kapitola

4

Postupy při sledování výkonu

Obsah kapitoly:

Pravidelné postupy sledování výkonu, které budete implementovat, musí sloužit více účelům, mezi něž patří následující:

- Detekování a diagnostika výkonnostních potíží,
- ověření naplňování odsouhlasených úrovní služeb,
- podpora aktivních iniciativ plánování kapacity, které očekávají a řeší hrozící nedostatky prostředků, ještě než ovlivní úroveň služeb.

V této kapitole jsou popsány ukázkové procedury sledování výkonu, jež vám pomohou s dosažením těchto důležitých cílů. Ukázkové postupy, které se zaměřují na snadno implementovatelný proces denního sběru výkonnostních údajů, jsou zevšeobecněny a vyhoví tak podnikovým konfiguracím malého i většího rozsahu. Do určité míry si je ale budete muset přizpůsobit, aby vyhovovaly vašemu konkrétnímu prostředí.

Ukázkové procedury vám rovněž pomohou začít s diagnostikou běžných výkonnostních problémů serverů. Další doporučení se budou týkat výkonnostních výstrah, hlášení pro správu a plánování kapacity.

Poslední oddíl této kapitoly dokumentuje postupy, které byste měli využít, když máte nějaký problém se shromažďováním určitých výkonnostních statistik pomocí nástroje Sledování výkonu. Kupříkladu aplikace, které instalují a odistalovávají výkonnostní čítače, občas postupují nesprávně – v takovém případě můžete k obnovení infrastruktury výkonnostních čítačů využít zde popisované postupy. Tím zajistíte, že Sledování výkonu bude správně hlásit výkonnostní statistiky.

Ukázkové procedury v této kapitole jsou vytvořeny tak, abyste mohli předvídat různé obvyklé výkonnostní potíže. Využívají automatizované nástroje příkazového řádku Log Manager a Relog popisované v kapitole 2, „Nástroje sledování výkonu". Log Manager a Relog vám pomohou protokolovat čítače z lokálního počítače na místní disk a zjišťovat nejdůležitější výkonnostní čítače, jak je zdůraznila kapitola 3, „Měření výkonu serveru".

Ukázkové automatizované procedury vám umožní detekovat a odhalovat řadu problémů s výkonností. Využívají na pozadí běžící relace shromažďování dat – data pak můžete bez časového omezení analyzovat po hlášení nějaké potíže. Doporučované množství dat získávané nestálým denním monitorováním výkonu však nemusí vždy odpovídat potřebám řešení každého výkonnostního problému. Občas budete muset tyto procedury sběru dat na pozadí rozšířit o zaměřené monitorování v reálném čase. Sledování v reálném čase používejte, potřebujete-li získávat detailnější údaje o konkrétních situacích.

Další informace Často není vůbec jednoduché pokoušet se o diagnostiku nějakého komplexnějšího výkonnostního problému. Ještě obtížnější je snažit se odhalit složitou potíž s výkonem v reálném čase prostřednictvím nástroje Sledování systému. Nevíte-li přesně, co vlastně hledáte, a není-li problém trvalejšího rázu, může být obtížně pracovat se Sledováním systému v reálném čase a určovat příčinu daného problému. V relaci monitorování v reálném čase může být zapotřebí sledovat a analyzovat tolik čítačů a jejich instancí, že problém zmizí dříve, než jej dokážete identifikovat. Diagnostika problémů je zaměřením kapitoly 5, „Řešení potíží s výkonem".

4.1 Jaké čítače protokolovat

Procedura denního monitorování výkonu, která vám má efektivně pomáhat při detekování, diagnostikování a řešení běžných výkonnostních potíží, musí shromažďovat velké množství užitečných výkonnostních dat, která pak můžete editovat, sumarizovat a používat v hlášení pro správu a při plánování kapacity. První rozhodnutí, které budete muset učinit, souvisí s volbou pravidelně zaznamenávaných výkonnostních čítačů z těch mnoha nabízených.

Sledování výkonu na pozadí

Denní monitorování výkonu, v jehož rámci se vytvářejí relace protokolování čítačů na pozadí, jež trvale sbírají výkonnostní data, vám dovoluje zjišťovat a řešit obvyklé potíže s výkonem, jak se začnou projevovat. Jelikož není možné předem vědět, jaké klíčové prostředky stroje s výkonnostními potížemi jsou nasycené, budete v reálném světě shromažďovat výkonnostní data všech klíčových prostředků, jakými jsou procesor, paměť, disk a síť. Narůstající dodatečné náklady však na běžném počítači se systémem Microsoft Windows Server 2003 neumožňují neustále sbírat všechny dostupné indikátory výkonu všech prostředků při všech pracovních zatíženích. Proto musíte selektivně vybrat data, jež budete sbírat, a stanovit si, jak často je budete sbírat. Je důležité dosáhnout rovnováhy mezi množstvím shromažďovaných a analyzovaných údajů a náklady souvisejícími s tímto procesem.

Abyste dokázali odhalit a stanovit běžné výkonnostní problémy související s přetíženými prostředky, musíte shromažďovat široký rozsah detailních výkonnostních údajů o využití procesoru, paměti, disku a sítě a také o pracovních zatíženích, jež tyto prostředky konzumují. Takové údaje musí zahrnovat všechny čítače výkonu indikující chybové stavy, zejména chyby vyplývající z nedostatku interních prostředků. Tato kapitola popisuje některé z klíčových indikátorů chyb, které byste měli sledovat.

Hlášení pro správu

Hlášení pro správu na úrovni služeb i v dalších formách obvykle vyžaduje mnohem méně podrobných údajů. Proto je k hlášení pro správu úroveň detailů, zajišťovaná postupy denního sledování výkonu vhodnými k detekci a řešení běžných výkonnostních potíží, více než dostatečná. Často platí, že chcete-li sestavit postupy hlášení pro správy efektivně škálovatelné ve velké organizaci, je vhodné sumarizovat nejprve denní protokoly výkonnostních čítačů a pak teprve sestavit konečnou zprávu. Sumarizace denních protokolů čítačů omezí velikost souborů, které je zapotřebí přenášet přes síť, a zlepší efektivitu postupů hlášení.

Hlášení na úrovni služeb se zaměřují na spotřebu prostředků a měření zatížení, jako jsou přihlašování, relace, rychlost transakcí a přijaté zprávy. Hlášení na úrovni služeb často zajímají více profesionálních správců systému, z nichž mnozí nemusejí být detailně obeznámeni se způsobem, jakým funguje systém Windows Server 2003 a jeho hlavní serverové aplikace. Je proto důležité vyhnout se ve zprávách pro správu, zaměřených na širší cílovou skupinu, příliš mnoha technickým detailům. Hlášení na úrovni služeb se musí místo toho zaměřit na několik klíčových ukazatelů využívání prostředků a zatížení, které jsou dobře známé a široce chápané.

Plánování kapacity

Abyste mohli implementovat aktivní plánování kapacity, v jehož rámci identifikujete trendy růstu pracovního zatížení, spolehlivě předvídáte nárůst pracovního zatížení a odhadujete budoucí požadavky, musíte sledovat a shromažďovat historické údaje o úrovních využívání a spotřeby základních prostředků v čase. To zřejmě bude zapotřebí jen u malého počtu klíčových počítačových komponent, jako je procesor, síť a disk, popřípadě i pro několik zásadních aplikací. Data protokolů čítačů, sloužící jako základ procesů hlášení pro správu, se znovu budou editovat a sumarizovat, aby je bylo možné využít pro plánování kapacity. V tomto okamžiku se důraz přenáší na sestavení historických záznamů dat využívání prostředků počítačů.

Budoucnost dokážete s rozumnou přesností předpovědět pouze po zaznamenání značného množství historických údajů zachycujících vzory růstu pracovního zatížení. Kupříkladu na každou jednotku času, u které chcete předpovědět budoucí růst pracovního zatížení, budete potřebovat historický záznam roven minimálně dvojnásobku příslušného časového období. Plánování kapacity tak vyžaduje, abyste zaznamenávali využívání prostředků v delším časovém horizontu. Většinou platí, že rozumně přesné odhady kapacitních požadavků, které budou podkladem pro každoroční rozpočet, lze provést až po nashromáždění dat odpovídajících dvěma nebo třem rokům provozu systému.

Při plánování budoucích kapacitních požadavků mají často zásadní dopad sezónní vzory činnosti. K sezónním proměnám dochází v mnoha produkčních pracovních zatíženích běžně v ročních nebo měsíčních cyklech. Kupříkladu na konci měsíce nebo roku dochází často ke zvýšeným tokům zpracování finančních transakcí. Budete muset zajistit počítačovou kapacitu dostačující ke zvládnutí takových špiček na konci měsíců nebo roků. V maloobchodních organizacích zřejmě odhalíte vzory nákupů odpovídající různým svátkům, kdy lze očekávat mnohem vyšší objem transakcí. Není ani zapotřebí zdůrazňovat, že i tyto špičkové rychlosti transakcí je zapotřebí nějakým způsobem zvládnout. Faktory sezónních proměn v požadavcích na pracovní zatížení budete moci zohlednit až poté, co nashromáždíte dostatečná historická data odrážející více cyklů takových sezónních činností.

4.2 Postupy denního sledování serverů

Tento oddíl se detailně zabývá modelem procedury denního monitorování výkonu, který je součástí komplexního programu aktivní správy výkonu. Tato procedura zahrnuje následující denní činnosti:

- Automatický sběr hloubkového pohledu na výkon systému prostřednictvím protokolů čítačů,
- monitorování klíčových indikátorů chyb systému a serverových aplikací,
- nastavení výstrah, jež automaticky spustí sběr detailních, diagnostických protokolů čítačů,
- vývoj hlášení klíčových výkonnostních metrik, které si mohou prohlédnout příslušní lidé ve vaší organizaci,

- zachovávání sumarizovaných výkonnostních statistik v rámci podpory plánování kapacity,
- správa protokolů čítačů, které se v rámci těchto postupů vytvářejí automaticky.

Pamatujte, že zde popisované modelové postupy a procedury sledování výkonu bude na základně požadavků specifických vašemu sídlu zapotřebí upravit pro použití ve vašem prostředí. Každá organizace IT má kupříkladu jiné požadavky na hlášení pro správu, které mají dopad na typ a množství generovaných zpráv a dat v nich zahrnutých. Zde uvedené postupy a procedury představují dobrý počáteční bod pro začátek tvorby funkcí efektivní správy výkonu ve vaší organizaci IT.

Denní protokoly čítačů

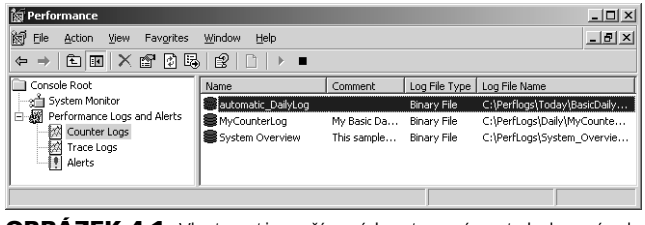
Prvním krokem monitorování počítačů se systémem Windows Server 2003 je vytvoření automatického protokolování dat pomocí nástroje příkazového řádku Log Manager (logman).

Příkaz uvedený ve výpisu 4.1 zavádí proceduru denního protokolování výkonu s využitím souboru nastavení, který definuje sbírané výkonnostní čítače. (Ukázkový soubor nastavení je popsán dále.) Příkaz se také odkazuje na příkazový soubor vykonávaný při uzavírání souborů denních protokolů čítačů. (Ukázkový skript zajišťující typické zpracování je rovněž ilustrován dále.)

VÝPIS 4.1: Ustavení denního protokolování výkonu

```
logman create counter automaticky_DenniProtokol
  -cf "c:\perflogs\zakladni-soubor-nastaveni-citacu.txt"
  -o C:\Perflogs\Dnes\ZakladniDenniProtokol -b 1.1.2004 00:00:00
  -cnf 24:00:00 -si 1:00 -f BIN -v mmddhhmm
  -rc c:\perflogs\nasledne-zpracovani.vbs
```

Po vykonání uvedeného příkazu bude definovaný protokol čítačů viditelný a bude se podobat zobrazení uvedenému na obrázku 4.1. Použijete-li grafické uživatelské rozhraní tohoto protokolu čítačů, můžete si ověřit vlastnosti aplikované na zavedení protokolovací relace, jak je tu ukázáno.



OBRÁZEK 4.1: Vlastnosti používané k ustavení protokolovací relace

Jak dokumentuje kapitola 2, „Nástroje sledování výkonu“, utilita Log Manager vám umožňuje nakonfigurovat relace sběru protokolových dat čítačů na pozadí. Tabulka 4.1 probírá parametry příkazu Log Manager použité ve výpisu 4.1 a vysvětluje jejich účel.

Parametr `-v` nástroje Log Manager vám umožňuje generovat jedinečné názvy souborů denně vytvářených protokolů čítačů. Doplněk snap-in Performance Logs and Alerts (Výstrahy a protokolování výkonu) podporuje další možnost označování ver-

zí souborů specifikující datum ve formátu `yyyymmdd`. Chcete-li nechat soubory protokolů čítačů automaticky pojmenovávat ve formátu `yyyymmdd`, můžete následně využít doplněk Výkon a ručně změnit vlastnosti protokolu tak, aby se k základnímu názvu souboru přidalo označení roku, měsíce a dne.

TABULKA 4.1: Parametry příkazu správce Log Manager

Parametr programu Log Manager	Vysvětlení
`-cf "c:\perflogs\zakladni-soubor-nastaveni-citacu.txt"`	V souboru nastavení jsou specifikovány čítače, jež se mají protokolovat. Příklad souboru zakladni-soubor-nastaveni-citacu najdete v oddílu „Ukázkový soubor nastavení čítačů" této kapitoly.
`-b 1.1.2004 00:00:00 -cnf 24:00:00`	Protokolování iniciuje automaticky protokolovací služba Sledování systému, jakmile je váš počítač restartován. Protokolování běží trvale po dobu 24 hodin.
`-si 1:00`	Vzorky dat se shromažďují jednou za minutu.
`-f BIN`	Protokolování dat probíhá průběžně do binárního souboru. Binární formát je využit kvůli výkonnosti a šetření množství spotřebovávaného diskového prostoru.
`-v mmddhhmm`	K vytváření jedinečných souborů denních protokolů čítačů se používá automatické označování verzí.
`-rc c:\perflogs\nasledne-zpracovani.vbs`	Na konci relace protokolování dat se po zavření protokolového souboru spustí určitý skript. Tento skript zajistí správu souborů a další následné zpracování. Takové následné zpracování zahrnuje odstranění starších kopií souborů protokolů čítačů, jež byly vytvořeny dříve, a sumarizaci aktuálního souboru protokolu pro denní hlášení. Příklad skriptu následného zpracování najdete v oddílu „Automatizované zpracování protokolů čítačů" této kapitoly.

Jakmile se přesvědčíte o správném specifikování vytvořené relace protokolování čítačů, vydejte příkaz uvedený ve výpisu 4.2, kterým spustíte vlastní protokolování dat.

VÝPIS 4.2: Spuštění protokolování dat

```
logman start counter automaticky_DenniProtokol
```

Protokolování místních čítačů na lokální disk

Zde doporučovaná procedura denního monitorování výkonu generuje protokoly čítačů obsahující lokální čítače zapisované v binárním formátu na místní disk. Náš příklad zapisuje soubory protokolů čítačů na lokální disk do adresáře nazvaného C:\Perflogs\Dnes, vyhoví ale jakákoli jiná rozumná složka na místním disku. Binární formát protokolového souboru se doporučuje kvůli své efektivnosti a menším dodatečným nákladům.

Protokolování místních čítačů na místní disk vám dovoluje implementovat jednotnou proceduru sledování výkonu na všech počítačích v síti, takže lze tyto procedury škálovat i na největších serverových konfiguracích bez ohledu na topologii sítě.

Zde dokumentované procedury denního sledování výkonu předpokládají zapisování protokolovacích dat místních čítačů do nějaké složky na lokálním disku, i když jsou možné i jiné konfigurace. Protokoly čítačů lze kupříkladu použít ke sběru dat ze vzdálených počítačů. Vzdálený sběr dat je vhodný v situaci, kdy nemáte k danému vzdálenému počítači fyzický přístup. Pak je jednodušší zajišťovat shromažďování výkonnostních údajů na jednom centrálním počítači, jehož úkolem je „vytahovat" data čítačů z jednoho nebo více vzdálených strojů. Taková procedura ale už z principu není tak robustní jako lokální běh protokolování čítačů, protože potíže se sběrem dat na *kterémkoli* ze vzdálených počítačů mohou mít dopad na všechny počítače, k jejichž monitorování je centrální proces vytvořen. Je zapotřebí také chápat dopad vzdáleného sledování počítačů na síť. Toto téma je detailně probíráno v oddílu „Scénáře protokolování čítačů".

Při protokolování dat čítačů na lokální disk musíte spravovat pravidelně vytvářené protokolové soubory, aby nespotřebovávaly nerozumné množství lokálního diskového prostoru. Bez určitého zastarávání souborů a čištění mohou denně generované protokoly čítačů zabírat 30 až 100 MB prostoru místního disku vašich serverů každý den, kdy běží. V oddílu „Automatizované zpracování protokolů čítačů" najdete ukázkový skript následného zpracování, který k zajištění takové denní správy souborů a čištění využívá jazyk Microsoft Visual Basic Scripting Edition (VBScript) a komponentu Windows Script Host (WSH).

Prostřednictvím utility příkazového řádku Relog můžete později převést soubory z binárního formátu na jakoukoli jinou formu. Máte kupříkladu možnost vytvářet souhrnné soubory, které lze přenášet na konsolidační server na síti. Také můžete vytvářet textové soubory s hodnotami oddělovanými středníky (nebo čárkami) a ty dále využívat v programech jako Microsoft Excel, jež dokážou generovat užitečné a atraktivní grafy. Pomocí nástroje Relog lze také sestavit a udržovat databázi SQL Serveru konsolidovaných dat protokolování čítačů z řady serverů, která pak poslouží při plánování kapacity.

Protokolování na místo sdílené v síti

Namísto generování protokolů čítačů v binárním formátu do složky místního disku upřednostňuje mnoho lidí zapisování protokolů čítačů do nějaké složky sdílené v síti. Jelikož může zjednodušit správu souborů, je tento přístup často preferován. Zvolíte-li ale tuto metodu, musíte zvážit také dále uvedené věci, jež mohou ovlivnit škálovatelnost vašich procedur monitorování výkonu na sídlech s větším počtem serverů:

- Dávejte si pozor na přetížení sítě. Spotřebovává-li váš denní protokol čítačů 50 MB diskového prostoru každý den, což je rozděleno na 24hodinová období, představuje tento objem pouze 600 bajtů na server za sekundu dodatečného zatížení, které musí vaše síť zvládnout. Přibližné zatížení sítě při vzdáleném protokolování dat zjistíte tím způsobem, že znásobíte uvedenou hodnotu počtem serverů zaznamenávajících údaje do téže složky sdílené na síti.
- Zajistěte, aby vaše relace protokolování čítačů běžela pod identifikátorem uživatele s oprávněním pro přístup k dané sdílené síťové složce. Tento identifikátor uživatele musí být zároveň členem vestavěné skupiny Performance Log Users (uživatelé protokolů výkonu). Chcete-li do relace nástroje Log Manager doplnit údaje o uživateli, zadejte uživatelské jméno a heslo parametrem `-u`. Budete-li

využívat Výstrahy a protokolování výkonu v konzole Performance Monitor (Sledování výkonu), musíte nastavit parametr Run As (Spustit jako) na kartě General (Obecné) vlastností protokolu čítačů.

- Zajistěte, aby služba Windows Time Service synchronizovala hodiny na všech serverech, které zapisují soubory protokolů čítačů do sdílené síťové složky.

Další informace Popis toho, jak používat časovou službu Windows k synchronizaci hodin na počítačích v síti systému Windows Server 2003, najdete v dokumentaci nazvané „Windows Time Service Technical Reference" umístěné na sídle TechNet na adrese `http://www.microsoft.com/resources/documentation/WindowsServ/2003/all/techref/en-us/W2K3TR_times_intro.asp`.

- Do názvů souborů protokolů čítačů zahrňte také název počítače, abyste mohli jednoduše zjistit, od kterého z nich shromážděné údaje pocházejí.

Všechny tyto věci lze snadno zajistit s využitím skriptu WSH. Kupříkladu kód VBScriptu z výpisu 4.3 vrací název lokálního počítače v proměnné nazvané PrihlasovaciServer.

VÝPIS 4.3: Identifikace zdrojového počítače

```
Set WshShell = CreateObject("Wscript.Shell")
Set objEnv = WshShell.Environment("Process")
PrihlasovaciServer = objENV("COMPUTERNAME")
```

Dále můžete zkonstruovat název souboru s vloženým názvem lokálního počítače. Stačí kód ve výpisu 4.4.

