# ASCII Data Types

| ID | Type | Alt Name | Width | Description |

|:--:|:----:|:---------:|:------:|:------------|

| 0 | NIL | null/void | 0 | Nil/null/void type.|

| 1 | SOH | Address | <=N | An address of an I2P device or function stored as a string.|

| 2 | STX | String | <=N | A UTF-8/ASCII string.|

| 3 | SI1 | int8\_t | -1 | An 8-bit signed integer.|

| 4 | UI1 | uint8\_t | 1 | An 8-bit unsigned integer.|

| 5 | BOL | bool | -1 | A non-zero false Boolean variable.|

| 6 | SI2 | int16\_t | -2 | A 16-bit signed varint.|

| 7 | UI2 | uint16\_t | 2 | A 16-bit unsigned integer.|

| 8 | HLF | half | 2 | A 16-bit floating-point number.|

| 9 | SI4 | int32\_t | -4 | A 32-bit signed varint.|

| 10 | UI4 | uint32\_t | 4 | A 32-bit unsigned integer.|

| 11 | FLT | float | 4 | A 32-bit floating-point number.|

| 12 | TMS | Time\_s | -4 | A 32-bit second since epoch timestamp.|

| 13 | SI8 | int64\_t | -8 | A 64-bit signed integer.|

| 14 | UI8 | uint64\_t | 8 | A 64-bit unsigned integer.|

| 15 | DBL | double | 8 | A 64-bit floating-point number.|

| 16 | TMU | Time\_us | -8 | A 64-bit microsecond since epoch timestamp.|

| 17 | SV2 | int16\_t | -1 - 3 | A 32-bit signed varint.|

| 18 | SV4 | int32\_t | -1 - 5 | A 32-bit signed varint.|

| 19 | SV8 | int64\_t | -1 - 9 | A 64-bit signed varint.|

| 20 | UV2 | uint16\_t | 1 - 3 | A 32-bit unsigned varint.|

| 21 | UV4 | uint32\_t | 1 - 5 | A 32-bit unsigned varint.|

| 22 | UV8 | uint64\_t | 1 - 9 | A 64-bit unsigned varint.|

| 23 | AR1 | Array-1 | N | An array of up to 2^8-1 like primitive types 2-14.|

| 24 | AR2 | Array-2 | N | An array of up to 2^16-1 like primitive types 2-14.|

| 25 | AR4 | Array-4 | N | An array of up to 2^32-1 like primitive types 2-14.|

| 26 | AR8 | Array-8 | N | An array of up to 2^64-1 like primitive types 2-14.|

| 27 | ESC | Sequence | ? | An atomic escape sequence of I2P procedure calls.|

| 28 | BK8 | Book-8 | N | A book with up to 2^30-1 members and less than 2^64 bytes data.|

| 29 | BK4 | Book-4 | N | A book with up to 2^14-1 members and less than 2^32 bytes data.|

| 30 | BK2 | Book-2 | N | A book with up to 2^14-1 members and less than 2^16 bytes data.|

| 31 | US | Unit | N | A unit separator for splitting data up into packets.|

## List of Types Key

| Width | Description |

|:-----:|:-----------:|

| -X | Signed integer type.|

| ? | Type of unknown size.|

| N | Has prespecified buffer of size N bytes.|

| <=N | Has prespecified buffer of size N bytes but can use less than that.|

# Books, Bags, and Sequences

A Book is a special type of dictionary that can be configured with or without a hash table.

#### Book Example

RS -8 256 ; Preallocate a 256 byte book with 8 reserved indexes and no hash table.

"Item 1" UI1 23

"Item 2" STR "Hello world!" ; String will be stored with the default buffer size.

"Item 3" 32 STR "Hello world!" ; Optional buffer size comes before the type.

"Item 4" 6 ESC 1 2 3 BS CR FF ; It’s faster to parse script with a preallocated buffer sizes.

“Item 5” BK2 ; This is a dictionary with a hash table that takes longer to parse.

"Item 1" UI4 2,747,572 ; Commas are allowed in numbers in I2P.

"Item 2" 23 STR "Hello world!"

NIL

; No need for a NIL to close this books of prespecified length.

# Escape Sequences

; Hierarchy Description:

; Let ‘A’ be a device named “A” that contains a single sub-device at index ‘B’ named “dev”.

; Let ‘B’ be a sub-device of “A” in index ‘B’ that has a function ‘foo’ at index ‘D’ that takes a UI1 and returns NIL.

DC1 A B D 1 CR ; A pushes the device at index ‘A’ onto the stack and CR pops both A and dev off the stack.

“A” dev BS ; “A” with quotes is the string “A”. BS pops “A” off the stack.

ESC A BS ; This is a nested escape sequence.

ESC “A” “foo” 1 CR ; Quotes around foo unnecessary but legal.

A foo 1 CR ; FF closes all the ESC.

CR CR ; Please note there is one FF per ESC.

0xXXXXXXXX ; followed by a 32-bit hash to verify data integrity.

#### Example C++ Remote Procedure Call Implementation

#include <chinese\_room/include/module.h>

using namespace \_;

/\*\* Script Operation evaluation result structure.

For devices, set result null to nullptr.

\*/

struct Evaluation {

const char \* name; //< The name of the operation.

const uint\_t\* params, //< B-Sequence operation parameters.

\* result; //< B-Sequence Set Evaluation.

const char \* metadata; //< Metadata.

const void \* set; //< Pointer to the evaluated set.  
};

const Evaluation\* Operation (char\_t index, Automata\* a) {

void\* params[2];

static const int rxHeader = { 2, UI4, STR, 32 },

txHeader[] = { 2, UI4, STR };

static const Evaluation evaluation = { “OperationName”,

params, txHeader,

“Description”,

, nullptr };

if (a == nullptr) return evaluation;

uint32\_t input\_a,

input\_b,

output\_a = 1,

output\_b = 2;

if (a->read (m.rxHeader, args (params, &input\_a, &input\_b))) return ReadError();

/// Do some function logic here.

return a->write (m.txHeader, args (params, &output\_a, &output\_b));

}