Virtualization Mode Switcher — спецификация и код

Статус: **реализовано**. В окне — страница `WSL И VM -> HYPER-V`. Решает конфликт «Hyper-V/WSL2 забирает VT-x → сторонняя быстрая VM (VMware/ VirtualBox/эмуляторы) деградирует или не стартует», и обратный случай.

Жанр — governance, не debloat: только документированные механизмы (`bcdedit`, DISM Optional Features, `Win32_DeviceGuard`), backup BCD перед мутацией, confirm через `kind: dangerous`, обязательный reboot в логе.

Механизмы под капотом

- `bcdedit /set hypervisorlaunchtype Auto | Off` — грузить ли гипервизор Windows на старте. `Off` → сторонние VM на полной скорости, но WSL2/Hyper-V/ Sandbox мертвы. `Auto` → наоборот.
- **DISM Optional Features**: `Microsoft-Hyper-V-All`, `VirtualMachinePlatform`, `HypervisorPlatform` (Windows Hypervisor Platform — API для сосуществования сторонних VM с Hyper-V), `Containers-DisposableClientVM` (Windows Sandbox), `Microsoft-Windows-Subsystem-Linux`.
- **VBS / Core Isolation (HVCI) / Credential Guard** (`Win32_DeviceGuard`) — тоже поднимают гипервизор и едят VT-x. Только диагностика: статус объясняет «почему VM тормозит даже при hypervisorlaunchtype Off». Модуль их не выключает (это отдельный security-домен).

Инварианты

- Каждая мутация: `bcdedit /export` бэкап в `backup\virtualization\` + reboot-флаг в логе.
- Все `id` уникальны по смыслу. Секция `id: virtualization`.
- `service:` один — `system_core.services.devops_tools:virtualization_switch`, диспетчеризация по `parameters.mode`.
- YAML: 2 пробела на уровень.