VÝPIS 4.4: Sestavení názvu souboru

```
Const SlozkaPerflogVcera = "Z:\SdilenyPerflog\Vcera"
Const TypProtokolu = "blg"
Const ParametryProtokolu = " -b 1.1.2004 00:00:00 -cnf 24:00:00 " & _
  "-si 1:00 -f BIN -v mmddhhmm -rc c:\perflogs\nasledne-zpracovani.vbs"
NazevDennihoSouboru = SlozkaPerflogVcera & "\" & PrihlasovaciServer & _
  "." & "zakladniDenniProtokol" & "." & TypProtokolu
NazevRelaceProtokolu = PrihlasovaciServer & "-denni-protokol"
```

Pak můžete vykonat nástroj Logman ze skriptu s využitím metody `Exec` objektu Shell WSH, jak ukazuje výpis 4.5.

VÝPIS 4.5: Vykonání nástroje Logman

```
Const NazevSouboruNastaveniProtokolu = _
  "Z:\SdilenyPerflog\soubor-nastaveni-citacu-protokolu.txt"
prikaz = "logman create counter " & NazevRelaceProtokolu & " -o " & _
  NazevDennihoSouboru & " -cf " & NazevSouboruNastaveniProtokolu & " " & _
  ParametryProtokolu
Set WshShell = Wscript.CreateObject("WScript.Shell")
Set vykonavanyPrikaz = WshShell.Exec(prikaz )
Wscript.Echo vykonavanyPrikaz.StdOut.ReadAll
```

Poslední řádek skriptu přejímá vlastnost `ReadAll` proudu `StdOut`, která obsahuje všechny zprávy vygenerované nástrojem Logman v metodě `Exec` objektu Shell. To vám umožňuje stanovit, zda se tato utilita vykonala úspěšně.

Další souvislosti vykonávání vzdáleného protokolování a zapisování protokolů čítačů na vzdálený disk najdete v oddílu „Scénáře protokolování čítačů".

Měření základů

Data měření základů jsou vlastně detailním výkonnostním profilem vašeho počítače, který si uložíte pro budoucnost, až budete chtít zjistit, co se ve vašem prostředí změnilo. Předchozí sadu dat měření základu můžete porovnat s aktuálními údaji a odhalit tak všechny významnější změny v pracovním zatížení. Občas se malé postupné změny, k nimž dochází pozvolna, najednou v čase projeví jako významná změna. Budete-li používat jen časově úzce zaměřené výkonnostní hlášení, můžete snadno přehlédnout rozsah a měřítko takových postupných změn. Jednu z dobrých možností ocenění rozsahu podobných změn v čase představuje porovnání detailů aktuálního prostředí s jednou z novějších sad měření základních údajů.

Zde doporučovaná procedura denního monitorování výkonu je vhodná pro zajištění sady základních měření vašeho počítače. Každých přibližně šest měsíců si tento soubor protokolů čítačů zkopírujte a uložte. Při kopírování protokolu čítačů využijte nějaké bezpečné místo umožňující dlouhodobé skladování. Je rovněž vhodné uložit si protokol čítačů základu před každou zásadnější změnou konfigurace hardwaru nebo softwaru systému.

Aby byla shromážděná data čítačů užitečná jako sada základních měření, musíte je zachovávat v původním binárním formátu. Může být vhodné uložit si tyto sady měření základů na nějaké archivní úložiště v režimu offline, aby mohly s minimálními náklady zůstat zachovány několik let.

Použití souboru nastavení čítačů

Když ke specifikaci čítačů, které chcete shromažďovat, využijete soubor nastavení čítačů, budete moci snadno vytvářet a udržovat jednotné postupy sledování výkonu na všech svých počítačích s podobnými požadavky. Můžete tak kupříkladu vytvořit soubor nastavení čítačů využívaný na strojích provozujících Microsoft SQL Server. Podobně budou vaše počítače webových serverů s aplikacemi Internet Information Services (IIS) a .NET Framework vyžadovat k dosažení nejlepších výsledků jiný soubor nastavení čítačů.

Každý vámi vytvářený soubor nastavení čítačů bude zřejmě zahrnovat základní množinu čítačů hlásících využití principiálních prostředků počítače – procesorů, pamětí, disků a síťových rozhraní. Jednoduchý soubor nastavení čítačů, který bude tvořit jádro téměř každého souboru nastavení čítačů vytvářeného pro specifické aplikace, může vypadat jako ve výpisu 4.6.

VÝPIS 4.6: Soubor nastavení čítačů podle jejich originálního, anglického označení

```
\LogicalDisk(*)\% Free Space
\LogicalDisk(*)\Free Megabytes
\PhysicalDisk(*)\*
\Cache\*
```

```
\Processor(*)\*
\Memory\*
\System\*
\Network Interface(*)\*
\IPv4\*
\TCPv4\*
```

Lokalizační poznámka Cesty čítačů lze v souboru nastavení čítačů specifikovat rovněž podle českého označení. V tomto případě se tedy jedná o následující zadání:

```
\Logický disk(*)\% volného místa
\Logický disk(*)\Volné megabajty
\Fyzický disk(*)\*
\Mezipaměť\*
\Procesor(*)\*
\Paměť\*
\Systém\*
\Rozhraní sítě(*)\*
\IPv4\*
\TCPv4\*
```

Grafické uživatelské rozhraní modulu Výkon uvádí české ekvivalenty cest a názvů čítačů.

Výpis 4.6 shromažďuje statistiku volného prostoru logických disků a všechny čítače objektů Mezipaměť, Paměť, Rozhraní sítě, Fyzický disk, Procesor a Systém. Měření volného prostoru na logickém disku vám dovoluje stanovit, kdy vašemu systému souborů dochází kapacita.

Doplnění čítačů specifických aplikací Soubor nastavení čítačů z výpisu 4.6 postrádá všechna aplikačně specifická výkonnostní data. Tento jednoduchý soubor nastavení čítačů lze výrazně vylepšit doplněním čítačů přidružených aplikacím, které na daném serveru běží (viz tabulka 4.2).

V zájmu snížení velikosti denně generovaných souborů protokolů čítačů nevytvářejte jediný soubor nastavení obsahující *všechny* čítače specifické aplikacím podle tabulky 4.2 a jim přiřazená data na úrovni procesů. Místo toho vytvořte řadu souborů nastavení specifických jednotlivým aplikacím.

Tip Pokud nevíte, jaké serverové aplikace jsou na určitém počítači instalované a aktivní, použijte nástroj příkazového řádku Typeperf popisovaný v kapitole 2, „Nástroje sledování výkonu". Ten generuje soubor nastavení uvádějící kompletní výpis rozšířených aplikačních čítačů, které lze na daném počítači sledovat.

TABULKA 4.2: Doplnění čítačů

Role vašeho počítače se systémem Windows Server 2003	Zaznamenávejte čítače z těchto objektů	Zaznamenávejte čítače z těchto procesů
Řadič domény	`NTDS`	`lsass, smss`
Server vzdáleného přístupu	`RAS Total, RAS Port`	`svchost`

Role vašeho počítače se systémem Windows Server 2003	Zaznamenávejte čítače z těchto objektů	Zaznamenávejte čítače z těchto procesů
Databázový server	SQL Server:General Statistics, SQL Server:Databases, SQL Server:Buffer Manager, SQL Server:Cache Manager, SQL Server:SQL Statistics, SQL Server:Lock	sqlserver, sqlagent
Webový server	Internet Information Services Global, FTP Service, Web Service, Web Service Cache	inetinfo, svchost
Souborový a tiskový server	Server, Server Work Queues, Print Queue, NBT Connection	svchost, spoolsv
Terminálový server	Terminal Services, Terminal Services Session	svchost; tssdis
Server Exchange	MSExchangeAL, MSExchangeDSAccess Caches, MSExchangeDSAccess Contexts, MSExchangeDSAccess Processes, Epoxy, MSExchangeIS Mailbox, Database ==> Instances, MSExchange Transport Store Driver, MSExchangeIS Transport Driver, MSExchangeSRS, MSExchange Web Mail, MSExchangeIMAP4, MSExchangePOP3, MSExchangeMTA, MSExchangeMTA Connections, SMTP Server, SMTP NTFS Store Driver	store, dsamain
Aplikační server	MSMQ Session, MSMQ Service, MSMQ Queue	dllhost

Ukázkový soubor nastavení čítačů Na základě principů uvedených v předchozím oddílu lze pro souborový a tiskový server vytvořit soubor nastavení čítačů, který může vypadat jako ukázka ve výpisu 4.7.

VÝPIS 4.7: Příklad souboru nastavení čítačů pro souborový a tiskový server

a) produkt v angličtině:

```
\LogicalDisk(*)\% Free Space
\LogicalDisk(*)\Free Megabytes
\LogicalDisk(*)\Current Disk Queue Length
\PhysicalDisk(*)\Current Disk Queue Length
\PhysicalDisk(*)\Avg. Disk Queue Length
\PhysicalDisk(*)\Avg. Disk sec/Transfer
\PhysicalDisk(*)\Avg. Disk sec/Read
\PhysicalDisk(*)\Avg. Disk sec/Write
\PhysicalDisk(*)\Disk Transfers/sec
\PhysicalDisk(*)\Disk Reads/sec
\PhysicalDisk(*)\Disk Writes/sec
```

```
\PhysicalDisk(*)\Disk Bytes/sec
\PhysicalDisk(*)\Disk Read Bytes/sec
\PhysicalDisk(*)\Disk Write Bytes/sec
\PhysicalDisk(*)\Avg. Disk Bytes/Transfer
\PhysicalDisk(*)\Avg. Disk Bytes/Read
\PhysicalDisk(*)\Avg. Disk Bytes/Write
\PhysicalDisk(*)\% Idle Time
\PhysicalDisk(*)\Split IO/Sec
\Server\Bytes Total/sec
\Server\Bytes Received/sec
\Server\Bytes Transmitted/sec
\Server\Sessions Timed Out
\Server\Sessions Errored Out
\Server\Sessions Logged Off
\Server\Sessions Forced Off
\Server\Errors Logon
\Server\Errors Access Permissions
\Server\Errors Granted Access
\Server\Errors System
\Server\Blocking Requests Rejected
\Server\Work Item Shortages
\Server\Pool Nonpaged Bytes
\Server\Pool Nonpaged Failures
\Server\Pool Nonpaged Peak
\Server\Pool Paged Bytes
\Server\Pool Paged Failures
\Server Work Queues(*)\Queue Length
\Server Work Queues(*)\Active Threads
\Server Work Queues(*)\Available Threads
\Server Work Queues(*)\Available Work Items
\Server Work Queues(*)\Borrowed Work Items
\Server Work Queues(*)\Work Item Shortages
\Server Work Queues(*)\Current Clients
\Server Work Queues(*)\Bytes Transferred/sec
\Server Work Queues(*)\Total Operations/sec
\Server Work Queues(*)\Context Blocks Queued/sec
\Cache\Data Maps/sec
\Cache\Data Map Hits %
\Cache\Data Map Pins/sec
\Cache\Pin Reads/sec
\Cache\Pin Read Hits %
\Cache\Copy Reads/sec
\Cache\Copy Read Hits %
\Cache\MDL Reads/sec
\Cache\MDL Read Hits %
\Cache\Read Aheads/sec
\Cache\Lazy Write Flushes/sec
\Cache\Lazy Write Pages/sec
\Cache\Data Flushes/sec
\Cache\Data Flush Pages/sec
\Processor(*)\% Processor Time
\Processor(*)\% User Time
```

```
\Processor(*)\% Privileged Time
\Processor(*)\Interrupts/sec
\Processor(*)\% DPC Time
\Processor(*)\% Interrupt Time
\Memory\Page Faults/sec
\Memory\Available Bytes
\Memory\Committed Bytes
\Memory\Commit Limit
\Memory\Write Copies/sec
\Memory\Transition Faults/sec
\Memory\Cache Faults/sec
\Memory\Demand Zero Faults/sec
\Memory\Pages/sec
\Memory\Pages Input/sec
\Memory\Page Reads/sec
\Memory\Pages Output/sec
\Memory\Pool Paged Bytes
\Memory\Pool Nonpaged Bytes
\Memory\Page Writes/sec
\Memory\Pool Paged Allocs
\Memory\Pool Nonpaged Allocs
\Memory\Free System Page Table Entries
\Memory\Cache Bytes
\Memory\Cache Bytes Peak
\Memory\Pool Paged Resident Bytes
\Memory\System Code Total Bytes
\Memory\System Code Resident Bytes
\Memory\System Driver Total Bytes
\Memory\System Driver Resident Bytes
\Memory\System Cache Resident Bytes
\Memory\% Committed Bytes In Use
\Memory\Available KBytes
\Memory\Available MBytes
\Memory\Transition Pages RePurposed/sec
\Paging File(*)\% Usage
\Paging File(*)\% Usage Peak
\System\Context Switches/sec
\System\System Up Time
\System\Processor Queue Length
\System\Processes
\System\Threads
\Process( svchost,*)\% Processor Time
\Process(svchost,*)\% User Time
\Process(svchost,*)\% Privileged Time
\Process(svchost,*)\Virtual Bytes Peak
\Process(svchost,*)\Virtual Bytes
\Process(svchost,*)\Page Faults/sec
\Process(svchost,*)\Working Set Peak
\Process(svchost,*)\Working Set
\Process(svchost,*)\Page File Bytes Peak
\Process(svchost,*)\Page File Bytes
\Process(svchost,*)\Private Bytes
```

```
\Process(svchost,*)\Thread Count
\Process(svchost,*)\Priority Base
\Process(svchost,*)\Elapsed Time
\Process(svchost,*)\ID Process
\Process(svchost,*)\Pool Paged Bytes
\Process(svchost,*)\Pool Nonpaged Bytes
\Print Queue(*)\Total Jobs Printed
\Print Queue(*)\Bytes Printed/sec
\Print Queue(*)\Total Pages Printed
\Print Queue(*)\Jobs
\Print Queue(*)\References
\Print Queue(*)\Max References
\Print Queue(*)\Jobs Spooling
\Print Queue(*)\Max Jobs Spooling
\Print Queue(*)\Out of Paper Errors
\Print Queue(*)\Not Ready Errors
\Print Queue(*)\Job Errors
\Print Queue(*)\Enumerate Network Printer Calls
\Print Queue(*)\Add Network Printer Calls
\Network Interface(*)\Bytes Total/sec
\Network Interface(*)\Packets/sec
\Network Interface(*)\Packets Received/sec
\Network Interface(*)\Packets Sent/sec
\Network Interface(*)\Current Bandwidth
\Network Interface(*)\Bytes Received/sec
\Network Interface(*)\Packets Received Unicast/sec
\Network Interface(*)\Packets Received Non-Unicast/sec
\Network Interface(*)\Packets Received Discarded
\Network Interface(*)\Packets Received Errors
\Network Interface(*)\Packets Received Unknown
\Network Interface(*)\Bytes Sent/sec
\Network Interface(*)\Packets Sent Unicast/sec
\Network Interface(*)\Packets Sent Non-Unicast/sec
\Network Interface(*)\Packets Outbound Discarded
\Network Interface(*)\Packets Outbound Errors
\Network Interface(*)\Output Queue Length
\IPv4\Datagrams/sec
\IPV4\Datagrams Received/sec
\IPV4\Datagrams Received Header Errors
\IPV4\Datagrams Received Address Errors
\IPV4\Datagrams Forwarded/sec
\IPV4\Datagrams Received Unknown Protocol
\IPV4\Datagrams Received Discarded
\IPV4\Datagrams Received Delivered/sec
\IPV4\Datagrams Sent/sec
\IPV4\Datagrams Outbound Discarded
\IPV4\Datagrams Outbound No Route
\IPV4\Fragments Received/sec
\IPV4\Fragments Re-assembled/sec
\IPV4\Fragment Re-assembly Failures
\IPV4\Fragmented Datagrams/sec
\IPV4\Fragmentation Failures
```

```
\IPV4\Fragments Created/sec
\TCPV4\Segments/sec
\TCPV4\Connections Established
\TCPV4\Connections Active
\TCPV4\Connections Passive
\TCPV4\Connection Failures
\TCPV4\Connections Reset
\TCPV4\Segments Received/sec
\TCPV4\Segments Sent/sec
\TCPV4\Segments Retransmitted/sec
```

b) produkt s českou lokalizací

```
\Logický disk(*)\% volného místa
\Logický disk(*)\Volné megabajty
\Logický disk(*)\Aktuální délka fronty disku
\Fyzický disk(*)\Aktuální délka fronty disku
\Fyzický disk(*)\Střední délka fronty disku
\Fyzický disk(*)\Střední doba disku/přenos
\Fyzický disk(*)\Střední doba disku/čtení
\Fyzický disk(*)\Střední doba disku/zápis
\Fyzický disk(*)\Přenosy disku/s
\Fyzický disk(*)\Čtení z disku/s
\Fyzický disk(*)\Zápisy na disk/s
\Fyzický disk(*)\Bajtů disku/s
\Fyzický disk(*)\Bajty čtení z disku/s
\Fyzický disk(*)\Bajty zapisování na disk/s
\Fyzický disk(*)\Střední počet bajtů disku/přenos
\Fyzický disk(*)\Střední počet bajtů disku/čtení
\Fyzický disk(*)\Střední počet bajtů disku/zápis
\Fyzický disk(*)\% času nečinnosti
\Fyzický disk(*)\Dělení vstupně-výstupních operací/s
\Server\Bajty celkem/s
\Server\Přijaté bajty/s
\Server\Odeslané bajty/s
\Server\Vypršelé relace
\Server\Chybné relace
\Server\Odhlášené relace
\Server\Odpojené relace
\Server\Chyby přihlášení
\Server\Chyby přístupových práv
\Server\Chyby udělených práv
\Server\Chyby systému
\Server\Zamítnuté žádosti zablokování
\Server\Nedostatky pracovních položek
\Server\Bajty nestránkovaného fondu
\Server\Chyby nestránkovaného fondu
\Server\Vrchol nestránkovaného fondu
\Server\Bajty stránkovaného fondu
\Server\Chyby stránkovaného fondu
\Pracovní fronty serveru(*)\Délka fronty
\Pracovní fronty serveru(*)\Aktivní podprocesy
\Pracovní fronty serveru(*)\Dostupné podprocesy
```

```
\Pracovní fronty serveru(*)\Dostupné pracovní položky
\Pracovní fronty serveru(*)\Vypůjčené pracovní položky
\Pracovní fronty serveru(*)\Nedostatky pracovních položek
\Pracovní fronty serveru(*)\Aktuální počet klientů
\Pracovní fronty serveru(*)\Bajty přenesené/s
\Pracovní fronty serveru(*)\Celkem operací/s
\Pracovní fronty serveru(*)\Bloky kontextu ve frontě/s
\Mezipaměť\Mapování dat/s
\Mezipaměť\Zásahy mapování dat %
\Mezipaměť\Držení mapování dat/s
\Mezipaměť\Čtení držení/s
\Mezipaměť\Zásahy čtení držení %
\Mezipaměť\Čtení kopií/s
\Mezipaměť\Zásahy čtení kopií %
\Mezipaměť\Čtení MDL/s
\Mezipaměť\Zásahy čtení MDL %
\Mezipaměť\Dopředná čtení/s
\Mezipaměť\Dopsání opožděných zápisů/s
\Mezipaměť\Stránky opožděných zápisů/s
\Mezipaměť\Dopsání dat/s
\Mezipaměť\Dopsání dat stránek/s
\Procesor(*)\% času procesoru
\Procesor(*)\% uživatelského času
\Procesor(*)\% privilegovaného času
\Procesor(*)\Přerušení/s
\Procesor(*)\% času DPC
\Procesor(*)\% času přerušení
\Paměť\Chyby stránek/s
\Paměť\Bajty k dispozici
\Paměť\Svěřené bajty
\Paměť\Mez svěření
\Paměť\Zápisy kopií/s
\Paměť\Chyby převodu stavu/s
\Paměť\Chyby mezipaměti/s
\Paměť\Chyby nulových požadavků/s
\Paměť\Stránky/s
\Paměť\Vstup stránek/s
\Paměť\Čtení stránek/s
\Paměť\Výstup stránek/s
\Paměť\Bajty stránkovaného fondu
\Paměť\Bajty nestránkovaného fondu
\Paměť\Zápisy stránek/s
\Paměť\Vyhrazení stránkovaného fondu
\Paměť\Vyhrazení nestránkovaného fondu
\Memory\Free System Page Tabulka Entries
\Paměť\Bajty mezipaměti
\Paměť\Vrchol bajtů mezipaměti
\Paměť\Rezidentní bajty stránkovaného fondu
\Paměť\Celkem bajtů kódu systému
\Paměť\Rezidentní bajty kódu systému
\Paměť\Celkem bajtů systémových ovladačů
\Paměť\Rezidentní bajty systémových ovladačů
```

```
\Paměť\Rezidentní bajty systémové mezipaměti
\Paměť\% využití svěřených bajtů
\Paměť\Počet kB k dispozici
\Paměť\Počet MB k dispozici
\Paměť\Změn účelu stran přenosu/s
\Stránkovací soubor(*)\% využití
\Stránkovací soubor(*)\% vrchol využití
\Systém\Přepnutí kontextu/s
\Systém\Doba provozu systému
\Systém\Délka fronty procesoru
\Systém\Procesy
\Systém\Podprocesy
\Proces(svchost,*)\% času procesoru
\Proces(svchost,*)\% uživatelského času
\Proces(svchost,*)\% privilegovaného času
\Proces(svchost,*)\Vrchol virtuálních bajtů
\Proces(svchost,*)\Virtuální bajty
\Proces(svchost,*)\Chyby stránek/s
\Proces(svchost,*)\Vrchol pracovní sady
\Proces(svchost,*)\Pracovní sada
\Proces(svchost,*)\Vrchol bajtů stránkovacího souboru
\Proces(svchost,*)\Bajty stránkovacích souborů
\Proces(svchost,*)\Nesdílené bajty
\Proces(svchost,*)\Počet podprocesů
\Proces(svchost,*)\Základní priorita
\Proces(svchost,*)\Uplynulý čas
\Proces(svchost,*)\ID procesu
\Proces(svchost,*)\Bajty stránkovaného fondu
\Proces(svchost,*)\Bajty nestránkovaného fondu
\Tisková fronta(*)\Celkový počet vytisknutých úloh
\Tisková fronta(*)\Počet bajtů vytisknutých za sekundu
\Tisková fronta(*)\Celkový počet vytisknutých stránek
\Tisková fronta(*)\Počet úloh
\Tisková fronta(*)\Počet odkazů
\Tisková fronta(*)\Maximální počet odkazů
\Tisková fronta(*)\Počet zařazovaných úloh
\Tisková fronta(*)\Maximální počet zařazovaných úloh
\Tisková fronta(*)\Počet chyb typu Došel papír
\Tisková fronta(*)\Počet chyb typu Tiskárna není připravena
\Tisková fronta(*)\Počet chyb úloh
\Tisková fronta(*)\Počet volání na výčet síťových tiskáren
\Tisková fronta(*)\Počet volání na přidání síťové tiskárny
\Rozhraní sítě(*)\Bajty celkem/s
\Rozhraní sítě(*)\Pakety/s
\Rozhraní sítě(*)\Přijaté pakety/s
\Rozhraní sítě(*)\Odeslané pakety/s
\Rozhraní sítě(*)\Aktuální šířka pásma
\Rozhraní sítě(*)\Přijaté bajty/s
\Rozhraní sítě(*)\Přijaté jednosměrové pakety/s
\Rozhraní sítě(*)\Přijaté nejednosměrové pakety/s
\Rozhraní sítě(*)\Vyřazené přijaté pakety
\Rozhraní sítě(*)\Chyby přijatých paketů
```

```
\Rozhraní sítě(*)\Přijaté pakety neznámé
\Rozhraní sítě(*)\Odeslané bajty/s
\Rozhraní sítě(*)\Odeslané jednosměrové pakety/s
\Rozhraní sítě(*)\Odeslané nejednosměrové pakety/s
\Rozhraní sítě(*)\Vyřazené odchozí pakety
\Rozhraní sítě(*)\Chyby odchozích paketů
\Rozhraní sítě(*)\Délka fronty výstupu
\IPv4\Datagramy/s
\IPv4\Přijaté datagramy/s
\IPv4\Chyby hlaviček přijatých datagramů
\IPv4\Chyby adres přijatých datagramů
\IPv4\Předané datagramy/s
\IPv4\Neznámý protokol přijatých datagramů
\IPv4\Vyřazené přijaté datagramy
\IPv4\Doručené přijaté datagramy/s
\IPv4\Odeslané datagramy/s
\IPv4\Vyřazené odchozí datagramy
\IPv4\Odchozí datagramy bez směrování
\IPv4\Přijaté fragmenty/s
\IPv4\Fragmenty zpětně složené/s
\IPv4\Chyby zpětného skládání fragmentů
\IPv4\Fragmentování datagramů/s
\IPv4\Chyby fragmentování
\IPv4\Fragmenty vytvořené/s
\TCPv4\Segmenty/s
\TCPv4\Navázaná připojení
\TCPv4\Aktivní připojení
\TCPv4\Pasivní připojení
\TCPv4\Chyby připojení
\TCPv4\Resetovaná připojení
\TCPv4\Přijaté segmenty/s
\TCPv4\Odeslané segmenty/s
\TCPv4\Segmenty odeslané znovu/s
```

