Especificação de API – Interpretador LUA

Meios de Captura

Fevereiro de 2015
P.01.00.03

1 CONTEÚDO

1 CONTEÚDO 2
2 HISTÓRICO 10
3 INTRODUÇÃO 11
3.1 CONVENÇÕES, TERMOS E ABREVIAÇÕES 11
4 ESPECIFICAÇÃO DA FUNÇÃO LUA 13
4.1 CHAMADA DO INTERPRETADOR LUA 13
4.2 FUNÇÕES BÁSICAS 13
4.2.1 dofile (filename) 13
4.2.2 gcinfo () 13
4.2.3 loadfile (filename) 13
4.2.4 print (e1 [, e2, ...]) 13
4.2.5 require (packagename) 14
4.2.6 _VERSION 14
4.3 FUNÇÕES MATEMÁTICAS 14
4.3.1 math.abs (n) 14
*4.3.2 math.ceil (n) ..... **Error! Bookmark not defined.***
*4.3.3 math.floor (n) ..... **Error! Bookmark not defined.***
4.3.4 math.max (n1, n2, ...) 14
4.3.5 math.min (n1, n2, ...) 15
4.3.6 math.mod (n1, n2) 15
4.3.7 math.pow (n1, n2) 15
*4.3.8 math.sqrt (n) ..... **Error! Bookmark not defined.***
4.3.9 math.random ([n1[, n2]]) 15
4.3.10 math.bitwiseand(val1, val2) 15
4.3.11 math.bitwiseyor(val1, val2) 15
4.3.12 math.bitwisenot(val1) 15
4.3.13 math.shiftl(val, quantidade) 16
4.3.14 math.shiftr(val, quantidade) 16
4.4 FUNÇÕES DE E/S 16
4.4.1 io.close ([file]) 17
4.4.2 io.input ([file]) 17
4.4.3 io.lines ([filename]) 17
4.4.4 io.open (filename [, mode]) 17

4.4.5 io.output ([file])....18
4.4.6 io.read ([format1, ...])....18
4.4.7 io.type (obj)....18
4.4.8 io.write (value1, ...)....18
4.4.9 file:close ()....18
4.4.10 file:lines ()....18
4.4.11 file:read ([format1, ...])....18
4.4.12 file:seek ([base], [, offset])....19
4.4.13 file:write (valor1, ...)....19
4.5 FUNCIONALIDADES DO SISTEMA OPERACIONAL....20
4.5.1 os.chdir (dirpath)....20
4.5.2 os.date ([format[, time]])....21
4.5.3 os.difftime (t2, t1)....21
4.5.4 os.exit ([code])....21
4.5.5 os.freediskspace ()....22
4.5.6 os.getcwd ()....22
4.5.7 os.loadimage (filename)....22
4.5.8 os.unloadimage (filename)....22
4.5.9 image:width ()....22
4.5.10 image:height ()....23
4.5.11 image:size ()....23
4.5.12 os.mkdir (dirname)....23
4.5.13 os.readdir ([dirpath])....23
4.5.14 os.remove (filename[,isglobal])....24
4.5.15 os.rename (oldname, newname[,isglobal])....24
4.5.16 os.isdir (path)....24
4.5.17 os.isglobal (filename)....24
4.5.18 os.exists (path[,isglobal])....25
4.5.19 os.filesize (filename[,isglobal])....25
4.5.20 os.filemtime (path[,isglobal])....25
4.5.21 os.time ([table])....25
4.5.22 os.setdatetime (table)....26
4.5.23 os.tmpname ()....26
4.5.24 os.getenv (id)....26
4.5.25 os.update ([table])....27
4.5.26 os.updateHF ([table, fone, caller_id])....27
4.5.27 os.batterystatus()....28
4.5.28 os.isbatteryinitialized()....28

4.5.29 os.alimoff () 28
4.5.30 os.getnextnsu ([flush]) 29
4.5.31 os.setnextnsu (newNsu) 29
4.5.32 os.flushnsu () 29
4.5.33 os.typeTouchScreen 29
4.5.34 os.calibrateTouchScreen (ponto0x, ponto0y, ponto1x, ponto1y, ponto2x, ponto2y) 30
4.6 SISTEMA DE IMPRESSÃO 30
4.6.1 printer.linefeed ([lines]) 30
4.6.2 printer.print (arg [,align [,bold]]) 30
4.6.3 printer.printimage (image [, align]) 31
4.6.4 printer.font ([font]) 31
4.6.5 printer.check () 32
4.6.6 printer. paperfeed () 32
4.7 SISTEMA DE LEITURA DE CARTÃO MAGNÉTICO 33
4.7.1 magcard.read (tracks, timeout, [cancel]) 33
4.7.2 magcard.cancelkey () 33
4.7.3 track:status () 33
4.7.4 track:length () 34
4.7.5 track:data ([index]) 34
4.8 SISTEMA DE COMUNICAÇÃO (MODEM) 35
4.8.1 comm.loadconfig ([filename]) 36
4.8.2 comm.config (key[, value]) 36
4.8.3 comm.saveconfig (filename) 37
4.8.4 comm.connect (phonenumber [,pabx]) 37
4.8.5 comm.maxdialnumbers () 37
4.8.6 comm.disconnect () 37
4.8.7 comm.check () 38
4.8.8 comm.buffersize () 38
4.8.9 comm.send (msg [, offset[, length]]) 38
4.8.10 comm.receive (maxlen, timeout) 39
4.9 SISTEMA DE COMUNICAÇÃO (GPRS) 41
4.9.1 gprs. loadconfig ([filename]) 41
4.9.2 gprs.config (key[, value]) 41
4.9.3 comm.connect (ip, port) 41
4.9.4 gprs.status() 42
4.9.5 gprs.send (msg [, offset[, length]]) 42
4.9.6 gprs.receive (maxlen, timeout) 42
4.9.7 gprs.signalstrength() 43

4.10 SISTEMA DE COMUNICAÇÃO (RS232) 43
4.10.1 rs232.open (port [,config]) 43
4.10.2 rs232.close (port) 43
4.10.3 port:send (msg [, offset[, length [,timeout]]]) 44
4.10.4 port:receive (maxlen, timeout) 44
4.11 MENSAGENS ISO 45
4.11.1 iso.create() 45
4.11.2 iso.destroy(handler) 46
4.11.3 iso:typeid 46
4.11.4 iso:version([pversion]) 46
4.11.5 iso:class([pclass]) 47
4.11.6 iso:fnc([pfnc]) 47
4.11.7 iso:transorign([ptransorign]) 47
4.11.8 iso:binaryfield (pbit[,pdata]) 47
4.11.9 iso:datefield (pbit,[pdatetime]) 48
4.11.10 iso:assemble (header) 48
4.11.11 iso.disassemble (pis, ptimeout) 48
4.12 DISPLAY 51
4.12.1 display.print (arg [,line [,col]]) 51
4.12.2 display.font ([font]) 51
4.12.3 display.printimage (image, x, y) 51
4.12.4 display.clear ([line]) 52
4.12.5 display.geometry () 52
4.12.6 display.pixels () 52
4.12.7 display.drawline (x1, y1, x2, y2) 52
4.12.8 display.drawpoint (x, y) 53
4.12.9 display.drawrect (x, y, width, height) 53
4.12.10 display.invertcolors ([value]) 53
4.12.11 display.autoflush ([value]) 54
4.12.12 display.flush () 54
4.12.13 display. backlight (percent) 54
4.13 KEYBOARD 54
4.13.1 keyboard.getkeystroke ([timeout]) 54
4.13.2 keyboard.getkeylabel (key) 55
4.13.3 keyboard.buzzer (length, frequency) 55
4.13.4 keyboard.beepon ([value]) 55
4.13.5 keyboard. getkeystrokenb ([timeout]) 55
4.14 ARQUIVOS DE PROPRIEDADES 56

4.14.1 property.open(filename[, location]) *56*
4.14.2 property.close (prop) *56*
4.14.3 property:getstring(key) *57*
4.14.4 property:getnumber(key) *57*
4.14.5 property:getboolean(key) *57*
4.15 INTERFACE DE USUÁRIO - UI 58
4.15.1 ui.textfield(title, label, [maxlenght [,minlenght [,ispassword]]]) *58*
4.15.2 ui.transient(title, message, duration) *58*
4.15.3 ui.message(title, message [,screennumber [,titlealign [,messagealign [,messagevalign]]]]) *59*
4.15.4 ui.menu(title,options,[shownumbers]) *59*
4.15.5 ui.destroy(uihandle) *59*
4.16 UI – TEXTFIELD 60
4.16.1 textfield:text ([text]) *60*
4.16.2 textfield:maxlenght() *60*
4.16.3 textfield:pattern(pattern, [jokechar]) *60*
4.16.4 textfield:show([timeout]) *61*
4.16.5 textfield:align(halign [, valign]) *61*
4.16.6 textfield:aligntext(align) *61*
4.16.7 textfield:alignlabel(align) *61*
4.16.8 textfield:screennumber(number) *62*
4.16.9 textfield:type([type]) *62*
4.17 UI - MENU 62
4.17.1 menu:accepted() *62*
4.17.2 menu:show([timeout]) *62*
4.17.3 menu:aligntitle(align) *63*
4.17.4 menu:alignoptions(align) *63*
4.17.5 menu:screennumber(number) *63*
4.18 COMPRESSÃO 64
4.18.1 compression.compress(inputfile, outputfile) *64*
4.18.2 compression.decompress(inputfile, outputfile) *64*
4.19 FUNÇÕES UTILITÁRIAS 65
4.19.1 util.asciintobcd(valor, indice) *65*
4.19.2 util.asciitobcd(valor) *65*
4.19.3 util.bcdtoascii(valor) *65*
4.19.4 util.bytearray(size [,fill]) *66*
4.19.5 bytearray:get(posição) *66*
4.19.6 bytearray:set(indice,valor) *66*
4.19.7 bytearray:size() *66*

4.19.8 util.timer(count) 67
4.19.9 timer:remaining() 67
4.19.10 timer:elapsed() 67
4.20 CRIPTOGRAFIA 67
4.20.1 Base 64 67
4.20.1.1 base64.encode(bytestring) 68
4.20.1.2 base64.decode(string) 68
4.20.2 Hash SHA-1 68
4.20.2.1 sha1.calculate(bytestring) 69
4.20.2.2 sha1.chaininit() 69
4.20.2.3 sha1.chainupdate(bytestring) 69
4.20.2.4 sha1.chainresult() 69
4.20.3 Checksum CRC 69
4.20.3.1 crc32.calculate(bytestring [, alternative]) 70
4.20.4 Criptografia RSA 70
4.20.4.1 rsa.loadkey(keypath) 71
4.20.4.2 key:unloadkey() 71
4.20.4.3 key:encrypt(bytestring) 72
4.20.4.4 key:decrypt(bytestring) 72
4.20.4.5 key:sign(hashstring) 72
4.20.4.6 key:verify(hashstring, signature) 72
4.20.5 Criptografia Triple DES 73
4.20.5.1 tdes.genkey([refstring[, mkindex]]) 73
4.20.5.2 tdes:encrypt(bytestring[, iv]) 73
4.20.5.3 tdes:decrypt(bytestring[, iv]) 74
4.20.5.4 tdes:fencrypt(fin, fout[, iv]) 74
4.20.5.5 tdes:fdecrypt(fin, fout[, iv]) 74
4.21 LOG 75
4.21.1 log.logsingle(preg, psevty) 75
4.21.2 log.logburst(preg, psevty) 76
4.21.3 log.close() 76
4.22 STREAMS 76
4.22.1 streams.openfileis(ppath) 76
4.22.2 streams.opencommis() 76
4.22.3 streams.opengprsis() 77
4.22.4 streams.openrs232is(port) 77
4.22.5 streams.openstringis(pstr) 77
4.22.6 streams.openbufferedis(pis) 77

4.22.7 streams.openfileos(ppath) 78
4.22.8 streams.opencommos() 78
4.22.9 streams.opengprsos() 78
4.22.10 streams.openrs232os(port) 78
4.22.11 streams.openstringos() 79
4.22.12 streams.openbufferedos(pos) 79
4.22.13 ios:close() 79
4.22.14 is:read (pbytenum, ptimeout) 79
4.22.15 os:write (pstring, ptimeout[, poffset[, plength]]) 80
4.22.16 stringos:content() 80
4.22.17 os:flush(ptimeout) 80
4.23 GERENCIAMENTO DE PROCESSOS 81
4.23.1 management.sleep(ptime) 81
4.24 LEITORA DE CÓDIGO DE BARRAS 81
4.24.1 barcode.loadconfig([filename]) 81
4.24.2 barcode.config(key [,value]) 81
4.24.3 barcode.saveconfig(filename) 82
4.24.4 barcode.read(timeout [,maxlen]) 82
4.25 EMV 82
4.25.1 emv.init() 82
4.25.2 emv.version() 83
4.25.3 emv.checkcard() 83
4.25.4 emv.starttransaction() 83
4.25.5 emv.processtransaction() 83
4.25.6 emv.completetransaction(onlineResult) 84
4.25.7 emv.getcandidatelist() 84
4.25.8 emv.defapp(ref, data) 85
4.25.9 emv.selectapp(appIndex) 85
4.25.10 emv.defdata(tag, data) 85
4.25.11 emv.getdata(tag) 86
4.26 MULTI APLICAÇÃO 87
4.26.1 ms.open(appname) 87
4.26.2 ms.call(appstate, params) equivalente a appstate:call(params) 87
5 APÊNDICE A – CONSTANTES E STRINGS ISO 93
6 APÊNDICE B – MENSAGENS E CÓDIGOS DE ERRO 96
7 APÊNDICE C – PARÂMETROS DE MODEM 98
8 APÊNDICE D – CONNLIB.LUA 99
8.1 CONNECTIONS.LOAD(FILENAME|TABLE) 99

8.2 CONN:OPEN() 100
8.3 CONN:CHECK() 100
8.4 CONN:OPENIS() 100
8.5 CONN:OPENOS() 101
8.6 CONN:CLOSE() 101
9 REFERÊNCIAS 102

2 HISTÓRICO

P.01.00 – Maio/2014 – Autor: REDE(Luciana Fujiki)

Alteração	Descrição	Onde procurar
01.00.00	Versão Inicial	
01.00.01	os.isbatteryinitialized() Multi aplicação	4.5.28 4.26
01.00.02	Removidas algumas funções de math	4.3

3 INTRODUÇÃO

Este documento especifica as assinaturas das funçoes do interpretador Lua a serem desenvolvidos sobre a API RedeFlex. Seu propósito é listar as funcionalidades que deverão ser portadas, assim como, as que deverão ser estendidas. Todas as funções descritas no Lua 5.2 Reference Manual e não referenciadas na seção "**Escopo negativo**" também estarão presentes no interpretador Lua.

Já os códigos de erro e strings retornadas em caso de erro encontram-se no apêndice B deste documento.

3.1 Convenções, termos e abreviações

Esta seção explica o conceito de alguns termos importantes que serão mencionados no decorrer deste documento. Estes termos são descritos na tabela a seguir, estando apresentados por ordem alfabética.

Termo	Descrição
API	Application Programming Interface – Interface de Programação da Aplicação
CPU	Unidade Central de Processamento
E/S	Entrada e Saída
Handle	Manipulador
Iterator	Padrão de projeto utilizado para percorrer uma coleção de dados.
Nil	Valor nulo
POS	Point Of Sale
Requisitos funcionais	Requisitos técnicos do software que compõe o sistema, que descrevem ações que o sistema deve estar apto a executar, ou seja, o que o sistema deve fazer.
Requisitos não funcionais	Requisitos técnicos do software que compõe o sistema, que descrevem atributos que o sistema deve possuir ou restrições sob as quais ele deve operar.
Requisitos não técnicos	Requisitos não relacionados ao software. Requisitos não técnicos estão fora do escopo deste documento, devendo, se necessário, serem incluídos apenas no Plano do Projeto.
Shell	A shell fornece uma interface onde o usuário pode

	interagir com o sistema.
SO	Sistema Operacional

4 ESPECIFICAÇÃO DA FUNÇÃO LUA

4.1 Chamada do interpretador lua

O interpretador lua é inicializado pela função RF_lua_interpret(filename). Esta, roda o programa lua "filename" e retorna os seguintes valores:

- RF_SUCCESS: quando o programa executa corretamente;
- RF_ERR_INVALIDARG: caso o arquivo seja invalido ou inexistente;
- RF_ERR_PATHERR: se o diretório do arquivo principal lua for invalido ou inexistente
- RF_ERR_NOMEMORY: quando houver estouro de memória
- RF_ERR_LUA_RUN: erro de execução;
- RF_ERR_LUA_SYNTAX: erro de sintaxe;
- RF_ERR_LUA: erro genérico;

4.2 Funções Básicas

A biblioteca básica provê algumas funções essenciais de Lua. As funções que carregam arquivos de código Lua (dofile, loadfile, require) carregam somente arquivos de uma mesma aplicação, ou seja, arquivos que estejam no diretório da aplicação ou em sub-diretórios

4.2.1 dofile (filename)

Abre o arquivo "filename" e executa como um bloco de código Lua. Retorna qualquer valor retornado pelo bloco. Em caso de erro, propaga o erro para quem a chamou (isto é, esta função não roda em modo protegido).

4.2.2 gcinfo ()

Retorna dois resultados: o número de Kbytes de memória dinâmica que Lua está utilizando e o limite do coletor de lixo (também em Kbytes).

4.2.3 loadfile (filename)

Carrega um arquivo como um bloco de código Lua (sem executá-lo). Se não houver erros, retorna um bloco compilado como uma função; caso contrário, retorna "nil" mais uma mensagem de erro. A função retornada pertence ao escopo global.

4.2.4 print (e1 [, e2, ...])

Recebe qualquer número de argumentos, e imprime seus valores no “stdout”, usando a função “tostring” para convertê-los para strings. Esta função não foi feita para saídas formatadas, mas sim para um modo rápido de exibir valores, tipicamente para depuração. Para saídas formatadas use “string.format”.

4.2.5 require (packagename)

Carrega o pacote dado. Esta função começa varrendo a tabela “_LOADED” para determinar se o pacote já foi carregado. Se sim, então “require” retorna o valor de quando o pacote foi carregado pela primeira vez. Caso contrário, procura no path predefinido (“shared\?;shared\?.lua;?;?.lua”) por um arquivo para carregar.

A função pára a busca assim que encontrar um arquivo que possa ser carregado, e então executa o arquivo. Em seguida, associa, na tabela “_LOADED”, o nome do pacote com o valor que o pacote retornou, e então retorna o valor. Se o pacote retornar “nil” (ou nenhum valor), “require” converte esse valor para “true”. Se o pacote retornar “false”, “require” retorna “false”. No entanto, como o campo na tabela “_LOADED” é “false”, qualquer nova tentativa de recarregar o arquivo irá ser realizada como se esse não tivesse sido carregado (i.e., o pacote será carregado novamente).

Se houver um erro durante o carregamento ou execução, ou se o arquivo não for encontrado no path, então “require” levanta um erro.

Durante a execução de um arquivo, “require” define a variável global “_REQUIREDNAME” com o nome do pacote. O pacote carregado sempre roda no escopo global.

4.2.6 _VERSION

Uma variável global (não uma função) que é uma string que contém a versão atual do interpretador. O conteúdo atual dessa string é “Lua 5.1”.

4.31Funções Matemáticas

As funções matemáticas estão agrupadas na tabela “**math**”.

No interpretados LUA do Redeflex os números representados como inteiros. Desta forma, as funções da biblioteca math que só fazem sentido para número de ponto flutuante foram removidas.

4.3.1 math.abs (n)

Retorna no valor absoluto do número “n” (| n |).

4.3.2 math.max (n1, n2, ...)

Retorna o maior dos números passados como parâmetro.

4.3.3 math.min (n1, n2, ...)

Retorna o menor dos números passados como parâmetro.

4.3.4 math.mod (n1, n2)

Retorna o resto da divisão de “n1” por “n2”.