Манифест

Вставить новый top-level group в `config/tool_manifest.yaml` **непосредственно перед** `- id: hosts` (то есть сразу после блока `wsl`). Отступы как у `wsl` (`- id` на 2 пробелах, children на 6).

```
  - id: virtualization
    title: "Virtualization"
    title_ru: "Виртуализация"
    description: "Switch the Windows hypervisor mode so Hyper-V/WSL2/Sandbox and
third-party VMs (VMware/VirtualBox) do not fight over VT-x."
    description_ru: "Переключение режима гипервизора Windows, чтобы Hyper-
V/WSL2/Sandbox и сторонние VM (VMware/VirtualBox) не конфликтовали за VT-x."
    children:
      - id: virt_status
        title: "Virtualization status"
        title_ru: "Статус виртуализации"
        description: "Show hypervisorlaunchtype, Hyper-V/VMPlatform/WHP/Sandbox/WSL
feature state, VBS/Core Isolation, and the interpreted current mode."
        description_ru: "Показать hypervisorlaunchtype, состояние фич Hyper-
V/VMPlatform/WHP/Sandbox/WSL, VBS/Core Isolation и интерпретированный текущий режим."
        service: "system_core.services.devops_tools:virtualization_switch"
        kind: "safe"
        risk_level: "readonly"
        parameters:
          mode: "status"
      - id: virt_mode_hyperv
        title: "Mode: Hyper-V / WSL2"
        title_ru: "Режим: Hyper-V / WSL2"
        description: "Set hypervisorlaunchtype=Auto and enable
VirtualMachinePlatform. Hyper-V/WSL2/Sandbox work; third-party VMs run only via WHP
or fail. Backup: backup\\virtualization. Needs reboot."
        description_ru: "hypervisorlaunchtype=Auto и включение
VirtualMachinePlatform. Заработают Hyper-V/WSL2/Sandbox; сторонние VM — только через
WHP или не стартуют. Backup: backup\\virtualization. Нужен reboot."
        service: "system_core.services.devops_tools:virtualization_switch"
        kind: "dangerous"
        risk_level: "system_change"
        parameters:
          mode: "mode_hyperv"
      - id: virt_mode_thirdparty
        title: "Mode: Third-party fast"
        title_ru: "Режим: Сторонние VM (быстро)"
        description: "Set hypervisorlaunchtype=Off. VMware/VirtualBox run at full
speed; WSL2/Hyper-V/Sandbox stop until you switch back. If VBS/Core Isolation is on,
VT-x can still be held. Backup: backup\\virtualization. Needs reboot."
        description_ru: "hypervisorlaunchtype=Off. VMware/VirtualBox на полной
скорости; WSL2/Hyper-V/Sandbox перестанут работать до обратного переключения. При
```

```
включённом VBS/Core Isolation VT-x всё равно может быть занят. Backup:
backup\\virtualization. Нужен reboot."
        service: "system_core.services.devops_tools:virtualization_switch"
        kind: "dangerous"
        risk_level: "system_change"
        parameters:
          mode: "mode_thirdparty"
      - id: virt_mode_coexist
        title: "Mode: Coexist (WHP)"
        title_ru: "Режим: Сосуществование (WHP)"
        description: "Keep hypervisorlaunchtype=Auto and enable Windows Hypervisor
Platform + VirtualMachinePlatform so modern VMware/VirtualBox run alongside Hyper-
V/WSL2 (slower). Backup: backup\\virtualization. Needs reboot."
        description_ru: "hypervisorlaunchtype=Auto + включение Windows Hypervisor
Platform и VirtualMachinePlatform, чтобы современные VMware/VirtualBox работали рядом
с Hyper-V/WSL2 (медленнее). Backup: backup\\virtualization. Нужен reboot."
        service: "system_core.services.devops_tools:virtualization_switch"
        kind: "dangerous"
        risk_level: "system_change"
        parameters:
          mode: "mode_coexist"
      - id: virt_hyperv_enable
        title: "Enable Hyper-V"
        title_ru: "Включить Hyper-V"
        description: "Enable Microsoft-Hyper-V-All (Hyper-V Manager + platform) and
set hypervisorlaunchtype=Auto. Backup: backup\\virtualization. Needs reboot."
        description_ru: "Включить Microsoft-Hyper-V-All (Hyper-V Manager + платформа)
и hypervisorlaunchtype=Auto. Backup: backup\\virtualization. Нужен reboot."
        service: "system_core.services.devops_tools:virtualization_switch"
        kind: "dangerous"
        risk_level: "system_change"
        parameters:
          mode: "hyperv_enable"
      - id: virt_hyperv_disable
        title: "Disable Hyper-V"
        title_ru: "Выключить Hyper-V"
        description: "Disable Microsoft-Hyper-V-All. To give third-party VMs full
speed also use Mode: Third-party fast (hypervisorlaunchtype=Off). Backup:
backup\\virtualization. Needs reboot."
        description_ru: "Выключить Microsoft-Hyper-V-All. Для полной скорости
сторонних VM также примените Режим: Сторонние VM (hypervisorlaunchtype=Off). Backup:
backup\\virtualization. Нужен reboot."
        service: "system_core.services.devops_tools:virtualization_switch"
```

```
        kind: "dangerous"
        risk_level: "system_change"
        parameters:
          mode: "hyperv_disable"
      - id: virt_sandbox_enable
        title: "Enable Windows Sandbox"
        title_ru: "Включить Windows Sandbox"
        description: "Enable Containers-DisposableClientVM (Windows Sandbox).
Requires the hypervisor (Hyper-V/WSL2 mode). Backup: backup\\virtualization. Needs
reboot."
        description_ru: "Включить Containers-DisposableClientVM (Windows Sandbox).
Требует включённый гипервизор (режим Hyper-V/WSL2). Backup: backup\\virtualization.
Нужен reboot."
        service: "system_core.services.devops_tools:virtualization_switch"
        kind: "dangerous"
        risk_level: "system_change"
        parameters:
          mode: "sandbox_enable"
      - id: virt_sandbox_disable
        title: "Disable Windows Sandbox"
        title_ru: "Выключить Windows Sandbox"
        description: "Disable Containers-DisposableClientVM (Windows Sandbox).
Backup: backup\\virtualization. Needs reboot."
        description_ru: "Выключить Containers-DisposableClientVM (Windows Sandbox).
Backup: backup\\virtualization. Нужен reboot."
        service: "system_core.services.devops_tools:virtualization_switch"
        kind: "dangerous"
        risk_level: "system_change"
        parameters:
          mode: "sandbox_disable"
```