Podle takových faktorů, jako je počet instalovaných fyzických procesorů, připojených fyzických disků, definovaných logických disků a počet adaptérů síťového rozhraní, bude celý objem čítačů uvedených ve výpisu 4.7 souboru nastavení čítačů generovat data o objemu dosahujícím 30 MB denně na malém počítači a až 100 MB nebo i více denně na velkém stroji.

Sledování indikátorů chyb Řada čítačů, zahrnutých v ukázkovém souboru nastavení čítačů ve výpisu 4.7, představuje indikátory určitého chybového stavu. Takové chybové podmínky se neomezují jen na nedostatek prostředků – některé odrážejí např. nesprávně nakonfigurované služby systému a sítě. Jiné mohou indikovat činnost související s pokusy o narušení zabezpečení počítače ze strany neoprávněných uživatelů. Význam zahrnutí těchto čítačů do denních monitorovacích procedur spočívá v tom, že máte možnost přesně stanovit dobu, kdy k takovým chybovým stavům v systémových službách dochází.

Výpis 4.8 ukazuje čítače obsažené již v souboru nastavení čítačů podle výpisu 4.7, jež slouží především jako indikátory chyb na souborovém a tiskovém serveru. Tato

množina čítačů zahrnuje chybové indikátory relací souborového serveru, chybové stavy tiskáren a obecné chyby sítě.

VÝPIS 4.8: Čítače, jež jsou ukazateli chyb na souborovém a tiskovém serveru

a) produkt v angličtině:

```
\Server\Sessions Timed Out
\Server\Sessions Errored Out
\Server\Sessions Logged Off
\Server\Sessions Forced Off
\Server\Errors Logon
\Server\Errors Access Permissions
\Server\Errors Granted Access
\Server\Errors System
\Server\Blocking Requests Rejected
\Server\Work Item Shortages
\Server\Pool Nonpaged Bytes
\Server\Pool Nonpaged Failures
\Server\Pool Paged Failures
\Print Queue(*)\Out of Paper Errors
\Print Queue(*)\Not Ready Errors
\Network Interface(*)\Packets Received Discarded
\Network Interface(*)\Packets Received Errors
\Network Interface(*)\Packets Received Unknown
\Network Interface(*)\Packets Outbound Discarded
\Network Interface(*)\Packets Outbound Errors
\IPV4\Datagrams Received Header Errors
\IPV4\Datagrams Received Address Errors
\IPV4\Datagrams Received Unknown Protocol
\IPV4\Datagrams Received Discarded
\IPV4\Datagrams Outbound Discarded
\IPV4\Datagrams Outbound No Route
\IPV4\Fragment Re-assembly Failures
\IPV4\Fragmentation Failures
\TCPV4\Connection Failures
\TCPV4\Connections Reset
```

b) produkt s českou lokalizací

```
\Server\Vypršelé relace
\Server\Chybné relace
\Server\Odhlášené relace
\Server\Odpojené relace
\Server\Chyby přihlášení
\Server\Chyby přístupových práv
\Server\Chyby udělených práv
\Server\Chyby systému
\Server\Zamítnuté žádosti zablokování
\Server\Nedostatky pracovních položek
\Server\Bajty nestránkovaného fondu
\Server\Chyby nestránkovaného fondu
\Server\Chyby stránkovaného fondu
\Tisková fronta(*)\Počet chyb typu Došel papír
```

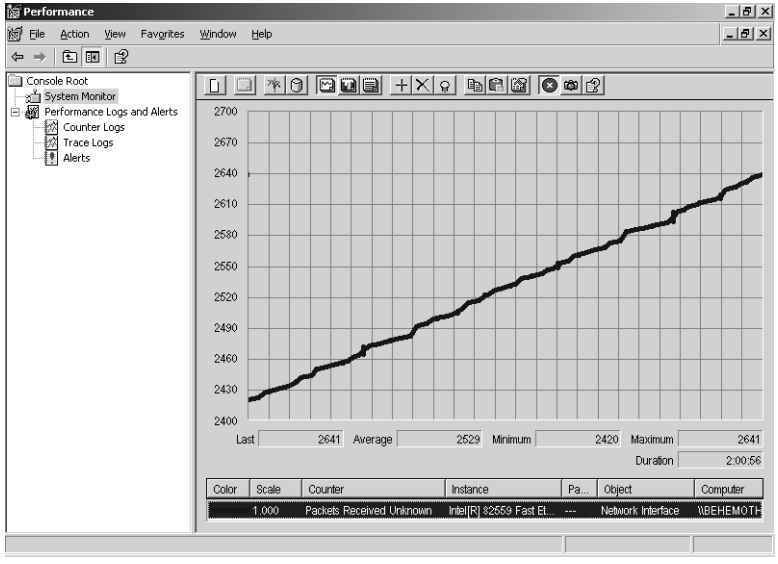
```
\Tisková fronta(*)\Počet chyb typu Tiskárna není připravena
\Rozhraní sítě(*)\Vyřazené přijaté pakety
\Rozhraní sítě(*)\Chyby přijatých paketů
\Rozhraní sítě(*)\Přijaté pakety neznámé
\Rozhraní sítě(*)\Vyřazené odchozí pakety
\Rozhraní sítě(*)\Chyby odchozích paketů
\IPv4\Chyby hlaviček přijatých datagramů
\IPv4\Chyby adres přijatých datagramů
\IPv4\Neznámý protokol přijatých datagramů
\IPv4\Vyřazené přijaté datagramy
\IPv4\Vyřazené odchozí datagramy
\IPv4\Odchozí datagramy bez směrování
\IPv4\Chyby zpětného skládání fragmentů
\IPv4\Chyby fragmentování
\TCPv4\Chyby připojení
\TCPv4\Resetovaná připojení
```

Čítače, jež jsou uvedeny ve výpisu 4.8 a které zaznamenávají počet vzniklých chybových stavů, jsou většinou okamžité neboli čisté. Obsahují ale kumulativní hodnoty. Jelikož počet chybových stavů, které nastávají, by měl být velmi nízký, při použití intervalového rozdílového čítače ke sledování chyb by byly hlášeny tak malé rychlosti chyb za sekundu, že by se objevovaly jako nulové hodnoty. Aby se tedy cenné metriky nezobrazovaly jako nuly, hlásí tyto čítače kumulativní počty chyb.

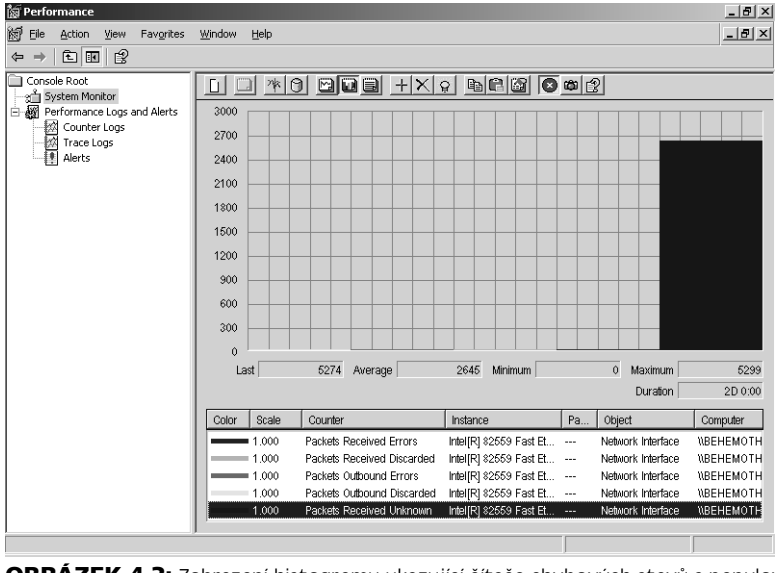
Jedna z nevýhod tohoto přístupu spočívá v tom, že nemůžete použít prvek Alerts (Výstrahy) nástroje Performance (Sledování výkonu) a nechat se upozornit na vznik chybového stavu. Graf na obrázku 4.2 vykresluje hodnoty chybových podmínek jednoho čítače síťového rozhraní v rámci dvouhodinové monitorovací relace. Všimněte si, že křivka vyznačující hodnoty čítače Network Interface\Packets Received Unknown (Rozhraní sítě(*)\Přijaté pakety neznámé) během sledovacího intervalu trvale roste. Ať už dochází k jakémukoli chybovému stavu, vyskytuje se pravidelně. Protože tento čítač udává kumulativní (nárůstový) počet vzniklých chybových podmínek od posledního spuštění systému, jakmile se objeví nějaký chybový stav, hodnota čítače již bude trvale nenulová. Je zřejmé, že když se jednou spustí výstraha na jednom z takových čítačů, bude se opakovat ve všech následujících měřicích intervalech. Takové výstrahy „zamoří“ protokol událostí aplikací.

Chcete-li zjistit, zda dochází k výskytu těchto chybových podmínek, je nejsnazší a nejrychlejší vyvinout zprávu využívající zobrazení histogramu, což vám dovolí rozpoznat všechny nenulové chybové stavy. Příklad tohoto přístupu je na obrázku 4.3. Ten ukazuje, že v rámci našeho sledovacího intervalu dochází pouze k chybám Přijaté pakety neznámé.

Jakmile z histogramu odstraníte všechny čítače chybových stavů, které hlásí nulové hodnoty, můžete se přepnout do zobrazení grafu a blíže prozkoumat zbývající nenulové čítače. S využitím zobrazení grafu lze stanovit, kdy se příslušné chybové stavy objevily a jakou rychlostí vznikaly.



OBRÁZEK 4.2: Hodnoty jednoho čítače chybových stavů síťového rozhraní v rámci dvouhodinové monitorovací relace



OBRÁZEK 4.3: Zobrazení histogramu ukazující čítače chybových stavů s nenulovými hodnotami

Efektivní využívání výstrah

Kvůli dodatečným výkonnostním nákladům a velikosti protokolů čítačů je jen zřídkakdy možné shromažďovat neustále všechny výkonnostní statistiky, které byste mohli potřebovat. K velikosti generovaných souborů protokolů čítačů nejvíce přispívá sběr dat na úrovni procesů – dokonce i na menších počítačích se systémem Windows Server 2003 většinou běží mnoho procesů, což znamená rozsáhlé soubo-

ry protokolů čítačů. V ukázkovém souboru nastavení čítačů ve výpisu 4.7 je tento potenciální problém vyřešen sběrem výkonnostních čítačů na úrovni procesů vždy jen u omezeného počtu procesů. Ve výpisu 4.7 a v příkladech uvedených v tabulce 4.1 byla vybrána výkonnostní data na úrovni procesů jen u těch procesů, které jsou pro provozovanou serverovou aplikaci nejzásadnější.

Tento selektivní sběr dat však není zcela uspokojivým řešením, protože výkonnostní data na úrovni jednotlivých procesů jsou nezbytná k detekování poškozených procesů, které si monopolizují procesor nebo kterým uniká virtuální paměť. Řešením tohoto rozporu je používat prvek Alerts (Výstrahy) konzoly Performance (Sledování výkonu) a zajistit automatické spouštění relace sběru dat čítačů nástroje Log Manager, jakmile nějaká výstraha překročí stanovený práh. Když procedury protokolování čítačů implementujete tímto způsobem, budete moci automaticky shromažďovat velmi detailní údaje o problémech, aniž by bylo nutné neustále sbírat všechna výkonnostní data.

Relaci sběru dat protokolů čítačů nástroje Log Manager automaticky spouštěnou, jakmile dojde ke spuštění nějaké výstrahy, nastavte těmito jednoduchými kroky:

1. Definujte protokol čítačů nástroje Log Manager, jehož použití plánujete pro sběr detailních výkonnostních dat o určité situaci.
2. Definujte podmínku výstrahy použitou ke spuštění dané relace protokolu čítačů.
3. V definici akce po výstraze zadejte spuštění příslušné relace protokolu čítačů definované v kroku 1.

Chcete-li například pomáhat se zjištěním stavu, kdy nějakému procesu uniká virtuální paměť, definujte relaci protokolu čítačů, která bude na úrovni procesů shromažďovat data související s alokováním virtuální paměti. Následně definujte chybový stav, který se spustí, kdykoli alokování virtuální paměti dosáhne kritické úrovně. Nakonec definujte akci této výstrahy zajišťující spuštění relace protokolu čítačů definované v kroku 1 (předchozího postupu), jakmile se daná podmínka výstrahy projeví.

Následující oddíl vás provede ukázkou implementace výstrahy, která se spustí, jakmile bude nedostatek virtuální paměti. Následně bude iniciovat relaci sběru dat protokolu čítačů, jež vám umožní stanovit, který proces zodpovídá za nedostatek virtuální paměti. V kapitole 5, „Řešení potíží s výkonem“, jsou probírány doplňkové techniky detekce a diagnostiky úniků virtuální paměti.

Upozornění Musíte si dávat pozor na to, aby vaše výkonnostní výstrahy nezahltily aplikační protokol událostí nadměrným počtem zpráv. Periodicky byste měli prohlížet aplikační protokol událostí a kontrolovat, zda vaše nastavení výstrah negenerují příliš mnoho položek protokolu událostí. Programy jako Microsoft Operations Manager dokážou aktivně spravovat protokoly událostí a konsolidovat i potlačovat duplicitní položky událostí, aby nedocházelo k neustálému generování zpráv výstrah souvisejících se stále stejným stavem.

Automatické spouštění protokolů čítačů

Tento oddíl vás krok za krokem provede procedurou generující diagnostické protokoly čítačů, kdykoli dojde k nedostatku virtuální paměti, který je možná způsoben poškozeným procesem, jemuž uniká virtuální paměť.

Krok 1: Definujte nastavení protokolu čítačů Jedná se o nastavení protokolu čítačů použitá, když výstraha spustí svou prahovou podmínku. Použijte například následující příkaz:

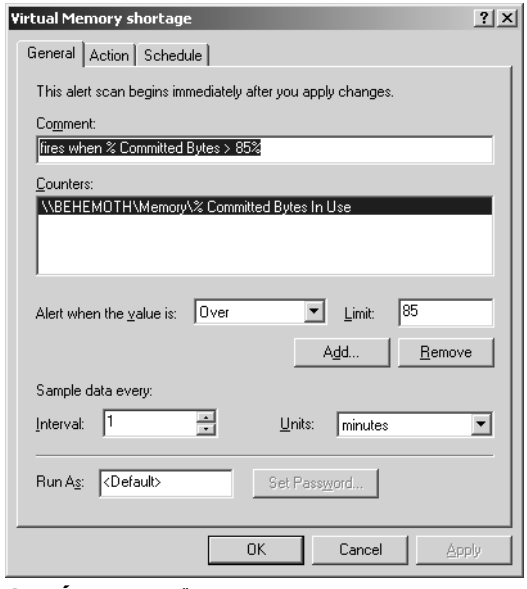
```
Logman create counter MemoryTroubleshooting -v mmddhhmm
  -c "\Memory\Available Bytes" "\Memory\% Committed Bytes in Use"
  "\Memory\Cache Bytes" "\Memory\Pool Nonpaged Bytes"
  "\Memory\Pool Paged Bytes" "\Process(*)\Pool Nonpaged Bytes"
  "\Process(*)\Pool Paged Bytes" "\Process(*)\Pool Virtual Bytes"
  "\Process(*)\Pool Private Bytes" "\Process(*)\Page Faults/sec"
  "\Process(*)\Pool Working Set" -si 00:30
  -o "c:\Perflogs\Alert Logs\MemoryTroubleshooting" -max 5
```

Respektive v českém prostředí:

```
Logman create counter ReseniPotiziPameti -v mmddhhmm
  -c "\Paměť\Bajty k dispozici" "\Paměť\% využití svěřených bajtů"
  "\Paměť\Bajty mezipaměti" "\Paměť\Bajty nestránkovaného fondu"
  "\Paměť\Bajty stránkovaného fondu"
  "\Proces(*)\Bajty nestránkovaného fondu"
  "\Proces(*)\Bajty stránkovaného fondu" "\Proces(*)\Virtuální bajty"
  "\Proces(*)\Nesdílené bajty" "\Proces(*)\Chyby stránek/s"
  "\Proces(*)\Pracovní sada" -si 00:30
  -o "c:\Perflogs\Protokoly vystrah\ReseniPotiziPameti" -max 5
```

Tento příkaz definuje relaci protokolu čítačů ReseniPotiziPameti, která shromažďuje čítače alokování virtuální paměti na úrovni procesů a také několik celosystémových čítačů alokování virtuální paměti. V našem příkladu vzorkujeme čítače ve 30sekundových intervalech. Zadání parametru `-max` zajistí ukončení sběru dat, jakmile protokol čítačů dosáhne velikosti 5 MB. Pomocí konzoly Performance Monitor byste rovněž mohli omezit dobu trvání sběru dat na hodnotu kupříkladu 10 až 15 minut. Všimněte si, že protokoly čítačů se budou vytvářet ve složce C:\Perflogs\Protokoly vystrah\, což je složka odlišná od té, kde vytváříte pravidelné denní protokoly výkonu. Tato samostatná složka vám umožňuje snáze odlišit relace protokolování čítačů spuštěné výstrahami a usnadňuje také jejich správu podle různých kritérií. Kupříkladu uvedený protokol ReseniPotiziPameti je detailním pohledem omezeným na data související s alokováním virtuální paměti. Takový typ protokolu čítačů nebudete nikdy muset sumarizovat, ani nebude obvykle zapotřebí zachovávat detailní historický záznam o podobných typech potíží.