4.3.5 math.pow (n1, n2)

Retorna “n1” elevado a “n2”. Essa função também acessível através do operador “^”, isto é, “math.pow(n1, n2)” é o mesmo que “n1 ^ n2”.

4.3.6 math.random ([n1[, n2]])

Quando chamada sem argumentos, retorna um número (pseudo-aleatório) no intervalo [0,1). Quando chamada com um número “n1”, retorna um número (pseudo-aleatório) no intervalo [1,n1]. Quando chamada com os argumentos “n1” e “n2”, retorna um número (pseudo-aleatório) no intervalo [n1,n2].

4.3.7 math.bitwiseand(val1, val2)

Aplica a operação de AND entre “**val1**” e “**val2**”.

Parâmetros:

- **val1**: 1º número.
- **val2**: 2º número

Retornos:

- retorna um inteiro como resultado da operação .

4.3.8 math.bitwiseyor(val1, val2)

Aplica a operação de OR entre “**val1**” e “**val2**”.

Parâmetros:

- **val1**: 1º número
- **val2**: 2º número

Retornos:

- retorna um inteiro como resultado da operação.

4.3.9 math.bitwisenot(val1)

Aplica a operação de NOT em “**val1**”.

Parâmetros:

- **val1**: número à ser aplicada a operação de not.

Retornos:

- retorna um inteiro como resultado da operação.

4.3.10 math.shiftl(val, quantidade)

Aplica a operação de SHIFT LEFT **"quantidade"** de vezes.

Parâmetros:

- **val**: número a ser aplicado a operação.
- **quantidade**: quantidade de deslocamentos a esquerda que serão realizados.

Retornos:

- retorna um inteiro como resultado da operação.

Obs.: Caso o parâmetro "**quantidade**" seja um valor negativo, o retorno da função será 0 (zero).

4.3.11 math.shiftr(val, quantidade)

Aplica a operação de SHIFT RIGHT **"quantidade"** de vezes.

Parâmetros:

- **val**: numero a ser aplicado a operação.
- **quantidade**: quantidade de deslocamentos a direita que serão realizados.

Retornos:

- retorna um inteiro como resultado da operação.

Obs.: Caso o parâmetro "**quantidade**" seja um valor negativo, o retorno da função será 0 (zero).

4.4 Funções de E/S

A biblioteca de E/S provê duas formas para a manipulação de arquivos. A primeira utiliza descritores de arquivos implícitos, isto é, existem operações para configurar a entrada e a saída padrão, e todas as operações de E/S são realizadas sobre esses arquivos padrão. A segunda forma utiliza descritores explícitos.

Quando utilizamos os descritores implícitos, as operações são acessadas através da tabela "io". Quando utilizamos os descritores explícitos, a operação "io.open" retorna um descritor de arquivo, e a partir desse momento, os métodos são acessados através desse descritor.

A tabela "io" também provê três descritores de arquivos predefinidos: io.stdin (keyboard), io.stdout (display) e io.stderr (display).

Ao ser inicializado, o interpretador configura a entrada padrão como o "io.stdin" e a saída padrão como "io.stdout".

A não ser que seja explicitado, todas as operações de E/S retornam "nil" em caso de falha (Mais uma mensagem de erro como segundo retorno) e algum resultado diferente de "nil" em casos de sucesso.

Obs.: Os nomes de arquivos devem seguir as especificações descritas na seção “Sistema de Arquivos” do Documento de Requisitos HiperFLEX API e HELPER Functions [3]

4.4.1 io.close ([file])

Equivalente a “file:close ()”. Sem o parâmetro “file”, fecha o arquivo de saída padrão.

Obs: Caso o arquivo de saída padrão seja o “io.stdout”, o mesmo não será fechado.

4.4.2 io.input ([file])

Quando chamado com um nome de arquivo, abre-se o arquivo e configura seu handle como entrada padrão (funciona apenas com arquivos locais). Quando chamado com um handle do arquivo, ele simplesmente configura este handle como entrada padrão. Quando chamado sem parâmetros, ele retorna a atual entrada padrão.

Em caso de erros, esta função levanta uma exceção, ao invés de retornar um código de erro.

4.4.3 io.lines ([filename])

Abre o arquivo indicado por “filename” em modo de leitura e retorna uma função iterator, que cada vez que é chamada, retorna uma nova linha desse arquivo. Então, a construção

```
for line in io.lines (filename) do ... end
```

vai percorrer todas as linhas do arquivo. Quando a função iterator detecta o fim do arquivo, ela retorna “nil” (para finalizar o laço) e automaticamente fecha o arquivo.

A chamada “io.lines()” (sem o nome do arquivo) é equivalente a “io.input():lines ()”, isto é, percorre todas as linhas da entrada padrão.

Obs.: chamadas passando “filename” funcionam apenas para arquivos locais.

4.4.4 io.open (filename [, mode])

Responsável pela abertura de um arquivo nos modos:

- “**r**” somente leitura (opção default);
- “**rg**” somente leitura (arquivo global);
- “**rw**” leitura e escrita. Se o arquivo não existir, será criado;
- “**rwg**” leitura e escrita. Se o arquivo não existir, será criado (arquivo global).

Essa função retorna um handle para o arquivo especificado no parâmetro. Em caso de erro o retorno da função é “nil”.

O cursor de leitura/escrita do arquivo é sempre colocada no início do mesmo.

4.4.5 io.output ([file])

Similar ao “io.input”, porém opera sobre o arquivo de saída padrão.

4.4.6 io.read ([format1, ...])

Equivalente à “io.input():read”.

4.4.7 io.type (obj)

Verifica se “obj” é um handle válido. Retorna a string “file” se “obj” é um handle para um arquivo aberto, “closed file” se for um handle para um arquivo fechado e “nil” se “obj” não for um handle válido.

4.4.8 io.write (value1, ...)

Equivalente à “io.output():write”.

4.4.9 file:close ()

Fecha o arquivo “file”.

4.4.10 file:lines ()

Retorna uma função iterator que, a cada vez que é chamada, retorna uma nova linha do arquivo (“**file**”). Então a construção

for line in file:lines () do … end

vai percorrer todas as linhas do arquivo. (Diferentemente de “**io.lines”**, esta função não fecha o arquivo quando o laço termina.).

4.4.11 file:read ([format1, ...])

Lê o arquivo (“**file**”) de acordo com os formatos dados, os quais especificam o que deve ser lido. Para cada formato, a função retorna uma string (ou número) com os caracteres lidos, ou “**nil**” caso não seja possível ler os dados com o formato especificado. Quando chamada sem formatos, a função usa o formato padrão, o qual lerá a próxima linha.

Os formatos disponíveis são

“***n**” – lê um número, esse é o único formato que retorna um número ao invés de uma string.

“***a**” – lê o arquivo inteiro iniciando da posição atual. No fim do arquivo, retorna uma string vazia.

“***l**” – lê a próxima linha (o caractere de fim de linha é ignorado), retornando “**nil**” no fim do arquivo. Este é o formato padrão.

number – lê uma string com até “**number**” caracteres, retornando “**nil**” no fim do arquivo. Se “**number**” for zero, nada é lido e uma string vazia é retornada.

4.4.12 file:seek ([base], [, offset])

Configura e retorna a posição do arquivo, medida a partir do início do arquivo, para a posição dada pelo “**offset**” (em bytes) adicionada à base, como a seguir:

“**set**” base igual a zero (início do arquivo);

“**cur**” base é igual à posição atual;

“**end**” base é igual ao fim do arquivo.

Em caso de sucesso a função “**seek**” retorna a posição final do arquivo, medida em bytes a partir do início do arquivo. Se a função falhar, ou a janela do arquivo ultrapassar o final do arquivo, a mesma retornará “**nil**” mais uma string descrevendo o erro.

O valor padrão de base é “**cur**” e o do “**offset**” é zero. Então, a chamada “**file:seek ()**” retorna a posição atual do arquivo, sem modificá-la; a chamada “**file:seek(“set”)**” configura a posição para o início do arquivo e retorna zero; e a função “**file:seek(“end”)**” configura a posição para o fim do arquivo, e retorna seu tamanho.

4.4.13 file:write (valor1, …)

Escreve o valor de cada um dos argumentos no arquivo representado por “**file**”. Os argumentos devem ser strings ou números. Para escrever outros valores, utilize “**tostring**” ou “**string.format**” antes de “**write**”.

Em caso de sucesso retorna-se “**true**”. Em caso de erro, retorna-se “**false**”, mais uma mensagem de erro.

Exemplo de uso:

```
display.print("Copiando arquivos")
local src = io.open("foto.jpg", "r")
local des = io.open("dest.jpg", "rw")
if src ~= nil and io.type(des) == "file" then -- garante a execução correta
io.input(des) –- configura o handler de des como entrada padrao
local size = src:seek("end") -- descobre o tamanho do arquivo
display.print(string.format("Tamanho: %d bytes", size))
src:seek("set") -- volta ao início do arquivo
local data = src:read(8192) -- le 8k
while data ~= nil do
des:write(data)
data = io.read(8192) -- equivalente a linha 9
end
src:close()
io.close(des) -- equivalente a linha 14
display.print("Copia finalizada");
else
display.print(“erro ao abrir arquivo”)
end
```

4.5 Funcionalidades do Sistema Operacional

Essa biblioteca é implementada através da tabela “**os**”.

4.5.1 os.chdir (dirpath)

Modifica o diretório corrente.

Parâmetros:

- **dirpath**: caminho do diretório.

Retornos:

- em caso de sucesso, retorna-se “**true**”;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro.

Obs.:

- O caminho **“/”** representa a raiz da aplicação LUA.

- Nenhum arquivo abaixo da raiz será acessível a uma aplicação LUA.
- Caso **dirpath** comece com **'/'** será considerado um caminho completo.
- "..", retorna para o diretório acima.

4.5.2 os.date ([format[, time]])

Formata data e hora de acordo com a string de formato dada.

Parâmetros:

- **format:** String de formatação.
- **time:** Se o argumento "**time**" estiver presente, esta é a hora a ser formatada (veja a função "**os.time()**" para descrição desse valor). Caso contrário, esta função formata a hora atual.

Retornos:

- em caso de sucesso, retorna-se a data formatada;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro.

Se o formato for igual à "***t**", então a função retornará uma tabela com os seguintes campos: "**year**" (4 dígitos), "**month**"(1-12), "**day**"(1-31), "**hour**"(0-23), "**min**"(0-59), "**sec**"(0-59), "**wday**"(dia da semana, onde domingo é 1), "**yday**"(dia do ano) e "**isdst**"(horário de verão que é um booleano).

Se o formato não for "***t**", então a função retorna a data como uma string, formatada de acordo com as mesmas regras da função "**strftime**" de C.

Quando chamada sem parâmetros, "**os.date**" retorna uma representação razoável da data e hora que dependem do sistema hospedeiro.

Obs: Se "**format**" começar com uma '!', então a data deveria ser formatada de acordo com as coordenadas universais de tempo (UTC). Porém, esta particularidade de "**os.date**" não será portada, pois a API HiperFlex 0 não disponibiliza a localidade em que se encontra o POS.

4.5.3 os.difftime (t2, t1)

Calculo da diferença entre as horas t2 e t1.

Parâmetros:

- **t2**: hora 2.
- **t1**: hora 1.

Retornos:

- retorna-se o número de segundos entre as horas "**t1**" e "**t2**";

4.5.4 os.exit ([code])

Esta função termina a interpretação do programa corrente.

Parâmetros:

- **code:** status de saída.

Retornos:

- em caso de sucesso, retorna-se o "**code**" como status de saída.;
  Obs: O valor padrão de "**code**" é o código de sucesso (EXIT_SUCCESS de C).

4.5.5 os.freediskspace ()

Retorna o espaço disponível (em bytes) no sistema de arquivos.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, o espaço disponível (em bytes) no sistema de arquivos:
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.6 os.getcwd ()

Retorna o nome absoluto do diretório corrente.

4.5.7 os.loadimage (filename)

Carrega uma imagem para a memória.

Parâmetros:

- **filename**: nome do arquivo de imagem a ser carregado.

Retornos:

- em caso de sucesso, retorna um handle para a imagem:
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

Obs.: A imagem deve ser um bitmap monocromático (1 bit/pixel) cuja largura seja múltiplo de 8. A imagem deve ser salva no formato BMP (Windows Bitmap Format).

4.5.8 os.unloadimage (filename)

Desaloca uma imagem da memória.

Parâmetros:

- **filename**: nome do arquivo de imagem a ser desalocado.

Retornos:

- em caso de sucesso, retorna "**true**";
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.9 image:width ()

Retorna a largura da imagem.

Retornos:

- em caso de sucesso, retorna a largura da imagem;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.10 image:height ()

Retorna a altura da imagem.

Retornos:

- em caso de sucesso, retorna a altura da imagem;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.11 image:size ()

Retorna a largura e altura da imagem.

Retornos:

- em caso de sucesso, retorna a largura e altura da imagem;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.12 os.mkdir (dirname)

Cria um novo diretório (com o nome "**dirname**") no diretório corrente.

Parâmetros:

- **dirname**: nome do diretório a ser criado.

Retornos:

- em caso de sucesso, retorna "**true**";
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.13 os.readdir ([dirpath])

Esta função lista os arquivos/diretórios do diretório especificado por "**dirpath**". A lista retornada contém apenas os nomes dos arquivos em ordem alfabética. O valor padrão de "**dirpath**" é CWD, ou seja, se chamada sem parâmetros, lê o diretório atual. Em caso de erro retorna **nil**.

Parâmetros:

- **dirname**: nome do diretório a ser listado, se não houver parâmetro, utiliza-se o valor padrão.

Retornos:

- em caso de sucesso, retorna a lista de arquivos/diretórios;

- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

Obs.: "**dirpath**" é o caminho completo (inicia com '**/**') e não possui ".." ou ".". Ex.: **/dados/**, representa o caminho real **/lua/<aplicação-lua>/dados/**

4.5.14 os.remove (filename[,isglobal])

Apaga o arquivo/diretório indicado por "**filename**".

Parâmetros:

- **filename**: nome do arquivo/diretório a ser apagado;
- **isglobal**: indica se o arquivo que se deseja remover é global (valor padrão é **false**).

Retornos:

- em caso de sucesso, retorna "**true**";
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.15 os.rename (oldname, newname[,isglobal])

Renomeia o arquivo/diretório "**oldname**" para "**newname**".

Parâmetros:

- **oldname**: nome do arquivo/diretório atual;
- **newname**: nome do arquivo/diretório futuro;
- **isglobal**: indica se o arquivo que se deseja renomear é global (valor padrão é **false**).

Retornos:

- em caso de sucesso, retorna "**true**";
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.16 os.isdir (path)

Checa se o caminho "**path**" passado é um diretório.

Parâmetros:

- **path**: caminho a ser checado.

Retornos:

- em caso de sucesso, retorna "**true**";
- em caso de falha, retorna "**false**".

4.5.17 os.isglobal (filename)

Checa se o arquivo de nome "**filename**" é global.

Parâmetros:

- **filename**: nome do arquivo a ser checado.

Retornos:

- em caso de sucesso, retorna "**true**";
- em caso de falha, retorna "**false**".

4.5.18 os.exists (path[,isglobal])

Checa se o caminho "**path**" passado representa um arquivo ou diretório válidos.

Parâmetros:

- **path**: nome do arquivo/diretório a ser checado.
- **isglobal**: indica se o arquivo referenciado por "**path**" é global (valor padrão é **false**).

Retornos:

- em caso de sucesso, retorna "**true**";
- em caso de falha, retorna "**false**".

4.5.19 os.filesize (filename[,isglobal])

Verifica o tamanho do arquivo "**filename**".

Parâmetros:

- **filename**: nome do arquivo a ser analizado;
- **isglobal**: indica se o arquivo referenciado por "**filename**" é global (valor padrão é **false**).

Retornos:

- em caso de sucesso, retorna o tamanho do arquivo;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.20 os.filemtime (path[,isglobal])

Retorna quando foi a última modificação realizada do arquivo ou diretório "**path**" em segundos passados a partir de 1º de janeiro de 1970.

Parâmetros:

- **path**: nome do arquivo/diretório a ser analizado;
- **isglobal**: indica se o arquivo referenciado por "**path**" é global (valor padrão é **false**).

Retornos:

- em caso de sucesso, retorna a quantidade de segundos passadas desde a última modificação;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.21 os.time ([table])

Retorna o timestamp atual quando chamada sem parâmetros, ou o timestamp representando a data e hora especificadas pela tabela dada. A tabela deve conter os campos "**year**", "**month**", e "**day**" e pode ter os campos "**hour**", "**min**", e "**sec**".

Timestamp é um número que conta os segundos passados a partir de 01/01/1970.

Parâmetros:

- **table**: tabela contendo data e se necessário hora.

Retornos:

- em caso de sucesso, retorna o timestamp;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.22 os.setdatetime (table)

Modifica a data e a hora do sistema. O parâmetro "table" é uma tabela que deve conter os campos "**year**", "**month**", "**day**", "**hour**", "**min**", e "**sec**".

Parâmetros:

- **table**: tabela contendo data e a hora.
  - year: quatro dígitos
  - month: valor entre 1 e 12 (janeiro = 1)

Retornos:

- em caso de sucesso, retorna o **true**;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.23 os.tmpname ()

Retorna um nome de arquivo que pode ser usado como arquivo temporário.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, retorna o nome do arquivo temporário;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

Obs:O arquivo deve ser explicitamente aberto antes de ser usado e removido quando não for mais necessário.

4.5.24 os.getenv (id)

Retorna uma propriedade do sistema a partir do seu ("**id**"). Os valores possíveis para **id** são:

- SYS_SERIALNU: *número de série do terminal;*
- SYS_HIPERFLEXVER: *versão do HiperFlex;*
- SYS_PLATFORMNAME: *nome da plataforma (Ex.: INGENICO);*

- SYS_PLATFORMVER: *versão da plataforma (Ex.: 5100).*
- SYS_GPRS: *suporte gprs ("suported" ou "unsuported").*

Parâmetros:

- **id**: identificador da propriedade do sistema.

Retornos:

- em caso de sucesso, retorna o valor da propriedade;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

Exemplo de uso:

```
serial = os.getenv(SYS_SERIALNU)
display.print(string.format("SERIAL: %s", serial))
```

4.5.25 os.update ([table])

Função que trata do agendamento de atualizações das aplicações Lua. Caso o parâmetro "table" seja passado, agenda uma atualização. Se o parâmetro for **nil**, apaga o agendamento existente. Se não houver um parâmetro, retorna o timestamp da próxima atualização (se a mesma existir).

Parâmetros:

- **table**: tabela contendo os campos "**year**", "**month**", "**day**", "**hour**", "**min**", e "**sec**".
  - year: quatro dígitos
  - month: valor entre 1 e 12 (janeiro = 1)

Retornos:

- em caso de sucesso, retorna um inteiro indicando o timestamp da próxima atualização. Caso não haja atualizações a serem feitas essa função retorna 0 (zero);
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.26 os.updateHF ([table, fone, caller_id])

Função para agendar a auto-tele-carga (atualização do Redeflex). A função cria um agendamento se receber os três argumentos (timestamp que corresponde à data e hora da atualização, telefone do servidor e o caller id). Se a função receber apenas um argumento nil, o agendamento corrente será apagado. Se não houver argumentos, retorna os parâmetros do agendamento corrente (caso ele exista).

Parâmetros:

- **table**: tabela contendo os campos "**year**", "**month**", "**day**", "**hour**", "**min**", e "**sec**".
  - year: quatro dígitos
  - month: valor entre 1 e 12 (janeiro = 1)
- **fone**: telefone do servidor (stringo contendo dígitos).

- **caller_id**: identificador usado pelo servidor de atualização para selecionar um pacote.

Retornos:

- em caso de sucesso, retorna o timestamp da próxima atualização, o telefone do servidor e o caller id. Se não houver um agendamento, a função retorna 0 (zero);
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.27 os.batterystatus()

Obtem o status da bateria.

Parâmetros:

- (sem parametros).

