Prerequisite Scanner
V1.2

用户指南

IBM

Prerequisite Scanner
V1.2

用户指南

IBM

注

在使用本资料及其支持的产品之前，请阅读第 149 页的『声明』中的信息。

本版本适用于 IBM Prerequisite Scanner V1.2 及所有后续发行版和修订版，直到在新版本中另有声明为止。

目录

图

表

第 1 章 Prerequisite Scanner 概述

IBM® Prerequisite Scanner 是一个扫描工具，用于在实际部署指定软件前对其执行先决条件标识、检查和验证。它根据那些针对先决条件属性设置的值来扫描硬件和软件先决条件。Scanner 在命令行界面中显示扫描结果，并且还将结果保存到文本文件以及可选的 XML 文件中。另外，它还将参考消息、跟踪消息和调试消息写入日志文件。

Prerequisite Scanner 可以检查机器的操作系统并验证其版本对于指定的软件而言是否正确。如果任何一项先决条件检查失败，那么整个扫描将失败。

您可以在安装后或任何时间运行 Prerequisite Scanner 以确认当前环境。Prerequisite Scanner 不要求运行要检查先决条件的软件的安装程序。

您可以扩展 Prerequisite Scanner，以扫描并非作为 Scanner 随附的核心先决条件检查集组成部分的先决条件。

Prerequisite Scanner 根据平台调用下列类型的脚本:

- Windows: VBScript 和批处理
- UNIX: Shell

注: 即使在 Windows 机器上安装了类似于 UNIX 的环境（例如 Cygwin），也无法在 Windows 系统上运行 UNIX 脚本。

Prerequisite Scanner 体系结构

IBM Prerequisite Scanner 由下列主要组件组成: 要在命令行界面中运行的脚本、一组用于先决条件检查的属性、先决条件属性配置文件、先决条件收集器和先决条件评估程序。Prerequisite Scanner 的运行结果以各种输出格式提供。

先决条件属性

先决条件属性是要安装的产品或解决方案所需的不同软件和硬件先决条件的期望值。先决条件属性的示例包括机器上可用的可用磁盘空间总量、机器上未被使用的一组端口以及当前已安装的应用程序的集合。

由于这些先决条件属性的值可能随产品不同而有所变化，因此这些属性及其值以具有可选限定符的名称/值对形式表示。它们包含在先决条件属性配置文件中。每个先决条件属性各占一行。

先决条件属性遵循以下格式:

[prefix_identifier.]property_name[.suffix_identifier]=
[[qualifier_name:qualifier_value]]property_value

其中:

- *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应标识。此前缀标识是某些预定义类别所必需。
- *property_name* 是先决条件属性的名称。
- *suffix_identifier* 是第 6 页的表 4 中概述的先决条件属性子类型的相应可选标识。
- *qualifier_name* 是先决条件属性的可选属性。IBM Prerequisite Scanner 使用此属性来限定先决条件属性或者对先决条件属性执行的检查类型。

  注: 可以指定多个以逗号分隔的限定符。这组限定符必须括在 [] 方括号中。
- *qualifier_value* 是可选属性的值。每个限定符及其值都必须以冒号 : 定界。
- *property_value* 是先决条件属性的值，并且可以是字符串或整数。

先决条件属性可以具有一个或多个值，具体取决于数据类型和限定符，如下所示:

- 单个整数，例如用于表示端口号值的 8080。
- 一组使用表 1 中描述的特殊字符表示的整数。

表 1. 表示范围类型的特殊字符

特殊字符	描述
*	用于标识多个值的占位符。例如，ports.* 可以表示数据库产品 (ports.DB) 端口和 IBM WebSphere® Application Server (ports.WAS) 端口的超集。
+	指示实际版本必须最多与预期版本的值匹配。例如，os.versionNumber=5.0+，表示版本必须是 5.0 或之后版本。
-	指示实际版本必须最多与预期版本的值匹配。例如，os.versionNumber=5.0- 表示版本必须为 5.0 或之前版本。
.*	指示实际版本可以与预期版本的任何通配值匹配。例如: os.versionNumber=5.*, means that the version can be 5.0, 5.0.1 or 5.5.