Krok 2: Definujte obecné vlastnosti výstrahy Test (nebo testy) prahové hodnoty čítače, který spouští určitý stav výstrahy, definujete na kartách dialogového okna vlastností výstrahy virtuální paměti. V našem příkladu je zapotřebí vytvořit výstrahu spouštěnou nedostatkem virtuální paměti – testovanou prahovou hodnotou čítače Memory\ % Committed Bytes In Use (Paměť\% využití svěřených bajtů) je 85 procent, jak ilustruje obrázek 4.4.



OBRÁZEK 4.4: Čítač % Committed Bytes In Use přesahující 85 %

Rychlost, jakou sledování výstrah vzorkuje vámi vybraný výkonnostní čítač (nebo čítače), určuje, jak často lze zprávy výstrahy generovat. V našem případě je běh sledování výstrahy naplánován jednou za minutu. Na počítači s omezenou virtuální pamětí, kde hodnota Paměť\% využití svěřených bajtů trvale přesahuje limitních 85 procent, se daná podmínka výstrahy spouští každou minutu. V závislosti na vybrané akci výstrahy se může tato frekvence projevit jako příliš vysoká.

Tip Je-li to možné, definujte takové podmínky výstrah, které se za normálního stavu nespouštějí více než několikrát za hodinu. Upravte nastavení těch stavů výstrahy, jež se spouštějí častěji než 5- až 10krát za hodinu, a to dokonce i za velmi nepříznivých podmínek, aby se spouštěly méně často.

Pokud se výstrahy vyskytují příliš často, jenom své příjemce zlobí. Z psychologického hlediska ztrácejí často spouštěné výstrahy svůj význam jako upozornění na důležité události. Nakonec budou považovány za běžné věci, které lze klidně ignorovat. Tyto lidské reakce na daný stav jsou pochopitelné, ale velmi nežádoucí, pokud zadaný práh výstrahy skutečně představuje výjimečný stav vyžadující pozornost a detailnější zkoumání.

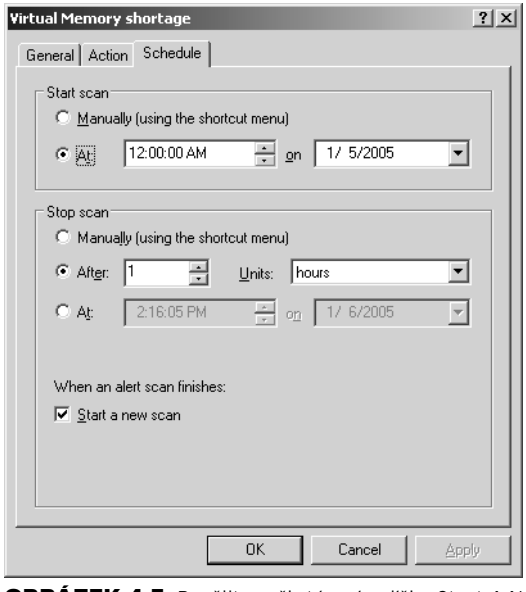
Frekvenci spouštění výstrah můžete snadno řídit jedním z následujících způsobů:

1. Upravte prahovou podmínku tak, aby se výstrahy spouštěly méně často.
2. Zpomalte frekvenci spouštění skenování výstrah.

Navíc můžete omezit akce výstrahy na ty, které úmyslně nebudou nikoho obtěžovat.

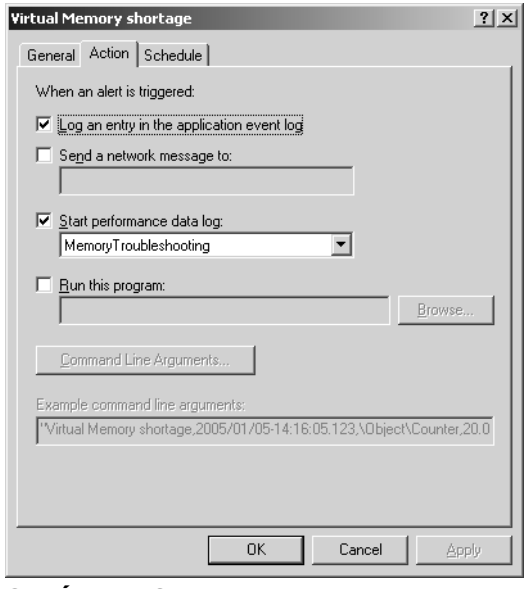
Krok 3: Definujte časový plán výstrahy Trvání sledování výstrah je dáno parametry časového plánu výstrah. Nejlepších výsledků dosáhnete omezením trvání sledování výstrah na 1 až 2 hodiny. V případě trvalého monitorování zajistěte, aby se nové sledování spustilo až po dokončení toho aktuálního, jak zachycuje obrázek 4.5.

Krok 4: Definujte akci výstrahy Akci vykonanou prvkem výstrah po naplnění zadané podmínky definují parametry akce výstrahy. My potřebujeme, aby prvek výstrah inicioval protokol čítačů definovaný v kroku 1 a zajistil tak sběr detailnějších údajů o výkonu aplikací na úrovni procesů v okamžiku spuštění výstrahy. Nezapomínejte, že zadaný protokol čítačů se spustí jen jednou na sledování výstrah. Oproti tomu položky protokolu událostí se generují v každém vzorkovacím intervalu, v němž je splněna podmínka výstrahy. Také zprávy výstrah se generují v každém vzorkovacím intervalu s naplněnou podmínkou výstrahy. Zvolíte-li spuštění nějakého programu (viz obrázek 4.6), naplánuje prvek výstrah jeho běh jen jednou v rámci sledování výstrah.



OBRÁZEK 4.5: Použijte zaškrtávací políčko Start A New Scan (Spustit nové sledování)

Protokol čítačů nebo zadaný program se spustí bezprostředně po prvním vzorkovacím intervalu sledování výstrah, v němž je zadaná podmínka výstrahy pravdivá. Doba trvání relace protokolování čítačů je určena parametry časového plánu, které zadáváte v rámci definování dané relace protokolu čítačů. Jak již bylo poznamenáno, pomocí rozhraní příkazového řádku nástroje Log Manager můžete omezit velikost souboru protokolu čítačů, který bude vytvořen. S využitím rozhraní konzoly Sledování výkonu máte možnost omezit dobu trvání relace protokolu čítačů. Dostatečně efektivní detailní protokol čítačů, který vám dovolí do hloubky prozkoumat stav počítače, má obvykle rozsah od 10 do 30 minut. Je zřejmé, že pokud se podmínky výstrah spouštějí dostatečně často a detailní relace protokolů čítačů jsou relativně dlouhé, riskujete shromáždění mnohem více dat o potenciálním problému, než dokážete později efektivně analyzovat. Navíc můžete spotřebovávat příliš mnoho diskového prostoru na počítači trpícím sledovaným problémem.



OBRÁZEK 4.6: Máte možnost zadat spouštěný program

Obecné procedury výstrah

Podobné procedury výstrah bude zapotřebí definovat pro spouštění takových protokolů čítačů, jež vám umožní zkoumat období nadměrného využívání procesoru, nedostatku fyzické paměti a možné potíže s výkonem disku. Tabulka 4.3 shrnuje základní procedury výstrah, které budou pro vaše produkční servery užitečné. Pojednání o nastavení konfiguračně závislých prahů výstrah stavu nadměrného stránkování najdete v kapitole 3.

Testy prahových podmínek výstrah uvedené na obrázku 4.3 zachycují reprezentativní hodnoty využitelné na mnoha počítačích. Specifičtější doporučení týkající se prahových podmínek výstrah těchto čítačů najdete v obsáhlém pojednání v kapitole 3, „Měření výkonu serveru“.

Aplikační výstrahy

Může být také vhodné definovat dodatečná sledování výstrah vycházející z kritérií specifických pro konkrétní aplikace. V každé situaci se musíte zaměřit na podmínky výstrah související s nadměrnou spotřebou prostředků, ukazatele nedostatku prostředků, měření zachycující velký počet požadavků čekajících ve frontě a další anomálie související s výkonem aplikací. U některých aplikací je vhodné generovat výstrahy jak při vysokých rychlostech transakcí, tak i při jejich nezvykle nízkých hodnotách, což může také znamenat, že daná aplikace přestala reagovat a vykonávání transakcí se blokuje.

TABULKA 4.3: Nastavení obecných stavů výstrah

Podmínka	Prahové testy	Frekvence sledování (v sekundách	Další protokolované čítače	Interval vzorkování (v sekundách)
Nadměrné využívání CPU; možný běh procesu v nekonečném cyklu	Processor(*)\% Processor Time > 98%	10–30	Processor(*)*; Process(*)\% Processor Time, Process(*)\% Privileged Time; Process(*)\% User Time	10–20 po 10–20 minut
Únik virtuální paměti procesu	Memory\% Committed Bytes In Use > 85%	10–30	Memory*; Process(*)\Private Bytes, Process(*)\Virtual Bytes, Process(*)\Pool Nonpaged Bytes, Process(*)\Pool Paged Bytes, Process(*)\Page File Bytes	15–30 po 10–30 minut
Nadměrné stránkování na disk	Memory\Available Kbytes < <práh> Memory\Pages/sec > <práh>	10–30	Memory*; Physical Disk(n)\% Idle Time, Physical Disk(n)\Avg. Disk Secs/Transfer, Physical Disk(n)\Transfers/sec; Process(*)\Page Faults/sec	10–20 po 10–20 minut
Slabý výkon disku	Physical Disk(n)\Avg. Disk Secs/Transfer > 20 Physical Disk(n)\Transfers/sec > 200 Physical Disk(n)\Current Disk Queue Length > 5	15	Physical Disk(n)\% Idle Time, Physical Disk(n)\Avg. Disk Secs/Transfer, Physical Disk(n)\Transfers/sec	10 po 10 minut

Tabulka 4.4 uvádí doporučená nastavení výstrah několika rozšířených serverových aplikací. Hodnoty používané v testech prahových podmínek výstrah u aplikačních serverů se obvykle na jednotlivých sídlech liší, jak je tu uvedeno.

Další informace Další pomoc s nastavením prahů výstrah konkrétních aplikací najdete v online dokumentaci Microsoft Operations Framework v oddílu TechNet Products and Technologies na adrese http://www.microsoft.com/technet/.

Denní hlášení pro správu

Hlášení pro správu je způsob, jakým zajistit přístup ke klíčovým údajům a zprávám popisujícím chování počítačů se systémem Windows Server 2003 všem účastníkům vaší instalace, které zajímá výkonnost serveru. Hlášení pro správu nelze zaměňovat s detailní a hloubkovou analýzou dat prováděnou zkušeným analytikem výkonu, jakmile se objeví výkonnostní potíže, které je zapotřebí diagnostikovat. Hlášení pro

správu se zaměřuje na několik klíčových metrik, které chápe mnoho lidí. Vytvářené grafy a tabulky by měly být relativně jednoduché a nekomplikované.

TABULKA 4.4: Ukázková nastavení aplikačních podmínek výstrah

Aplikace	Podmínka	Prahové testy	Další protokolované objekty
Řadiče domény	Nadměrné požadavky na Active Directory	NTDS\LDAP Searches/sec > <práh>	NTDS*
Stránky ASP (nebo aplikace ASPX rámce .NET Framework)	Nadměrné řazení požadavků ASP do front	ASP\Requests Queued > <práh>	ASP; Internet Information System Global; Web Service; Process(w3wp)*
Souborový server	Nedostatky prostředků	Server Work Queues(*)\Queue Length > <práh>	Server; Server Work Queues; Cache; Memory; Processor; Process(svchost)\% Processor Time
SQL Server	Nadměrné databázové transakce	SQL Server:Databases(n)\Transactions/sec > <práh>	SQL Server:Buffer Manager; SQL Server:Cache Manager; SQL Server:Memory Manager; SQL Server:Locks; SQL Server:SQL Statistics;SQL Server:Databases

Metriky, o které se techničtí pracovníci a další pověřené osoby nejvíce zajímají, zahrnují následující:

- Míry využití klíčových prostředků, jako jsou procesory, paměť, disky a síť.
- Dostupnost a rychlost transakcí základních aplikací, jako jsou webové servery, databázové servery a servery Exchange elektronické pošty a zpráv.
- Doby obsluhy a reakce transakcí v oblasti klíčových aplikací, pokud je lze zjistit.

Sumarizace denních protokolů čítačů

Protože hlášení pro správu by měla poskytovat náhled na výkon, bude zapotřebí generovat denní soubory obsahující výkonnostní data sumarizovaná nástrojem Relog. Detailní generované denní protokoly čítačů nejsou tím nejlepším k vytváření hlášení pro správu. Takový denní protokol čítačů, jenž sbírá jednominutové vzorky trvale po dobu 24 hodin, nakumuluje každý den 1440 pozorování každého protokolovaného čítače výkonu. (Skutečný počet denně vygenerovaných vzorků bude 1441, protože jeden další vzorek je zapotřebí na začátku k získání počátečních hodnot všech čítačů.) To je mnohem více dat, než dokáže nástroj System Monitor (Sledování systému) vměstnat do zobrazení grafu, jenž je omezen na vykreslení 100 datových bodů na své časové ose x. Chcete-li vytvořit zobrazení grafu pro 24hodinové období, bude muset Sledování systému destilovat 14 samostatných měření a vykreslit místo nich souhrnné statistiky, což může mít snadno za následek pokřivený pohled na výkonnostní údaje.

Použití nástroje Relog k zajištění sumarizované verze shromažďovaných denních protokolů čítačů zjednoduší proces vytváření užitečných denních hlášení pro správu. Kupříkladu zadáním dále uvedeného příkazu vytvoříte kompaktní verzi jednoho ze svých denních protokolů čítačů, sumarizovanou do ideálních 15minutových intervalů:

```
relog zakladniDenniProtokol_20031228.blg
  -o <nazev-pocitace>.zakladniDenniProtokol.blg -f BIN -t 15
```

Takové sumarizované verze denních protokolů čítačů jsou velmi užitečné k rychlému a výhodnému sestavování hlášení pro správu.

Když ve svých hlášeních pro správu použijete sumarizovaná data, eliminují se všechna pokřivení, jež se mohou objevovat, jakmile je zobrazení grafu nuceno vypouštět tolik dodatečných pozorování. Shrnutí 24hodinové periody do 15minutových intervalů nakonec poskytne trochu méně než 100 pozorování na každý čítač, což je hodnota ideálně zapadající do zobrazení grafu nástroje Sledování systému. Nezapomínejte ale také na to, že shrnování naměřených dat na této úrovni vyhladí mnoho špiček a propadů zachycených v originálním, detailním protokolu čítačů. Když zkoumáte nějaký výkonnostní problém, budete využívat původní data protokolu čítačů a také všechny detailní protokoly čítačů daného intervalu, které automaticky vygenerovaly vaše procedury výstrah.

Tip Podrobné denně vytvářené protokoly čítačů jsou vhodné pro generování hlášení pro správu zaměřující se na špičkové 1- až 2hodinové období. Jakmile hodinová rychlost transakcí překročí dvojnásobek průměrné rychlosti transakcí, jsou hlášení pro správu zaměřená na taková špičková období extrémně užitečná. Jsou-li výkonnostně úzká místa patrná jen během takových špičkových zatížení, velmi zajímavá jsou rovněž hlášení zaměřující se na tato úzká časová období.

Konsolidování výkonnostních dat z více serverů Zodpovídáte-li za výkon velkého počtu serverů, pak budete zřejmě chtít shromažďovat protokoly čítačů z mnoha takových serverů, abyste mohli sestavovat potřebná hlášení z jediného místa. Kvůli úspoře diskového prostoru na centrálním místě bude vhodné provádět tuto konsolidaci s využitím sumarizovaných protokolů čítačů a nikoli s objemnými, původně nasbíranými detailními protokoly čítačů. Takovou konsolidaci lze provádět denně s využitím řady automatizovaných procedur následujícím způsobem:

1. Pomocí nástroje Relog vytvořte na lokálním serverovém počítači sumarizovanou (souhrnnou) verzi generovaných souborů denních protokolů čítačů.
2. Do názvu sumarizovaného souboru, který je výstupem nástroje Relog, začleňte název počítače, odkud protokol čítačů pochází.
3. Zkopírujte soubor sumarizovaného protokolu čítačů na nějaké centrální místo.
4. Na konsolidačním serveru využijte nástroj Relog a zkombinujte všechny protokoly čítačů do jediného výstupního souboru, který lze využívat k dennímu hlášení.

Dále v tomto oddílu najdete příklady skriptů, které lze použít k automatizování naznačených funkcí denního zpracování. Alternativou ke konsolidaci protokolů čítačů nashromážděných na mnoha počítačích v jednom centrálním místě během následného zpracování je vytváření všech protokolů čítačů na centrálním umístění již od začátku. Jak je zdůrazněno v oddílu „Protokolování na místo sdílené v síti“ v této kapitole, tato volba má dopady na škálovatelnost rozsáhlých serverových farem,

takže je zapotřebí implementovat ji s rozvahou. Podrobnější pojednání o těchto věcech najdete v oddílu nazvaném „Protokolování místních čítačů na lokální disk“ dříve v této kapitole.

Při vykonávání nástroje Relog s cílem vytvořit sumarizované denní protokoly čítačů vhodné k hlášení pro správu a archivaci můžete získané výkonnostní metriky ještě dále editovat a odstranit čítače, o kterých se v hlášeních pro správu nechcete zmiňovat. Při vykonávání nástroje Relog s cílem vytvořit sumarizované denní protokoly čítačů se můžete odkazovat na nějaký soubor nastavení protokolu čítačů, jenž potřebnou editaci zajistí.

Máte kupříkladu možnost zavolat Relog dále uvedeným způsobem, kde `soubor-prenastaveni-citacu-protokolu.txt` představuje užší podmnožinu původního `zakladni-soubor-nastaveni-citacu.txt` použitého ke generování plných denních protokolů čítačů.

```
relog zakladniDenniProtokol_20031228.blg
  -o <nazev-pocitace>.zakladniDenniProtokol.blg
  -cf soubor-prenastaveni-citacu-protokolu.txt -f BIN -t 15
```

Výpis 4.9 ukazuje doporučený obsah souboru nastavení čítačů pro nástroj Relog vhodný k vytváření sumarizovaných souborů pro denní hlášení pro správu.