Retornos:

- em caso de sucesso, retorna um inteiro indicando o nivel de carga da bateria (valor de 0 a 100) e um inteiro indicando o status da bateria, o qual pode ser SYS_BATTERY_CHARGING (bateria em carga), SYS_BATTERY_NO_CHARGE (bateria não sendo carregada), SYS_BATTERY_LOW_ALARM (alarme de bateria em nível baixo);
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.28 os.isbatteryinitialized()

Algumas baterias necessitam que sua primeira carga seja feita por pelo menos 6 hora para ser considerada “inicializada”.

Parâmetros:

- (sem parametros).

Retornos:

- Caso a bateria esteja inicializada ou não tenha este conceito, retorna “**true”**;
- **“false”**, caso não esteja inicializada.

4.5.29 os.alimoff ()

Desativa a alimentação do POS. Esta função faz com que o POS seja desligado caso o mesmo seja alimentado por uma bateria e a mesma não esteja sendo carregada.

Parâmetros:

- (sem parametros).

Retornos:

- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.30 os.getnextnsu ([flush])

Retorna o NSU do terminal.

Parâmetros:

- flush – se o nsu deve ser salvo na memória física após incremento (Padrão true).

Retornos:

- em caso de sucess, o nsu em um inteiro.
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.31 os.setnextnsu (newNsu)

Configura o NSU do terminal.

Parâmetros:

- newNsu – o novo NSU.

Retornos:

- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.32 os.flushnsu ()

Salva o valor do NSU na memória física do terminal.

Parâmetros:

- (sem parametros).

Retornos:

- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.5.33 os.typeTouchScreen

Retorna o tipo de display touch screen

Parâmetros:

- (sem parametros).

Retornos:

- “resistivo”: dispositivo possui display resistivo.
- “capacitivo”: dipositivo possui display capacitivo.

4.5.34 os.calibrateTouchScreen (ponto0x, ponto0y, ponto1x, ponto1y, ponto2x, ponto2y)

Se o display do dispositivo for resistivo, realiza a calibragem da tela conforme coordenadas passadas.

Parâmetros:

- ponto0x – coordenada x do ponto 0.
- ponto0y – coordenada y do ponto 0.
- ponto1x – coordenada x do ponto 1.
- ponto1y – coordenada y do ponto 1.
- ponto2x – coordenada x do ponto 2.
- ponto2y – coordenada y do ponto 2.

Retornos:

- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.6Sistema de Impressão

Essa biblioteca é implementada através da tabela "**printer**".

4.6.1 printer.linefeed ([lines])

Imprime linhas em branco.

Parâmetros:

- **lines**: número de linhas em branco (o valor padrão de **lines** é um).

Retornos:

- em caso de sucesso, temos **true** como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.6.2 printer.print (arg [,align [,bold]])

Imprime uma seqüência de caracteres na impressora, essa seqüência pode ser impressa em 1 ou mais linhas.

Parâmetros:

- **arg**: seqüência de caracteres a ser impressa (caso esse parâmetro não seja uma string, então a chamada equivale à **printer.print (tostring(arg)[,align[,bold]])**);
- **align**: alinhamento: "left", "center" e "right". O valor padrão de **align** é "left";
- **bold**: impressão em negrito. O valor padrão de **bold** é **false**.

Retornos:

- em caso de sucesso, temos **true** como retorno;

- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

Obs.: Esta função também suporta caracteres de escape para centralizar o texto (PRINTER_ESCCENTER), para alinhar à direita (PRINTER_ESCRIGHT) e para voltar ao normal (PRINTER_ESCNORMAL). Além disso, existe o caracter para impressão em negrito (PRINTER_ESCBOLD). O caractere de seqüência de escape deve sempre preceder um desses quatro caracteres de escape (PRINTER_ESCSEQ). Veja o exemplo abaixo:

```
str = string.format("texo normal %c%ctexto em negrito",
                                    PRINTER_ESCSEQ, PRINTER_ESCBOLD)
printer.print(str)


-- saída:
-- texto normal texto em negrito
```

4.6.3 printer.printimage (image [, align])

Imprime uma imagem (carregada por "os.loadimage") na impressora.

Parâmetros:

- **image**: imagem a ser impressa;
- **align**: posição na qual a imagem deve ser alinhada.
  - "left" – alinhado a partir da margem esquerda (padrão);
  - "right" – alinhado a partir da margem direita;
  - "center" - alinhado a partir da coluna central.

Retornos:

- em caso de sucesso, temos **true** como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.6.4 printer.font ([font])

Modifica a fonte da impressora e retorna a fonte corrente.

Parâmetros:

- **font**: as fontes disponíveis são: "small", "normal", "large". Se esse parâmetro não for passado, a fonte não será modificada.

Retornos:

- em caso de sucesso, temos a fonte corrente como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

Obs.: A fonte padrão é “small”.

4.6.5 printer.check ()

Verifica o status da impressora.

Parâmetros:

- (sem parâmetros)

Retornos:

- **“ok”**: impressora pronta;
- **“oop”:** impressora sem papel;
- “**error**”: erro no dispositivo.

4.6.6 printer. paperfeed ()

Avança o papel em uma quantidade necessária para que a ultima linha impressa seja visível.

Parâmetros:

- (sem parâmetros)

Retornos:

- (sem retornos).

Exemplo de uso:

```
if printer.check() == “ok” then
   printer.print(“Teste de impressão”, “center”)
   printer.linefeed()
   local img = os.loadimage(“teste.bmp”)
   if img ~= nil then
      local offset = -(img:width()/2)
      printer.printimage(img, offset, “center”)
      printer.linefeed(5) -- avança 5 linhas
      printer.print(img) -- imprime dados da imagem
   else
      printer.print(“imagem não carregada!”)
   end
else if printer.check() == “oop” then
   display.print(“impressora sem papel”)
else
   display.print(“erro na impressora”)
end
```

4.7 Sistema de Leitura de Cartão Magnético

Essa biblioteca é responsável pela leitura dos cartões magnéticos e será implementada através da tabela "**magcard**".

4.7.1 magcard.read (tracks, timeout, [cancel])

Retorna a informação contida nas trilhas do cartão magnético.

Parâmetros:

- **tracks**: trilhas que devem ser lidas (para lermos as trilhas devemos usar a soma das constantes: MAGCARD_TRACK1, MAGCARD_TRACK2 e MAGCARD_TRACK3);
- **timeout**: tempo, em segundos, que se deve aguardar que o cartão seja passado;
- **cancel**: um booleano indicando se a operação deve ser cancelada caso a tecla CANCEL seja pressionada (padrão é **true**).

Retornos:

- em caso de sucesso, temos as trilhas lidas como o retorno, ou seja, o número de retornos é variável.
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.7.2 magcard.cancelkey ()

Retorna o código da tecla que causou o cancelamento da última solicitação de leitura do cartão.

Parâmetros:

- (sem parâmetros).

Retornos:

- em caso de sucesso, temos o código da tecla como o retorno.
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.7.3 track:status ()

Retorna a situação da leitura da trilha.

Parâmetros:

- (sem parâmetros)

Retornos:

- "track_ok": leitura com sucesso;
- "error_swipe": erro na passagem do cartão;
- "error_track": erro nos dados da trilha;
- "error_unavaliable": a trilha solicitada não está disponível;
- "error": erro de paridade, de checksum ou do dispositivo.

4.7.4 track:length ()

Retorna o tamanho dos dados armazenados na trilha.

Parâmetros:

- (sem parâmetros)

Retornos:

- o tamanho dos dados armazenados na trilha.

4.7.5 track:data ([index])

Retorna os dados armazenados na trilha. Caso **index** seja passado, então essa função retorna apenas o valor do byte cuja posição é igual a **index**.

Parâmetros:

- **index**: posição do byte a ser retornado.

Retornos:

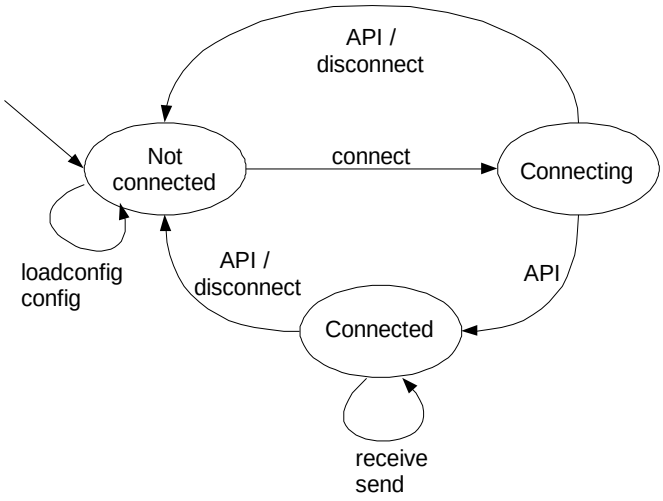
- os dados armazenados na trilha na forma de uma string ou o byte cuja posição é igual a **index**.
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

Exemplo de uso:

```
display.print("Teste de cartao")
t1, t2 = magcard.read(MAGCARD_TRACK1 + MAGCARD_TRACK2, 15, true)
if t1 ~= nil then
   if t1:status() == "track_ok" then
      printer.print("t1 ok")
      printer.print(string.format("length = %d", t1:length()))
      x = t1:data(1) -- lê o primeiro byte da trilha
      printer.print(string.format("t1[1] = %d", x))
   end
   if t2:status() == "track_ok" then
      printer.print("t2 ok")
      printer.print(string.format("length = %d", t2:length()))
      data = t2:data() -- lê todos os dados da trilha
      str = string.format("t2[1] = %d", string.byte(data, 1))
      printer.print(str)
   end
else if t2 == "read canceled" then
   display.print("leitura cancelada")
   local key = magcard.cancelkey()
   display.print(string.format("tecla precionada = %d", key))
                  else
   display.print(t2) -- t2 contém a mensagem de erro
end
```

4.8Sistema de Comunicação (Modem)

Essa biblioteca é responsável pela comunicação através do modem e será implementada através da tabela "comm". A utilização dessas funções é restrita pela máquina de estados do modem, ilustrado a seguir:



Ela é composta por três estados, sendo "*Not connected*" o estado inicial. Para cada estado, são listadas as funções que podem ser chamadas enquanto o modem estiver naquele estado. As funções que não estão listadas podem ser chamadas independentemente do estado do modem.

Note que algumas mudanças de estado são causadas diretamente pelo código lua (ex.: a mudança do estado "*Not connected*" para "*Connecting*" pela chamada a *connect*), enquanto que outras mudanças são causadas de forma indireta (ex.: do estado "*Connecting*" para "*Connected*"). Essas mudanças de estado indiretas ocorrem quando o código Lua faz chamadas a esta biblioteca (ex.: chamadas subseqüentes a check).

4.8.1 comm.loadconfig ([filename])

Configura o modem a partir do arquivo de configuração apontado por **filename**. Caso **filename** não seja passado, então carrega a configuração padrão da plataforma.

Parâmetros:

- **filename**: caminho do arquivo de propriedades.

Retornos:

- caso a função execute com sucesso, temos **true** como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.8.2 comm.config (key[, value])

Configura apenas uma propriedade do modem. Se o parâmetro opcional *value* não for informado, a função apenas retorna o valor corrente. As chaves e os valores permitidos para os parâmetros dessa função podem ser vistos em *Apêndice C – Parâmetros de modem*.

Parâmetros:

- **key**: nome da propriedade
- **value**: valor da propriedade.

Retornos:

- caso a função execute com sucesso, temos **o valor atribuído à propriedade**;

- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.8.3 comm.saveconfig (filename)

Salva a configuração atual do modem no arquivo apontado por **filename**. Se ele já existir, será sobrescrito.

Parâmetros:

- **filename**: nome do arquivo.

Retornos:

- caso a função execute com sucesso, temos **true** como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.8.4 comm.connect (phonenumber [,pabx])

Inicia uma conexão com o modem.

Parâmetros:

- **phonenumber**: uma string contendo o número a ser discado;
- **pabx**: uma string indicando o número do PABX.

Retornos:

- caso a função execute com sucesso, temos **true** como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

  Obs.: Esta função é assíncrona. Portanto, para verificar se a conexão foi aberta use a função "**comm.check()**".

4.8.5 comm.maxdialnumbers ()

Retorna o número máximo de dígitos suportados específicos da plataforma para pabx e número de discagem.

Parâmetros:

- (sem parâmetros).

Retornos:

- número máximo de dígitos suportados (pabx, phonenumber).

4.8.6 comm.disconnect ()

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso na finalização da conexão, temos **true** como retorno;

- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.8.7 comm.check ()

Verifica o estado do modem. O modem estará no estado *Connected* se o resultado dessa função for "**connected**", no *Connecting* se o resultado dessa função for "waiting" e no *Not Connected* para o restante dos retornos.

Parâmetros:

- (sem parâmetros)

Retornos:

- "connected": no caso de haver uma conexão estabelecida;
- "waiting": no caso de uma conexão estar sendo estabelecida;
- "disconnected": no caso de estar desconectado e não houver nenhum erro associado
- "no carrier": no caso de não ser detectado o sinal da portadora
- "no dial tone": no caso de não ser detectado tom de discagem
- "busy line": se a linha estiver ocupada
- "no answer": se a ligação não foi atendida
- "connection lost": se a conexão foi perdida
- "failed": no caso de falha desconhecida ao estabelecer a conexão.

4.8.8 comm.buffersize ()

Retorna os tamanhos do buffer de envio e de recepção. Se o modem não tiver sido configurado previamente, os tamanhos padrões dos buffers para a plataforma serão retornados.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, retorna o tamanho do buffer de envio mais o tamanho do buffer de recepção;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.8.9 comm.send (msg [, offset[, length]])

Envia uma mensagem através do modem.

Parâmetros:

- **msg**: uma string representando os dados a serem enviados (os caracteres da string serão tratados como bytes);
- **offset**: ponto de **msg** a partir do qual serão enviados os bytes (seu valor padrão é zero);

- **length**: a quantidade de bytes a serem enviados, esse valor não pode ultrapassar o tamanho do buffer de recepção (ver "**comm.buffersize()**").

Retornos:

- em caso de sucesso, retorna a quantidade de bytes enviados;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.8.10 comm.receive (maxlen, timeout)

Recebe uma mensagem através do modem.

Parâmetros:

- **maxlen**: número máximo de bytes que devem ser lidos, esse valor não pode ultrapassar o tamanho do buffer de recepção (ver "**comm.buffersize()**");
- **timeout** – tempo máximo, em milisegundos, de espera pelos dados.

Retornos:

- em caso de sucesso, retorna os bytes lidos na forma de uma string;
- em caso de erro ou timeout, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

Exemplo de uso:

```
function conectar(sNumber)
   result = false
   if comm.connect(sNumber) then
      display.print("conectando")
      while comm.check() == "waiting" do
         display.print(".")
      end
      if comm.check() == "connected" then
         result = true
      end
   end
   return result
end

display.print("Teste de modem")
if conectar("32332143") then
   sent, msg = comm.send("dados sendo enviados")
   if sent ~= nil then
      printer.print(string.format("bytes enviados = %d", sent))
   else
      printer.print(msg) -- mostra msg de erro
   end
   msg, errmsg = comm.receive(1024, 2000)
   if msg ~= nil then -- recebeu com sucesso
      printer.print(msg)
   else
      printer.print(string.format("erro: %s", errmsg))
   end
else
   display.print("erro ao conectar")
end
comm.disconnect()
```

4.9 Sistema de Comunicação (GPRS)

Essa biblioteca é responsável pela comunicação através da rede GPRS e será implementada através da tabela "**gprs**".

4.9.1 gprs. loadconfig ([filename])

Configura os parâmetros da conexão gprs a partir do arquivo de configuração apontado por **filename**. Caso **filename** não seja passado, então o arquivo "gprs.cfg" da aplicação em execução é carregado. Para o caso ainda de não existir um "gprs.cfg" para a aplicação, então as configurações gerais de gprs do POS são carregadas ("/cfg/gprs.cfg")

Parâmetros:

- **filename**: caminho do arquivo de propriedades.

Retornos:

- caso a função execute com sucesso, temos **true** como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.9.2 gprs.config (key[, value])

Configura apenas uma propriedade da conexão gprs. Se o parâmetro opcional *value* não for informado, a função apenas retorna o valor corrente. As chaves e os valores permitidos para os parâmetros dessa função podem ser vistos no documento de requisitos da API Hiperflex.

Parâmetros:

- **key**: nome da propriedade
- **value**: valor da propriedade.

Retornos:

- caso a função execute com sucesso, temos **o valor atribuído à propriedade**;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.9.3 comm.connect (ip, port)

Inicia uma conexão gprs.

Parâmetros:

- **ip**: uma string contendo o endereço ip do host a ser connectado;
- **port**: uma string indicando a porta de conexão com o host.

Retornos:

- caso a função execute com sucesso, temos **true** como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.9.4 gprs.status()

Verifica o estado da conexão gprs. A conexão gprs estará no estado *Connected* se o resultado dessa função for "**connected**", no *Connecting* se o resultado dessa função for "waiting" e no *Not Connected* para o restante dos retornos.

Parâmetros:

- (sem parâmetros)

Retornos:

- "connected": no caso de haver uma conexão estabelecida;
- "waiting" : no caso de uma conexão estar sendo estabelecida;
- "disconnected": no caso de estar desconectado e não houver nenhum erro associado
- "registrationerror": se o POS não consegui se registrar na rede GPRS
- "hostnotfound": se o ip ao qual o POS está tentando se conectar não foi encontrado na rede
- "connrefused": se o host não aceitar a conexão na porta desejada
- "connection lost": se a conexão foi perdida
- "failed": no caso de falha desconhecida ao estabelecer a conexão.

4.9.5 gprs.send (msg [, offset[, length]])

Envia uma mensagem através da conexão gprs.

Parâmetros:

- **msg**: uma string representando os dados a serem enviados (os caracteres da string serão tratados como bytes);
- **offset**: ponto de **msg** a partir do qual serão enviados os bytes (seu valor padrão é zero);
- **length**: a quantidade de bytes a serem enviados.

Retornos:

- em caso de sucesso, retorna a quantidade de bytes enviados;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.9.6 gprs.receive (maxlen, timeout)

Recebe uma mensagem através da conexão gprs.

Parâmetros:

- **maxlen**: número máximo de bytes que devem ser lidos;
- **timeout** – tempo máximo, em milisegundos, de espera pelos dados.

Retornos:

- em caso de sucesso, retorna os bytes lidos na forma de uma string;

- em caso de erro ou timeout, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.9.7 gprs.signalstrength()

Obtem a potência do sinal GPRS.

Parâmetros:

- (sem parametros).

Retornos:

- em caso de sucesso, a potencia do sinal em uma escala de 0 a 100;
- em caso de erro ou timeout, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.10 Sistema de Comunicação (RS232)

Essa biblioteca é responsável pela comunicação através de portas seriais (RS232) e será implementada através da tabela "**rs232**".

4.10.1 rs232.open (port [,config])

Abre uma porta serial.

Parâmetros:

- **port**: uma string indicando a porta a ser aberta ("COM1" ou "COM2");
- **config**: uma tabela com as configuações da porta. Os possíveis campos são os seguintes:
  - **baud:** valor numérico (300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600*, 115200);
  - **datasize:** valor numérico (7 ou 8*)
  - **parity:** string ("none"*, "odd" ou "even")
  - **flow:** string ("none"* ou "hard")
  - **stopbits:** valor numérico (1* ou 2)

  * Valor padrão utilizado caso o campo não exista

Retornos:

- caso a função execute com sucesso, temos o **handle** da porta como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.10.2 rs232.close (port)

Fecha a porta serial. Também podemos fechar a porta chamando *port:close().*

Parâmetros:

- **port**: handle da porta serial a ser fechada

Retornos:

- em caso de sucesso, temos **true** como retorno;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.10.3 port:send (msg [, offset[, length [,timeout]]])

Envia uma mensagem através da porta serial.