Сервис

Добавить в `system_core/services/devops_tools.py` (module-level, рядом с WSL- функциями — например сразу после `wsl_register_all_vhdx`). Использует уже существующие хелперы `run_ps_command`, `ps_quote`, `JobContext`, `Path`.

```
_VIRTUALIZATION_STATUS_PS = r"""
$ErrorActionPreference = 'Continue'
$PSNativeCommandUseErrorActionPreference = $false

$isAdmin = ([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.Wind
owsBuiltInRole]::Administrator)
Write-Host "Administrator: $isAdmin"
Write-Host ''

Write-Host '=== Hypervisor launch type (bcdedit) ==='
$hltLine = (& bcdedit /enum '{current}' 2>$null | Select-String -Pattern
'hypervisorlaunchtype')
if ($hltLine) {
  $hlt = ($hltLine.Line.Trim() -split '\s+')[-1]
} else {
  $hlt = 'Auto'
  Write-Host '(hypervisorlaunchtype not explicitly set; default is Auto on Hyper-V-
capable systems)'
}
Write-Host "hypervisorlaunchtype: $hlt"
Write-Host ''

Write-Host '=== Optional features ==='
if ($isAdmin) {
  $features = @(
    'Microsoft-Hyper-V-All',
    'VirtualMachinePlatform',
    'HypervisorPlatform',
    'Microsoft-Windows-Subsystem-Linux',
    'Containers-DisposableClientVM'
  )
  $rows = foreach ($f in $features) {
    try {
      Get-WindowsOptionalFeature -Online -FeatureName $f -ErrorAction Stop | Select-
Object FeatureName, State
    } catch {
      [pscustomobject]@{ FeatureName = $f; State = 'NotPresent/Unknown' }
    }
  }
  $rows | Format-Table FeatureName, State -AutoSize | Out-String -Width 200
} else {
```

```
  Write-Host 'Feature table needs Administrator rights; skipped.'
}
Write-Host ''

Write-Host '=== Virtualization-based security (VBS / Core Isolation) ==='
try {
  $dg = Get-CimInstance -Namespace 'root\Microsoft\Windows\DeviceGuard' -ClassName
Win32_DeviceGuard -ErrorAction Stop
  $running = @($dg.SecurityServicesRunning)
  $vbs = switch ($dg.VirtualizationBasedSecurityStatus) { 0 { 'Off' } 1 {
'Configured' } 2 { 'Running' } default { 'Unknown' } }
  Write-Host "VBS status: $vbs"
  Write-Host ("HVCI (Memory Integrity) running: " + $(if ($running -contains 2) {
'Yes' } else { 'No' }))
  Write-Host ("Credential Guard running: " + $(if ($running -contains 1) { 'Yes' }
else { 'No' }))
} catch {
  Write-Host "VBS query failed: $($_.Exception.Message)"
}
Write-Host ''

Write-Host '=== Interpreted mode ==='
if ($hlt -ieq 'Off') {
  Write-Host 'Mode: THIRD-PARTY FAST. Hyper-V/WSL2/Sandbox are OFF; VMware/VirtualBox
run at full speed.'
  Write-Host 'Note: if VBS/Core Isolation is Running, the hypervisor may still hold
VT-x. Disable Core Isolation for true full speed.'
} else {
  Write-Host 'Mode: HYPER-V / WSL2. The Windows hypervisor owns VT-x; third-party VMs
run only via Windows Hypervisor Platform (slower) or fail.'
}
exit 0
"""


def virtualization_switch(context: JobContext) -> dict[str, object]:
    params = context.operation.parameters
    mode = str(params.get("mode") or "status").strip().lower()

    if mode == "status":
        run_ps_command(context, _VIRTUALIZATION_STATUS_PS, check=False,
progress_seconds=60.0)
        return {"mode": mode}
```

```
    backup_dir = context.paths.root / "backup" / "virtualization"
    backup_dir.mkdir(parents=True, exist_ok=True)

    steps: list[str] = []
    if mode == "mode_hyperv":
        context.log("Switching to Hyper-V / WSL2 mode (hypervisor ON).")
        steps += [
            "& bcdedit /set hypervisorlaunchtype Auto | Out-Host",
            "dism.exe /online /enable-feature /featurename:VirtualMachinePlatform
/all /norestart",
        ]
    elif mode == "mode_thirdparty":
        context.log("Switching to third-party fast mode (hypervisor OFF). WSL2/Hyper-
V/Sandbox will stop until you switch back.")
        steps += [
            "& bcdedit /set hypervisorlaunchtype Off | Out-Host",
        ]
    elif mode == "mode_coexist":
        context.log("Enabling coexistence: Windows Hypervisor Platform ON, hypervisor
ON (third-party VMs run alongside Hyper-V, slower).")
        steps += [
            "& bcdedit /set hypervisorlaunchtype Auto | Out-Host",
            "dism.exe /online /enable-feature /featurename:HypervisorPlatform /all
/norestart",
            "dism.exe /online /enable-feature /featurename:VirtualMachinePlatform
/all /norestart",
        ]
    elif mode == "hyperv_enable":
        context.log("Enabling Hyper-V (Microsoft-Hyper-V-All).")
        steps += [
            "dism.exe /online /enable-feature /featurename:Microsoft-Hyper-V-All /all
/norestart",
            "& bcdedit /set hypervisorlaunchtype Auto | Out-Host",
        ]
    elif mode == "hyperv_disable":
        context.log("Disabling Hyper-V (Microsoft-Hyper-V-All).")
        steps += [
            "dism.exe /online /disable-feature /featurename:Microsoft-Hyper-V-All
/norestart",
        ]
    elif mode == "sandbox_enable":
        context.log("Enabling Windows Sandbox (Containers-DisposableClientVM).")
```

```
        steps += [
            "dism.exe /online /enable-feature /featurename:Containers-
DisposableClientVM /all /norestart",
        ]
    elif mode == "sandbox_disable":
        context.log("Disabling Windows Sandbox (Containers-DisposableClientVM).")
        steps += [
            "dism.exe /online /disable-feature /featurename:Containers-
DisposableClientVM /norestart",
        ]
    else:
        raise RuntimeError(f"Unsupported virtualization mode: {mode}")

    prologue = "\n".join(
        [
            "$ErrorActionPreference = 'Stop'",
            "$PSNativeCommandUseErrorActionPreference = $false",
            f"$BackupDir = {ps_quote(backup_dir)}",
            "New-Item -ItemType Directory -Force -Path $BackupDir | Out-Null",
            "$Stamp = Get-Date -Format 'yyyyMMdd_HHmmss'",
            f"$BcdBackup = Join-Path $BackupDir ('bcd_before_{mode}_' + $Stamp +
'.bcd')",
            "& bcdedit /export $BcdBackup | Out-Host",
            "Write-Host ('BCD backup: ' + $BcdBackup)",
            "Write-Host ''",
        ]
    )
    epilogue = "\n".join(
        [
            "Write-Host ''",
            "Write-Host 'Done. A reboot is REQUIRED before the new virtualization
mode is active.'",
        ]
    )
    script = prologue + "\n" + "\n".join(steps) + "\n" + epilogue
    run_ps_command(context, script, progress_seconds=300.0, elevated=True)
    return {"mode": mode, "reboot_required": True}
```

