🛠️ Documentação Técnica Completa – PowerCare Internals

Linguagem: PowerShell (System.Windows.Forms)
Última Atualização: 21/01/2026
Responsável Técnico: Gustavo Moreira – Suporte de TI

---

1. Camada de Segurança e Elevação de Privilégios (Auto-Elevador)

O PowerCare depende de privilégios administrativos para executar rotinas críticas do Windows, como DISM, NetSH e operações avançadas de limpeza. Por esse motivo, o script inicia com um subsistema de segurança projetado para verificar permissões, bloquear execução parcial e se autoelevar se necessário.

1.1 Detecção de Token Administrador

A verificação é feita através de:

- WindowsIdentity.GetCurrent (consulta do token da sessão atual)
- WindowsPrincipal (validação da associação ao grupo Administradores)

Esse mecanismo garante a análise do **S.I.D.** (Security Identifier) para identificar se o processo está realmente em um contexto elevado, independentemente de políticas de execução.

1.2 Autoelevação via RunAs

Caso o script detecte ausência de privilégios administrativos:

- Uma nova instância do PowerShell é iniciada com **Start-Process –Verb RunAs**
- O parâmetro **ExecutionPolicy Bypass** assegura execução mesmo onde políticas de TI bloqueiam scripts externos
- O próprio script é passado como argumento, continuando automaticamente a partir do ponto correto

Por que isso é crítico?

Comandos essenciais simplesmente **falham em silêncio** sem elevação:

- netsh
- ipconfig /renew
- dism.exe
- remoções em pastas protegidas do sistema

A camada de segurança previne inconsistências, falhas parciais e danos na integridade da manutenção.

---

2. Engine de Diagnóstico (WMI / CIM Layer)

Essa camada fornece todos os dados exibidos no painel azul do PowerCare e alimenta funções internas de monitoramento.

2.1 Uso de Get-CimInstance (Sucessor Moderno de WMI)

Motivos da escolha:

- Protocolo WS-Man/WinRM (mais seguro e rápido)
- Compatível com Windows 10 e 11
- Evita namespaces depreciados do WMI antigo
- Melhor desempenho em consultas repetidas (como CPU a cada 1s)

2.2 Diagnóstico da CPU em Tempo Real

A coleta é realizada por:

- Win32_PerfFormattedData_PerfOS_Processor

Esse namespace acessa **contadores reais de performance**, diferentemente do Win32_Processor (que é estático).

Benefícios:

- Amostragem de CPU a cada 1 segundo
- Leitura estável, evitando picos falsos
- Base futura para:
    - Temperatura da CPU
    - Clocks e estados de throttling

2.3 Diagnóstico de RAM

A lógica do cálculo:

1. Soma de todos os módulos físicos instalados
2. Leitura da memória livre (sistema operacional)
3. RAM em uso = Total – Livre

Esse método evita leituras incorretas causadas por:

- Memória reservada para GPU
- Cache agressivo do Windows
- Compressão de memória

2.4 Identificação do Tipo de Disco

A leitura do MediaType classifica o armazenamento como:

- HDD
- SSD SATA
- NVMe

Essa distinção é essencial porque cada tipo responde de forma diferente a:

- Fragmentação
- TRIM
- Otimização do sistema
- Operações realizadas pelo DISM

---

3. Engenharia de Limpeza (Deep Cleaning Engine)

Essa é uma das partes mais fortes e impactantes do PowerCare, responsável por liberar espaço e restaurar desempenho real.

3.1 Estrutura Baseada em Tabela Hash

A variável de alvos amigáveis (hash table):

- Mapeia caminhos críticos do Windows para nomes legíveis
- Apaga apenas conteúdo interno usando "*", preservando a estrutura raiz
- Evita erros de remoção acidental de pastas importantes

3.2 Tratamento de Arquivos em Uso

O PowerCare utiliza estratégias defensivas:

- Remoção com ErrorAction SilentlyContinue
- Evita que arquivos em uso (Chrome, Outlook, Teams) travem o processo
- Mantém continuidade da limpeza

3.3 DISM – Componente Central da Limpeza Pesada

O comando utilizado:

- /StartComponentCleanup

Principais efeitos:

- Limpeza profunda da pasta WinSxS
- Remoção de versões antigas de atualizações
- Liberação de espaço que normalmente chega a **gigas**, não megabytes
- Execução de 2 a 5 minutos dependendo do acúmulo

O DISM é a parte mais importante e mais avançada tecnicamente de toda limpeza.