Parâmetros:

- **msg**: uma string representando os dados a serem enviados (os caracteres da string serão tratados como bytes);
- **offset**: ponto de **msg** a partir do qual serão enviados os bytes (seu valor padrão é zero);
- **length**: a quantidade de bytes a serem enviados. Caso não seja passado seu valor corresponde ao tamanho de **msg**;
- **timeout**: tempo máximo (em milissegundos) para que os dados sejam enviados. Caso não seja passado seu valor será o máximo de um inteiro de 32bits.

Retornos:

- em caso de sucesso, retorna a quantidade de bytes enviados;
- em caso de falha, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

4.10.4 port:receive (maxlen, timeout)

Recebe uma mensagem através da porta serial.

Parâmetros:

- **maxlen**: número máximo de bytes que devem ser lidos;
- **timeout**: tempo máximo, em milisegundos, de espera pelos dados.

Retornos:

- em caso de sucesso, retorna os bytes lidos na forma de uma string;
- em caso de erro ou timeout, temos **nil**, mais uma mensagem de erro e mais o código do erro como retorno.

Exemplo de uso:

```
display.print("Teste de RS232")

config = {}
config.baud = 4800
config.flow = "hard"

p = rs232.open("COM1",config)

if p ~= nil then
   msg = p:receive(10, 2000)
   if msg ~= nil then
      printer.print(msg)
      sent = p:send(msg)
      if sent ~= nil then
        printer.print(string.format("enviados: %d", sent))
      end
   end
   p:close()
end
```

4.11 Mensagens ISO

Essa biblioteca descreve funções para manipulação de mensagens no formato ISO8583:1993 (os campos de bitmap secundário foram deixados de lado). Descrições e referencias podem ser encontradas, também, no documento da "Especificação Funcional_Mapa ISO". As constantes ISO podem ser encontradas no apêndice A deste mesmo documento. As strings retornadas pelas funções fnc, class e transorign também se encontram no apêndice A.

4.11.1 iso.create()

Cria uma mensagem ISO que deve ser destruída após o uso, com a função iso.destroy(handler). Demais propriedades da mensagem devem ser modificadas usando suas funções.

Parâmetros:

- sem parâmetros.

Retornos:

- em caso de sucesso, retorna um handle para a mensagem de ISO criada.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.11.2 iso.destroy(handler)

Destrói a mensagem de ISO passada como parâmetro. Desta forma, desaloca os recursos do mesmo.

Parâmetros:

- **handler:** o handler da mensagem ISO a ser destruído.

Retornos:

- em caso de sucesso, retorna "**true**".
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.11.3 iso:typeid

Modifica a propriedade *type identifier* da mensagem ISO (versão, classe, função e originador de transação) de uma só vez e retorna o seu valor. Se o parâmetro opcional *ptypeid* não for passado, simplesmente retorna o valor atual.

Parâmetros:

- **ptypeid:** novo tipo idendificador da mensagem (uma string com 4 caracteres numéricos).

Retornos:

- o valor atual do tipo identificador da mensagem (uma string)
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

Obs.: Se a função for chamada sem parâmetros e se ao menos um dos quatro campos que compõe o tipo identificador estiver indefinido, a função retorna um erro**.**

4.11.4 iso:version([pversion])

Modifica a propriedade *version* da mensagem de ISO e retorna o seu valor. Se o parâmetro opcional *pversion* não for passado, simplesmente retorna o valor atual.

Parâmetros:

- **pversion:** novo conteúdo da propriedade *version*.

Retornos:

- o valor atual da propriedade *version* (um número) (*Multi Aplicação*
- *ms.open(appname)*
- *Abre* um serviço externo para futuras chamadas

Parâmetros:

- **appname**: é o nome do aplicativo/serviço (diretório no AMS)

Retornos:

- Retorna um tipo que identifica o estado do serviço externo

4.11.5 ms.call(appstate, params) equivalente a appstate:call(params)

Realiza uma chamada ao serviço externo passado em appstate. É importante observar que o estado do serviço será preservado para futuras chamadas.

Parâmetros:

- **params**: tabela de parâmetros

Retornos:

- Retorna uma tabela contendo o resultado da chamada

4.11.6 ms.close(appstate) equivalente a appstate:close()

Fecha um serviço externo e libera todos os recursos alocados

4.11.7 ms.singlecall(appname, params)

Realiza uma chamada simples a um serviço externo

É equivalente a execução consecutiva das seguintes chamadas:

```
local appstate = ms.open(appname)
appstate:call(params)
appstate:close()
```

4.12 Capacidades do dispositivo

4.12.1 device.screen_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da tela do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da tela do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)
  - 2 – Gráfico monocromático
  - 3 – Gráfico colorido

- **pixels_x**: Resolução horizontal (em pixels) da configuração.
- **pixels_y**: Resolução vertical (em pixels) da configuração.
- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela tela do dispositivo.
- **pixels_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da tela do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.12.2 device.screen_get_current_settings()

Retorna a configuração gráfica atual de tela do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da tela do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.screen_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.screen_get_capabilities).**
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.12.3 device.screen_set_current_settings(settings)

Seleciona a configuração gráfica de tela a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a tela do dispositivo deve operar.

  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.screen_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.12.4 device.printer_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da impressora do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da impressora do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)

- 2 – Gráfico monocromático
- 3 – Gráfico colorido

- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela impressora do dispositivo.
- **dots_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da impressora do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.12.5 device.printer_get_current_settings()

Retorna a configuração gráfica atual da impressora do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da impressora do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.printer_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.printer_get_capabilities).**
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.12.6 device.printer_set_current_settings(settings)

Seleciona a configuração gráfica de impressora a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a impressora do dispositivo deve operar.
  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.printer_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.12.7 device.input_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica os métodos de entrada disponíveis no dispositivo.

Cada registro da tabela é uma tabela contendo um possível método de entrada.

Os seguintes métodos de entrada podem ser retornados:

- 1 – Teclado
- 2 – TouchScreen (ou Mouse)
- 3 – Teclado na tela
- 4 – Teclado externo
- 5 – Impressão digital
- 6 – Voz
- 7 – Leitor de trilhas magnéticas
- 8 – Leitor de cartão com chip
- 9 – Porta serial
- 10 – Porta serial USB

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam os mecanismos de entrada disponíveis do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.12.8 device.comm_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica as tecnologias de comunicação do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível tecnologia de comunicação.

Os seguintes valores podem ser retornados:

- 1 – Dial síncrono
- 2 – Dial assíncrono
- 3 – GSM CSD
- 4 – GSM GPRS
- 5 – GSM EDGE
- 6 – 3G
- 7 – 4G
- 8 – WIFI
- 9 – Ethernet
- 10 – Bluetooth
- 11 - NFC

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam as tecnologias de comunicação disponíveis no dispositivo.
- Em caso de erro, retorna ”nil”, uma mensagem de erro e o código do erro.
- *Apêndice A – Constantes e Strings ISO*)
- em caso de erro, retorna “**nil”,** mais uma mensagem de erro, mais o código do erro.

  Obs.: O valor default de version *é a **“ISO_VER_1987”.***

4.12.9 iso:class([pclass])

Modifica a propriedade *class* da mensagem de ISO e retorna o seu valor. Se o parâmetro opcional *pclass* não for passado, simplesmente retorna o valor atual.

Parâmetros:

- **pclass:** novo conteúdo da propriedade *class*.

Retornos:

- o valor atual da propriedade *class* (um número) (*Multi Aplicação*
- *ms.open(appname)*
- *Abre* um serviço externo para futuras chamadas

Parâmetros:

- **appname**: é o nome do aplicativo/serviço (diretório no AMS)

Retornos:

- Retorna um tipo que identifica o estado do serviço externo

4.12.10 ms.call(appstate, params) equivalente a appstate:call(params)

Realiza uma chamada ao serviço externo passado em appstate. É importante observar que o estado do serviço será preservado para futuras chamadas.

Parâmetros:

- **params**: tabela de parâmetros

Retornos:

- Retorna uma tabela contendo o resultado da chamada

4.12.11 ms.close(appstate) equivalente a appstate:close()

Fecha um serviço externo e libera todos os recursos alocados

4.12.12 ms.singlecall(appname, params)

Realiza uma chamada simples a um serviço externo

É equivalente a execução consecutiva das seguintes chamadas:

```
local appstate = ms.open(appname)
appstate:call(params)
appstate:close()
```

4.13 Capacidades do dispositivo

4.13.1 device.screen_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da tela do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da tela do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)
  - 2 – Gráfico monocromático
  - 3 – Gráfico colorido

- **pixels_x**: Resolução horizontal (em pixels) da configuração.
- **pixels_y**: Resolução vertical (em pixels) da configuração.
- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela tela do dispositivo.
- **pixels_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da tela do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.13.2 device.screen_get_current_settings()

Retorna a configuração gráfica atual de tela do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da tela do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.screen_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.screen_get_capabilities).**
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.13.3 device.screen_set_current_settings(settings)

Seleciona a configuração gráfica de tela a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a tela do dispositivo deve operar.

  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.screen_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.13.4 device.printer_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da impressora do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da impressora do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)

- 2 – Gráfico monocromático
- 3 – Gráfico colorido

- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela impressora do dispositivo.
- **dots_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da impressora do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.13.5 device.printer_get_current_settings()

Retorna a configuração gráfica atual da impressora do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da impressora do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.printer_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.printer_get_capabilities).**
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.13.6 device.printer_set_current_settings(settings)

Seleciona a configuração gráfica de impressora a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a impressora do dispositivo deve operar.
  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.printer_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.13.7 device.input_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica os métodos de entrada disponíveis no dispositivo.

Cada registro da tabela é uma tabela contendo um possível método de entrada.

Os seguintes métodos de entrada podem ser retornados:

- 1 – Teclado
- 2 – TouchScreen (ou Mouse)
- 3 – Teclado na tela
- 4 – Teclado externo
- 5 – Impressão digital
- 6 – Voz
- 7 – Leitor de trilhas magnéticas
- 8 – Leitor de cartão com chip
- 9 – Porta serial
- 10 – Porta serial USB

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam os mecanismos de entrada disponíveis do terminal.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.13.8 device.comm_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica as tecnologias de comunicação do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível tecnologia de comunicação.

Os seguintes valores podem ser retornados:

- 1 – Dial síncrono
- 2 – Dial assíncrono
- 3 – GSM CSD
- 4 – GSM GPRS
- 5 – GSM EDGE
- 6 – 3G
- 7 – 4G
- 8 – WIFI
- 9 – Ethernet
- 10 – Bluetooth
- 11 - NFC

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam as tecnologias de comunicação disponíveis no dispositivo.
- Em caso de erro, retorna ”nil”, uma mensagem de erro e o código do erro.
- *Apêndice A – Constantes e Strings ISO*)
- em caso de erro, retorna “**nil”,** mais uma mensagem de erro, mais o código do erro.

Obs.: O valor default de class é a **“ISO_CLASS_UNDEF”.**

4.13.9 iso:fnc([pfnc])

Modifica a propriedade *fnc* (função) da mensagem de ISO e retorna o seu valor. Se o parâmetro opcional *pfnc* não for passado, simplesmente retorna o valor atual.

Parâmetros:

- **pfnc**: novo conteúdo da propriedade *fnc*.

Retornos:

- o valor atual da propriedade *fnc* (um número) (*Multi Aplicação*
- *ms.open(appname)*
- *Abre* um serviço externo para futuras chamadas

Parâmetros:

- **appname**: é o nome do aplicativo/serviço (diretório no AMS)

Retornos:

- Retorna um tipo que identifica o estado do serviço externo

4.13.10 ms.call(appstate, params) equivalente a appstate:call(params)

Realiza uma chamada ao serviço externo passado em appstate. É importante observar que o estado do serviço será preservado para futuras chamadas.

Parâmetros:

- **params**: tabela de parâmetros

Retornos:

- Retorna uma tabela contendo o resultado da chamada

4.13.11 ms.close(appstate) equivalente a appstate:close()

Fecha um serviço externo e libera todos os recursos alocados

4.13.12 ms.singlecall(appname, params)

Realiza uma chamada simples a um serviço externo

É equivalente a execução consecutiva das seguintes chamadas:

```
local appstate = ms.open(appname)
appstate:call(params)
appstate:close()
```

4.14 Capacidades do dispositivo

4.14.1 device.screen_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da tela do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da tela do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)
  - 2 – Gráfico monocromático
  - 3 – Gráfico colorido

- **pixels_x**: Resolução horizontal (em pixels) da configuração.
- **pixels_y**: Resolução vertical (em pixels) da configuração.
- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela tela do dispositivo.
- **pixels_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da tela do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.14.2 device.screen_get_current_settings()

Retorna a configuração gráfica atual de tela do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da tela do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.screen_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.screen_get_capabilities).**
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.14.3 device.screen_set_current_settings(settings)

Seleciona a configuração gráfica de tela a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a tela do dispositivo deve operar.

  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.screen_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.14.4 device.printer_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da impressora do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da impressora do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)

- 2 – Gráfico monocromático
- 3 – Gráfico colorido

- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela impressora do dispositivo.
- **dots_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da impressora do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.14.5 device.printer_get_current_settings()

Retorna a configuração gráfica atual da impressora do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da impressora do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.printer_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.printer_get_capabilities).**
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.14.6 device.printer_set_current_settings(settings)

Seleciona a configuração gráfica de impressora a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a impressora do dispositivo deve operar.
  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.printer_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.14.7 device.input_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica os métodos de entrada disponíveis no dispositivo.

Cada registro da tabela é uma tabela contendo um possível método de entrada.

Os seguintes métodos de entrada podem ser retornados:

- 1 – Teclado
- 2 – TouchScreen (ou Mouse)
- 3 – Teclado na tela
- 4 – Teclado externo
- 5 – Impressão digital
- 6 – Voz
- 7 – Leitor de trilhas magnéticas
- 8 – Leitor de cartão com chip
- 9 – Porta serial
- 10 – Porta serial USB

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam os mecanismos de entrada disponíveis do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.14.8 device.comm_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica as tecnologias de comunicação do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível tecnologia de comunicação.

Os seguintes valores podem ser retornados:

- 1 – Dial síncrono
- 2 – Dial assíncrono
- 3 – GSM CSD
- 4 – GSM GPRS
- 5 – GSM EDGE
- 6 – 3G
- 7 – 4G
- 8 – WIFI
- 9 – Ethernet
- 10 – Bluetooth
- 11 - NFC

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam as tecnologias de comunicação disponíveis no dispositivo.
- Em caso de erro, retorna ”nil”, uma mensagem de erro e o código do erro.
- *Apêndice A – Constantes e Strings ISO*)
- em caso de erro, retorna “**nil”,** mais uma mensagem de erro, mais o código do erro.

Obs.: O valor default de função é a **“ISO_FNC_UNDEF”.**

4.14.9 iso:transorign([ptransorign])

Modifica a propriedade *transorign* (iniciador da transação) da mensagem de ISO e retorna o seu valor. Se o parâmetro opcional *ptransorign* não for passado, simplesmente retorna o valor atual.

Parâmetros:

- **ptransorign**: novo conteúdo da propriedade *transorign*.

Retornos:

- o valor atual da propriedade *transorign* (um número) (*Multi Aplicação*
- *ms.open(appname)*
- *Abre* um serviço externo para futuras chamadas

Parâmetros:

- **appname**: é o nome do aplicativo/serviço (diretório no AMS)

Retornos:

- Retorna um tipo que identifica o estado do serviço externo

4.14.10 ms.call(appstate, params) equivalente a appstate:call(params)

Realiza uma chamada ao serviço externo passado em appstate. É importante observar que o estado do serviço será preservado para futuras chamadas.

Parâmetros:

- **params**: tabela de parâmetros

Retornos:

- Retorna uma tabela contendo o resultado da chamada

4.14.11 ms.close(appstate) equivalente a appstate:close()

Fecha um serviço externo e libera todos os recursos alocados

4.14.12 ms.singlecall(appname, params)

Realiza uma chamada simples a um serviço externo

É equivalente a execução consecutiva das seguintes chamadas:

```
local appstate = ms.open(appname)
appstate:call(params)
appstate:close()
```

4.15 Capacidades do dispositivo

4.15.1 device.screen_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da tela do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da tela do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)
  - 2 – Gráfico monocromático
  - 3 – Gráfico colorido

- **pixels_x**: Resolução horizontal (em pixels) da configuração.
- **pixels_y**: Resolução vertical (em pixels) da configuração.
- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela tela do dispositivo.
- **pixels_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da tela do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.15.2 device.screen_get_current_settings()

Retorna a configuração gráfica atual de tela do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da tela do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.screen_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.screen_get_capabilities).**
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.15.3 device.screen_set_current_settings(settings)

Seleciona a configuração gráfica de tela a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a tela do dispositivo deve operar.

  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.screen_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.15.4 device.printer_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da impressora do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da impressora do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)

- 2 – Gráfico monocromático
- 3 – Gráfico colorido

- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela impressora do dispositivo.
- **dots_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da impressora do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.15.5 device.printer_get_current_settings()

Retorna a configuração gráfica atual da impressora do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da impressora do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.printer_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.printer_get_capabilities).**
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.15.6 device.printer_set_current_settings(settings)

Seleciona a configuração gráfica de impressora a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a impressora do dispositivo deve operar.
  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.printer_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.15.7 device.input_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica os métodos de entrada disponíveis no dispositivo.

Cada registro da tabela é uma tabela contendo um possível método de entrada.

Os seguintes métodos de entrada podem ser retornados:

- 1 – Teclado
- 2 – TouchScreen (ou Mouse)
- 3 – Teclado na tela
- 4 – Teclado externo
- 5 – Impressão digital
- 6 – Voz
- 7 – Leitor de trilhas magnéticas
- 8 – Leitor de cartão com chip
- 9 – Porta serial
- 10 – Porta serial USB

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam os mecanismos de entrada disponíveis do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.15.8 device.comm_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica as tecnologias de comunicação do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível tecnologia de comunicação.

Os seguintes valores podem ser retornados:

- 1 – Dial síncrono
- 2 – Dial assíncrono
- 3 – GSM CSD
- 4 – GSM GPRS
- 5 – GSM EDGE
- 6 – 3G
- 7 – 4G
- 8 – WIFI
- 9 – Ethernet
- 10 – Bluetooth
- 11 - NFC

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam as tecnologias de comunicação disponíveis no dispositivo.
- Em caso de erro, retorna ”nil”, uma mensagem de erro e o código do erro.
- *Apêndice A – Constantes e Strings ISO*)
- em caso de erro, retorna “**nil”,** mais uma mensagem de erro, mais o código do erro.

Obs.: O valor default de transorign é a **“ISO_TRANSORIGN_UNDEF”.**

4.15.9 iso:binaryfield (pbit[,pdata])

Modifica o conteúdo do bit **pbit** da mensagem ISO e retorna o seu valor. Se o parâmetro opcional **pdata** não for passado, simplesmente retorna o valor atual do bit. Se **pdata** for **nil**, então o bit será removido da mensagem (caso exista). Para bit não setado na mensagem, o valor **nil** é retornado sem qualquer sinalização de erro.

Esta função coloca um bit ISO que tenha qualquer formato, exceto data e hora.

Parâmetros:

- **pbit**: o bit a ser colocado (*Multi Aplicação*
- *ms.open(appname)*
- *Abre* um serviço externo para futuras chamadas

Parâmetros:

- **appname**: é o nome do aplicativo/serviço (diretório no AMS)

Retornos:

- Retorna um tipo que identifica o estado do serviço externo

4.15.10 ms.call(appstate, params) equivalente a appstate:call(params)

Realiza uma chamada ao serviço externo passado em appstate. É importante observar que o estado do serviço será preservado para futuras chamadas.