VÝPIS 4.9: Soubor-prenastaveni-citacu-protokolu.txt

a) produkt v angličtině:

```
\LogicalDisk(*)\% Free Space
\LogicalDisk(*)\Free Megabytes
\LogicalDisk(*)\Current Disk Queue Length
\PhysicalDisk(*)\Current Disk Queue Length
\PhysicalDisk(*)\Avg. Disk Queue Length
\PhysicalDisk(*)\Avg. Disk sec/Transfer
\PhysicalDisk(*)\Avg. Disk sec/Read
\PhysicalDisk(*)\Avg. Disk sec/Write
\PhysicalDisk(*)\Disk Transfers/sec
\PhysicalDisk(*)\Disk Reads/sec
\PhysicalDisk(*)\Disk Writes/sec
\PhysicalDisk(*)\Disk Bytes/sec
\PhysicalDisk(*)\Disk Read Bytes/sec
\PhysicalDisk(*)\Disk Write Bytes/sec
\PhysicalDisk(*)\Avg. Disk Bytes/Transfer
\PhysicalDisk(*)\Avg. Disk Bytes/Read
\PhysicalDisk(*)\Avg. Disk Bytes/Write
\PhysicalDisk(*)\% Idle Time
\Processor(*)\% Processor Time
\Processor(*)\% User Time
\Processor(*)\% Privileged Time
\Processor(*)\Interrupts/sec
\Processor(*)\% DPC Time
\Processor(*)\% Interrupt Time
\Memory\Page Faults/sec
\Memory\Available Bytes
\Memory\Committed Bytes
```

```
\Memory\Commit Limit
\Memory\Transition Faults/sec
\Memory\Cache Faults/sec
\Memory\Demand Zero Faults/sec
\Memory\Pages/sec
\Memory\Pages Input/sec
\Memory\Page Reads/sec
\Memory\Pages Output/sec
\Memory\Pool Paged Bytes
\Memory\Pool Nonpaged Bytes
\Memory\Page Writes/sec
\Memory\Cache Bytes
\Memory\Pool Paged Resident Bytes
\Memory\System Code Resident Bytes
\Memory\System Driver Resident Bytes
\Memory\System Cache Resident Bytes
\Memory\% Committed Bytes In Use
\Memory\Available KBytes
\Memory\Available MBytes
\Memory\Transition Pages RePurposed/sec
\System\Context Switches/sec
\System\Processor Queue Length
\Process(sqlserver)\% Processor Time
\Process(sqlserver)\% User Time
\Process(sqlserver)\% Privileged Time
\Process(sqlserver)\Virtual Bytes
\Process(sqlserver)\Page Faults/sec
\Process(sqlserver)\Working Set
\Process(sqlserver)\Private Bytes
\Process(sqlserver)\Elapsed Time
\Process(sqlserver)\Pool Paged Bytes
\Process(sqlserver)\Pool Nonpaged Bytes
\Process(inetinfo)\% Processor Time
\Process(inetinfo)\% User Time
\Process(inetinfo)\% Privileged Time
\Process(inetinfo)\Virtual Bytes
\Process(inetinfo)\Page Faults/sec
\Process(inetinfo)\Working Set
\Process(inetinfo)\Private Bytes
\Process(inetinfo)\Elapsed Time
\Process(inetinfo)\Pool Paged Bytes
\Process(inetinfo)\Pool Nonpaged Bytes
\RAS Total\Bytes Transmitted/Sec
\RAS Total\Bytes Received/Sec
\RAS Total\Total Errors/Sec
\Print Queue(*)\Total Jobs Printed
\Print Queue(*)\Bytes Printed/sec
\Print Queue(*)\Total Pages Printed
\Print Queue(*)\Jobs
\Network Interface(*)\Bytes Total/sec
\Network Interface(*)\Packets/sec
\Network Interface(*)\Packets Received/sec
```

```
\Network Interface(*)\Packets Sent/sec
\Network Interface(*)\Current Bandwidth
\Network Interface(*)\Bytes Received/sec
\Network Interface(*)\Bytes Sent/sec
\Network Interface(*)\Output Queue Length
\IP\Datagrams/sec
\IP\Datagrams Received/sec
\IP\Datagrams Sent/sec
\TCP\Segments/sec
\TCP\Connections Established
\TCP\Connections Active
\TCP\Connection Failures
\TCP\Connections Reset
\TCP\Segments Received/sec \TCP\Segments Sent/sec
\TCP\Segments Retransmitted/sec
```

b) produkt s českou lokalizací

```
\Logický disk(*)\% volného místa
\Logický disk(*)\Volné megabajty
\Logický disk(*)\Aktuální délka fronty disku
\Fyzický disk(*)\Aktuální délka fronty disku
\Fyzický disk(*)\Střední délka fronty disku
\Fyzický disk(*)\Střední doba disku/přenos
\Fyzický disk(*)\Střední doba disku/čtení
\Fyzický disk(*)\Střední doba disku/zápis
\Fyzický disk(*)\Přenosy disku/s
\Fyzický disk(*)\Čtení z disku/s
\Fyzický disk(*)\Zápisy na disk/s
\Fyzický disk(*)\Bajtů disku/s
\Fyzický disk(*)\Bajty čtení z disku/s
\Fyzický disk(*)\Bajty zapisování na disk/s
\Fyzický disk(*)\Střední počet bajtů disku/přenos
\Fyzický disk(*)\Střední počet bajtů disku/čtení
\Fyzický disk(*)\Střední počet bajtů disku/zápis
\Fyzický disk(*)\% času nečinnosti
\Procesor(*)\% času procesoru
\Procesor(*)\% uživatelského času
\Procesor(*)\% privilegovaného času
\Procesor(*)\Přerušení/s
\Procesor(*)\% času DPC
\Procesor(*)\% času přerušení
\Paměť\Chyby stránek/s
\Paměť\Bajty k dispozici
\Paměť\Svěřené bajty
\Paměť\Mez svěření
\Paměť\Chyby převodu stavu/s
\Paměť\Chyby mezipaměti/s
\Paměť\Chyby nulových požadavků/s
\Paměť\Stránky/s
\Paměť\Vstup stránek/s
\Paměť\Čtení stránek/s
\Paměť\Výstup stránek/s
```

```
\Paměť\Bajty stránkovaného fondu
\Paměť\Bajty nestránkovaného fondu
\Paměť\Zápisy stránek/s
\Paměť\Bajty mezipaměti
\Paměť\Rezidentní bajty stránkovaného fondu
\Paměť\Rezidentní bajty kódu systému
\Paměť\Rezidentní bajty systémových ovladačů
\Paměť\Rezidentní bajty systémové mezipaměti
\Paměť\% využití svěřených bajtů
\Paměť\Počet kB k dispozici
\Paměť\Počet MB k dispozici
\Paměť\Změn účelu stran přenosu/s
\Systém\Přepnutí kontextu/s
\Systém\Délka fronty procesoru
\Proces(sqlserver)\% času procesoru
\Proces(sqlserver)\% uživatelského času
\Proces(sqlserver)\% privilegovaného času
\Proces(sqlserver)\Virtuální bajty
\Proces(sqlserver)\Chyby stránek/s
\Proces(sqlserver)\Pracovní sada
\Proces(sqlserver)\Nesdílené bajty
\Proces(sqlserver)\Uplynulý čas
\Proces(sqlserver)\Bajty stránkovaného fondu
\Proces(sqlserver)\Bajty nestránkovaného fondu
\Proces(inetinfo)\% času procesoru
\Proces(inetinfo)\% uživatelského času
\Proces(inetinfo)\% privilegovaného času
\Proces(inetinfo)\Virtuální bajty
\Proces(inetinfo)\Chyby stránek/s
\Proces(inetinfo)\Pracovní sada
\Proces(inetinfo)\Nesdílené bajty
\Proces(inetinfo)\Uplynulý čas
\Proces(inetinfo)\Bajty stránkovaného fondu
\Proces(inetinfo)\Bajty nestránkovaného fondu
\Celkový počet připojení RAS\Odeslané bajty/s
\Celkový počet připojení RAS\Přijaté bajty/s
\Celkový počet připojení RAS\Celkový počet chyb/s
\Tisková fronta(*)\Celkový počet vytisknutých úloh
\Tisková fronta(*)\Počet bajtů vytisknutých za sekundu
\Tisková fronta(*)\Celkový počet vytisknutých stránek
\Tisková fronta(*)\Počet úloh
\Rozhraní sítě(*)\Bajty celkem/s
\Rozhraní sítě(*)\Pakety/s
\Rozhraní sítě(*)\Přijaté pakety/s
\Rozhraní sítě(*)\Odeslané pakety/s
\Rozhraní sítě(*)\Aktuální šířka pásma
\Rozhraní sítě(*)\Přijaté bajty/s
\Rozhraní sítě(*)\Odeslané bajty/s
\Rozhraní sítě(*)\Délka fronty výstupu
\IPv4\Datagramy/s
\IPv4\Přijaté datagramy/s
\IPv4\Odeslané datagramy/s
```

```
\TCPv4\Segmenty/s
\TCPv4\Navázaná připojení
\TCPv4\Aktivní připojení
\TCPv4\Chyby připojení
\TCPv4\Resetovaná připojení
\TCPv4\Přijaté segmenty/s
\TCPv4\Odeslané segmenty/s
\TCPv4\Segmenty odeslané znovu/s
```

Ukázková hlášení pro správu

Následující oddíl ilustruje hlášení pro správu, která v rámci prezentování souhrnných výkonnostních statistik zřejmě shledáte nejužitečnějšími. Uvedené příklady používají k zachycení výkonnostních dat hlavně zobrazení grafu konzoly System Monitor (Sledování systému). Samozřejmě máte možnost sestavit propracovanější grafy a tabulky, než jak je nabízí Sledování výkonu, použijete-li nástroje jako Microsoft Excel a podobné.

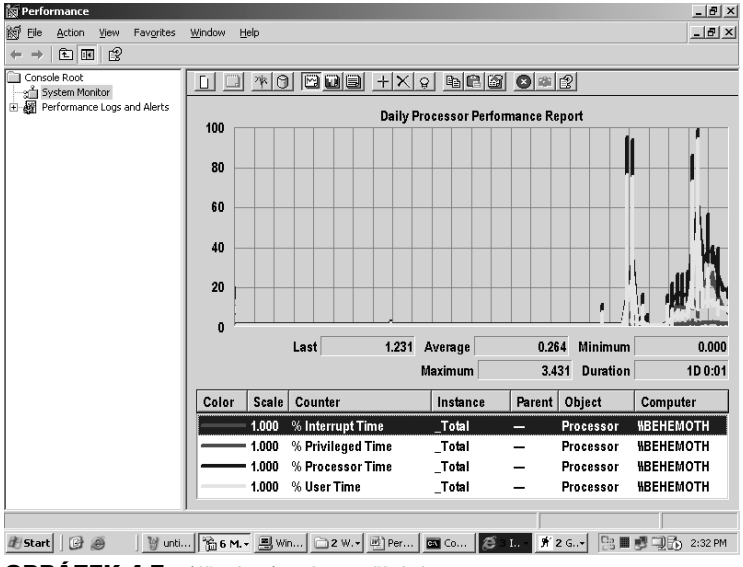
Zde představená ukázková hlášení pro správu zachycují klíčové výkonností čítače, které byste měli uvádět. Zároveň nejsou grafy a tabulky přeplněné údaji, takže je lze snadno číst a dešifrovat. Protokoly čítačů použité k vygenerování grafů byly shromažďovány na relativně nevýznamném počítači, který v intervalu hlášení vykonával jen minimum činností. Smyslem je zaměřit vaši pozornost na *prezentaci* dat a nikoli na údaje samotné. Pokud chcete vidět příklady zajímavějších tabulek a grafů zachycující počítače vykazující problémy s výkonem, podívejte se do kapitoly 5, „Řešení potíží s výkonem“, kde najdete mnoho takových ukázek.

Informace o tom, jak používat automatizační rozhraní Sledování systému k automatickému generování podobných hlášení pro správu najdete v oddílu nazvaném „Automatizační rozhraní Sledování systému“ v kapitole 6, „Pokročilá výkonnostní témata“.

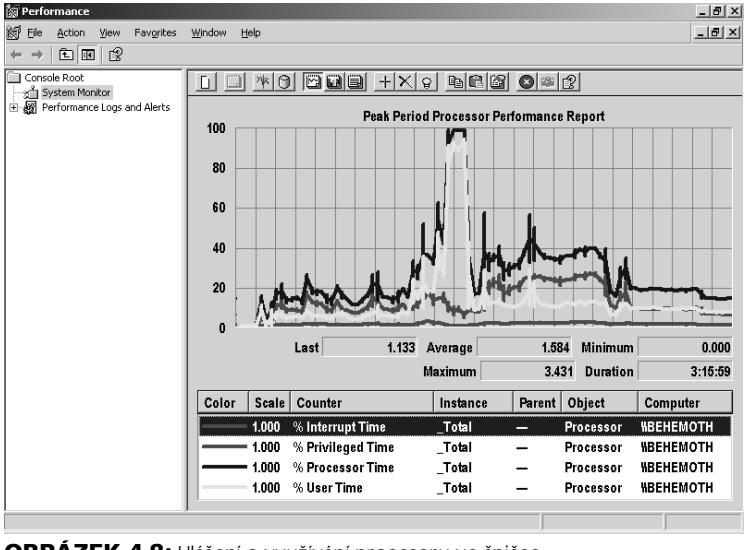
Využití procesoru Obrázek 4.7 ilustruje základní šablonu zobrazení grafu využitelnou v mnoha hlášeních pro správu. Zpráva zachycuje celkové využití procesoru, které zároveň rozděluje na jeho složkové části. Bylo vybráno velké, snadno čitelné písmo, doplněny mřížky os x a y a byl rovněž použit popisný titulek. Po vytvoření šablon hlášení, výběru požadovaných čítačů a přidání příslušných prezentačních prvků můžete nastavení uložit jako soubor `.msc` konzoly správy, jenž je opakovaně použitelný.

Podobný graf, který se zaměřuje na dvouhodinovou dobu špičkového zpracování, je uveden na obrázku 4.8. Opakované používání šablon grafů pro podobná hlášení správy zjednodušuje váš úkol vysvětlit význam různých grafů a tabulek.

Hlášení pro správu denního využívání procesoru uvedené na obrázku 4.7 využívá jako vstup sumarizovaný soubor denního protokolu čítačů vytvořený nástrojem Relog. Hlášení o špičkové hodině na obrázku 4.8 pracuje s plným, nesumarizovaným denním protokolem čítačů, přičemž časové okno je upraveno tak, aby zachycovalo přibližně dvě hodiny dat špičkového zatížení.



OBRÁZEK 4.7: Hlášení o denním využívání procesoru

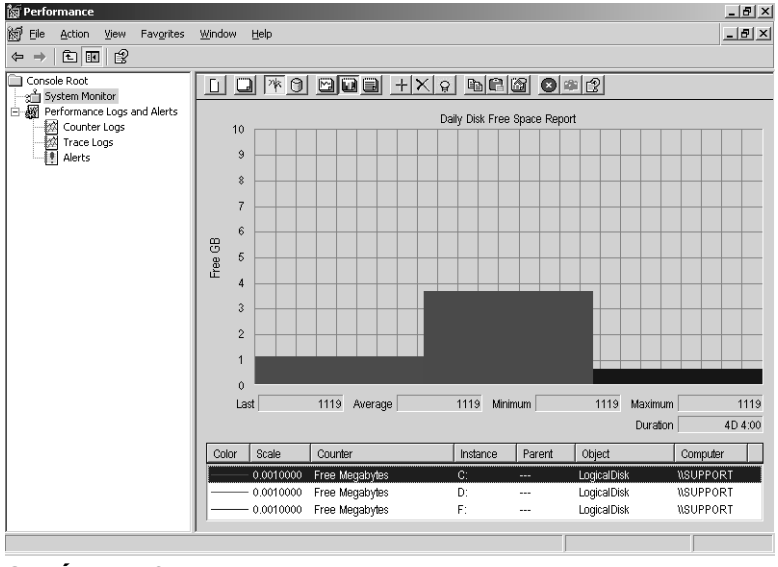


OBRÁZEK 4.8: Hlášení o využívání procesoru ve špičce

Dostupný diskový prostor Zobrazení histogramu podle obrázku 4.9 je vhodné pro hlášení dat z čítačů, které mají tendenci k velmi pomalým změnám, jako například Logical Disk(*)\Free Megabytes [Logický disk(*)\Volné megabajty]. Pomocí zobrazení histogramu můžete velmi rychle ukázat množství volného prostoru na velkém počtu serverových disků.

Výkon disku Obrázek 4.10 zachycuje problémy s čitelností, k nimž dochází, když musíte vynášet více čítačů na jednu osu y. Vše lze samozřejmě napravit rozumným využíváním škálovacích hodnot. Budete-li ale hlásit metriky využívající různé hod-

noty měřítek tak, aby je bylo možné vynášet na jedinou osu y, můžete tím čtenáře hlášení snadno zmást. Obrázek 4.10 zachycuje nastíněný problém, když zároveň uvádí měření výkonu disku ve formě doby nečinnosti, poměrů činnosti zařízení a reakční doby disku. Tyto míry společně přesně charakterizují výkonnost fyzického disku. Každý jednotlivý čítač ale obvykle vyžaduje aplikování jiného škálovacího měřítka, což často vede ke zmatkům.



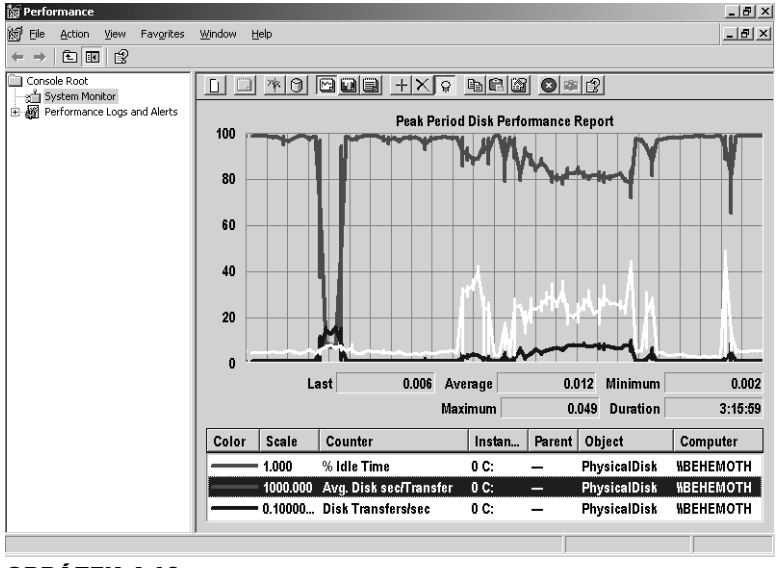
OBRÁZEK 4.9: Hlášení o volném diskovém prostoru

% Idle Time (Čítač % času nečinnosti) ideálně spadá do výchozího rozsahu osy y od nuly do sta; to už ale neplatí pro zbývající metriky. Transfers/sec (Přenosy/s) fyzického disku často přesahují rychlost 100 za sekundu, takže tento čítač nelze vždy správně vykreslit v grafu se stejným měřítkem osy y, jaké využívá % času nečinnosti. Navíc reakční dobu disku, jež se měří v milisekundách, je zapotřebí znásobit škálovací hodnotou 1000, aby mohla být rozumně zachycena na téže ose jako % času nečinnosti. Umístění několika čítačů využívajících různé faktory měřítka je doslova koledováním si o nesprávnou interpretaci.

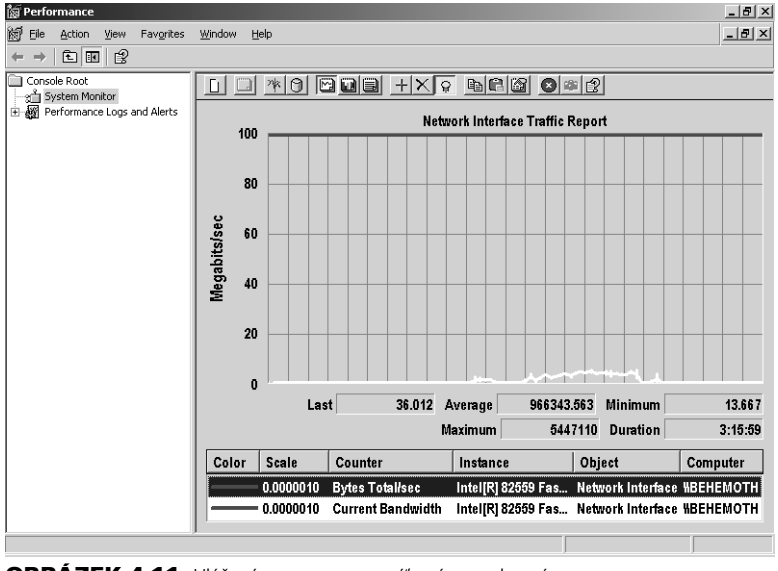
Tento přístup je ale někdy nevyhnutelný. Zmatek lze občas výrazně omezit zadáním vhodného popisku osy y. Alternativa, tedy uvedení tří samostatných grafů, každého pro jedno měření, není atraktivní, protože obsah grafů s hodnotami jednotlivých čítačů je výrazně rozdrobený.

Síťový provoz Síťový provoz lze hlásit z hlediska jednotlivých síťových rozhraní nebo jako sumarizovaná data všech existujících rozhraní sítě. Jelikož čítače objektu Network Interface (Rozhraní sítě) nehlásí využití sítě přímo, umožní vizualizaci relativního využívání daného rozhraní spíše taková šablona hlášení, jakou ilustruje obrázek 4.11. Ten zachycuje jak celkový počet bajtů přenesených přes síťové rozhraní, tak i aktuální nominální šířku pásma dané karty. Změňte výchozí maximum osy y tak, aby čítač Current Bandwidth (Aktuální šířka pásma) představoval úroveň 100% využití. Čítač Bytes Total/sec (Bajty celkem/s) dané instance síťového rozhraní si

pak lze představit jako zlomek šířky pásma této karty. Nezapomeňte osu y příslušným způsobem označit, jak je tomu na obrázku 4.11.



OBRÁZEK 4.10: Měření výkonu disku s využitím tří metrik



OBRÁZEK 4.11: Hlášení o provozu na síťovém rozhraní

Historická data pro plánování kapacity

Plánování kapacity představuje postupy a zásady zavedené za účelem prevence vzniků výkonnostních potíží při zvyšování a změnách pracovního zatížení. Když si budete ukládat historické záznamy využívání prostředků počítačů různými důleži-

tými produkčními zatíženími, extrapolací z historických trendů dokážete předpovídat budoucí požadavky na prostředky.

Procesy plánování kapacity využívají rovněž předpovědi růstu, které vám poskytnou lidé zabývající se klíčovými produkčními pracovními zatíženími ve vaší organizaci. Tyto předpovědi růstu jsou často projekcemi vycházejícími z přidávání dalších klientů, uživatelů takových produkčních systémů. Následně musíte transformovat tyto předpovědi růstu na sadu požadavků na počítačové prostředky vycházející z profilu využívání prostředků aktuálními klienty. Plánování kapacity obvykle využívá oba druhy předpovídaných údajů při odhadování budoucích požadavků pracovního zatížení a druhů počítačových prostředků nezbytných ke zvládnutí takových pracovních zatížení.