限制: 在 Windows 系统上，* 通配符只有在 OS Version 先决条件属性的正则表达式中使用时才受支持。

- 可以表示先决条件类型的下列任何值的字符串:
  - 带有单位的数字值，例如 8GB 或 10MB
  - 应用程序、操作系统、体系结构或软件包，例如 IBM Lotus Symphony, RedHat Enterprise Linux 5.4, 32-bit 或 ftp

    注: 字符串还可以包含多个以逗号分隔的值，例如应用程序列表。
  - 由下列其中一个组合表示的二者择一值，例如 True|False, Available|Unavailable 或 Enabled|Disabled

第 3 页的表 2 对先决条件属性的示例作了概述。

表 2. 先决条件属性示例

先决条件属性	说明
`Disk=1GB`	这是可用磁盘空间量，其中: • *property_name* 为 `Disk` • *property_value* 为 `1GB`
`user.isAdmin=True`	表示登录用户是否属于管理员组，其中: • *prefix_identifier* 为 `user`（表示用户先决条件属性） • *property_name* 为 `isAdmin` • *property_value* 为 `True`

`network.availablePorts.DB=60000-60005`
`network.availablePorts.WAS=8080`
`network.availablePorts.FTP=21`
	用于检查端口 60000-60005 是否可供数据库服务器使用，端口 8080 是否可供 WebSphere Application Server 使用，以及端口 21 是否可供 FTP 使用，其中:
	• *prefix_identifier* 为 `network`（表示常规先决条件属性）
	• *property_name* 为 `availablePorts`
	• *suffix_identifier* 为 `DB`（表示可用的数据库端口）、`WAS`（表示可用的 WebSphere Application Server 端口）和 FTP（表示可用的 `FTP` 端口）
	• *property_value* 为 `60000-60005`、`8080` 或 `21`

`os.dir.home=[dir:/home,type:permission]755+`	用于检查主目录是否具有 `drwxr-xr-x` 许可权，其中: • *prefix_identifier* 为 `os`（表示操作系统先决条件属性） • *property_name* 为 `dir` • *suffix_identifier* 为 `home`（表示要检查的目录） • *qualifier_name* 为 `dir` 和 `type`（用于限定先决条件属性以及检查类型） • *qualifier_value* 为 `home` 和 `permission`（限定符的值） • *property_value* 为 `755+`，即主目录的访问许可权的八进制数表示

您可以为每个要针对其运行 Prerequisite Scanner 的产品添加或编辑预定义先决条件属性。另外，您还可以创建定制先决条件属性，并根据需要使用 Prerequisite Scanner 收集器和评估程序来扫描和比较先决条件属性。

相关概念:

第 8 页的『先决条件属性的预定义限定符』

IBM Prerequisite Scanner 为预定义类别中的一些先决条件属性提供了一组基本限定符。限定符表示先决条件属性的属性，Prerequisite Scanner 使用这些属性来限定先决条件属性或者限定要对该先决条件属性执行的检查类型。

先决条件属性的预定义类别

IBM Prerequisite Scanner 为不同类别（公共、已安装的软件、操作系统、用户、连通性、Internet Explorer、数据库服务器、环境变量和网络）的数据提供了一组基本先决条件属性，包括用于 Windows 和 UNIX 的特定于平台的属性。