Parâmetros:

- **params**: tabela de parâmetros

Retornos:

- Retorna uma tabela contendo o resultado da chamada

4.15.11 ms.close(appstate) equivalente a appstate:close()

Fecha um serviço externo e libera todos os recursos alocados

4.15.12 ms.singlecall(appname, params)

Realiza uma chamada simples a um serviço externo

É equivalente a execução consecutiva das seguintes chamadas:

```
local appstate = ms.open(appname)
appstate:call(params)
appstate:close()
```

4.16 Capacidades do dispositivo

4.16.1 device.screen_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da tela do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da tela do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)
  - 2 – Gráfico monocromático
  - 3 – Gráfico colorido

- **pixels_x**: Resolução horizontal (em pixels) da configuração.
- **pixels_y**: Resolução vertical (em pixels) da configuração.
- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela tela do dispositivo.
- **pixels_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da tela do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.16.2 device.screen_get_current_settings()

Retorna a configuração gráfica atual de tela do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da tela do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.screen_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.screen_get_capabilities).**
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.16.3 device.screen_set_current_settings(settings)

Seleciona a configuração gráfica de tela a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a tela do dispositivo deve operar.

  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.screen_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.16.4 device.printer_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da impressora do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da impressora do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)

- 2 – Gráfico monocromático
- 3 – Gráfico colorido

- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela impressora do dispositivo.
- **dots_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da impressora do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.16.5 device.printer_get_current_settings()

Retorna a configuração gráfica atual da impressora do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da impressora do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.printer_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.printer_get_capabilities).**
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.16.6 device.printer_set_current_settings(settings)

Seleciona a configuração gráfica de impressora a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a impressora do dispositivo deve operar.
  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.printer_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.16.7 device.input_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica os métodos de entrada disponíveis no dispositivo.

Cada registro da tabela é uma tabela contendo um possível método de entrada.

Os seguintes métodos de entrada podem ser retornados:

- 1 – Teclado
- 2 – TouchScreen (ou Mouse)
- 3 – Teclado na tela
- 4 – Teclado externo
- 5 – Impressão digital
- 6 – Voz
- 7 – Leitor de trilhas magnéticas
- 8 – Leitor de cartão com chip
- 9 – Porta serial
- 10 – Porta serial USB

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam os mecanismos de entrada disponíveis do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.16.8 device.comm_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica as tecnologias de comunicação do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível tecnologia de comunicação.

Os seguintes valores podem ser retornados:

- 1 – Dial síncrono
- 2 – Dial assíncrono
- 3 – GSM CSD
- 4 – GSM GPRS
- 5 – GSM EDGE
- 6 – 3G
- 7 – 4G
- 8 – WIFI
- 9 – Ethernet
- 10 – Bluetooth
- 11 - NFC

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam as tecnologias de comunicação disponíveis no dispositivo.
- Em caso de erro, retorna ”nil”, uma mensagem de erro e o código do erro.
- *Apêndice A – Constantes e Strings ISO*).
- **data**: o novo conteúdo do bit.

Retornos:

- em caso de sucesso, retorna o conteúdo do bit *pbit*.
- em caso de erro, retorna “**nil”,** mais uma mensagem de erro, mais o código do erro.

4.16.9 iso:datefield (pbit,[pdatetime])

Coloca o bit pbit na mensagem ISO com o conteúdo **pdatetime**. Se o parâmetro opcional **pdatetime** não for passado, simplesmente retorna o valor atual do bit. Se **pdatetime** for **nil**, então o bit será removido da mensagem (caso exista).

Esta função valida apenas os campos de datetime que são necessários para o bit (ex.: se o campo é AAMMDD, uma hora de -90 não é validada).

Parâmetros:

- **pbit**: o bit a ser colocado (*Multi Aplicação*
- *ms.open(appname)*
- *Abre* um serviço externo para futuras chamadas

Parâmetros:

- **appname**: é o nome do aplicativo/serviço (diretório no AMS)

Retornos:

- Retorna um tipo que identifica o estado do serviço externo

4.16.10 ms.call(appstate, params) equivalente a appstate:call(params)

Realiza uma chamada ao serviço externo passado em appstate. É importante observar que o estado do serviço será preservado para futuras chamadas.

Parâmetros:

- **params**: tabela de parâmetros

Retornos:

- Retorna uma tabela contendo o resultado da chamada

4.16.11 ms.close(appstate) equivalente a appstate:close()

Fecha um serviço externo e libera todos os recursos alocados

4.16.12 ms.singlecall(appname, params)

Realiza uma chamada simples a um serviço externo

É equivalente a execução consecutiva das seguintes chamadas:

local appstate = ms.open(appname)

appstate:call(params)

appstate:close()

4.17 Capacidades do dispositivo

4.17.1 device.screen_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da tela do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da tela do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)
  - 2 – Gráfico monocromático
  - 3 – Gráfico colorido

- **pixels_x**: Resolução horizontal (em pixels) da configuração.
- **pixels_y**: Resolução vertical (em pixels) da configuração.
- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela tela do dispositivo.
- **pixels_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da tela do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.17.2 device.screen_get_current_settings()

Retorna a configuração gráfica atual de tela do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da tela do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.screen_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.screen_get_capabilities).**
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.17.3 device.screen_set_current_settings(settings)

Seleciona a configuração gráfica de tela a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a tela do dispositivo deve operar.

  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.screen_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.17.4 device.printer_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da impressora do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da impressora do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)

- 2 – Gráfico monocromático
- 3 – Gráfico colorido

- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela impressora do dispositivo.
- **dots_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da impressora do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.17.5 device.printer_get_current_settings()

Retorna a configuração gráfica atual da impressora do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da impressora do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.printer_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.printer_get_capabilities).**
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.17.6 device.printer_set_current_settings(settings)

Seleciona a configuração gráfica de impressora a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a impressora do dispositivo deve operar.
  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.printer_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.17.7 device.input_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica os métodos de entrada disponíveis no dispositivo.

Cada registro da tabela é uma tabela contendo um possível método de entrada.

Os seguintes métodos de entrada podem ser retornados:

- 1 – Teclado
- 2 – TouchScreen (ou Mouse)
- 3 – Teclado na tela
- 4 – Teclado externo
- 5 – Impressão digital
- 6 – Voz
- 7 – Leitor de trilhas magnéticas
- 8 – Leitor de cartão com chip
- 9 – Porta serial
- 10 – Porta serial USB

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam os mecanismos de entrada disponíveis do terminal.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.17.8 device.comm_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica as tecnologias de comunicação do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível tecnologia de comunicação.

Os seguintes valores podem ser retornados:

- 1 – Dial síncrono
- 2 – Dial assíncrono
- 3 – GSM CSD
- 4 – GSM GPRS
- 5 – GSM EDGE
- 6 – 3G
- 7 – 4G
- 8 – WIFI
- 9 – Ethernet
- 10 – Bluetooth
- 11 - NFC

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam as tecnologias de comunicação disponíveis no dispositivo.
- Em caso de erro, retorna ”nil”, uma mensagem de erro e o código do erro.
- *Apêndice A – Constantes e Strings ISO*).
- **datetime:** data e hora a serem colocados.

Retornos:

- em caso de sucesso, retorna os dados do campo (apenas os campos que o bit ISO de datetime contém serão retornados)
- em caso de erro, retorna “**nil”,** mais uma mensagem de erro, mais o código do erro.

4.17.9 iso:assemble (header)

Monta um buffer, no formato ISO8583, com as propriedades previamente setadas e os dados passados em “**header**”.

Parâmetros:

- **header**: uma tabela com os campos “**tpdu_id**”, “**destiny**”, “**origin**”.

Retornos:

- retorna o buffer montado ou “**nil”,** mais uma mensagem de erro, mais o código do erro.

Obs: Uma mensagem ISO com valores indefinidos para classe, função ou iniciador de transação não pode ser montada.

Os campos de header são strings numéricas e tamanho fixo: **tpdu_id**: 2; **destiny**: 4; **origin**: 4.

4.17.10 iso.disassemble (pis, ptimeout)

Desmonta uma lista de bits ISO a partir de um fluxo de entrada de dados, criando uma nova estrutura ISO.

Parâmetros:

- **pis**: fluxo de entrada de dados;
- **ptimeout**: tempo máximo a esperar pela mensagem ISO (em milissegundos). A depender do fluxo de dados (ex.: arquivo), o parâmetro timeout será desprezado.

Retornos:

- retorna uma tabela representando o cabeçalho da mensagem e preenche os campos de bits da mensagem ISO. Em caso de erro, retorna "**nil"**, mais mensagem de erro e código do erro.

Exemplo de uso:

```
display.print("Teste de iso")
local data = {}
    data.year = 2006
    data.month = 04
    data.day = 18
    data.hour = 20
    data.min = 06
    data.sec = 10
local header = {}
    header.origin = "1232"
    header.destiny = "4565"
    header.tpdu_id = "78"
local ret, err = iso.create()
local ret2, err2 = iso.create()
local buffer, err3
if ret ~= nil then
    ret:class(ISO_CLASS_AUTHORIZATION)
    ret:fnc(ISO_FNC_NOTIFICATION)
    ret:transorig(ISO_TRANSORIGN_OTHER)
    ret:binaryfield(ISO_BIT_003,"123456")
    ret:datefield(ISO_BIT_012,data)
    buffer = ret:assemble(header)
else
    display.print(err) -- err contém a mensagem de erro
end
if ret2 ~= nil then
    if buffer ~= nil then
        -- abre um fluxo de entrada para a mensagem gerada acima
        local stris = streams. openstringis(buffer)
        local header2 = ret2:disassemble(stringis, 2000)
        local valor = ret2:class()
        printer.print("CLASS_VALUE:")
        printer.print(valor)
        printer.print("HEADER_VALUES:")
        printer.print(header2.origin)
        printer.print(header2.destiny)
        printer.print(header2.tpdu_id)
    else
        printer.print(err3) - err3 contém a mensagem de erro
    end
    iso.destroy(ret)
    iso.destroy(ret2)
else
   printer.print(err2) -- err2 contém a mensagem de erro
end
```

4.18 Display

Essa biblioteca é um meio mais direto de acessar a tela do POS. Também é possível imprimir textos na tela do POS através das funções “**print**” e “**io.write**”.

4.18.1 display.print (arg [,line [,col]])

Imprime uma string na tela do POS atualizando a posição de seu “cursor”. O caractere ‘\n’ causa uma quebra de linha, sendo os outros caracteres especiais ignorados.

Parâmetros:

- **arg**: seqüência de caracteres a ser impressa na tela (caso esse parâmetro não seja uma string, então a chamada equivale à **display.print (tostring(arg) [,line [,col]]);**
- **line**: linha a partir da qual a string “**text**” será impressa (o valor padrão é a posição do “cursor” - relativo às linhas - retornado na última chamada desta função);
- **col**: coluna a partir da qual a string “**text**” será impressa (o valor padrão é 0 (zero)). Caso não queira passar a coluna exata e sim o alinhamento passe as strings:
  - **“left”** – para alinhar à esquerda;
  - **“center”** – para centralizar;
  - **“right”** – para alinhar à direita.

Retornos:

- retorna um par (linha, coluna) indicando a posição atual do “cursor” na tela. Caso o “cursor” chegue ao final da tela e nada mais possa ser impresso o “cursor” vai para a posição (0, 0) .

Obs: Caso se tente imprimir fora do display, nada será exibido e o retorno será o “cursor” vai para a posição (0, 0).

4.18.2 display.font ([font])

Modifica a fonte do display e retorna a fonte corrente.

Parâmetros:

- **font**: as fontes disponíveis são: “small”, “normal”, “large” e “large_bold". Se esse parâmetro não for passado, a fonte não será modificada.

Retornos:

- em caso de sucesso, temos a fonte corrente como retorno;
- em caso de falha, temos “**nil”**, mais uma mensagem de erro e mais o código do erro como retorno.

4.18.3 display.printimage (image, x, y)

Imprime uma imagem (carregada por “os.loadimage”) na tela do POS.

Parâmetros:

- **image**: tabela contendo os campos “width”, “height” e “data”;
- **x**: posição horizontal da imagem;
- **y**: posição vertical da imagem.

Retornos:

- retorna **true** em caso de sucesso ou **false** em caso de falha.

4.18.4 display.clear ([line])

Limpa a tela do POS. Se for chamada sem parâmetros, limpa toda a tela e posiciona o cursor para a posição (0,0).

Parâmetros:

- **line**: indica a linha a ser limpa.

Retornos:

- (sem retornos)

4.18.5 display.geometry ()

Retorna as dimensões da tela do POS (colunas x linhas).

Parâmetros:

- (sem parâmetros).

Retornos:

- retorna a dimensão (colunas, linhas) da tela do POS.

4.18.6 display.pixels ()

Retorna as dimensões da tela do POS em pixels (largura x altura).

Parâmetros:

- (sem parâmetros).

Retornos:

- retorna a dimensão (largura, altura) da tela do POS.

4.18.7 display.drawline (x1, y1, x2, y2)

Desenha uma linha no display.

Parâmetros:

- **x1**: coordenada x do primeiro ponto;
- **y1**: coordenada y do primeiro ponto;
- **x2**: coordenada x do segundo ponto;
- **y2**: coordenada y do segundo ponto.

Retornos:

- Retorna **true** em caso de sucesso. Caso contrário, retorna "**nil"**, mais mensagem de erro e código do erro.

4.18.8 display.drawpoint (x, y)

Desenha um ponto no display.

Parâmetros:

- **x**: coordenada x do ponto;
- **y**: coordenada y do ponto;

Retornos:

- Retorna **true** em caso de sucesso. Caso contrário, retorna "**nil"**, mais mensagem de erro e código do erro.

4.18.9 display.drawrect (x, y, width, height)

Desenha um retangulo no display.

Parâmetros:

- **x**: coordenada x do vértice superior esquerdo;
- **y**: coordenada y do vértice superior esquerdo;
- **width**: largura do retangulo;
- **height**: altura do retangulo.

Retornos:

- Retorna **true** em caso de sucesso. Caso contrário, retorna "**nil"**, mais mensagem de erro e código do erro.

4.18.10 display.invertcolors ([value])

Inverte as cores do display caso o parâmetro "**value**" seja passado, caso contrário, simplesmente devolve um booleano indicando se as cores estão invertidas. Caso o "**value**" seja **true** então a cor de fundo passa a ser preto enquanto a cor da caneta passa a ser branco. Essa inversão afeta as cores de todas as funções que desenham na tela, exceto **display.printimage** [RF094].

Parâmetros:

- **value**: se for **true** inverte as cores do display.

Retornos:

- Retorna **true** caso as cores estejam invertidas, caso contrário, **false**.

Obs.: Chamadas consecutivas a display.invertcolors(true) não desfaz a inversão. Para desfazer a inversão das cores basta chamar display.invertcolors(false)

4.18.11 display.autoflush ([value])

Se “value” for passado como parametro, então a opção de autoflush é modificada. Caso contrário, apenas retorna o valor da opção.

Parâmetros:

- **value**: booleano indicando se a opção de autoflush deve ser habilidata ou não.

Retornos:

- Retorna o valor corrente do autoflush.

4.18.12 display.flush ()

Imprime o conteúdo do buffer do display no display em si. Essa função deve ser chamada quando a opção de autoflush estiver desabilitada e se deseja atualizar o conteúdo do display.

Parâmetros:

(sem parâmetros).

Retornos:

(sem retorno).

4.18.13 display. backlight (percent)

Seta a luminosidade do display do terminal. O parâmetro percent é um valor percentual e, como tal, deve estar entre 0 e 100.

Parâmetros:

- **percent**: valor percentual da luminosidade do display.

Retornos:

- (sem retorno).

4.19 Keyboard

Essa biblioteca é um meio mais direto de acessar o teclado do POS. Também é possível ler do teclado através da função “**io.read**”.

4.19.1 keyboard.getkeystroke ([timeout])

Espera e retorna a tecla pressionada no teclado pelo usuário.

Parâmetros:

- **timeout**: tempo máximo de espera pela tecla em segundos (se não for passado como parâmetro, espera para sempre).

Retornos:

- em caso de sucesso, retorna número contendo o valor da tecla pressionada;

- em caso de erro ou timeout, retorna “**nil**”.

Obs.: Para realizar comparações podem-se usar as constantes abaixo.

KEY_CANCEL, KEY_CLEAR, KEY_UP, KEY_DOWN, KEY_MENU, KEY_STAR, KEY_HASH, KEY_ENTER, KEY_ONE, KEY_TWO, KEY_THREE, _KEY_FOUR, KEY_FIVE, KEY_SIX, KEY_SEVEN, KEY_EIGHT, KEY_NINE, KEY_ZERO, KEY_F2.

4.19.2 keyboard.getkeylabel (key)

Retorna o rótulo de uma tecla a partir do seu valor (“**key**”).

Parâmetros:

- **key**: o valor da tecla.

Retornos:

- em caso de sucesso, retorna uma string associada à tecla;
- em caso de erro, retorna “**nil**”.

4.19.3 keyboard.buzzer (length, frequency)

Ativa o sonorizador do teclado.

Parâmetros:

- **length**: tempo, em milissegundos, de duração do som (pode-se usar as constantes BEEP_CLICK, BEEP_SHORT e BEEP_LONG);
- **frequency**: freqüência do som, em Hertz (pode-se usar as constantes BEEP_LOW, BEEP_MIDTONE, BEEP_HIGH). Ao informar diretamente o valor, lembre-se de que o ouvido humano só escuta sons entre 20Hz e 20.000Hz. Caso a plataforma não suporte a freqüência passada como argumento ela irá tocar a freqüência mais próxima suportada pela plataforma.

Retornos:

- em caso de sucesso, retorna “**true**”;
- em caso de erro, retorna “**nil”,** mais uma mensagem de erro, mais o código do erro.

4.19.4 keyboard.beepon ([value])

Caso seja chamada com argumentos, ativa/desativa o beep após uma tecla ser pressionada. Caso contrário apenas retorna se esta opção está ativada ou desativada.

Parâmetros:

- **value**: booleano indicando onde “**true**” ativa e “**false**” desativa os beeps.

Retornos:

- retorna “**true**” caso o beep esteja ativado e “**false**” caso contrário.

4.19.5 keyboard. getkeystrokenb ([timeout])

Esta função tem o mesmo comportamento da funcção keyboard.getkeystroke, com a única diferença que ela desconsidera teclas que tenham sido pressionadas pelo usuário

antes da chamada da função. Ou seja, ela utiliza descarta o buffer de teclas pressionadas anteriormente.

Parâmetros:

- **timeout**: tempo máximo de espera pela tecla em segundos (se não for passado como parâmetro, espera para sempre).

Retornos:

- em caso de sucesso, retorna número contendo o valor da tecla pressionada;
- em caso de erro ou timeout, retorna "**nil**".

Obs.: Para realizar comparações podem-se usar as constantes abaixo.

KEY_CANCEL, KEY_CLEAR, KEY_UP, KEY_DOWN, KEY_MENU, KEY_STAR, KEY_HASH, KEY_ENTER, KEY_ONE, KEY_TWO, KEY_THREE, _KEY_FOUR, KEY_FIVE, KEY_SIX, KEY_SEVEN, KEY_EIGHT, KEY_NINE, KEY_ZERO, KEY_F2.

4.20 Arquivos de Propriedades

Essa biblioteca descreve funções referentes aos arquivos de propriedades.

4.20.1 property.open(filename[, location])

Abre um arquivo de propriedades especificado por filename.

Parâmetros:

- **filename**: nome do arquivo de propriedades a ser aberto.
- **location:** localização do arquivo de propriedades.
    - "local" – o filename é considerado a partir do diretório da aplicação (valor padrão);
    - "shared" – o filename é considerado a partir do diretório de arquivos globais do lua (/lua/shared/);
    - "cfg" – o filename é considerado a partir do diretório padrão para arquivos de configuração (/cfg/).

Obs.: Tanto para "shared" quando para "cfg" o filename deve conter apenas o nome do arquivo.

Retornos:

- em caso de sucesso, retorna um handle para o property especificado no parâmetro.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.20.2 property.close (prop)

Equivalente a "prop:close ()". Fecha o arquivo de propriedade aberto, liberando os recursos alocados por ele.

Parâmetros:

- **prop**: arquivo de property a ser fechado.

Retornos:

- em caso de sucesso, retorna "**true**";
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.20.3 property:getstring(key)

Procura a chave **key** dentre as propriedades existentes no property e retorna o valor da string correspondente a propriedade.

Parâmetros:

- **key**: o valor da chave a ser procurada.