Tento oddíl se zaměřuje na procedury využitelné ke spojení generovaných denních protokolů výkonnostních čítačů při vytváření a udržování databáze obsahující historické záznamy dat využívání počítačových prostředků. Tato data lze dále použít k podpoře plánování kapacity a dalších funkcí správy systému. Historický záznam výkonnostních dat počítačů se označuje za *výkonnostní databázi* (Performance Database neboli PDB). Tento oddíl také popisuje procedury použitelné k sestavení výkonnostní databáze SQL Serveru s využitím nástrojů příkazového řádku popisovaných v kapitole 2, „Nástroje sledování výkonu“. Další oddíl této kapitoly nazvaný „Práce s repozitářem SQL Serveru“ probírá použití nástrojů jako Microsoft Excel k analýze dat v PDB SQL Serveru a nabízí příklad využití Microsoft Excelu k předvídání budoucích kapacitních požadavků.

Proč SQL Server?

Microsoft SQL Server je ideální výkonnostní databází. Nemáte-li s touto aplikací zatím zkušenosti, možná nebudete mít chuť pouštět se do implementování výkonnostní databáze SQL Serveru. Můžete být ale úplně klidní – při použití nástrojů příkazového řádku, jakými jsou Logman a Relog, můžete docela jednoduše využívat SQL Server k tomuto účelu. Microsoft SQL Server je pokročilým systémem správy relačních databází (Relational Database Management System – RDBMS) vhodným pro tento úkol, zejména pokud se jedná o možnost zpracovávání velkého množství dat pro plánování kapacity, která nakonec shromáždíte. Při využití SQL Serveru se nemusíte starat o správu velkého počtu jednotlivých souborů protokolů čítačů. Místo toho můžete konsolidovat všechna svá historická data na jednom místě v jedné nebo více tabulkách SQL Serveru.

Další výhodou využití aplikace SQL Server jako repozitáře dat pro plánování kapacity je to, že při hlášení nejste omezeni na práci s nástrojem Sledování systému. Jakmile budete mít data protokolů čítačů zavedena do SQL Serveru, můžete využívat různé nástroje přístupu k datům, hlášení a analýzy. Jedním z neoblíbenějších a nejvýkonnějších nástrojů analytických hlášení je Microsoft Excel. Oddíl této kapitoly nazvaný „Práce s repozitářem SQL Serveru“ popisuje, jak používat Microsoft Excel k analýze, hlášení a předvídání pomocí výkonnostních dat uložených do PDB SQL Serveru.

Vytváření historických souhrnných protokolů

K plánování kapacity jsou zapotřebí historická data shromažďovaná a ukládaná po delší dobu, aby se projevily historické trendy. Tento oddíl ukazuje, jak lze použít

nástroj Relog k sumarizaci dat a akumulaci těchto údajů protokolů čítačů v zájmu podpory plánování kapacity.

Soubory sumarizovaných denních protokolů čítačů, jež se používají v hlášení pro správu, obsahují příliš mnoho podrobností na to, aby je bylo možné ukládat celé týdny, měsíce a roky. Zvažte opakované spuštění nástroje Relog na souborech sumarizovaných denních protokolů čítačů vytvářených vaší procedurou denního sledování výkonu, aby došlo k jejich další editaci a sumarizaci. Příklad nabízí výpis 4.10.

VÝPIS 4.10: Editace a sumarizace denních protokolů čítačů

```
relog <computer-name>.basicDailyLog.blg -o <computer-
name>.dailyHistoryLog.blg
 -cf summary-counters-setting-file.txt -f BIN -t 4
```

Respektive v českém prostředí:

```
relog <nazev-pocitace>.zakladniDenniProtokol.blg
  -o <nazev-pocitace>.denniHistorickyProtokol.blg
  -cf soubor-nastaveni-souhrnnych-citacu.txt -f BIN -t 4
```

Protože soubory protokolů čítačů definované v našem kódu jako vstup již byly sumarizovány na 15minutové intervaly, zajišťuje parametr `-t 4` další sumarizaci na jednohodinové úrovni.

Pro dlouhodobé kapacitní plánování má význam jen malý počet čítačů. Soubor nastavení protokolu čítačů, na který se odkazuje uvedený příkaz Relog, vypouští mnoho hodnot čítačů, které mají z dlouhodobého hlediska jen minimální nebo nulový přínos. Příklad souboru nastavení protokolu čítačů, vhodného pro plánování kapacity na souborovém a tiskovém serveru, je uveden ve výpisu 4.11.

VÝPIS 4.11: Soubor nastavení protokolu čítačů databáze pro plánování kapacity

a) produkt v angličtině

```
\LogicalDisk(*)\% Free Space
\LogicalDisk(*)\Free Megabytes
\PhysicalDisk(*)\Avg. Disk sec/Transfer
\PhysicalDisk(*)\Avg. Disk sec/Read
\PhysicalDisk(*)\Avg. Disk sec/Write
\PhysicalDisk(*)\Disk Transfers/sec
\PhysicalDisk(*)\Disk Reads/sec
\PhysicalDisk(*)\Disk Writes/sec
\PhysicalDisk(*)\Disk Bytes/sec
\PhysicalDisk(*)\Disk Read Bytes/sec
\PhysicalDisk(*)\Disk Write Bytes/sec
\PhysicalDisk(*)\Avg. Disk Bytes/Transfer
\PhysicalDisk(*)\Avg. Disk Bytes/Read
\PhysicalDisk(*)\Avg. Disk Bytes/Write
\PhysicalDisk(*)\% Idle Time
\Processor(*)\% Processor Time
\Processor(*)\% User Time
\Processor(*)\% Privileged Time
\Processor(*)\Interrupts/sec
\Processor(*)\% DPC Time
```

```
\Processor(*)\% Interrupt Time
\Memory\Page Faults/sec
\Memory\Available Bytes
\Memory\Committed Bytes
\Memory\Commit Limit
\Memory\Transition Faults/sec
\Memory\Cache Faults/sec
\Memory\Demand Zero Faults/sec
\Memory\Pages/sec
\Memory\Pages Input/sec
\Memory\Page Reads/sec
\Memory\Pages Output/sec
\Memory\Pool Paged Bytes
\Memory\Pool Nonpaged Bytes
\Memory\Page Writes/sec
\Memory\Cache Bytes
\Memory\Pool Paged Resident Bytes
\Memory\System Cache Resident Bytes
\Memory\% Committed Bytes In Use
\Memory\Available KBytes
\Memory\Available MBytes
\Memory\Transition Pages RePurposed/sec
\Process(svchost,*)\% Processor Time
\Process(svchost,*)\% User Time
\Process(svchost,*)\% Privileged Time
\Process(svchost,*)\Virtual Bytes
\Process(svchost,*)\Page Faults/sec
\Process(svchost,*)\Working Set
\Print Queue(*)\Bytes Printed/sec
\Print Queue(*)\Total Pages Printed
\Print Queue(*)\Jobs
\Network Interface(*)\Bytes Total/sec
\Network Interface(*)\Bytes Received/sec
\Network Interface(*)\Bytes Sent/sec
\IPv4\Datagrams/sec
\IPv4\Datagrams Received/sec
\IPv4\Datagrams Sent/sec
\TCPv4\Segments/sec
\TCPv4\Segments Received/sec
\TCPv4\Segments Sent/sec
```

b) produkt s českou lokalizací

```
\Logický disk(*)\% volného místa
\Logický disk(*)\Volné megabajty
\Fyzický disk(*)\Střední doba disku/přenos
\Fyzický disk(*)\Střední doba disku/čtení
\Fyzický disk(*)\Střední doba disku/zápis
\Fyzický disk(*)\Přenosy disku/s
\Fyzický disk(*)\Čtení z disku/s
\Fyzický disk(*)\Zápisy na disk/s
\Fyzický disk(*)\Bajtů disku/s
\Fyzický disk(*)\Bajty čtení z disku/s
```

```
\Fyzický disk(*)\Bajty zapisování na disk/s
\Fyzický disk(*)\Střední počet bajtů disku/přenos
\Fyzický disk(*)\Střední počet bajtů disku/čtení
\Fyzický disk(*)\Střední počet bajtů disku/zápis
\Fyzický disk(*)\% času nečinnosti
\Procesor(*)\% času procesoru
\Procesor(*)\% uživatelského času
\Procesor(*)\% privilegovaného času
\Procesor(*)\Přerušení/s
\Procesor(*)\% času DPC
\Procesor(*)\% času přerušení
\Paměť\Chyby stránek/s
\Paměť\Bajty k dispozici
\Paměť\Svěřené bajty
\Paměť\Mez svěření
\Paměť\Chyby převodu stavu/s
\Paměť\Chyby mezipaměti/s
\Paměť\Chyby nulových požadavků/s
\Paměť\Stránky/s
\Paměť\Vstup stránek/s
\Paměť\Čtení stránek/s
\Paměť\Výstup stránek/s
\Paměť\Bajty stránkovaného fondu
\Paměť\Bajty nestránkovaného fondu
\Paměť\Zápisy stránek/s
\Paměť\Bajty mezipaměti
\Paměť\Rezidentní bajty stránkovaného fondu
\Paměť\Rezidentní bajty systémové mezipaměti
\Paměť\% využití svěřených bajtů
\Paměť\Počet kB k dispozici
\Paměť\Počet MB k dispozici
\Paměť\Změn účelu stran přenosu/s
\Proces(svchost,*)\% času procesoru
\Proces(svchost,*)\% uživatelského času
\Proces(svchost,*)\% privilegovaného času
\Proces(svchost,*)\Virtuální bajty
\Proces(svchost,*)\Chyby stránek/s
\Proces(svchost,*)\Pracovní sada
\Tisková fronta(*)\Počet bajtů vytisknutých za sekundu
\Tisková fronta(*)\Celkový počet vytisknutých stránek
\Tisková fronta(*)\Počet úloh
\Rozhraní sítě(*)\Bajty celkem/s
\Rozhraní sítě(*)\Přijaté bajty/s
\Rozhraní sítě(*)\Odeslané bajty/s
\IPv4\Datagramy/s
\IPv4\Přijaté datagramy/s
\IPv4\Odeslané datagramy/s
\TCPv4\Segmenty/s
\TCPv4\Přijaté segmenty/s
\TCPv4\Odeslané segmenty/s
```

Jak bylo právě zmíněno, jelikož má pro plánování kapacity význam jen malý počet čítačů, soubor nastavení protokolu čítačů může být ještě stručnější než náš příklad ve výpisu 4.11. Zde se zachovávají jen vysokoúrovňové statistiky využití procesoru, paměti, disku a sítě společně s určitými statistikami procesů aplikací serveru a statistik souvisejících s pracovním zatížením a propustností tiskárny.

Akumulování historických dat Procedura příkazu Relog (viz výpis 4.10) konstruuje binární protokol čítačů z denního souboru a sumarizuje jej na jednohodinové intervaly. Hodinové intervaly jsou vhodné pro dlouhodobé plánování kapacity, předvídání zatížení a zachycování trendů. V případě plánování kapacity budete muset sestavit a spravovat soubor sumarizovaných historických protokolů čítačů, který bude obsahovat informace v rozsahu týdnů, měsíců a dokonce i roků. Toho dosáhnete použitím volby připojení dat v nástroji Relog. Kupříkladu dále uvedený příkaz využívá Relog parametr `-a` k doplnění denního sumarizovaného protokolu čítačů k souboru protokolu čítačů, jenž obsahuje nashromážděné historické údaje:

```
relog <computer-name>.dailyHistoryLog.blg -o
<computer-name>.historyPerformanceLog.blg -f BIN -a
```

Respektive v českém prostředí:

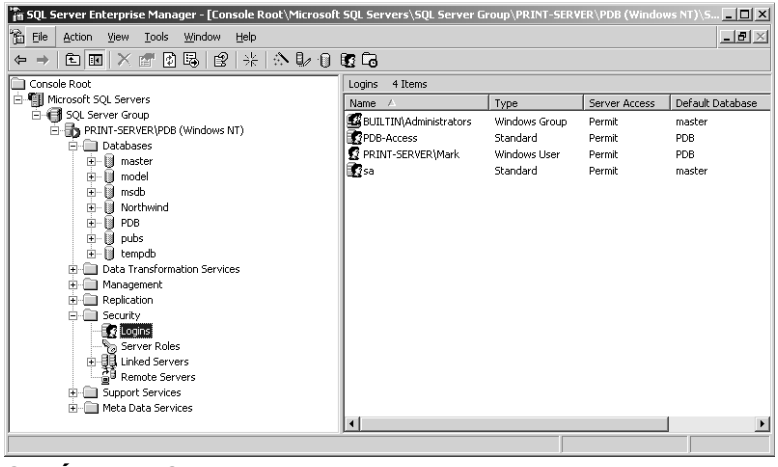
```
relog <nazev-pocitace>.denniHistorickyProtokol.blg
  -o <nazev-pocitace>.historickyProtokolVykonu.blg -f BIN -a
```

Volba `-a` předpokládá existenci výstupního souboru protokolu čítačů.

Vytvoření výkonnostní databáze SQL Serveru

Než budete moci použít nástroje příkazového řádku jako Logman a Relog k naplnění výkonnostní databáze SQL Serveru, musíte si nainstalovat instanci tohoto programu a definovat a nakonfigurovat databázi SQL Serveru, kterou budete používat jako PDB. Můžete si nainstalovat samostatnou instanci SQL Serveru používanou výhradně jako PDB, nebo sdílet SQL Server s ostatními aplikacemi. Jakmile vyberete instanci SQL Serveru, kterou máte v úmyslu používat, definujte výkonnostní databázi následujícími kroky:

1. **Nadefinujte databázi, do níž chcete ukládat výkonnostní data.** Pomocí konzoly SQL Enterprise Manager definujte novou databázi a alokujte diskový prostor pro ni a jí přidružený protokol obnovy databáze. V našem příkladu jsme využívanou databázi nazvali PDB. Jakmile je tato databáze vytvořená, můžete přistupovat k systémovým tabulkám, jež se sestavují automaticky.
2. **Definujte pravidla zabezpečení této databáze.** Zabezpečení je zásadní prvek správy databází SQL Serveru. Dokud specificky nezadáte přístupová oprávnění, žádní externí uživatelé se nemohou k databázi připojit a získat přístup k informacím uloženým v databázi PDB. Musíte nadefinovat přinejmenším jedno nové přihlášení k databázi a danému uživateli umožnit přístup k PDB, jak to zachycuje obrázek 4.12. V našem případě je nové přihlášení k databázi označeno PDB-Access a má standardně přístup k databázi PDB.



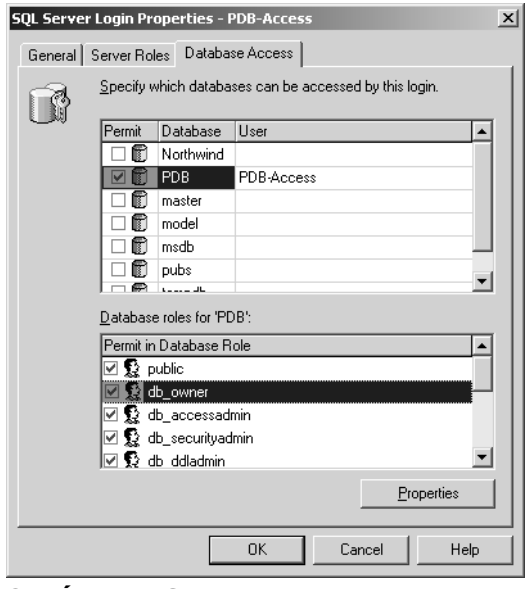
OBRÁZEK 4.12: Definování nového přihlášení k databázi

Vaše používaná instance SQL Serveru musí být nastavena tak, aby využívala pouze ověřování Microsoft Windows NT. Navíc je nutné definovat pojmenovaného uživatele s odpovídajícími údaji, kterým je v našem případě uživatel nazvaný PDB-Access. Na úrovni databáze PDB musíte rovněž definovat oprávnění platná pro právě zadané přihlášení. Obrázek 4.13 ukazuje vlastnosti uživatele databáze PDB pro přihlášení PDB-Access. Účet má v našem případě oprávnění používat všechny dostupné bezpečností role databáze. Tomuto přihlášení musíte přiřadit přinejmenším oprávnění `db_owner` a `db_datawriter`. Role `db_owner` mu umožní definovat, přidávat a měnit databázové tabulky a `db_datawriter` umožňuje uživateli aktualizovat data a přidávat je do definovaných tabulek.

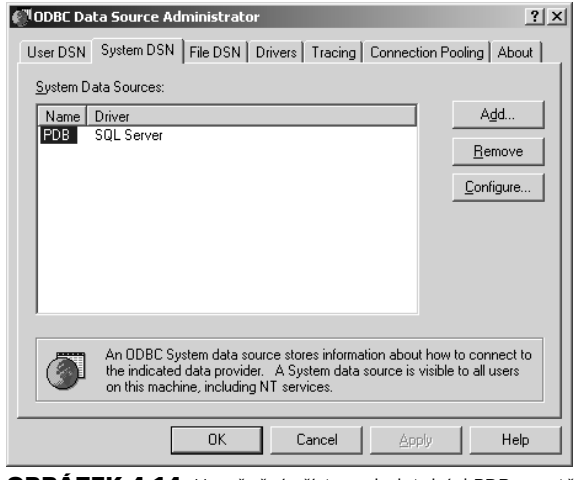
3. **Definujte připojení ODBC používané pro přístup k dané databázi PDB.** Aby mohly k databázi přistupovat programy jako utilita Relog, služba Výstrahy a protokolování výkonu, konzola Sledování systému kvůli hlášení a Microsoft Excel při extrakci dat a jejich analýze, musíte definovat systémové připojení DSN. Pomocí správce ODBC přidejte nové systémové připojení DSN, které bude používat pro umožnění přístupu k právě definované databázi PDB ovladač SQL Serveru – podívejte se na obrázek 4.14.

   Tím se spustí průvodce připojením ODBC, jenž vám umožní toto připojení nastavit, jak jej zachycuje obrázek 4.15.

   Musíte zadat název připojení, jak je tu uvedeno, a připojení nasměrovat na příslušnou instanci a databázi SQL Serveru. V našem příkladu je názvem připojení PDB. Dále je zapotřebí zadat údaje o zabezpečení tohoto připojení, jak zachycuje obrázek 4.16.

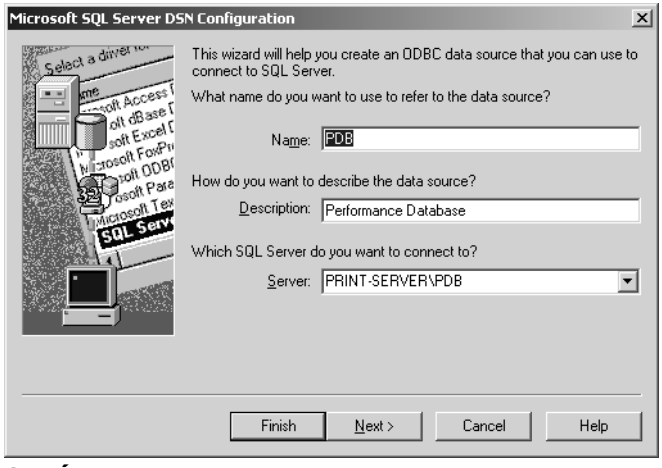


OBRÁZEK 4.13: Vlastnosti databázového uživatele s přihlášením PDB-Access

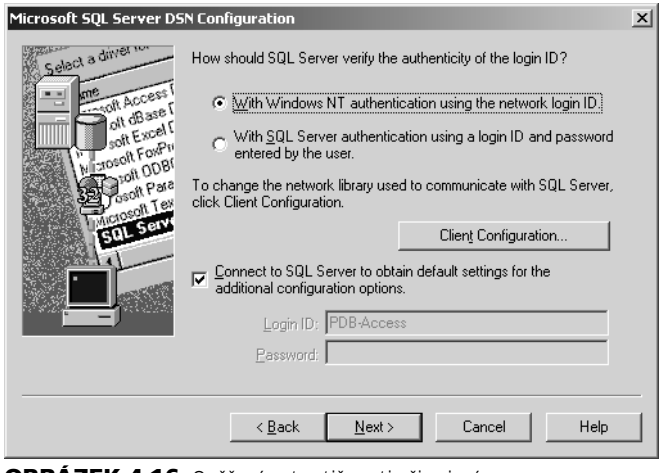


OBRÁZEK 4.14: Umožnění přístupu k databázi PDB prostřednictvím ovladače SQL Serveru

Zaškrtávací políčko **Connect To SQL Server To Obtain Default Settings For The Additional Configuration Options**, které je na obrázku 4.16 aktivované, vám dovoluje plně otestovat oprávnění daného připojení, jakmile je jeho definice dokončena. Pokračujte ve vyplňování formulářů nabízených průvodcem připojení ODBC a testování připojení uzavřete stiskem tlačítka Finish.



OBRÁZEK 4.15: Práce s průvodcem připojení ODBC



OBRÁZEK 4.16: Ověření autentičnosti připojení

Zaplnění repozitáře

Po nadefinování připojení ODBC můžete začít s nahráváním dat protokolů čítačů do vytvořené databáze PDB. Protože máte v plánu v PDB nashromáždit velké množství historických informací o výkonu, dokonce i sumarizovaný denní protokol čítačů, který jste vytvořili kvůli sestavování denních hlášení pro správu, obsahuje mnohem více dat, než je pro účely plánování kapacity zapotřebí. Podobně jako v případě zpracování zadaného při údržbě historického protokolu čítačů v binárním formátu, jak bylo probíráno v oddílu „Vytváření historických souhrnných protokolů“, bude vhodné tyto údaje dále sumarizovat a vypustit všechny čítače, jejichž zachovávání po delší dobu nemá význam.

