Módulo System

O módulo **system** fornece funções para obter informações sobre o sistema operacional, rede e ambiente de execução do agente. Este módulo é útil para scripts que precisam adaptar seu comportamento com base no ambiente, coletar informações de inventário ou realizar diagnósticos do sistema.

Características principais:

- Informações detalhadas do sistema operacional
- Nome do host e informações de rede
- Dados do agente (UUID, versão)
- Informações de kernel e distribuição
- Interface unificada para diferentes sistemas operacionais

Funções Disponíveis

1. `system.env()`

Retorna informações completas sobre o ambiente de execução do agente.

Parâmetros:

- Nenhum

Retorno:

- **tabela**: Estrutura com os seguintes campos:
  - `os_type` (string): Tipo do sistema operacional (ex: "linux", "windows", "macos")
  - `os_name` (string): Nome do sistema operacional (ex: "Ubuntu", "Windows 10", "macOS")
  - `os_version` (string): Versão do sistema operacional
  - `kernel_version` (string): Versão do kernel
  - `agent` (tabela ou nil): Informações do agente Monagent (se disponível):
    - `uuid` (string): UUID único do agente
    - `version` (string): Versão do Monagent

Exemplo de Uso:

```
-- Obter todas as informações do ambiente
local env_info = system.env()

print("Tipo do SO:", env_info.os_type)
print("Nome do SO:", env_info.os_name)
print("Versão do SO:", env_info.os_version)
print("Versão do Kernel:", env_info.kernel_version)

if env_info.agent then
    print("UUID do Agente:", env_info.agent.uuid)
    print("Versão do Monagent:", env_info.agent.version)
else
    print("Informações do agente não disponíveis")
end

-- Adaptar comportamento baseado no SO
local env = system.env()

if env.os_type == "linux" then
    -- Comandos específicos para Linux
    local saida, erro = process.exec("ps", "aux")
elseif env.os_type == "windows" then
    -- Comandos específicos para Windows
    local saida, erro = process.exec("tasklist")
elseif env.os_type == "macos" then
    -- Comandos específicos para macOS
    local saida, erro = process.exec("ps", "-ef")
end
```

2. `system.hostname()`

Retorna o nome do host do sistema.

Parâmetros:

- Nenhum

Retorno:

- **string**: Nome do host do sistema, ou "unknown" se não puder ser determinado

Exemplo de Uso:

```
-- Obter nome do host
local nome_host = system.hostname()
print("Nome do host:", nome_host)

-- Usar em identificadores únicos
local identificador = nome_host .. "_" .. os.date("%Y%m%d_%H%M%S")
print("Identificador único:", identificador)

-- Verificar se é um host específico
if nome_host == "servidor-producao" then
    log.info("Executando em servidor de produção")
    config.modo = "producao"
elseif string.find(nome_host:lower(), "test") then
    log.info("Executando em ambiente de teste")
    config.modo = "teste"
else
    log.info("Executando em host desconhecido:", nome_host)
    config.modo = "desenvolvimento"
end

-- Criar tags para métricas
local tags_metricas = {
    host = nome_host,
    ambiente = config.modo,
    timestamp = os.time()
}
```

3. `system.networks()`

Retorna uma lista de endereços IP de rede não-loopback do sistema.

Parâmetros:

- Nenhum

Retorno:

- **array de strings**: Lista de endereços IP das interfaces de rede (excluindo loopback)

Comportamento:

- Exclui endereços de loopback (127.0.0.1, ::1, etc.)
- Inclui endereços IPv4 e IPv6
- Atualiza a lista de interfaces antes de retornar
- Retorna apenas endereços IP, não nomes de interface

Exemplo de Uso:

```
-- Obter todos os endereços IP do sistema
local enderecos_ip = system.networks()

print("Endereços IP disponíveis:")
for i, ip in ipairs(enderecos_ip) do
    print("  " .. i .. ". " .. ip)
end

-- Identificar endereços IPv4 vs IPv6
local function classificar_enderecos()
    local enderecos = system.networks()
    local classificacao = {
        ipv4 = {},
        ipv6 = {},
        outros = {}
    }

    for _, ip in ipairs(enderecos) do
        if string.find(ip, ":") then
            -- IPv6
            table.insert(classificacao.ipv6, ip)
        elseif string.match(ip, "^%d+%.%d+%.%d+%.%d+$") then
            -- IPv4
            table.insert(classificacao.ipv4, ip)
        else
            -- Outro formato
            table.insert(classificacao.outros, ip)
        end
    end

    return classificacao
end
```

```
local ips = classificar_enderecos()
print("IPv4:", #ips.ipv4, "endereços")
print("IPv6:", #ips.ipv6, "endereços")

-- Encontrar endereço preferido para comunicação
local function encontrar_endereco_preferido()
    local enderecos = system.networks()

    -- Prioridade: IPv4 privado > IPv4 público > IPv6
    for _, ip in ipairs(enderecos) do
        -- Verificar se é IPv4 privado
        if string.match(ip, "^10%.") or
           string.match(ip, "^172%.%d?%d?%d?%.") or
           string.match(ip, "^192%.168%.") then
```

Revision #1
Created 3 February 2026 16:46:22 by Marc
Updated 3 February 2026 16:46:32 by Marc