Retornos:

- em caso de sucesso, retorna uma string com o valor da propriedade;
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.20.4 property:getnumber(key)

Procura a chave **key** dentre as propriedades existentes no property e retorna o valor correspondente a propriedade.

Parâmetros:

- **key**: o valor da chave a ser procurada.

Retornos:

- em caso de sucesso, retorna o valor da propriedade;
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.20.5 property:getboolean(key)

Procura a chave **key** dentre as propriedades existentes no property e retorna o valor "**true"** ou "**false"** correspondente à propriedade.

Parâmetros:

- **key**: o valor da chave a ser procurada.

Retornos:

- em caso de sucesso, retorna "**true"** ou "**false";**

  em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

Observação: os valores "0" e "false" mapeiam no booleano "**false**", o restante das possibilidades será mapeada para o booleano "**true**".

Exemplo de uso:

```
local prop = property.open("config.txt")
if prop ~= nil then
    valor = prop:getstring("chave")
    property.close(prop)
    display.print(valor)
else
    display.print("erro ao carregar propriedades")
end
```

4.21 Interface de Usuário - UI

Essa biblioteca descreve funções utilizadas na interação com o usuário. Utiliza as bibliotecas encontradas na API de display e teclado para fornecer funções mais alto nível.

4.21.1 ui.textfield(title, label, [maxlenght [,minlenght [,ispassword]]])

Cria um Textfield (caixa de texto) que pode ser mostrada ao usuário para entrada de dados. Demais propriedades do TextField devem ser modificadas usando suas funções.

Parâmetros:

- **title**: título do TextField;
- **label**: rótulo do TextField;
- **maxlenght**: quantidade máxima de caracteres suportados (valor padrão = 16);
- **minlenght**: quantidade mínima de caracteres que o usuário deve entrar (valor padrão = 0);
- **ispassword**: booleano indicando se é um campo de senha (valor padrão = false).

Retornos:

- em caso de sucesso, retorna um handle para o TextField criado.

  em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.21.2 ui.transient(title, message, duration)

Mostra um alerta transiente para o usuário como texto de "**message**", por um período igual a "**duration**".

Parâmetros:

- **title**: título do alerta;
- **message**: mensagem do alerta;
- **duration**: duração em milisegundos que deve ser mostrado o alerta;

Retornos:

- em caso de sucesso, "**true"**.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.21.3 ui.message(title, message [,screennumber [,titlealign [,messagealign [,messagevalign]]]])

Mostra uma mensagem para o usuário e retorna o fluxo de execução, não assumindo o controle da thread.

Parâmetros:

- **title**: título da mensagem (para exibir uma mensagem sem título passe uma string vazia);
- **message**: texto a ser exibido;
- **screennumber**: número da tela (use UI_NOSCREENNUMBER se não houver número);
- **titlealign**: alinhamento do título (“left”, “center”, “right”);
- **messagealign**: alinhamente do conteúdo da mensagem (“left”, “center”, “right”);
- **messagevalign**: alinhamente vertical do conteúdo da mensagem (“top”, “middle”, “bottom”).

Retornos:

- em caso de sucesso, retorna “**true**”.
- em caso de erro, retorna “**nil”,** mais uma mensagem de erro, mais o código do erro.

4.21.4 ui.menu(title,options,[shownumbers])

Cria um menu de opções que pode ser mostrado ao usuário para seleção de opções. Demais propriedades do Menu devem ser modificadas usando suas funções.

Parâmetros:

- **title**: texto título do menu de opções;
- **options**: opções do menu;
- **shownumbers**: mostrar as opções de forma numerada.(“**false**” para apenas listá-las).

Retornos:

- em caso de sucesso, retorna um handle para o menu criado.
- em caso de erro, retorna “**nil”,** mais uma mensagem de erro, mais o código do erro.

Obs.: Se o parâmetro **shownumbers** não for passado, seu valor default será “**true**”.

4.21.5 ui.destroy(uihandle)

Destrói o componente de menu passado como parâmetro. Este pode ser um menu ou TextField. Desta forma, desaloca os recursos do componente.

Parâmetros:

- **uihandle**: TextField ou Menu a ser destruído.

Retornos:

- (sem retorno).

4.22 UI – TextField

Parte da Biblioteca de UI relacionada às funções de TextField.

4.22.1 textfield:text ([text])

Modifica a propriedade text do TextField e retorna o seu valor. Caso o parâmetro opcional ***text*** não seja passado, simplesmente retorna o conteúdo da caixa de texto.

Parâmetros:

- **text**: novo conteúdo da caixa de texto.

Retornos:

- o valor do text em caso de sucesso.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.22.2 textfield:maxlenght()

Retorna a quantidade máxima de caracteres que o TextField suporta.

Parâmetros:

- **(sem parâmetros)**.

Retornos:

- em caso de sucesso, retorna-se o valor do maxlenght;
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.22.3 textfield:pattern(pattern, [jokechar])

Cria um formatador de string a ser usado junto ao TextField.

Parâmetros:

- **pattern**: padrão de formatação;
- **jokechar**: caractere coringa (máscara que vai ser substituída), o valor padrão é '#';

Retornos:

- em caso de sucesso, "**true**".
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

Obs.: Caso o valor da passado de pattern seja "**nil**", a mascara deixará de ser exibida.

4.22.4 textfield:show([timeout])

Mostra o TextField para o usuário. A função assume o controle da execução da thread até que o usuário aceite ou rejeite a tela.

Parâmetros:

- timeout – tempo em segundos que o textfield permanece na tela caso o usuário não entre com dados. Se o parâmetro não for passado, espera para sempre.

Retornos:

- retorna "**true**" caso seja aceita e "**false**" caso seja rejeitada. Caso ocorra timeout, o retorno é "**nil**" + código de erro (*-5*) + mensagem de erro ("timeout").

4.22.5 textfield:align(halign [, valign])

Altera a disposição do campo de texto na tela. Para alinhar o conteúdo do TextField, ver [4.22.6].

Parâmetros:

- **halign:** alinhamento horizontal ("left", "center", "right");
- **valign:** alinhamento vertical ("top", "middle", "botton"). Se "**valign**" não for passado então terá o valor "middle".

Retornos:

- retorna "**true**" se executou com sucesso ou "**nil**", mais uma mensagem descrevendo o erro, mais o código do erro.

4.22.6 textfield:aligntext(align)

Altera a disposição do conteúdo do campo de texto.

Parâmetros:

- **align:** alinhamento do conteúdo do campo de texto ("left", "right").

Retornos:

- retorna "**true**" se executou com sucesso ou "**nil**", mais uma mensagem descrevendo o erro, mais o código do erro.

4.22.7 textfield:alignlabel(align)

Altera a disposição do label do campo de texto.

Parâmetros:

- **align:** alinhamento do label ("left", "center", "right").

Retornos:

- **retorna** "**true**" se executou com sucesso ou "**nil**", mais uma mensagem descrevendo o erro, mais o código do erro.

4.22.8 textfield:screennumber(number)

Modifica o número da tela do TextField.

Parâmetros:

- **number**: novo número da tela (caso queira que a tela não tenha número passe UI_NOSCREENNUMBER).

Retornos:

- retorna "**true**" em caso de sucesso;
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.22.9 textfield:type([type])

Modifica a propriedade **type** do TextField e retorna o seu valor. Caso o parâmetro opcional type não seja passado, simplesmente retorna o tipo atual.

Parâmetros:

- **type**: o novo tipo de conteúdo da caixa de texto. Pode ser uma das seguintes strings: "phone", "number", "ip" ou "text"

Retornos:

- o valor de type em caso de sucesso.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.23 UI - Menu

Parte da Biblioteca de UI relacionada às funções de menu.

4.23.1 menu:accepted()

Retorna o valor do item selecionado no momento.

Parâmetros:

- (sem parâmetros)

Retornos:

- retorna o valor do índice selecionado em caso de sucesso;
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.23.2 menu:show([timeout])

Mostra o menu para o usuário. A função assume o controle da execução da thread até que o usuário aceite ou rejeite a tela.

Parâmetros:

- timeout – tempo em segundos que o menu permanece na tela caso o usuário não entre com dados. Se o parâmetro não for passado, espera para sempre.

Retornos:

- retorna "**true**" caso seja aceita e "**false**" caso seja rejeitada. Caso ocorra timeout, o retorno é "**nil**" + código de erro (*-5*) + mensagem de erro ("timeout").

4.23.3 menu:aligntitle(align)

Altera a disposição do título do menu.

Parâmetros:

- **align:** alinhamento do título ("left", "center", "right").

Retornos:

- retorna "**true**" se executou com sucesso ou "**nil**", mais uma mensagem descrevendo o erro, mais o código do erro.

4.23.4 menu:alignoptions(align)

Altera a disposição das opções do menu.

Parâmetros:

- **align:** alinhamento das opções ("left", "center", "right").

Retornos:

- retorna "**true**" se executou com sucesso ou "**nil**", mais uma mensagem descrevendo o erro, mais o código do erro.

4.23.5 menu:screennumber(number)

Modifica o número da tela do Menu.

Parâmetros:

- **number**: novo número da tela (caso queira que a tela não tenha número passe UI_NOSCREENNUMBER).

Retornos:

- retorna "**true**" em caso de sucesso;
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

Exemplo de uso:

```
function func1()
    ui.message("func1", "pressione uma tecla")
    keyboard.getkeystroke()
end
function func2()
    display.print("func2")
    local tf = ui.textfield("Codigo")
    tf:minlength(3) -- entrar no mínimo 3 caracteres
    if tf:show() then
        display.print(tf:text())
    else
        display.print("rejected")
    end
    ui.desrtoy(tf)
end

funcoes = {func1, func2}
menu = ui.menu("Menu Demo", {"Opcao1", "Opcao2", "Sair"})
aceitou = ui.transient("HiperFlex", "Bem Vindo", 2, m)
while true do
    if aceitou then
        local idx = menu:accepted()
        if idx == 3 then -- Sair
            break
        end
        funcoes[idx]()
    end
    aceitou = menu:show()
end
ui.destroy(menu)
```

4.24 Compressão

Essa biblioteca descreve funções utilizadas na compressão de dados.

4.24.1 compression.compress(inputfile, outputfile)

Gera um arquivo compactado a partir do arquivo passado.

Parâmetros:

- **inputfile**: nome do arquivo a ser compactado. Se o arquivo já estiver compactado, é considerado como caso de erro;
- **outputfile**: nome do arquivo de saída que recebe os dados compactados. Se o arquivo já existir, será sobrescrito.

Retornos:

- em caso de sucesso, retorna **true**.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.24.2 compression.decompress(inputfile, outputfile)

Descompacta um arquivo a partir do arquivo passado.

Parâmetros:

- **inputfile**: nome do arquivo a ser descompactado. Se o arquivo não estiver compactado, é considerado como caso de erro;
- **outputfile**: nome do arquivo de saída. Se o arquivo já existir, será sobreposto.

Retornos:

- em caso de sucesso, retorna **true**.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.25 Funções Utilitárias

Essa biblioteca descreve funções gerais que facilitam o uso de outras APIs.

4.25.1 util.asciintobcd(valor, indice)

Transforma os **"indice"** primeiros caracteres de "**valor**" em BCD.

Parâmetros:

- **valor**. String ascii a ser transformada em bcd.
- **indice**. Quantidade de caracteres a serem convertidos.

Retornos:

- em caso de sucesso, retorna uma string com o "**valor**" convertido em BCD.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.25.2 util.asciitobcd(valor)

Transforma "**valor**" de ASCII em BCD.

Parâmetros:

- **valor**. String ASCII a ser transformada em BCD.

Retornos:

- em caso de sucesso, retorna uma string com o "**valor**" convertido em BCD.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.25.3 util.bcdtoascii(valor)

Transforma "**valor**" de BCD em ASCII.

Parâmetros:

- **valor**. String BCD a ser transformada em ASCII.

Retornos:

- em caso de sucesso, retorna uma string com o "**valor**" convertido em ASCII.

- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.25.4 util.bytearray(size [,fill])

Cria um array de bytes com tamanho inicial "**size**".

Parâmetros:

- **size**: tamanho inicial do array;
- fill: valor com o qual o array será preenchido inicialmente.

Obs.: Caso seja passada uma string com parâmetro, o array terá o tamanho e conteúdo da mesma.

Retornos:

- em caso de sucesso, retorna o handle do array de bytes.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.25.5 bytearray:get(posição)

Retorna o byte localizado em "**posição**".

Parâmetros:

- **posição**: posição do array a ser retornada (o primeiro byte do array possui índice 1).

Retornos:

- em caso de sucesso, retorna o valor de posição.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.25.6 bytearray:set(indice,valor)

Seta o "**valor**" na posição "**índice**".

Parâmetros:

- **valor**: valor a ser setado.
- **indice**: posição do array a ser modificado (o primeiro byte do array possui índice 1).

Retornos:

- em caso de sucesso, retorna "**true**".
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.25.7 bytearray:size()

Retorna o tamanho do array de bytes.

Parâmetros:

(sem parâmetros).

Retornos:

- o tamanho do array de bytes.

4.25.8 util.timer(count)

Cria um timer, contador de tempo. Sua função é marcar quanto tempo passou desde sua criação e quanto tempo falta para o valor de "**count**" zerar.

Parâmetros:

- **count**: tempo em milisegundos que serve como base para a função timer:reamining [RF172]. Caso se deseje criar um timer infinito utilize a constante TIMER_INFINITE.

Retornos:

- em caso de sucesso, um objeto timer.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.25.9 timer:remaining()

Retorna quanto tempo falta para o contador zerar. A conta realizada é similar à "count" (ver [4.25.8]) menos timer:elapsed [4.25.9].

Parâmetros:

- sem parâmetros

Retornos:

- Retorna quanto tempo falta para o contador zerar, em milisegundos.
- Caso o timer tenha sido criado utilizando a constante TIMER_INFINITE, então esta função sempre retornará 1000 (1 segundo).
- Caso o contador do timer tenha zerado, então esta função retornará TIMER_EXPIRED

4.25.10 timer:elapsed()

Retorna quanto tempo passou desde a criação do timer.

Parâmetros:

- sem parâmetros

Retornos:

- Retorna quanto tempo se passou desde a criação do timer, em milisegundos.

4.26 Criptografia

Essa biblioteca descreve funções utilizadas para criptografia.

4.26.1 Base 64

Esta seção lista as funções referentes à codificação base64. O seguinte trecho de código mostra como codificar e decodificar uma string base64.

```
enc = base64.encode("abc");

printer.print(enc); -- imprime "YWJj"

dec = base64.decode(enc);

printer.print(dec); -- imprime "abc"
```

4.26.1.1 base64.encode(bytestring)

Codifica a string binária passada por parâmetro utilizando base-64.

Parâmetros:

- **bytestring**. String binária a ser codificada.

Retornos:

- em caso de sucesso, retorna uma string codificada em base-64.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.26.1.2 base64.decode(string)

Decodifica a string base-64 passada por parâmetro, transformando numa string binária.

Parâmetros:

- **string**. String em base-64 a ser decodificada.

Retornos:

- em caso de sucesso, retorna uma string (binária) decodificada.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.26.2 Hash SHA-1

Esta seção lista a função referente ao hash SHA-1. O seguinte trecho de código mostra como calcular um hash SHA-1.

```
str1 = "hello world";
str2 = "Hello world";

hash1 = sha1.calculate(str1);
-- hash1:
-- 2A AE 6C 35 C9 4F CF B4 15 DB E9 5F 40 8B 9C E9 1E E8 46 ED

hash2 = sha1.calculate(str2);
-- hash2:
-- 7B 50 2C 3A 1F 48 C8 60 9A E2 12 CD FB 63 9D EE 39 67 3F 5E
```

4.26.2.1 sha1.calculate(bytestring)

Calcula o hash SHA-1 da string binária.

Parâmetros:

- **bytestring**. String binária cujo hash será calculado.

Retornos:

- em caso de sucesso, retorna uma string binária representando o hash (sempre de tamanho 20).
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.26.2.2 sha1.chaininit()

Inicializa o cálculo incremental do hash SHA-1.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, retorna **true**.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.26.2.3 sha1.chainupdate(bytestring)

Realiza a atualização do cálculo de hash SHA-1 em andamento.

Parâmetros:

- **bytestring**. bloco de dados que será submetido ao cálculo SHA-1.

Retornos:

- em caso de sucesso, retorna **true**.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.26.2.4 sha1.chainresult()

Retorna o resultado do cálculo incremental de hash SHA-1.

Parâmetros:

(sem parâmetros)

Retornos:

- em caso de sucesso, retorna uma string binária representando o hash (sempre de tamanho 20).
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.26.3 Checksum CRC

Esta seção lista a função referente ao CRC32. O seguinte trecho de código mostra como calcular um checksum CRC32.

```
crc = crc32.calculate("The quick brown fox jumped over the lazy dog.")
print(string.format("crc: %X %X %X %X", string.byte(crc, 4), string.byte(crc, 3), string.byte(crc, 2),
string.byte(crc, 1)))
-- crc: 82 A3 46 42
```

4.26.3.1 crc32.calculate(bytestring [, alternative])

Calcula o CRC32 checksum da string binária.

Parâmetros:

- **bytestring**: string binária cujo CRC32 será calculado;
- **alternative**: booleano indicando se deve ser usada a tabela alternativa definida pela E-Capture para o cálculo do CRC32.

Retornos:

- em caso de sucesso, retorna uma string binária representando o checksum (sempre de tamanho 4).
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.26.4 Criptografia RSA

Esta seção lista as funções referentes a RSA. O seguinte trecho de código mostra como carregar uma chave, criptografar, decriptografar, assinar e verificar uma assinatura.

```
-- carregando chaves
pvk = rsa.loadkey("ingpvt64.key"); -- private key
pbk = rsa.loadkey("ingpbl64.key"); -- public key

-- criptografando com chave pública
ciphered = pbk:encrypt("abc");

-- descriptografando
deciphered = pvk:decrypt(ciphered);
printer.print(deciphered); -- imprime "abc"

-- assinando com chave privada
signature = pvk:sign("abc");

-- verificando assinatura
isvalid = pbk:verify("abc", signature);

if isvalid then
  printer.print("signature is valid");
else
  printer.print("signature is not valid");
end

-- descarregando as chaves
pvk:unload();
pbk:unload();
```

4.26.4.1 rsa.loadkey(keypath)

Carrega uma chave RSA apontada pelo caminho *keypath*. Esta chave precisa estar no formato HiperFlex0.

Parâmetros:

- **keypath**. Caminho da chave a ser carregada.

Retornos:

- em caso de sucesso, retorna a chave RSA carregada.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.26.4.2 key:unloadkey()

Descarrega a chave RSA da memória.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, **true**.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.26.4.3 key:encrypt(bytestring)

Criptografa a string binária passada por parâmetro utilizando a chave pública referenciada por *key*.

Parâmetros:

- **bytestring**. String binária a ser criptografada.

Retornos:

- em caso de sucesso, retorna a string criptografada.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.26.4.4 key:decrypt(bytestring)

Descriptografa a string binária passada por parâmetro utilizando a chave privada referenciada por *key*.

Parâmetros:

- **bytestring**. String binária a ser descriptografada.

Retornos:

- em caso de sucesso, retorna uma string binária representando o dado original.
- em caso de erro, retorna "**nil",** mais uma mensagem de erro, mais o código do erro.

4.26.4.5 key:sign(hashstring)

Assina o dado passado em *hashstring* utilizando a chave privada *key*. Um hash SHA-1 pode ser obtido por **sha1.calculate**.

Parâmetros:

- **hashstring**. O hash a ser assinado.

Retornos:

- em caso de sucesso, retorna uma string binária representando a assinatura do hash pela chave privada *key*.
- em caso de erro, retorna "nil", mais uma mensagem de erro, mais o código do erro.

4.26.4.6 key:verify(hashstring, signature)

Verifica se a assinatura *signature* bate com o *hashstring* utilizando uma chave pública.

Parâmetros:

- **hashstring**. Hash a ser comparado com o resultado da descriptografia da assinatura.
- **signature**: A assinatura a ser verificada

Retornos:

- se a assinatura for válida, **true**.

