message-digest egg

Message Digest Support
Extension for Chicken Scheme
Version 1.5

Kon Lovett

Table of Contents

1 About this egg

1.1 Version history

`1.5` Exports

`1.4` Removed ->integer

`1.3` Added name field to primitive record

`1.2` Added Auxiallary Procedures

`1.1` Added un-managed context object support

`1.0` Initial release

1.2 Usage

Load this egg like so:

```
(require-extension message-digest)
```

2 Documentation

Message Digest provides support for message digest primitives. A message-digest is a function taking arbitrary-length input and returning a fixed-length hash.

`make-binary-message-digest` [procedure]

```
(make-binary-message-digest OBJECT CONTEXT-INFO DIGEST-LENGTH INIT UPDATE FINAL [
```

Returns the message-digest for `OBJECT` as a binary string. `CALLER` is the symbol for the calling context.

Acceptable objects are string, input-port, byte-vector, or anything that can be converted into a byte-vector. Lists, vectors, and homogeneous-vectors can be converted. Lists and vectors must only be composed of characters, fixnums, or booleans. A boolean is converted as #f = 0 and #t = 1. An input-port is not closed, but is read to end-of-file.

The arguments `CONTEXT-INFO DIGEST-LENGTH INIT UPDATE FINAL` are the components of a generic message-digest primitive. The return value of the phase procedures is ignored.

`INIT` [procedure]

```
(INIT CONTEXT)
```

Initialization phase procedure. Usually sets up the `CONTEXT`.

`UPDATE` [procedure]

```
(UPDATE CONTEXT BYTES COUNT)
```

Accumulation phase procedure. Must accumulate the `COUNT BYTES`. Will be called one or more times.

`FINAL` [procedure]

```
(FINAL CONTEXT RESULT)
```

Finalization phase procedure. Must build the resulting message-digest in the supplied `RESULT` string.

`CONTEXT-INFO`
: When a fixnum, the size in bytes of the memory block to contain the state between the phases. When a procedure, it returns the context object.

`DIGEST-LENGTH`
: The fixnum count of bytes in the result string.

The optional `CALLER` should be the name of the calling procedure.

`make-message-digest` [procedure]

```
(make-message-digest OBJECT CONTEXT-INFO DIGEST-LENGTH INIT UPDATE FINAL [CALLER]
```

Exactly as above but returns the message-digest for `OBJECT` as a hexadecimal encoded string of length 2 * `DIGEST-LENGTH`.

`message-digest-primitive` [record]

```
(message-digest-primitive context-info digest-length init update final name)
```

The meaning of the fields are exactly as above. `name` is an optional string or symbol for identification. The usual generated record procedures are available, except '*-set!'. An immutable object.

`message-digest-primitive-apply` [procedure]

```
(message-digest-primitive-apply MESSAGE-DIGEST-PRIMITIVE OBJECT [CALLER])
```

Returns a binary-message-digest of `OBJECT` using `MESSAGE-DIGEST-PRIMITIVE`.

2.1 Auxillary Procedures

`string->substring-list/shared` [procedure]

```
(string->substring-list/shared STRING CHUNK-SIZE [START [END]])
```

Returns a list of `CHUNK-SIZE` substrings of `STRING`, on the interval [`START` `END`). Defaults are [0 string-length). Any remaining substring less than chunk-size is appended to the list.

The substrings share storage with the `STRING`!

`string->substring-list` [procedure]

```
(string->substring-list STRING CHUNK-SIZE [START [END]])
```

Returns a list of `CHUNK-SIZE` substrings of `STRING`, on the interval [`START` `END`). Defaults are [0 string-length). Any remaining substring less than chunk-size is appended to the list.

`->byte-vector` [procedure]

```
(->byte-vector OBJECT)
```

Converts the `OBJECT` into a byte-vector.

`string->hexadecimal` [procedure]

```
(string->hexadecimal STRING [LENGTH])
```

Returns the `STRING` as a hex-encoded string. When `LENGTH` missing string-length is used. The returned string is 2 * string-length.

3 License

Index