Kupříkladu dále uvedený příkaz Relog přebírá jeden nebo více sumarizovaných denních souborů protokolů čítačů, které vytvářejí soubor historie, a vkládá výstup do databáze PDB:

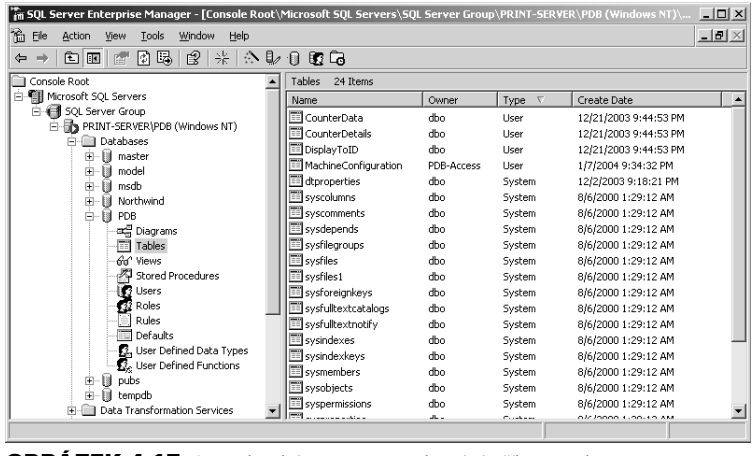
```
relog <computer-name>.historyPerformanceLog.blg -o "SQL:PDB! ByHour" -f SQL
  -cf c:\perflogs\summary-counters-setting-file.txt
```

Respektive v českém prostředí:

```
relog <nazev-pocitace>.historickyProtokolVykonu.blg -o "SQL:PDB!Hodinove"
  -f SQL -cf c:\perflogs\soubor-nastaveni-souhrnnych-citacu.txt
```

Specifikace výstupního souboru, `-o "SQL:PDB!Hodinove"`, identifikuje připojení ODBC nazvané PDB a definuje pododdíl databáze PDB nazvaný Hodinove. V jedné databázi PDB můžete definovat několik databází pro plánování kapacity, když je budete identifikovat odlišnými názvy.

Jakmile se vykoná uvedený příkaz, vytvoří tabulky databáze PDB používané k ukládání dat protokolu čítačů. Prostřednictvím konzoly SQL Enterprise Manager si můžete ověřit, že tabulky protokolů čítačů byly správně vytvořeny. Obrázek 4.17 zachycuje stav databáze PDB po vykonání příkazu Relog, který se odkazuje na připojení ODBC v databázi PDB aplikace SQL Server.



OBRÁZEK 4.17: Stav databáze PDB po vykonání příkazu Relog

Data protokolu čítačů jsou uložena ve třech tabulkách SQL Serveru – CounterData, CounterDetails a DisplayToID. Formát těchto tabulek SQL Serveru odráží datový model používaný nástroji a službami sledování výkonu k udržování repozitáře dat protokolu čítačů SQL Serveru. Tento datový model je detailně probírán v oddílu nazvaném „Práce s repozitářem SQL Serveru“.

Automatizované zpracování protokolů čítačů

V předchozích oddílech této kapitoly byly načrtnuty procedury vykonávající následující funkce:

- Shromažďování denních protokolů výkonnostních čítačů,
- sběr diagnostických protokolů čítačů v reakci na výstrahy,
- sumarizování denních protokolů výkonnostních čítačů při sestavování hlášení pro správu,
- zaplnění repozitáře v SQL Serveru, který lze používat při plánování kapacity.

Jedná se o funkce, jež je zapotřebí vykonávat automaticky na všech počítačích se systémem Windows Server 2003, které vyžadují trvalé monitorování výkonu. V tomto oddílu najdete ukázkový skript zajišťující automatizaci uvedených denních procedur. Tento skript je napsán v jazyce VBScript a můžete jej snadno přizpůsobit svému prostředí. Bude se odkazovat na soubory nastavení čítačů ukázané v předchozích oddílech, jež vytvářejí sumarizované soubory protokolů čítačů a využívají databázi PDB v SQL Serveru definovanou v předchozím oddílu.

Úplný skript následného zpracování je uveden ve výpisu 4.12. Je rozsáhle okomentovaný a naplněný zprávami `Wscript.Echo`; všechny tyto informace vám umožní skript jednoduše upravit tak, aby lépe vyhověl konkrétním požadavkům vašeho prostředí. Chcete-li aktivovat zmíněné diagnostické zprávy `Wscript.Echo`, odstraňte značku komentáře Microsoft Visual Basicu, což je znak apostrofu (').

Výpis 4.12 Ukázkový skript následného zpracování

```
'VBscript for post-processing daily performance Counter Logs
'
'Initialization
CreatedTodayDate = Now
Const OverwriteExisting = True
Const LogType = "blg"
Const OldLogType = "Windows Backup File"
Const PMFileType = "Performance Monitor File"
Const relogSettingsFileName = "c:\perflogs\relog-counters-setting-file.txt"
Const relogParms = "-f BIN -t 15"
Const relogSummaryParms = "-f BIN -t 4"
Const relogHistoryParms = "-f BIN -a"
Const HistorySettingsFileName = "c:\perflogs\summary-counters-setting-
file.txt"
Const PerflogFolderToday = "C:\Perflogs\Today"
Const PerflogFolderYesterday = "C:\Perflogs\Yesterday"
Const PerflogFolderAlerts = "C:\Perflogs\Alert Logs"
Const PerflogFolderHistory = "C:\Perflogs\History"

Set WshShell = CreateObject("Wscript.Shell")
Set objEnv = WshShell.Environment("Process")
LogonServer = objENV("COMPUTERNAME")
DailyFileName = PerflogFolderYesterday & "\" & LogonServer & "." & _
  "basic_daily_logDailyLog" & "." & LogType
SummaryFileName = PerflogFolderYesterday & "\" & LogonServer & "." & _
  "basic_history_logHistoryLog" & "." & LogType
HistoryFileName = PerflogFolderHistory & "\" & LogonServer & "." & _
  "history_performance_logPerformanceLog" & "." & LogType

'WScript.Echo DailyFileName
'WScript.Echo SummaryFileName
'WScript.Echo HistoryFileName
Const AlertDaysOld = 3 'Number of days to keep Alert-generated
Counter Log files
Set objFSO1 = CreateObject("Scripting.FileSystemObject")
```

```
If objFSO1.FolderExists(PerflogFolderYesterday) Then
  Set objYesterdayFolder = objFSO1.GetFolder(PerflogFolderYesterday)
Else
  Wscript.Echo "Yesterday folder does not exist. Will create " &
  PerflogFolderYesterday
  Set objYesterdayFolder = objFSO1.CreateFolder(PerflogFolderYesterday)
End If

Set objYesterdayFolder = objFSO1.GetFolder(PerflogFolderYesterday)
Set objTodayFolder = objFSO1.GetFolder(PerflogFolderToday)
Set objAlertsFolder = objFSO1.GetFolder(PerflogFolderAlerts)

'Wscript.Echo "Begin Script Body"
objYesterdayFolder.attributes = 0
Set fc1 = objYesterdayFolder.Files
'Wscript.Echo "Look for Yesterday's older backup files..."
For Each f1 in fc1
  'Wscript.Echo "Found " & f1.name & " in " & PerflogFolderYesterday
  'Wscript.Echo "File type is " & f1.type
  If f1.type = OldLogType Then
    'Wscript.Echo "Old files of type " & f1.type & " will be deleted."
    filename = PerflogFolderYesterday & "\" & f1.name
    'Wscript.Echo "Delete " & filename
    objFSO1.DeleteFile(filename)
  End If
Next
'Wscript.Echo "Look for Yesterday's .blg files..."
For Each f1 in fc1
  'Wscript.Echo "Found " & f1.name & " in " & PerflogFolderYesterday
  If f1.type = PMFileType Then
    NewName = PerflogFolderYesterday & "\" & f1.name & "." & "bkf"
    'Wscript.Echo f1.name & " will be renamed to " & NewName
    filename = PerflogFolderYesterday & "\" & f1.name
    'Wscript.Echo "Rename " & filename
    objFSO1.MoveFile filename, NewName
  End If
Next
objYesterdayFolder.attributes = 0
'Wscript.Echo "Look at Today's files..."
'Wscript.Echo "Today is " & CStr(CreatedTodayDate)
Set fc2 = objTodayFolder.Files
For Each f2 in fc2
  filename = PerflogFolderToday & "\" & f2.name
  FileCreatedDate = CDate (f2.DateCreated)
  'Wscript.Echo filename & " was created on " & CStr (FileCreatedDate)
  If DateDiff ("d", CreatedTodayDate, FileCreatedDate) = 0 Then
    'Wscript.Echo "Skipping the file currently in use: " & filename
  Else
    'Wscript.Echo filename & " is " & CStr(DateDiff("d", FileCreatedDate,
    CreatedTodayDate)) & " day(s) old."
    'Wscript.Echo "Copying " & filename & " to " & PerflogFolderYesterday
    objFSO1.CopyFile filename, PerflogFolderYesterday & "/"
```

```
    relogfiles = relogfiles & PerflogFolderYesterday & "\" & f2.name & " "
  End If
Next
'Wscript.Echo "Today's files to send to relog: " & relogfiles
command = "relog " & relogfiles & "" -o " & DailyFileName & " -cf " _
  & relogSettingsFileName & " " & relogParms
'Wscript.Echo "Relog command string: " & command
Set WshShell = Wscript.CreateObject("WScript.Shell")
Set execCommand = WshShell.Exec(command)
Wscript.Echo execCommand.StdOut.ReadAll
command = "relog " & DailyFileName & _
  " -o " & SummaryFileName & " -cf " & HistorySettingsFileName _
  & " " & relogSummaryParms
'Wscript.Echo "Relog command string: " & command
Set WshShell = Wscript.CreateObject("WScript.Shell")
Set execCommand = WshShell.Exec(command)
Wscript.Echo execCommand.StdOut.ReadAll
If (objFSO1.FileExists(HistoryFileName)) Then
  command = "relog " & HistoryFileName & " " & SummaryFileName & _
    " -o " & HistoryFileName & " " & relogHistoryParms
  'Wscript.Echo "Relog command string: " & command
  Set WshShell = Wscript.CreateObject("WScript.Shell")
  Set execCommand = WshShell.Exec(command)
  Wscript.Echo execCommand.StdOut.ReadAll
Else
  objFSO1.CopyFile SummaryFileName, HistoryFileName
End If
'Copy the summarized daily file to a Counter Log data consolidation server
' objFSO1.CopyFile DailyFileName, <somewhere>
 'Wscript.Echo "Deleting files after processing"
For Each f2 in fc2
  filename = PerflogFolderToday & "\" & f2.name
  FileCreatedDate = CDate (f2.DateCreated)
  If DateDiff ("d", CreatedTodayDate, FileCreatedDate) = 0 Then
    'Wscript.Echo "Skipping the file currently in use: " & filename
  Else
    'Wscript.Echo "Deleting " & filename & " from " & PerflogFolderToday
    objFSO1.DeleteFile(filename)
  End If
Next
Set fc3 = objAlertsFolder.Files
'Wscript.Echo "Look for older Alert-generated log files ..."
For Each f3 in fc3
  'Wscript.Echo "Found " & f3.name & " in " & PerflogFolderAlerts
  filename = PerflogFolderAlerts & "\" & f3.name
  FileCreatedDate = CDate (f3.DateCreated)
  'Wscript.Echo filename & " is " & CStr(DateDiff("d", FileCreatedDate,
  CreatedTodayDate)) & " day(s) old."
  If DateDiff ("d", FileCreatedDate, CreatedTodayDate) < AlertDaysOld Then
    'Wscript.Echo "Skipping recently created Alert Counter Log file: " &
    filename
  Else
```

```
    'Wscript.Echo "Deleting " & filename & " from " &
    PerflogFolderToday
    objFSO1.DeleteFile(filename)
  End If
Next
```

Respektive v českém prostředí:

```
'VBScript pro další zpracování denních protokolů výkonnostních čítačů
'
'Inicializace

DnesniDatumVytvoreni = Now

Const PrepsatExistujici = True
Const TypProtokolu = "blg"
Const StaryTypProtokolu = "Windows Backup File"
Const TypSouboruSV = "Performance Monitor File"

Const NazevSouboruPrenastaveniProtokolu = "c:\perflogs\soubor-prenastaveni-
citacu-protokolu.txt"
Const parametryPrenastaveni = "-f BIN -t 15"
Const parametryPrenastaveniSouhrnu = "-f BIN -t 4"
Const parametryPrenastaveniHistorie = "-f BIN -a"
Const SouborNastaveniSouhrnnychCitacu = "c:\perflogs\soubor-nastaveni-
souhrnnych-citacu.txt"

Const SlozkaPerflogDnes = "C:\Perflogs\Dnes"
Const SlozkaPerflogVcera = "C:\Perflogs\Vcera"
Const SlozkaPerflogVystrahy = "C:\Perflogs\Protokoly vystrah"
Const SlozkaPerflogHistorie = "C:\Perflogs\Historie"

Set WshShell = CreateObject("Wscript.Shell")
Set objEnv = WshShell.Environment("Process")
PrihlasovaciServer = objENV("COMPUTERNAME")
NazevDennihoSouboru = SlozkaPerflogVcera & "\" & PrihlasovaciServer & _
  "." & "zakladni_denni_protokol" & "." & TypProtokolu
NazevSouboruSouhrnu = SlozkaPerflogVcera & "\" & PrihlasovaciServer & _
  "." & "zakladni_historicky_protokol" & "." & TypProtokolu
NazevSouboruHistorie = SlozkaPerflogHistorie & "\" & PrihlasovaciServer & _
  "." & "protokol_historie_vykonnosti" & "." & TypProtokolu
'WScript.Echo NazevDennihoSouboru
'WScript.Echo NazevSouboruSouhrnu
'WScript.Echo NazevSouboruHistorie

'Počet dnů zachování souborů protokolů čítačů generovaných výstrahami
Const PocetDnuVystrahy = 3

Set objFSO1 = CreateObject("Scripting.FileSystemObject")

If objFSO1.FolderExists(SlozkaPerflogVcera) Then
  Set objSlozkaVcera = objFSO1.GetFolder(SlozkaPerflogVcera)
Else
```

```
  Wscript.Echo "Složka Vcera neexistuje. Dojde k vytvoření složky " & _
    SlozkaPerflogVcera
  Set objSlozkaVcera = objFSO1.CreateFolder(SlozkaPerflogVcera)
End If

Set objSlozkaVcera = objFSO1.GetFolder(SlozkaPerflogVcera)
Set objDnesFolder = objFSO1.GetFolder(SlozkaPerflogDnes)
Set objAlertsFolder = objFSO1.GetFolder(SlozkaPerflogVystrahy)

'Wscript.Echo "Začátek těla skriptu"
objSlozkaVcera.attributes = 0
Set fc1 = objSlozkaVcera.Files

'Wscript.Echo "Hledání starších záložních souborů ve složce Vcera..."
For Each f1 in fc1
  'Wscript.Echo "Nalezen soubor " & f1.name & " v " & SlozkaPerflogVcera
  'Wscript.Echo "Typem souboru je " & f1.type
  If f1.type = StaryTypProtokolu Then
    'Wscript.Echo "Staré soubory typu " & f1.type & " budou odstraněny."
    nazev_souboru = SlozkaPerflogVcera & "\" & f1.name
    'Wscript.Echo "Odstranění souboru " & nazev_souboru
    objFSO1.DeleteFile(nazev_souboru)
  End If
Next

'Wscript.Echo "Hledání souborů .blg ve složce Vcera..."
For Each f1 in fc1
  'Wscript.Echo "Nalezen soubor " & f1.name & " v " & SlozkaPerflogVcera
  If f1.type = TypSouboruSV Then
    NovyNazev = SlozkaPerflogVcera & "\" & f1.name & "." & "bkf"
    'Wscript.Echo f1.name & " bude přejmenován na " & NovyNazev
    nazev_souboru = SlozkaPerflogVcera & "\" & f1.name
    'Wscript.Echo "Přejmenování souboru " & nazev_souboru
    objFSO1.MoveFile nazev_souboru, NovyNazev
  End If
Next

objSlozkaVcera.attributes = 0

'Wscript.Echo "Hledání souborů ve složce Dnes..."
'Wscript.Echo "Dnes je " & CStr(DnesniDatumVytvoreni)
Set fc2 = objDnesFolder.Files
For Each f2 in fc2
  nazev_souboru = SlozkaPerflogDnes & "\" & f2.name
  DatumVytvoreniSouboru = CDate(f2.DateCreated)
  'Wscript.Echo nazev_souboru & " byl vytvořen " & CStr
(DatumVytvoreniSouboru)
  If DateDiff ("d", DnesniDatumVytvoreni, DatumVytvoreniSouboru) = 0 Then
    'Wscript.Echo "Přeskočení aktuálně používaného souboru: " & _
      nazev_souboru
  Else
```

```
    'Wscript.Echo nazev_souboru & " je " & CStr(DateDiff("d",
DatumVytvoreniSouboru, DnesniDatumVytvoreni) ) & " dnu stary."
    'Wscript.Echo "Kopírování " & nazev_souboru & " do " & _
      SlozkaPerflogVcera
    objFSO1.CopyFile nazev_souboru, SlozkaPerflogVcera & "/"
    prenastavene_soubory = prenastavene_soubory & SlozkaPerflogVcera & _
      "\" & f2.name & " "
  End If
Next

'Wscript.Echo "Soubory složky Dnes odesílané na přeprotokolování: " & _
  prenastavene_soubory

prikaz = "relog " & prenastavene_soubory & "" -o " & _
  NazevDennihoSouboru & " -cf " & NazevSouboruPrenastaveniProtokolu & _
  " " & parametryPrenastaveni
'Wscript.Echo "Řetězec příkazu Relog: " & prikaz

Set WshShell = Wscript.CreateObject("WScript.Shell")
Set vykonavanyPrikaz = WshShell.Exec(prikaz)
Wscript.Echo vykonavanyPrikaz.StdOut.ReadAll

prikaz = "relog " & NazevDennihoSouboru & " -o " & _
  NazevSouboruSouhrnu & " -cf " & SouborNastaveniSouhrnnychCitacu & " " _
  & parametryPrenastaveniSouhrnu
'Wscript.Echo "Řetězec příkazu Relog: " & prikaz

Set WshShell = Wscript.CreateObject("WScript.Shell")
Set vykonavanyPrikaz = WshShell.Exec(prikaz)
Wscript.Echo vykonavanyPrikaz.StdOut.ReadAll

If (objFSO1.FileExists(NazevSouboruHistorie)) Then
  prikaz = "relog " & NazevSouboruHistorie & " " & NazevSouboruSouhrnu & _
    " -o " & NazevSouboruHistorie & " " & parametryPrenastaveniHistorie
  'Wscript.Echo "Řetězec příkazu Relog: " & prikaz
  Set WshShell = Wscript.CreateObject("WScript.Shell")
  Set vykonavanyPrikaz = WshShell.Exec(prikaz)
  Wscript.Echo vykonavanyPrikaz.StdOut.ReadAll
Else
  objFSO1.CopyFile NazevSouboruSouhrnu, NazevSouboruHistorie
End If

'Zkopírovat souhrnný denní soubor na konsolidační server dat protokolů
čítačů
'  objFSO1.CopyFile NazevDennihoSouboru, <někde>

'Wscript.Echo "Odstranění souborů po zpracování"
For Each f2 in fc2
  nazev_souboru = SlozkaPerflogDnes & "\" & f2.name
  DatumVytvoreniSouboru = CDate(f2.DateCreated)
  If DateDiff ("d", DnesniDatumVytvoreni, DatumVytvoreniSouboru) = 0 Then
```

```
    'Wscript.Echo "Přeskočení aktuálně používaného souboru " & _
      nazev_souboru
  Else
    'Wscript.Echo "Odstranění " & nazev_souboru & " z " & SlozkaPerflogDnes
    objFSO1.DeleteFile(nazev_souboru)
  End If
Next

Set fc3 = objAlertsFolder.Files

'Wscript.Echo "Hledání starších souborů protokolů generovaných
výstrahami..."
For Each f3 in fc3
  'Wscript.Echo "Nalezen " & f3.name & " v " & SlozkaPerflogVystrahy
  nazev_souboru = SlozkaPerflogVystrahy & "\" & f3.name
  DatumVytvoreniSouboru = CDate(f3.DateCreated)
  'Wscript.Echo nazev_souboru & " je " & CStr(DateDiff("d",
DatumVytvoreniSouboru, DnesniDatumVytvoreni)) & " dnů starý."
  If DateDiff ("d", DatumVytvoreniSouboru, DnesniDatumVytvoreni) <
PocetDnuVystrahy Then
    'Wscript.Echo "Přeskok nedávno vytvořeného souboru protokolů čítačů
výstrah: " & nazev_souboru
  Else
    'Wscript.Echo "Odstranění " & nazev_souboru & " z " & SlozkaPerflogDnes
    objFSO1.DeleteFile(nazev_souboru)
  End If
Next
```


Uvedený ukázkový skript bude automaticky spouštět služba Performance Logs and Alerts (Výstrahy a protokolování výkonu), jakmile smlogsvc uzavře jeden denní protokol čítačů a otevře následující. K automatickému spuštění skriptu v okamžiku, kdy chcete vykonat kroky následného zpracování, můžete využít také Task Scheduler (Plánovač úloh).