- caso contrário, retorna “**nil**”, mais uma mensagem de erro, mais o código do erro.

4.26.5 Criptografia Triple DES

Esta seção lista as funções referentes à criptografia Triple DES. O seguinte trecho de código mostra como gerar uma chave, criptografar, descriptografar, pegar os bytes de uma chave e regerar a chave.

```
-- gerando uma chave aleatoria
key1 = tdes.genkey()

ciphered = key1:encrypt("abc")     -- criptografando
deciphered = key1:decrypt(ciphered) -- descriptografando

printer.print(deciphered) -- imprime "abc"

-- pegandos os bytes da chave (não funciona com chaves MasterKey)
strkey1 = tostring(key1)

-- regerando a chave
key2 = tdes.genkey(strkey1)

deciphered2 = key2:decrypt(ciphered); -- descriptografando

printer.print(deciphered2); -- imprime "abc"
```

4.26.5.1 tdes.genkey([refstring[, mkindex]])

Cria um “**handle**” para uma chave 3-DES (simples ou utilizando “MasterKey”). Caso nenhum parâmetro seja informado, gera uma chave randômica.

Parâmetros:

- **refstring**. string binária que representa a chave 3-DES (precisa ter 16 bytes).
- **mkindex:** índice da MasterKey. Utilize esse parâmetro somente se *refstring* for uma “WorkingKey”, ou seja, uma chave 3-DES que foi criptografada pela MasterKey *mkindex.* Para utilizar a ***MasterKey Fixa*** utilize a constante *MK3DES_INCODE_MK_INDEX.*

Retornos:

- em caso de sucesso, um “**handle**” representando a chave.
- em caso de erro, retorna **“nil”,** mais uma mensagem de erro, mais o código do erro.

4.26.5.2 tdes:encrypt(bytestring[, iv])

Criptografa a string **bytestring** com a chave Triple DES referenciada por **tdes**.

Parâmetros:

- **bytestring**. String binária a ser criptografada.
- **iv:** vetor de inicialização. Esse argumento deve ser passado apenas quando se deseja utilizar criptografia 3DES em modo CBC. Caso não seja passado os dados serão criptografados utilizando 3DES em modo ECB. Um vetor de inicialização é um buffer/string de 8 bytes.

Retornos:

- em caso de sucesso, uma string binária criptografada;
- em caso de erro, retorna **"nil",** mais uma mensagem de erro, mais o código do erro.

4.26.5.3 tdes:decrypt(bytestring[, iv])

Descriptografa a string **bytestring** com a chave Triple DES referenciada por **tdes**.

Parâmetros:

- **bytestring**. String binária a ser descriptografada;
- **iv:** vetor de inicialização. Esse argumento deve ser passado apenas quando o bloco de dados a ser descriptografado foi criptografado utilizando 3DES em modo CBC. Caso não seja passado os dados serão descriptografados utilizando 3DES em modo ECB. Um vetor de inicialização é um buffer/string de 8 bytes.

Retornos:

- em caso de sucesso, uma string binária descriptografada.
- em caso de erro, retorna **"nil",** mais uma mensagem de erro, mais o código do erro.

4.26.5.4 tdes:fencrypt(fin, fout[, iv])

Criptografa o arquivo **fin** com a chave Triple DES referenciada por **tdes** armazenando o resultado em **fout**. Caso **fout** já exista, ele será sobrescrito.

Parâmetros:

- **fin**: caminho do arquivo a ser criptografado;
- **fout**: caminho do arquivo onde os dados criptografados serão armazenados;
- **iv:** vetor de inicialização. Esse argumento deve ser passado apenas quando se deseja utilizar criptografia 3DES em modo CBC. Caso não seja passado os dados serão criptografados utilizando 3DES em modo ECB. Um vetor de inicialização é um buffer/string de 8 bytes.

Retornos:

- em caso de sucesso, **true**.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.26.5.5 tdes:fdecrypt(fin, fout[, iv])

Descriptografa o arquivo **fin** com a chave Triple DES referenciada por **tdes** armazenando o resultado em **fout**. Caso o arquivo **fout** já exista, ele será sobrescrito.

Parâmetros:

- **fin**: caminho do arquivo a ser descriptografado;
- **fout**: caminho do arquivo onde os dados descriptografados serão armazenados;
- **iv:** vetor de inicialização. Esse argumento deve ser passado apenas quando o bloco de dados a ser descriptografado foi criptografado utilizando 3DES em modo CBC. Caso não seja passado os dados serão descriptografados utilizando 3DES em modo ECB. Um vetor de inicialização é um buffer/string de 8 bytes.

Retornos:

- em caso de sucesso, **true**.
- em caso de erro, retorna **"nil",** mais uma mensagem de erro, mais o código do erro.

4.27 Log

Esta seção descreve as funcionalidades de log disponibilizadas ao código LUA. Mais detalhes sobre configuração podem ser encontrados no apêndice C de 0.

Um script Lua pode gerar log tanto um a um, como também em rajadas. Logar um a um significa que o meio de log será aberto e fechado a cada chamada à função de log (caso seja necessário abrir o meio de log, como no caso de arquivo); logar em rajada significa abrir o meio de log e deixá-lo aberto até a explícita invocação à função que fecha o meio de log. O seguinte trecho de código ilustra essas duas maneiras:

```
-- log um a um
log.logsingle("log 1", LOG_FATAL);
log.logsingle("log 2", LOG_DEBUG);


-- log em rajada
log.logburst("log 3", LOG_FATAL);
log.logburst("log 4", LOG_CRITICAL);
-- logs em rajada devem explicitamente chamar log.close
log.close();
```

O primeiro parâmetro de logsingle e logburst refere-se à mensagem a ser logada, enquanto que o segundo parâmetro à severidade do log. As opções para severidades de log são: LOG_FATAL, LOG_ALERT, LOG_CRITICAL, LOG_WARNING, LOG_NOTICE, LOG_INFO e LOG_DEBUG.

4.27.1 log.logsingle(preg, psevty)

Abre o meio de log, loga a string **preg** e fecha o meio.

Parâmetros:

- **preg**: a string a ser logada.
- **psevty**: a severidade do log.

Retornos:

- (sem retornos)

4.27.2 log.logburst(preg, psevty)

Abre o meio de log, caso já não esteja aberto, loga a string **preg** sem fechar o meio. Essa função é indicada para utilizações da API de log em rajadas. Ao final, deve-se chamar a função log.close().

Parâmetros:

- **preg**: a string a ser logada.
- **psevty**: a severidade do log.

Retornos:

- (sem retornos)

4.27.3 log.close()

Fecha o meio de log aberto para logs em rajada.

Parâmetros:

- (sem parâmetros)

Retornos:

- (sem retornos)

4.28 Streams

Essa biblioteca descreve funções utilizadas para se trabalhar com fluxos de dados (streams).

4.28.1 streams.openfileis(ppath)

Abre o arquivo apontado por ppath como um fluxo de dados de entrada.

Parâmetros:

- **ppath**: o caminho do arquivo a ser aberto.

Retornos:

- em caso de sucesso, um fluxo de dados de entrada.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.28.2 streams.opencommis()

Abre um fluxo de dados de entrada para o modem.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, um fluxo de dados de entrada.

- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.3 streams.opengprsis()

Abre um fluxo de dados de entrada para a conexão gprs.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, um fluxo de dados de entrada.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.4 streams.openrs232is(port)

Abre um fluxo de dados de entrada para uma porta serial.

Parâmetros:

- **port:** handle da porta serial aberta

Retornos:

- em caso de sucesso, um fluxo de dados de entrada.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.5 streams.openstringis(pstr)

Abre um fluxo de dados de entrada cujos dados são retirados a partir dos bytes de uma string.

Parâmetros:

- **pstr:** a string de onde os dados do fluxo de dados serão lidos

Retornos:

- em caso de sucesso, um fluxo de dados de entrada.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.6 streams.openbufferedis(pis)

Abre um fluxo de dados de entrada bufferizado. Seus dados são obtidos a partir de **pis**.

Parâmetros:

- **pis:** o fluxo de entrada de dados que alimentará o fluxo de dados bufferizado.

Retornos:

- em caso de sucesso, um fluxo de dados de entrada (bufferizado).

- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.7 streams.openfileos(ppath)

Abre o arquivo apontado por **ppath** como um fluxo de dados de saída.

Parâmetros:

- **ppath:** o caminho do arquivo a ser aberto.

Retornos:

- em caso de sucesso, um fluxo de dados de saída.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.8 streams.opencommos()

Abre um fluxo de dados de saída para o modem.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, um fluxo de dados de saída.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.9 streams.opengprsos()

Abre um fluxo de dados de saída para a conexão gprs.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, um fluxo de dados de saída.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.10 streams.openrs232os(port)

Abre um fluxo de dados de saída para uma porta serial.

Parâmetros:

- **port:** handle da porta serial aberta

Retornos:

- em caso de sucesso, um fluxo de dados de saída.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.11 streams.openstringos()

Abre um fluxo de dados de saída em que os dados são escritos na memória.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, um fluxo de dados de saída.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.12 streams.openbufferedos(pos)

Abre um fluxo de saída bufferizado. Um fluxo bufferizado posterga a escrita dos dados até que uma quantidade razoável esteja disponível. Entretanto, eles podem ser escritos imediatamente mediante a chamada à função :flush.

Parâmetros:

- **pos:** o fluxo de saída de dados em que o fluxo de dados bufferizado irá escrever.

Retornos:

- em caso de sucesso, um fluxo de saída de dados (bufferizado).
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.13 ios:close()

Fecha um fluxo de dados (de entrada ou de saída). Se o fluxo de dados for de arquivo, o arquivo será fechado, se for um fluxo de saída bufferizado, seus dados serão escritos.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, ”**true**”.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.14 is:read (pbytenum, ptimeout)

Lê no máximo pbytenum bytes do fluxo de entrada e retorna como string. A depender do fluxo de dados (ex.: arquivo), o parâmetro timeout será desprezado.

Com um timeout negativo, a função só retorna quando pbytenum bytes forem lidos, não existirem mais bytes no fluxo ou um erro ocorrer.

Parâmetros:

- **pbytenum:** o número máximo de bytes a serem lidos.
- **ptimeout:** o tempo (em milissegundos) máximo a ser esperado pelos bytes.

Retornos:

- em caso de sucesso, uma string representando os dados lidos.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.15 os:write (pstring, ptimeout[, poffset[, plength]])

Escreve a maior quantidade de bytes de pstring que conseguir em ptimeout milissegundos no fluxo de saída. A depender do fluxo de dados (ex.: arquivo), o parâmetro timeout será desprezado.

Parâmetros:

- **pstring:** a string que contém os bytes que serão escritos.
- **ptimeout:** o tempo (em milissegundos) máximo a ser esperado enquanto os bytes são escritos.
- **poffset:** o índice a partir do qual os dados de pstring serão escritos (o primeiro caractere está no índice 1).
- **plength:** a quantidade de bytes a serem escritos (não deve ser maior que pstring:len() – poffset + 1).

Retornos:

- em caso de sucesso, o número de bytes escritos.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.16 stringos:content()

Retorna o conteúdo de um fluxo de saída em string.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, uma string com o conteúdo do que foi escrito no fluxo de saída em string.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.28.17 os:flush(ptimeout)

Descarrega o buffer do fluxo de saída, caso ele esteja bufferizado. A depender do fluxo de dados (ex.: arquivo), o parâmetro ptimeout será desprezado.

Com um timeout negativo, a função só retorna quando não existirem mais bytes a descarregar ou um erro ocorrer.

Um flush ocorre de forma encadeada. Se o fluxo de saída for formado por vários níveis de fluxos de saída, o flush vai sendo transmitido até chegar ao fluxo de saída primário (arquivo, modem ou array de bytes).

Parâmetros:

- **ptimeout:** o tempo (em milissegundos) máximo a ser esperado enquanto os bytes são descarregados.

Retornos:

- em caso de sucesso, o número de bytes descarregados.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.29 Gerenciamento de Processos

Essa biblioteca descreve funções utilizadas para o gerenciamento de threads/ processos.

4.29.1 management.sleep(ptime)

Suspende a execução da thread por **ptime** milissegundos.

Parâmetros:

- **ptime:** o tempo, em milissegundos, que a thread deve ficar suspensa.

Retornos:

- em caso de sucesso, ”**true**”.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.30 Leitora de Código de Barras

Essa biblioteca descreve funções utilizadas para a leitura de código de barras.

4.30.1 barcode.loadconfig([filename])

Configura a leitora a partir de um arquivo de configuração. Caso o **filename** não seja passado, então carrega a configuração padrão.

Parâmetros:

- **filename:** caminho do arquivo de propriedades.

Retornos:

- em caso de sucesso, **true**.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.30.2 barcode.config(key [,value])

Configura uma propriedade da leitora. Se o valor da propriedade nao for informado, retorna o valor setado para atual.

Parâmetros:

- **key:** nome da propriedade.
- **value:** valor da propriedade

Retornos:

- em caso de sucesso, o valor da propriedade '**key**'.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.30.3 barcode.saveconfig(filename)

Salva a configuração atual da leitora no arquivo apontado por **filename**. Se ele já existir, será sobrescrito.

Parâmetros:

- **filename:** caminho do arquivo de propriedades a ser salvo.

Retornos:

- em caso de sucesso, **true**.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

4.30.4 barcode.read(timeout [,maxlen])

Lê um código de barras.

Parâmetros:

- **timeout:** tempo máximo, em milisegundos, de espera pelos dados;
- **maxlen:** quantidade máxima de bytes a serem lidos.

Retornos:

- em caso de sucesso, os dados recebidos em forma de uma string.
- em caso de erro, retorna "**nil**", mais uma mensagem de erro, mais o código do erro.

Obs.: Para exemplos de arquivos de configuração do módulo de código de barras, assim como as possíveis propriedades a serem configuradas, ver Apêndice I – Configuração para leitora de código de barras do Documento de Requisitos HiperFLEX API e HELPER Functions [3].

4.31 EMV

Essa biblioteca descreve funções utilizadas para o uso de chip EMV.

4.31.1 emv.init()

Esta função inicializa a Camada HEMV, e deve ser chamada pela aplicação uma única vez, antes de chamar qualquer outra função do kernel.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, retorna código especifico.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.2 emv.version()

Esta função retorna uma string contendo a versão do Kernel EMV do terminal.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, String de final NULL contendo a versão do Kernel EMV.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.3 emv.checkcard()

Esta função verifica se um cartão está inserido no acoplador de cartão com chip. Ela pode ser chamada a qualquer momento, não interferindo com a transação caso esta esteja em andamento.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, Retorn 0 caso cartao esteja presente.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.4 emv.starttransaction()

Esta função inicializa a Camada HEMV, limpando as variáveis e recursos do processamento, assim como todos os parâmetros definidos pela função HEMV_iDefData devendo ser chamada antes de iniciar cada uma das transações EMV.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, Retorn 0 caso tenha sucesso.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.5 emv.processtransaction()

Esta função efetua os seguintes processamentos regidos pela norma EMV até chegar à primeira decisão do cartão:

Offline Data Authentication;

Processing Restrictions;

Cardholder Verification;

Terminal Risk Management;

Terminal Action Analisys;

Card Action Analisys;

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, Result code + Transaction Result + CVM Result.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.6 emv.completetransaction(onlineResult)

Caso a função HEMV_iProcessTransaction indique que a autorização deverá ser feita pelo Emissor (online), esta função deverá ser chamada ao final para completar a transação, efetuando os seguintes passos regidos pela norma EMV:

Issuer Authentication;

Script Processing;

Completion;

Parâmetros:

- **onlineResult:** Resultado da transação online:
  - 0 Negada pelo host
  - 1 Aprovada pelo host
  - 2 Não foi possível comunicar com o host

Retornos:

- em caso de sucesso, Result Code + Transaction Result.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.7 emv.getcandidatelist()

Esta função deve ser chamada pela aplicação após a inserção da lista de aplicações usando-se chamadas à função HEMV_iDefApp. A Camada HEMV ativa o cartão e efetua o processo de obtenção da lista de candidatos.

Os labels das aplicações candidatas são retornados, juntamente com seus códigos proprietários de referência, já em ordem de prioridade para que a aplicação possa exibir um menu (se necessário). Antes do processamento de HEMV_iGetCandidateList, deve-se fornecer através da função HEMV_iDefData a tag HTAG_ADDTRMCAP (caso já sejam conhecidos da aplicação).

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, Result Code + Numero de Apps + Primeiro Label + Labels.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.8 emv.defapp(ref, data)

Esta função insere uma aplicação conhecida pelo terminal na lista das aplicações que podem participar do processo de seleção no cartão. HEMV_iDefApp pode ser chamada diversas vezes, sendo este processo iniciado por HEMV_iStartTransaction..

Parâmetros:

- **ref:** Código proprietário de referência (0 a 255).
- **data:** AID - identificador da aplicação.

Retornos:

- em caso de sucesso, retorna código especifico.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.9 emv.selectapp(appIndex)

Em HEMV_iSelectApp a Camada HEMV deverá efetuar os seguintes processamentos regidos pela norma:

Initiate Application Processing;

Read Application Data;

Caso o kernel identifique um problema que demande a retirada da aplicação da lista de candidatas para uma nova seleção, essa situação é informada pelo código de retorno HEMVERR_SELECTFAILED de maneira que a aplicação principal chame novamente HEMV_iGetCandidateList..

Parâmetros:

- **appIndex:** Índice da aplicação selecionada (de 0 em diante) conforme seqüência devolvida por HEMV_iGetCandidateList.

Retornos:

- em caso de sucesso, retorna código especifico.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.10 emv.defdata(tag, data)

Esta função insere um parâmetro (objeto) EMV na Camada HEMV.

Esta função pode ser chamada diversas vezes, a qualquer momento, de forma a definir parâmetros necessários ao processamento dos cartões.

Parâmetros:

- **tag:** Tag que identifica o parâmetro sendo inserido (EMV_xxxx).

- **data:** Valor do parâmetro.

Retornos:

- em caso de sucesso, retorna código especifico.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.31.11 emv.getdata(tag)

Esta função obtém um dado (objeto) EMV da Camada HEMV.

Esta função pode ser chamada diversas vezes, a qualquer momento, de forma a obter dados resultantes do processamento dos cartões.

Parâmetros:

- **tag:** Tag que identifica o parâmetro a ser obtido (EMV_xxxx).

Retornos:

- em caso de sucesso, Result Code + Tag data.
- em caso de erro, retorna ”**nil**”, mais uma mensagem de erro, mais o código do erro.

4.32 Multi Aplicação

4.32.1 ms.open(appname)

Abre um serviço externo para futuras chamadas

Parâmetros:

- **appname**: é o nome do aplicativo/serviço (diretório no AMS)

Retornos:

- Retorna um tipo que identifica o estado do serviço externo

4.32.2 ms.call(appstate, params) equivalente a appstate:call(params)

Realiza uma chamada ao serviço externo passado em appstate. É importante observar que o estado do serviço será preservado para futuras chamadas.

Parâmetros:

- **params**: tabela de parâmetros

Retornos:

- Retorna uma tabela contendo o resultado da chamada

4.32.3 ms.close(appstate) equivalente a appstate:close()

Fecha um serviço externo e libera todos os recursos alocados

4.32.4 ms.singlecall(appname, params)

Realiza uma chamada simples a um serviço externo

É equivalente a execução consecutiva das seguintes chamadas:

local appstate = ms.open(appname)

appstate:call(params)

appstate:close()

4.33 Capacidades do dispositivo

4.33.1 device.screen_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da tela do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da tela do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)
  - 2 – Gráfico monocromático
  - 3 – Gráfico colorido

- **pixels_x**: Resolução horizontal (em pixels) da configuração.
- **pixels_y**: Resolução vertical (em pixels) da configuração.
- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela tela do dispositivo.
- **pixels_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da tela do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.33.2 device.screen_get_current_settings()

Retorna a configuração gráfica atual de tela do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da tela do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.screen_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.screen_get_capabilities).**
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.33.3 device.screen_set_current_settings(settings)

Seleciona a configuração gráfica de tela a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a tela do dispositivo deve operar.

