accessor, etc., creation
- record inspection

Four records libraries:

- (r6rs records procedural)
- (r6rs records explicit)
- (r6rs records implicit)
- (r6rs records inspection)
Exceptions/conditions Libraries

Where R5RS said “it is an error” . . .

. . . R6RS often requires exception be raised

. . . with specific condition type

Improves predictability, portability, recovery
Exceptions/conditions Libraries

Library (r6rs exceptions) provides

   exception raising, catching operators

Library (r6rs conditions) provides

   condition-type creation, creation operators
   condition creation, access operators
   various condition types
   exception raising for specific conditions
I/O Libraries

(r6rs i/o primitive)
primitive, unbuffered, and custom I/O

(r6rs i/o ports)
port-level I/O

(r6rs i/o simple)
R5RS compatible textual I/O
Arithmetic libraries

(r6rs arithmetic fixnum)
fixed-precision exact integer ops (wrap on overflow)

(r6rs arithmetic fx)
fixed-precision exact integer ops (exception on overflow)

(r6rs arithmetic flonum)
float-only operations

(r6rs arithmetic exact)
exact-only operations

(r6rs arithmetic inexact)
inexact-only operations
Library (r6rs syntax-case) provides

- syntax-case transformers
- associated operators
- quasisyntax, unsyntax, unsyntax-splicing
- syntax-violation exception-raising operator
Hash-Tables Library

Library (r6rs hash-tables) provides

- `eq?` and `eqv?` hash table creation operators
- General hash table creation operator
- Equal, string, and symbol hash functions
Enum Library

Library (r6rs enum) provides

- enumeration creation operator
- enumeration definition syntax
- enumeration manipulation operators

More efficient, robust than sets of symbols

Used in the I/O libraries
Miscellaneous Libraries

Library (r6rs when-unless) provides alternatives to one-armed if

Library (r6rs case-lambda) provides creating procedures with multiple interfaces

Library (r6rs promises) provides R5RS delay and force

Library (r6rs scripts) provides access to command-line arguments
Mutable Pairs Library

Library (r6rs mutable-pairs) provides

\[ \text{set-car! and set-cdr!} \]

Isolation may help whole-program compilation
Eval Library

Library (r6rs eval) provides

- eval procedure, as in R5RS
- environment-creation procedure
  (forms environment from specified libraries)
R5RS compatibility Library

Library (r6rs r5rs) provides

- exact→inexact and inexact→exact
  (new names are →inexact, →exact)

- quotient, remainder, modulo
  (replaced by div, mod operators)

- scheme-report-environment, null-environment
Some optional R5RS features completely gone:

- transcript-on, transcript-off
- interaction-environment
- top-level definitions (outside libraries, scripts)
Composite Library

Library (r6rs) provides

   Everything from all standard libraries except:

   (r6rs mutable-pairs)

   (r6rs eval)

   (r6rs records explicit)

   (r6rs r5rs)

*One library to rule them all ...*

*... and in the darkness bind them*
Design Process

Email—mixed blessing
Face-to-face meetings
SRFIs
Discord and change
End game: Feb–Aug 2006
End Game

September 1 deadline
Articulated principles
Cumulative status reports
High tech communication
Focused effort
End Game

September 1 deadline
Articulated principles
Cumulative status reports
High tech communication
Focused effort
Public Review

First part of our job is done—now it’s your turn

Six month review period—through mid March 2007

Draft posted at http://www.r6rs.org

Discussion list: discuss@r6rs.org
must be subscribed to post (link at www.r6rs.org)

Formal comments: formal-comment@r6rs.org
Formal Comment Requirements

Must be submitted to formal-comment@r6rs.org

Comments accepted only from discuss@r6rs subscribers

Submissions must include

• name, email address
• type of issue (defect, enhancement, simplification)
• priority (critical, major, minor, trivial)
• R6RS component (base, arithmetic, etc.)
• report version (2.91, 2.92, etc.)
• one-sentence summary
• full description

Details: http://www.r6rs.org/process.html
Reference Implementations

Implementations of major subsystems

library and syntax-case, arithmetic, records, Unicode library, Unicode reader, exceptions & conditions, bytes, I/O, hash tables, list library, enumerations

Intended to be portable, reasonably efficient

Currently preliminary

See http://www.r6rs.org/refimpl/

No unfunded mandates here