Následující oddíly detailně probírají logiku využívanou ve skriptu dalšího zpracování pro případ, že si jej budete chtít přizpůsobit.

Inicializace skriptu

Oddíl inicializace našeho ukázkového skriptu nastavuje počáteční hodnoty řady používaných konstant. Změnou počátečních hodnot těchto konstant můžete snadno změnit chování skriptu tak, aby vyhovovalo vašemu prostředí, aniž by bylo zapotřebí měnit vlastní logiku skriptu. Kupříkladu následující řádky nastavují počáteční hodnoty čtyř řetězcových proměnných, jež se odkazují na složky obsahující protokoly čítačů spravované skriptem:

```
Const SlozkaPerflogDnes = "C:\Perflogs\Dnes"
Const SlozkaPerflogVcera = "C:\Perflogs\Vcera"
Const SlozkaPerflogVystrahy = "C:\Perflogs\Protokoly vystrah"
Const SlozkaPerflogHistorie = "C:\Perflogs\Historie"
```

Předpokládá se, že `SlozkaPerflogDnes` ukazuje na místo, kam služba Výstrahy a protokolování výkonu zapisuje denní protokoly čítačů. Také se předpokládá, že

`SlozkaPerflogVystrahy` směřuje na složku, kam se zapisují protokoly čítačů spuštěné hodinovým sledováním výstrah. Skript bude přesunovat včerejší protokoly čítačů ze složky `C:\Perflogs\Dnes` do `C:\Perflogs\Vcera`, přičemž složku `C:\Perflogs\Vcera` vytvoří, jestliže zatím neexistuje. Skript bude také odstraňovat protokoly čítačů ze složky C:\Perflogs\Protokoly vystrah, jakmile jsou starší než tři dny. Další konstanta, nazvaná `PocetDnuVystrahy`, obsahuje kritérium zastarávání platné pro složku Protokoly vystrah. Chcete-li zachovávat diagnostické protokoly čítačů vygenerované automaticky spuštěním podmínek výstrah po dobu 10 dnů, stačí změnit inicializační hodnotu proměnné `PocetDnuVystrahy` takto:

```
Const PocetDnuVystrahy = 10
```

Pro přístup k vestavěné proměnné prostředí `COMPUTERNAME` používá skript objekt `WshShell`. Zmíněná proměnná se používá k vytvoření názvu souboru sumarizovaného protokolu čítačů, který nástroj relog vytvoří z jakéhokoli nalezeného detailního protokolu čítačů. Když bude mít soubor protokolu čítačů ve svém názvu označení počítače, z něhož pochází, bude snáze identifikovatelný.

```
Set WshShell = CreateObject("Wscript.Shell")
Set objEnv = WshShell.Environment("Process")
PrihlasovaciServer = objENV("COMPUTERNAME")
NazevDennihoSouboru = SlozkaPerflogVcera & "\" & PrihlasovaciServer & _
  "." & "zakladni_denni_protokol" & "." & TypProtokolu
'WScript.Echo NazevDennihoSouboru
```

Po inicializaci těchto a dalších konstant skript inicializuje objekt `FileSystemObject`, který používá k vykonání různých operací správy souborů v určených složkách a v nich obsažených souborech protokolů čítačů:

```
Set objFSO1 = CreateObject("Scripting.FileSystemObject")
```

Objekt `FileSystemObject` se pak používá k identifikaci složek souborů, v nichž skript pracuje. Kupříkladu dále uvedený kód inicializuje proměnnou nazvanou `fc1`, což je kolekce obsahující soubory ve složce `C:\Perflog\Vcera`:

```
Set objSlozkaVcera = objFSO1.GetFolder(SlozkaPerflogVcera)
Set fc1 = objSlozkaVcera.Files
```

Odstranění včerejších souborů zálohy

Tělo skriptu začíná výčtem souborů ve složce C:\Perflogs\Vcera, kde jsou uloženy starší protokoly čítačů. V tomto okamžiku se odstraní všechny soubory ve složce C:\Perflogs\Vcera, které odpovídají `StaryTypProtokolu`. Následující oddíl skriptu přejmenuje všechny soubory `.blg`, které nalezne ve složce C:\Perflogs\Vcera, na názvy s příponou `StaryTypProtokolu`, která se k názvu souboru přidá v následující sekci kódu:

```
For Each f1 in fc1
  If f1.type = TypSouboruSV Then
    nazev_souboru = SlozkaPerflogVcera & "\" & f1.name
    NovyNazev = SlozkaPerflogVcera & "\" & f1.name & "." & "bkf"
    objFSO1.MoveFile nazev_souboru, NovyNazev
  End If
Next
```

Ukázkový skript inicializuje hodnotu `StaryTypProtokolu` tak, aby odpovídala souboru zálohy systému Windows (Windows Backup File), který používá příponu `.bkf`. Tento údaj můžete změnit na jakoukoli hodnotu používanou ve vašem prostředí. (Alternativně lze rozdělit řetězec názvu souboru na části a zjistit jeho příponu.)

Důsledkem tohoto zpracování souborů ve složce C:\Perflogs\Vcera je to, že ta bude nyní obsahovat protokoly čítačů ze včerejška a záložní verze protokolů čítačů použitých předevčírem. Tím vzniká určité bezpečnostní pásmo umožňující vám vrátit se přinejmenším celý jeden den zpět, když se v proceduře následného zpracování denních protokolů čítačů něco pokazí.

Správa včerejších denních protokolů čítačů

Skript pak prochází soubory protokolů čítačů ve složce C:\Perflogs\Dnes. Zde bude binární soubor protokolů čítačů, kam se zapisují aktuální naměřené údaje, a také všechny předchozí protokoly čítačů, které lze nyní zpracovat. Skript stanoví, zda jsou soubory protokolů čítačů ve složce `C:\Perflogs\Dnes` připraveny ke zpracování, porovnáním jejich data vytvoření s aktuálním datem. Soubory ve složce `C:\Perflogs\Dnes` z předchozího dne se pak s využitím funkce `DateDiff` zkopírují do složky `C:\Perflogs\Vcera`. Mezitím se názvy dále kopírovaných souborů kupí v řetězcové proměnné nazvané `prenastavene_soubory`, která se použije dále ve skriptu.

Vytvoření sumarizovaných protokolů čítačů

Jakmile jsou zpracovány všechny starší protokoly čítačů ve složce `C:\Perflogs\Dnes`, skript vykoná nástroj Relog prostřednictvím metody `Exec` objektu `WshShell`:

```
prikaz = "relog " & prenastavene_soubory & "-o " & NazevDennihoSouboru & _
  " -cf " & NazevSouboruPrenastaveniProtokolu & " " & parametryPrenastaveni
Set WshShell = Wscript.CreateObject("WScript.Shell")
Set vykonavanyPrikaz = WshShell.Exec(prikaz)
```

Tím se vytvoří binární protokol čítačů označený řetězcovou proměnnou `NazevDennihoSouboru` a sumarizovaný do patnáctiminutových intervalů.

Řádek následující za voláním utility Relog vám zajišťuje přístup ke všem výstupním zprávám nástroje Relog, jakmile odstraníte znak komentáře a volání `Wscript.Echo` aktivujete:

```
'Wscript.Echo vykonavanyPrikaz.StdOut.ReadAll
```

Jakmile nástroj Relog vytvoří tento sumarizovaný denní protokol čítačů, bude zřejmě zapotřebí přidat krok zpracování, v němž se denní protokol zkopíruje do protokolu čítačů na konsolidačním serveru někde ve vaší síti. Nahraďte následující řádky komentářů, které vyhrazují místo pro toto zpracování, kódem zajišťujícím přenos souboru na určený konsolidační server ve vašem prostředí:

```
'Zkopírovat souhrnný denní soubor na konsolidační server dat protokolů
čítačů
'  objFSO1.CopyFile NazevDennihoSouboru, <někde>
```

Skript vydá druhý příkaz Relog zajišťující sumarizaci denního protokolu na jednohodinové intervaly. Následně znovu použije Relog k přidání tohoto protokolu čítačů k souboru historického souhrnu ve složce C:\Perflogs\Historie, který je rovněž

hodinově sumarizován. Na konci tohoto kroku bude ve složce C:\Perflogs\Historie soubor nazvaný <nazev-pocitace>.protokol_historie_vykonnosti.blg, který obsahuje všechna nashromážděná historická data sumarizovaná do hodinových intervalů. Protože souhrnný historický soubor bude růst tempem přibližně 300 až 500 KB za den, bude vhodné periodicky jej zálohovat na dlouhodobé úložiště na jiném místě. Kupříkladu po jednom roce naroste historický sumarizovaný protokol čítačů na přibližně 100 až 200 MB na počítač.

Alternativně lze v tomto místě procedury zvolit ukládání historických, sumarizovaných dat protokolu čítačů do výkonnostní databáze SQL Serveru. Rozhodnete-li se využívat pro historické údaje čítačů repozitář SQL Serveru, nejprve vložte následující inicializační kód:

```
Const parametryPrenastaveniSQL = "-f SQL -t 4"
Const DBPrenastaveniSQL = """SQL:PDB!Hodinove"""
Const NazevSouboruNastaveniSQL = "c:\perflogs\soubor-nastaveni-souhrnnych-
citacu.txt"
```

Dále nahraďte poslední dva příkazy Relog jediným příkazem Relog zajišťujícím vložení vygenerovaných hodinových dat protokolů čítačů přímo do SQL Serveru takto:

```
prikaz = "relog " & NazevDennihoSouboru & " -o " & DBPrenastaveniSQL & _
  " -cf " & NazevSouboruNastaveniSQL & " " & parametryPrenastaveniSQL
```

Tip Maximální flexibility dosáhnete, když budete sestavovat a udržovat sumarizovaná i historická data v protokolech binárního formátu, které lze snadno přenášet mezi počítači na síti a do konsolidační historické databáze PDB využívající SQL Server.

Skript následně obsahuje komentář vyžadující po vás zkopírování právě vytvořeného souboru sumarizovaných denních protokolů čítačů na konsolidační server někde v síti, kde jej použijete v rámci svého procesu hlášení pro správu:

```
'  objFSO2.CopyFile NazevDennihoSouboru, <někde>
```

Skript se pak vrátí do složky `C:\Perflogs\Dnes` a odstraní starší soubory, které byly předtím zkopírovány do složky `C:\Perflogs\Vcera` a zpracovány nástrojem Relog.

Správa protokolů čítačů automaticky generovaných výstrahami

Nakonec skript prochází všemi soubory ve složce `C:\Perflogs\Protokoly` vystrah a odstraňuje ty, které jsou starší než hodnota nastavená v konstantě `PocetDnuVystrahy`, jež standardně odpovídá třem dnům. Důvodem k odstraňování starších protokolů čítačů ve složce Protokoly vystrah je to, že příslušná krize již zřejmě pominula. Když tu podobně jako ve složce `C:\Perflogs\Dnes`, kam se zapisují denní protokoly čítačů, zanedbáte vykonávání nějaké formy automatické správy souborů, nakonec vám na počítači dojde diskový prostor.

Plánované měsíční zprávy a archivace

Další přeprotokolování lze zajišťovat plánovaně pomocí nástroje Scheduled Tasks (Naplánované úlohy). Úlohy, jejichž běh je naplánován jinou službou než Výstrahy a protokolování výkonu, musejí používat aktivní soubory protokolů velmi opatrně. Když si aktivní soubor protokolu čítačů otevře jiný proces, může tím zabránit pro-

tokolovací službě v jeho řádném uzavření nebo znemožní správné fungování příkazového souboru vykonávaného po zavření aktuálního protokolu čítačů.

Na konci měsíce lze soubory protokolů nashromážděné v jednotlivých dnech konsolidovat do jediného archivního souhrnného protokolového souboru. Ve většině případů tak bude zajištěna dostatečná detailnost údajů, aniž by spotřebovávaly příliš diskového prostoru.

Zpracování na konci měsíce sestává ze dvou hlavních funkcí:

1. Omezení počtu vzorků a
2. konsolidování denních souborů do jediného protokolového souboru představujícího daný měsíc.

Nejlepšího výkonu zpracování dosáhnete, když nejprve zkomprimujete jednotlivé soubory a pak je pospojujete. Protože nástroj Relog se snaží spojit dohromady všechny vstupní soubory, ještě než je zpracuje, práce s menšími vstupními soubory zajistí mnohem rychlejší spojení. Výpis 4.13 je příkladem typického příkazového souboru měsíčního zpracování.

VÝPIS 4.13: Příklad měsíčního zpracování

```
Rem ************************************************************************
Rem * arg 1 is the name of the month that the performance data logs
Rem * are being compiled for.
Rem *
Rem * arg 2 is the directory path in which to put the output file
Ren *
Ren * NOTE: This procedure should not run when a daily log is being
Ren * processed. It should run after the last daily log of the month has
Rem * been processed and moved to the SAVE_DIR
Rem *
Rem ************************************************************************
set LOG_DIR <directory path containing these files>
set SAVE_DIR <directory path where daily log files are saved>
set TEMP_DIR <directory path where compressed daily log files are saved>
echo Archiving logs in %SAVE_DIR% at %date% %time%>>
%LOG_DIR%\SAMPLE_RELOG.LOG
Rem compress each daily log and store the output files in the TEMP_DIR
for %a in (%SAVE_DIR%) do %LOG_DIR%\Montlhy_Compress.bat "%a" "%TEMP_DIR%"
if errorlevel 1 goto COMPRESS_ERROR
Rem concatenate the compressed log files into monthly summary file
relog %TEMP_DIR%\*.blg -o %2\%1.blg
if errorlevel 1 goto RELOG_ERROR
Rem clear out the temp directory to remove the temp files
Del /q %TEMP_DIR%\*.*
Rem clear out the original files if you want to free up the space
Rem or you may wish to do this manually after insuring the files were
Rem compressed correctly
Del /q %SAVE_DIR%\*.*
exit :COMPRESS_ERROR
echo Compress error at %date% %time%>> %LOG_DIR%\SAMPLE_RELOG.LOG
exit
```

```
:RELOG_ERROR
echo Relog error at %date% %time%>> %LOG_DIR%\SAMPLE_RELOG.LOG
exit
```

Respektive v českém prostředí:

```
Rem ***************************************************************************
Rem * arg 1 je název měsíce, jehož data protokolů výkonu se kompilují.
Rem *
Rem * arg 2 je adresářová cesta pro uložení výstupního souboru.
Ren *
Ren * POZN.: Tato procedura nesmí běžet, když je zpracováván denní
Ren * protokol. Měla by běžet až po zpracování posledního denního protokolu
Rem * daného měsíce a jeho přesunutí do SAVE_DIR.
Rem *
Rem ***************************************************************************
set LOG_DIR <adresářová cesta obsahující tyto soubory>
set SAVE_DIR <adresářová cesta ukládání denních souborů protokolů>
set TEMP_DIR <adresářová cesta ukládání zkomprimovaných denních souborů
rotokolů>
echo Archivace protokolu %SAVE_DIR% v %date% %time%>>
%LOG_DIR%\SAMPLE_RELOG.LOG
Rem Zkomprimovat jednotlivé denní protokoly a uložit výstupní soubory do
EMP_DIR.
for %a in (%SAVE_DIR%) do %LOG_DIR%\Mesicni_komprimace.bat "%a"
"%TEMP_DIR%"
if errorlevel 1 goto COMPRESS_ERROR
Rem Sloučit zkomprimované soubory protokolů do souboru měsíčního souhrnu.
relog %TEMP_DIR%\*.blg -o %2\%1.blg
if errorlevel 1 goto RELOG_ERROR
Rem Vymazat dočasný adresář a odstranit dočasné soubory
Del /q %TEMP_DIR%\*.*
Rem Nyní lze vymazat původní soubory, aby se uvolnilo místo. To můžete také
Rem udělat manuálně po ověření, že došlo ke správnému zkomprimování
souborů.
Del /q %SAVE_DIR%\*.*
exit :COMPRESS_ERROR
echo Chyba komprimace v %date% %time%>> %LOG_DIR%\SAMPLE_RELOG.LOG
exit
:RELOG_ERROR
echo Chyba přeprotokolování v %date% %time%>> %LOG_DIR%\SAMPLE_RELOG.LOG
exit
```

V příkazovém souboru ve výpisu 4.13 se soubory protokolů uložené v adresáři SAVE_DIR komprimují dalším příkazovým souborem a pak se slučují.

VÝPIS 4.14: Příkazový soubor volaný úlohou měsíčního zpracování

```
Rem ***************************************************************************
Rem * arg 1 is the filename of the daily performance data log file to
Rem * compress for subsequent concatenation.
Rem *
Rem * arg 2 is the directory path in which to put the output file
Ren *
```

```
Rem ********************************************************************
set LOG_DIR <directory path containing these files>
set SAVE_DIR <directory path where daily log files are saved>
set TEMP_DIR <directory path where compressed daily log files are saved>
echo Compressing file: %1 at %date% %time%>> %LOG_DIR%\SAMPLE_RELOG.LOG
Rem compress each daily log and store the output files in the TEMP_DIR
Relog %1 -config %LOG_DIR%\COMPRESS_CFG.TXT -o %2\%~nx1 >>
%LOG_DIR%\SAMPLE_RELOG.LOG
```

Respektive v českém prostředí:

```
Rem ********************************************************************
Rem * arg 1 je název souboru protokolu denních výkonnostních dat, který
Rem * se má zkomprimovat pro následné sloučení.
Rem *
Rem * arg 2 je adresář, kam se má uložit výstupní soubor.
Ren *
Rem ********************************************************************
set LOG_DIR <adresářová cesta obsahující tyto soubory>
set SAVE_DIR <adresářová cesta ukládání denních souborů protokolů>
set TEMP_DIR _
  <adresářová cesta ukládání zkomprimovaných denních souborů protokolů>
echo Komprimace souboru: %1 v %date% %time%>> %LOG_DIR%\SAMPLE_RELOG.LOG
Rem Komprimace jednotlivých denních protokolů a uložení výstupních souborů
do TEMP_DIR.
Relog %1 -config %LOG_DIR%\COMPRESS_CFG.TXT -o %2\%~nx1 >>
%LOG_DIR%\SAMPLE_RELOG.LOG
```

Příkazový soubor ve výpisu 4.14 je oddělen od hlavního příkazového souboru, aby bylo možné využít funkce rozdělování názvu souboru interpreteru příkazů.

Jakmile jsou data nashromážděná během měsíce konsolidována do zkomprimovaného souhrnného souboru, lze jeho pomocí vytvářet měsíční hlášení. Stejné soubory konfigurace protokolů čítačů lze dokonce používat k měsíčnímu souhrnu stejných údajů hlášených v denních zprávách. Jediným rozdílem je práce s odlišným vstupním souborem – souborem měsíčního souhrnu místo souboru denních výkonnostních dat.

Definice konfigurací protokolů pro více serverů

Konfigurační soubory lze rovněž využít k zavedení jednotných procedur shromažďování protokolů čítačů na více počítačích. Následující konfigurační soubor ukazuje základní nastavení protokolu, která lze použít v konfiguraci sestávající z mnoha serverů. V případě potřeby lze samozřejmě doplnit další čítače.

```
[name]
Basic Performance Data Log

[sample]
15

[format]
bin
```

```
[--max]
[--append]
[version]
nnnnnn

[runcmd] <insert the full path name of command file to run
          when this log file is closed>

[counters]
\Processor(_Total)\% Processor Time
\LogicalDisk(_Total)\% Disk Time
\Memory\Pages/sec
\Network Interface (*)\Bytes Total/sec
```

Respektive v českém prostředí:

```
[ name ]
Zakladni protokol vykonnostnich dat

[sample]
15

[format]
bin

[--max]

[--append]

[version]
nnnnnn

[runcmd] <vložte úplnou cestu příkazového souboru spuštěného po zavření
          tohoto protokolového souboru >

[counters]
\Procesor(_Total)\% času procesoru
\Logický disk(_Total)\% času disku
\Paměť\Stránky/s
\Rozhraní sítě(*)\Bajty celkem/s
```

4.3 Práce s repozitářem SQL Serveru

Podpora dat protokolů čítačů ze strany SQL Serveru je flexibilní a nabízí dobrou možnost sestavování a udržování dlouhodobého repozitáře dat protokolů čítačů jak pro analýzu, tak i plánování kapacity. Tento oddíl popisuje použití repozitáře SQL Serveru k hlášení výkonnostních dat protokolů čítačů. Pojednává o databázovém schématu využívaném k ukládání dat a protokolů čítačů. Následně popisuje, jak používat nástroje extrakce dat, jako je Microsoft Excel, pro přístup k datům protokolů čítačů v repozitáři SQL Serveru a jejich zpracování do hlášení. Nakonec je uveden příklad přejímající dat protokolů čítačů z aplikace SQL Server prostřednic-