  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.screen_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna ”**nil**”, uma mensagem de erro e o código do erro.

4.33.4 device.printer_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica a capacidade gráfica da impressora do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível configuração gráfica da impressora do dispositivo e contém os seguintes campos:

- **struct_size**: Campo interno da plataforma que não deve ser modificado pela aplicação.

  Valor 0(zero) indica que o dispositivo implementa versão antiga desta API.

- **mode**: Modo gráfico. Valores possíveis:
  - 1 – Modo caracter (texto)

- 2 – Gráfico monocromático
- 3 – Gráfico colorido

- **bits_per_pixel**: Quantidade necessária para representar cada pixel. Quanto maior este valor, maior a quantidade de diferentes cores suportadas pela impressora do dispositivo.
- **dots_per_inch**: Quantidade pixels que podem ser "armazenados" por polegada. Quanto maior este valor maior a resolução gráfica da impressora do dispositivo.
- **orientation**: Retrato (valor 1) ou Paisagems (valor 2)
- **bmp_format_mask**: Formatos de arquivo de bitmap suportados nativamente pelo dispositivo para esta configuração gráfica.

  Este campo é uma máscara de bits (32 bits), com cada formato representando um bit.

  Este campo só deve ser utilizado caso struct_size for diferente de zero.

  Valores possíveis:
  - 1 – Windows BMP Monocromático
  - 2 – Windows BMP Colorido
  - 4 – Verifone VMP Monocromático
  - 8 – Verifone VMP Colorido
  - 16 – JPEG
  - 32 – GIF
  - 64 - PNG

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam a capacidade gráfica do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.33.5 device.printer_get_current_settings()

Retorna a configuração gráfica atual da impressora do dispositivo.

Parâmetros:

- **Não tem.**

Retornos:

- Retorna uma tabela que indica a configuração gráfica atual da impressora do dispositivo.

  Os campos desta tabela são iguais aos de um registro retornado pela função **device.printer_get_capabilities** (a tabela retornada deve ser uma cópia de algum dos registros retornados na **device.printer_get_capabilities).**
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.33.6 device.printer_set_current_settings(settings)

Seleciona a configuração gráfica de impressora a ser utilizada pelo dispositivo.

Parâmetros:

- **settings:** Configuração gráfica na qual a impressora do dispositivo deve operar.
  Este parâmetro deve ser igual a algum dos registros retornados pela função **device.printer_get_capabilities.**

Retornos:

- true, em caso de sucesso.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.33.7 device.input_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica os métodos de entrada disponíveis no dispositivo.

Cada registro da tabela é uma tabela contendo um possível método de entrada.

Os seguintes métodos de entrada podem ser retornados:

- 1 – Teclado
- 2 – TouchScreen (ou Mouse)
- 3 – Teclado na tela
- 4 – Teclado externo
- 5 – Impressão digital
- 6 – Voz
- 7 – Leitor de trilhas magnéticas
- 8 – Leitor de cartão com chip
- 9 – Porta serial
- 10 – Porta serial USB

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam os mecanismos de entrada disponíveis do terminal.
- Em caso de erro, retorna "**nil**", uma mensagem de erro e o código do erro.

4.33.8 device.comm_get_capabilities()

Retorna uma tabela (que pode estar vazia) que indica as tecnologias de comunicação do dispositivo.

Cada registro da tabela é uma tabela contendo uma possível tecnologia de comunicação.

Os seguintes valores podem ser retornados:

- 1 – Dial síncrono
- 2 – Dial assíncrono
- 3 – GSM CSD
- 4 – GSM GPRS
- 5 – GSM EDGE
- 6 – 3G
- 7 – 4G
- 8 – WIFI
- 9 – Ethernet
- 10 – Bluetooth
- 11 - NFC

Parâmetros:

- **Não tem**

Retornos:

- Tabela (que pode estar vazia) contendo registros que indicam as tecnologias de comunicação disponíveis no dispositivo.
- Em caso de erro, retorna ”nil”, uma mensagem de erro e o código do erro.

5 APÊNDICE A – CONSTANTES E STRINGS ISO

Esta seção descreve as constantes utilizadas na formação de mensagens ISO e as strings retornadas pelas funções class, transorign, fnc. Sendo elas as seguintes:

As versões ISO:

ISO_VER_UNDEF,
ISO_VER_1987,
ISO_VER_1993,
ISO_VER_PRIVATERESERVED_9

As classes ISO:

ISO_CLASS_UNDEF,
ISO_CLASS_AUTHORIZATION,
ISO_CLASS_FINANCIAL,
ISO_CLASS_FILEACTION,
ISO_CLASS_REVERSAL_CHARGEBACK,
ISO_CLASS_RECONCILIATION,
ISO_CLASS_ADMINISTRATIVE,
ISO_CLASS_FEECOLLECTION,
ISO_CLASS_NETWORKMANAGEMENT

As funções ISO:

ISO_FNC_UNDEF,
ISO_FNC_REQUEST,
ISO_FNC_REQUESTRESPONSE,
ISO_FNC_ADVICE,
ISO_FNC_ADVICERESPONSE,
ISO_FNC_NOTIFICATION

As opções para iniciador da transação (*transaction originator*) ISO:

ISO_TRANSORIGN_UNDEF,
ISO_TRANSORIGN_ACQUIRER,
ISO_TRANSORIGN_ACQUIRERREPT,
ISO_TRANSORIGN_CARDISSUER,
ISO_TRANSORIGN_CARDISSUERREPT,
ISO_TRANSORIGN_OTHER,
ISO_TRANSORIGN_OTHERREPT

Os bits ISO:

ISO_BIT_002, // PRIMARY_ACCOUNT_NUMBER,
ISO_BIT_003, // PROCESSING_CODE,
ISO_BIT_004, // AMOUNT_TRANSACTION,
ISO_BIT_005, // AMOUNT_SETTLEMENT,
ISO_BIT_006, // AMOUNT_CARDHOLDER_BILLING,
ISO_BIT_007, // TRANSMISSION_DATE_AND_TIME,
ISO_BIT_008, // AMOUNT_CARDHOLDER_BILLING_FEE,
ISO_BIT_009, // CONVERSION_RATE_SETTLEMENT,
ISO_BIT_010, // CONVERSION_RATE_CARDHOLDER_BILLING,
ISO_BIT_011, // SYSTEM_TRACE_AUDIT_NUMBER,
ISO_BIT_012, // TIME_LOCAL_TRANSACTION,
ISO_BIT_013, // DATE_LOCAL_TRANSACTION,
ISO_BIT_014, // DATE_EXPIRATION,

ISO_BIT_015, // DATE_SETTLEMENT,
ISO_BIT_016, // DATE_CONVERSION,
ISO_BIT_017, // DATE_CAPTURE,
ISO_BIT_018, // MERCHANTS_TYPE,
ISO_BIT_019, // ACQUIRING_INSTITUTION_COUNTRY_CODE,
ISO_BIT_020, // PAN_EXTENDED_COUNTRY_CODE,
ISO_BIT_021, // FORWARDING_INSTITUTION_COUNTRY_CODE,
ISO_BIT_022, // POINT_OF_SERVICE_ENTRY_MODE,
ISO_BIT_023, // CARD_SEQUENCE_NUMBER,
ISO_BIT_024, // NETWORK_INTERNATIONAL_IDENTIFIEER,
ISO_BIT_025, // POINT_OF_SERVICE_CONDITION_CODE,
ISO_BIT_026, // POINT_OF_SERVICE_PIN_CAPTURE_CODE,
ISO_BIT_027, // AUTHORIZATION_IDENTIFICATION_RESP_LEN,
ISO_BIT_028, // AMOUNT_TRANSACTION_FEE,
ISO_BIT_029, // AMOUNT_SETTLEMENT_FEE,
ISO_BIT_030, // AMOUNT_TRANSACTION_PROCESSING_FEE,
ISO_BIT_031, // AMOUNT_SETTLEMENT_PROCESSING_FEE,
ISO_BIT_032, // ACQUIRING_INSTITUTION_IDENT_CODE,
ISO_BIT_033, // FORWARDING_INSTITUTION_IDENT_CODE,
ISO_BIT_034, // PAN_EXTENDED,
ISO_BIT_035, // TRACK_2_DATA,
ISO_BIT_036, // TRACK_3_DATA,
ISO_BIT_037, // RETRIEVAL_REFERENCE_NUMBER,
ISO_BIT_038, // AUTHORIZATION_IDENTIFICATION_RESPONSE,
ISO_BIT_039, // RESPONSE_CODE,
ISO_BIT_040, // SERVICE_RESTRICTION_CODE,
ISO_BIT_041, // CARD_ACCEPTOR_TERMINAL_IDENTIFICACION,
ISO_BIT_042, // CARD_ACCEPTOR_IDENTIFICATION_CODE,
ISO_BIT_043, // CARD_ACCEPTOR_NAMELOCATION,
ISO_BIT_044, // ADITIONAL_RESPONSE_DATA,
ISO_BIT_045, // TRACK_1_DATA,
ISO_BIT_046, // ADITIONAL_DATA_ISO,
ISO_BIT_047, // ADITIONAL_DATA_NATIONAL,
ISO_BIT_048, // ADITIONAL_DATA_PRIVATE,
ISO_BIT_049, // CURRENCY_CODE_TRANSACTION,
ISO_BIT_050, // CURRENCY_CODE_SETTLEMENT,
ISO_BIT_051, // CURRENCY_CODE_CARDHOLDER_BILLING,
ISO_BIT_052, // PIN_DATA,
ISO_BIT_053, // SECURITY_RELATED_CONTROL_INFORMATION,
ISO_BIT_054, // ADDITIONAL_AMOUNTS,
ISO_BIT_055, // RESERVED_ISO,
ISO_BIT_056, // RESERVED_ISO,
ISO_BIT_057, // RESERVED_NATIONAL,
ISO_BIT_058, // RESERVED_NATIONAL,
ISO_BIT_059, // RESERVED_NATIONAL,
ISO_BIT_060, // RESERVED_PRIVATE,
ISO_BIT_061, // RESERVED_PRIVATE,
ISO_BIT_062, // RESERVED_PRIVATE,
ISO_BIT_063, // RESERVED_PRIVATE,
ISO_BIT_064 // Message authentication code field

Strings de class:

“AUTHORIZATION"
"FINANCIAL"
"FILEACTION"
"REVERSAL_CHARGEBACK"
"RECONCILIATION"
"ADMINISTRATIVE"
"FEECOLLECTION"
"NETWORKMANAGEMENT"
"UNDEFINED"

Strings de fnc:

"REQUEST"
"REQUESTRESPONSE"
"ADVICE"
"ADVICERESPONSE"
"NOTIFICATION"
"UNDEFINED"

Strings de transorign:

```
"ACQUIRER"
"ACQUIRERREPT"
"CARDISSUER"
"CARDISSUERREPT"
"OTHER"
"OTHERREPT"
"UNDEFINED"
```

6 APÊNDICE B – MENSAGENS E CÓDIGOS DE ERRO

Esta seção descreve as mensagens de erro e códigos de erro retornadas pelas funções de lua.

CÓDIGOS GERAIS:

"invalid argument"	-1
"no memory"	-2
"device fault"	-3
"resource allocation"	-4
"timeout"	-5
"small buffer"	-6
"invalid state"	-7

BASE 64:

"invalid char"	-1001

DATA E HORA:

"invalid date”	-2001
"invalid time"	-2002

SISTEMA DE ARQUIVOS:

"file/directory not found"	-3001
"max length"	-3002
"file already open"	-3003
"max files opened"	-3004
"header corrupted"	-3005
"no space"	-3006
"path error"	-3007
"invalid accessmode"	-3008
"invalid description"	-3009
"read only"	-3010
"out of bounds"	-3011
"no access right"	-3012
"path too deep"	-3013
"directory already exists"	-3014
"not empty"	-3015
"is a file"	-3016
"end of file"	-3017
"not global"	-3018
"buffer over flow"	-3019
"is a directory"	-3020
"access forbiden"	-3021
"empty directory"	-3022

DISPLAY

"unknow format"	-5001

KEYBOARD

"key mismatch"	-6001

"key not supported"	-6002
CARTÂO MAGNÉTICO	
"read canceled"	-8001
"swipe error"	-8002
"track error"	-8003
"unavailable track"	-8004
"data parity error”	-8005
"LRC to data checksum error"	-8006
"parity LRC checksum error"	-8007
COMUNICAÇÂO	
"comm property not suported"	-9004
"invalid comm property"	-9005
"no carrier"	-9006
"no dial tone"	-9007
"busy line"	-9008
"no answer"	-9009
"connection lost"	-9010
ARQUIVOS DE PROPRIEDADES	
"key not found"	-10001
UI	
"element not supported"	-11004
"align not supported"	-11005
"no item accepted"	-11006
CRIPTOGRAFIA	
"invalid rsa key"	-12001
"invalid signature"	-12002
"not a private key"	-12003
"not a public key"	-12004
PROPRIEDADES DO SISTEMA	
"invalid system property"	-13001
COMPRESSÃO	
"invalid buffer format"	-14001
ISO	
"invalid bit format"	-15001
"max bit length"	-15002
"bit not set"	-15003
"incomplete iso mensage"	-15004
"invalid TPDU"	-15005
"invalid iso format"	-15006
UTIL	
"invalid BDC char"	-16001

7 APÊNDICE C – PARÂMETROS DE MODEM

Esta seção lista os parâmetros que podem ser utilizados para configurar o modem (chamadas à função comm.config()).

chaves	valores
"baud_rate"	"50", "75", "150", "300", "600", "1200", "2400", "4800", "9600", "19200", "38400", "57600", "76800", "115200"
"parity"	"none", "odd", "even"
"data_size"	"5", "6", "7", "8"
"stop_bits"	"1", "2"
"rbuffsz" "sbuffsz"	"[número]". ex.: "1024"
"dial_mode"	"pulse", "tone"
"wait_dial_tone"	"used", "not_used"
"init_string" "config_string" "pre_dial_fast_string" "pre_dial_std_string" "dial_string"	"[comandos AT]". ex.: "ATX4"
"flow_control"	"none", "xon_xoff"
"x_on_char""x_off_char"	"[caractere]". ex.: "a", "x"
"modulation_protocol"	"v21", "v22", "v23", "v32", "v32b", "v34", "v90", "v90d", "v92u", "k56", "bell103", "bell212", "v29"
"data_link_protocol"	"none", "hdlc_v80", "sdlc", "apacs40", "spdh_base24"
"compression_protocol"	"none", "v42b", "mnp5", "v42b_mnp5"
"err_correction_protocol"	"none", "lapm", "mnp", "lapm_mnp"
"min_rec_speed" "max_rec_speed" "min_send_speed" "max_send_speed"	"300", "1200", "2400", "4800", "7200", "9600", "12000", "14400", "16800", "19200", "21600", "24000", "25333", "26400", "26667", "28000", "28800", "29333", "30667", "31200", "32000", "33333", "33600", "34000", "34667", "36000", "37333", "38000", "38400", "38667", "40000", "41333", "42000", "42667", "44000", "45333", "46000", "46667", "48000", "49333", "50000", "50667", "52000", "53333", "54000", "54667", "56000"

As configurações-padrão podem ser encontradas em Apêndice D - Configuração de modem 0.

8 APÊNDICE D – CONNLIB.LUA

connlib.lua é uma biblioteca para abstrair a forma de conexão a ser utlizada pelo POS.

8.1 connections.load(filename|table)

Cria um handle para uma conexão com as configurações disponívies no arquivo "filename" ou a partir de uma tabela. Esse arquivo deve seguir o formato "chave=valor" como nos exemplos abaixo:

Exemplo de configuração para modem:

```
device=modem
device_config=modem.cfg
dial_number=28765432
dial_prefix=0
```

Arquivo modem.cfg:

```
data_size=8
parity=none
baud_rate=2400
stop_bits=1
flow_control=none
dial_mode=tone
compression_protocol=none
err_correction_protocol=none
modulation_protocol=v22
data_link_protocol=hdlc_v80
```

Exemplo de configuração para RS232:

```
device=rs232
device_config=rs232.cfg
port=COM1
```

Arquivo rs232.cfg:

```
baud_rate=2400
parity=odd
data_size=7
flow_control=none
stop_bits=2
```

Exemplo de configuração usando tabela:

```
conf = {}
conf.device="modem"
conf.device_config="default"
conf.dial_number="28765432"
conf.dial_prefix="0"
```

Obs.: ao utilizar a palavra "default" em "device_config" o dispositivo será configurado com seus valores padrão.

Parâmetros:

- **filename|table**: caminho para o arquivo ou tabela contendo as configurações da conexão.

Retornos:

- em caso de sucesso, retorna o handle (conn) para manipular a conexão.
- em caso de erro, retorna **nil,** mais uma mensagem de erro, mais o código do erro.

8.2 conn:open()

Inicia o processo para estabelecer uma conexão.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, retorna **true**.
- em caso de erro, retorna **nil,** mais uma mensagem de erro, mais o código do erro.

8.3 conn:check()

Retorna o estado da conexão.

Parâmetros:

- (sem parâmetros)

Retornos:

- “connected”: no caso de haver uma conexão estabelecida;
- “waiting”: no caso de uma conexão estar sendo estabelecida;
- “disconnected”: no caso de estar desconectado e não houver nenhum erro associado
- “no carrier”: no caso de não ser detectado o sinal da portadora
- “no dial tone”: no caso de não ser detectado tom de discagem
- “busy line”: se a linha estiver ocupada
- “no answer”: se a ligação não foi atendida
- “connection lost”: se a conexão foi perdida
- “failed”: no caso de falha desconhecida ao estabelecer a conexão.

8.4 conn:openis()

Retorna um Stream de entrada para a conexão. Esse método deve ser chamado apenas após a conexão ter sido estabelecida.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, retorna um Stream de entrada.
- em caso de erro, retorna **nil,** mais uma mensagem de erro, mais o código do erro.

8.5 conn:openos()

Retorna um Stream de saída para a conexão. Esse método deve ser chamado apenas após a conexão ter sido estabelecida.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, retorna um Stream de saída.
- em caso de erro, retorna **nil,** mais uma mensagem de erro, mais o código do erro.

8.6 conn:close()

Fecha a conexão e libera seus recursos.

Parâmetros:

- (sem parâmetros)

Retornos:

- em caso de sucesso, retorna **true**.
- em caso de erro, retorna **nil,** mais uma mensagem de erro, mais o código do erro.

Exemplo de uso:

```
local conn = connections.load("conn.cfg") -- carrega o exemplo 1 acima
conn:open() -- inicia o processo para estabelecer a conexão
while conn:status() == "waiting" do
   os.sleep(200)
end

if (conn:status() == "connected") then
  local istream = conn:openis() -- abre o stream de entrada
  local ostream = conn:openos() -- abre o stream de saída

  --Montando a mensagem ISO: HEADER
  local headerIso = {}
  headerIso.tpdu_id = "60"
  headerIso.origin  = "0000"
  headerIso.destiny = compra.nii

  --Montando a mensagem ISO: Bitmap
  local hIso = iso.create()
  hIso:typeid("0200")
  hIso:binaryfield(ISO_BIT_003,"300000")
  -- (...)

  -- Enviando a mensagem
  local buff = hIso:assemble(headerIso)
  ostream:write(buff, 10000)
  ostream:flush(10000)

  --Recebendo mensagem
  headerIso, msgErro = hIso:disassemble(ostream, 20000)
```

```
    -- (realiza o processamento)

    ostream:close()
    istream:close()
    conn:close()
end
```

9 REFERÊNCIAS

[1] Lua 5.2 Reference Manual: http://www.lua.org/manual/5.2/

[2] System Architecture Specification; Version 01.01; 15/09/2005;

/hiperflex/produtos/hiperflex/documentos/aep/HFLEX-ARQ.doc

[3] Documento de Requisitos HiperFLEX API e HELPER Functions; Version 11.00; 27/07/2007;

/hiperflex/produtos/hiperflex/documentos/requisitos/HFLEX-APIHELPER-SRS.doc