Проверка после правки

```
runtime\python.exe -m py_compile system_core\services\devops_tools.py
runtime\python.exe -m pytest -q
install\verify_portable_env.cmd
```

Ручной обход:

- страница `HYPER-V` стоит в разделе `WSL И VM` после страниц WSL;
- `Статус виртуализации` печатает hypervisorlaunchtype, таблицу фич, VBS/Core Isolation и интерпретированный режим;
- `Режим: Сторонние VM` и `Режим: Hyper-V / WSL2` показывают confirm-плашку, создают `backup\virtualization\bcd_before_*.bcd` и пишут «reboot REQUIRED»;
- после реального переключения + reboot — `wsl --status` и сторонняя VM ведут себя согласно выбранному режиму.

Заметки

- VBS/Core Isolation модуль только диагностирует. Если пользователь жалуется на медленную VM при `hypervisorlaunchtype=Off` — статус подскажет, что виноват включённый Core Isolation; его выключение остаётся ручным/отдельным.
- Тексты подтверждения отдельно править не нужно: общий confirm + `description_ru` уже несут предупреждение «что отвалится» и про reboot.

Аддон: Optimization status (readonly)

Статус: **реализовано поверх готовой** `virtualization_switch`. Диагностическая команда «всё в одном»: что мешает VM/WSL работать быстро. Только чтение, ноль мутаций.