---

4. Stack de Rede e Recuperação de Conexão

A função “Otimizar Rede” atua em três camadas principais da pilha de rede.

4.1 DNS – Camada de Aplicação

Flush da tabela DNS corrige:

- Sites que não abrem
- Mudanças de DNS que não propagaram
- Problemas de rede interna

4.2 Winsock – Camada de Sockets

Reset do Winsock:

- Corrige bibliotecas corrompidas
- Restaura acesso à internet em apps
- Resolve problemas pós-VPN/antivírus

4.3 TCP/IP – Camada de Rede e Transporte

Reset dos parâmetros TCP/IP restaura:

- Pilha de roteamento
- Configurações corrompidas
- Erros de pacote e lentidão

4.4 Ajuste Automático (AutoTuning)

Configuração de autotuninglevel no modo normal:

- Ajuste dinâmico da janela de recepção TCP
- Aumenta velocidade de downloads em empresas
- Resolve gargalos em conexões onde há limitação por latência

---

5. Engine da Interface Gráfica (GUI e Threading)

A interface é construída totalmente com System.Windows.Forms, garantindo:

- Zero dependências externas
- Compatibilidade nativa com Windows
- Execução silenciosa e rápida

5.1 Evitando Congelamento – DoEvents

PowerShell é single-thread. Sem medidas adicionais:

- A GUI ficaria travada durante DISM
- Os botões não responderiam
- O usuário veria “Não Respondendo”

Para evitar isso, é usado:

- Application.DoEvents (permite que a GUI continue viva)

5.2 Log com Segurança de Thread

A atualização do log utiliza:

- Chamada segura via Invoke
- Execução sempre na thread correta
- Autoscroll inteligente
- Cores de status (Theme.Success, Theme.Accent, etc.)

O log é o coração da interação com o usuário final.

---

6. Gerenciador de Inicialização (Startup Manager)

O módulo mais técnico e complexo da aplicação.

6.1 Mesclagem de Fontes

O script coleta dados de:

- Registro (HKCU e HKLM)
- Pastas de inicialização do usuário
- Pastas de inicialização global

6.2 Construção de Objetos

Cada entrada vira um objeto contendo:

- Nome
- Caminho
- Origem
- Tipo (registro ou arquivo)
- Método adequado de remoção

Essa estrutura evita erros como:

- Remover valor de registro usando lógica de arquivo
- Remover arquivo usando lógica de registro

---

7. Theming e Identidade Visual

O PowerCare utiliza um objeto central ($Theme) com as seguintes cores:

- BG = fundo principal
- Accent = cor de destaque
- Success = mensagens de sucesso

Alterar essas cores muda instantaneamente toda a identidade visual da aplicação sem refatorar componentes da GUI.

---

8. Recomendações Futuras e Melhorias Planejadas

8.1 Diagnóstico de Bateria

- Uso do Win32_Battery para saúde, ciclos e capacidade real

8.2 Módulo de Reparo

- Botão de SFC /scannow

8.3 Sistema de Atualização Automática

- Verificação de versão em servidor interno

8.4 Garbage Collection

- Execução periódica de System.GC.Collect para janelas com Timer ativo

8.5 Auditoria Avançada de Logs

- Registro externo automático em arquivo .LOG datado

---

9. Checklist de Compilação (ps2exe)

Para gerar o executável final:

- Runtime64Bit
- Admin
- NoConsole
- Ícone personalizado (opcional)

Benefícios:

- Garantia de execução em modo administrador
- Compatibilidade total com sistemas modernos
- Estética mais profissional e limpa