Манифест

Leaf находится в секции `virtualization`, сразу **после** `virt_status`:

```
      - id: virt_optimization_status
        title: "Optimization status"
        title_ru: "Статус оптимизации"
        description: "Read-only check of what slows VM/WSL: Core Isolation/VBS,
active power plan, Defender exclusions for WSL/VM paths, .wslconfig presence and WSL
VHDX placement."
        description_ru: "Только чтение: что тормозит VM/WSL — Core Isolation/VBS,
активный план питания, Defender-исключения для путей WSL/VM, наличие .wslconfig и где
лежат WSL VHDX."
        service: "system_core.services.devops_tools:virtualization_switch"
        kind: "safe"
        risk_level: "readonly"
        parameters:
          mode: "optimization_status"
```

Сервис

В `virtualization_switch`, **сразу после ветки** `if mode == "status":` (до блока с backup/мутациями) находится ветка ниже. Использует существующий хелпер `wsl_registered_vhdx_paths`.

```
    if mode == "optimization_status":
        run_ps_command(context, _VIRTUALIZATION_OPTIMIZATION_PS, check=False,
progress_seconds=90.0)
        try:
            vhdx = wsl_registered_vhdx_paths(context.paths.root)
        except Exception:
            vhdx = {}
        context.log("")
        context.log("=== WSL VHDX placement ===")
        if vhdx:
            for path, name in vhdx.items():
                context.log(f"{name}: {path}")
            context.log("Tip: keep VHDX on a fast NVMe drive; a slow/system drive
throttles WSL IO.")
        else:
            context.log("No registered WSL VHDX paths found.")
        return {"mode": mode}
```

И рядом с `_VIRTUALIZATION_STATUS_PS` добавить константу:

```
_VIRTUALIZATION_OPTIMIZATION_PS = r"""
$ErrorActionPreference = 'Continue'

Write-Host '=== Core Isolation / VBS (VM speed) ==='
try {
  $dg = Get-CimInstance -Namespace 'root\Microsoft\Windows\DeviceGuard' -ClassName
Win32_DeviceGuard -ErrorAction Stop
  $running = @($dg.SecurityServicesRunning)
  if ($running -contains 2) {
    Write-Host 'HVCI (Memory Integrity): ON -> adds VM overhead. Turn OFF for max
third-party VM speed (security tradeoff).'
  } else {
    Write-Host 'HVCI (Memory Integrity): OFF -> good for VM speed.'
  }
  Write-Host ("Credential Guard: " + $(if ($running -contains 1) { 'ON -> holds VT-x'
} else { 'OFF' }))
} catch {
  Write-Host "VBS query failed: $($_.Exception.Message)"
}
Write-Host ''

Write-Host '=== Active power plan ==='
& powercfg /getactivescheme
Write-Host 'Tip: use High performance / Ultimate for heavy virtualization.'
Write-Host ''

Write-Host '=== Defender real-time exclusions ==='
try {
  $ex = (Get-MpPreference).ExclusionPath
  if ($ex) {
    foreach ($p in $ex) { Write-Host ("  " + $p) }
  } else {
    Write-Host '  (none) -> consider excluding WSL VHDX and VM disk folders for IO
speed (security tradeoff).'
  }
} catch {
  Write-Host '  Get-MpPreference unavailable (no Defender or insufficient rights).'
}
Write-Host ''

Write-Host '=== .wslconfig ==='
$cfg = Join-Path $env:USERPROFILE '.wslconfig'
```

```
if (Test-Path -LiteralPath $cfg) {
  Write-Host ("Found: " + $cfg)
  Get-Content -LiteralPath $cfg | ForEach-Object { Write-Host ("  " + $_) }
} else {
  Write-Host '  Not found -> create %UserProfile%\.wslconfig to cap RAM/CPU and
enable sparseVhd / nestedVirtualization.'
}
exit 0
"""
```

Проверка

```
runtime\python.exe -m py_compile system_core\services\devops_tools.py
devops_cli.cmd run --operation virt_optimization_status
```

- `WSL И VM -> HYPER-V -> Статус оптимизации` печатает 5 блоков (Core Isolation, power plan, Defender exclusions, .wslconfig, VHDX placement) с tip-строками;
- команда `safe/readonly`, без confirm и без изменений системы.