<*prefix_identifier*> 是先决条件属性的预定义类别的标识。

第 4 页的表 3 对硬件和软件先决条件的预定义类别作了概述。

表 3. 基本先决条件属性类别

数据类别	描述	必需的前缀标识
公共	此类别用于检查公共先决条件，例如处理器速度、内存量、磁盘空间量和临时空间量。以下示例是用于检查操作系统的先决条件属性: `OS Version=RedHat Enterprise Linux 5.4`	None
已安装的软件	此类别用于检查已安装的软件先决条件（例如在 Windows 注册表中注册的程序）以及是否安装了 cygwin 和 gskit。以下示例是用于扫描操作系统注册表以查找已安装的程序及其位置的先决条件属性: `installedSoftware=`*`list_of_installed_programs`*	None
用户	此类别用于检查用户先决条件，例如登录用户是否具有管理权以及是否为 root 用户。以下示例是用于检查登录用户是否为管理员组成员的先决条件属性: `user.isAdmin=True`	`user`
操作系统	此类别用于检查操作系统先决条件，例如版本、体系结构、内存总量、可用内存量和物理内存总量。以下示例是用于检查远程注册服务是否处于运行状态的先决条件属性: `os.isServiceRunning.remoteRegistry=True`	`os`
连通性	此类别用于检查连通性先决条件，例如 Telnet 是否处于运行状态以及 Scanner 所能够连接到的 IP 地址和端口。	None
网络	此类别用于检查所有平台的公共网络先决条件，例如是否有可用的端口。以下示例是用于检查 8080 端口是否可供 IBM WebSphere Application Server 使用的先决条件属性: `network.availablePorts.was=8080`	`network`
Windows 网络	此类别用于检查 Windows 网络先决条件，例如是否在机器上启用了 NetBIOS 和 DHCP 以及 Ping 属性。以下示例是用于检查是否至少有一个具有有效 IP 地址的适配器启用了 NetBIOS 作为协议的先决条件属性: `network.netBIOSEnabled=True`	`network`
UNIX 网络	此类别用于检查 UNIX 网络先决条件，例如是否在机器上启用了 NetBIOS 和 DHCP 以及 Ping 属性。以下示例是用于检查本地主机是否对 Ping 协议作出响应的先决条件属性: `network.pingLocalhost=True`	`network`
Internet Explorer	此类别用于检查 Microsoft Internet Explorer 先决条件，例如版本。以下示例是用于检查 Internet Explorer 版本是否为 `7.0` 的先决条件属性: `internetExplorer.version=7.0`	`internetExplorer`
数据库服务器: DB2®	此类别用于检查 DB2 先决条件，例如版本。以下示例是用于检查 DB2 版本是否至少为 `9.5` 的先决条件属性: `DB2 Version=9.5.*`	`DB2`
数据库服务器: Oracle	此类别用于检查 Oracle 先决条件，例如版本。以下示例是用于检查 Oracle 客户机版本是否至少为 `9.2.0.8` 的先决条件属性: `oracle.Client=9.2.0.8+`	`Oracle`
环境变量	此类别用于检查环境变量先决条件，例如是否设置了环境变量。以下示例是用于检查类路径是否包含 Derby JAR 文件的先决条件属性: `env.classpath.derbyJAR=False`	`env`

表 3. 基本先决条件属性类别 (续)

数据类别	描述	必需的前缀标识
Autonomic Deployment Engine	此类别用于检查 Autonomic Deployment Engine 先决条件，例如是否安装了 Autonomic Deployment Engine 以及 Tivoli® Integrated Portal 的安装单元。以下示例是用于检查是否在 Windows 系统上安装了 Tivoli Integrated Portal V2.1.1.0 或 2.1.1.1 安装单元的先决条件属性。 `de.installationUnit=regex{.*C37109911C8A11D98E1700061BDE7AEA.*|.*TIP 2.1.1.0.*|.*TIP 2.1.1.1.*}`	`de`
数据库服务器: MS SQL	此类别用于检查 MS SQL 先决条件，例如版本。以下示例是用于检查 MS SQL Server 版本是否为 SQL Server 2008 R2 Developer Edition 的先决条件属性: `mssql.Server=10.50.1600.1`	`mssql`

先决条件属性的预定义子类型

IBM Prerequisite Scanner 为预定义类别中的一些先决条件属性提供了一组基本子类型。子类型按应用程序、实用程序或服务子类型对先决条件属性（例如类别）进行进一步分类。

例如，您可能使用了针对可用网络端口的先决条件属性。您可以对该先决条件属性进行进一步分类，以检查数据库服务器和应用程序服务器的可用端口或协议。

<suffix_identifier> 是先决条件属性名中的子类型的可选标识。

表 4 对不同类别先决条件属性的预定义子类型（包括 *<suffix_identifier>*）作了概述。

表 4. 预定义子类型

先决条件属性子类型	后缀标识	平台	描述	子类型的有效值
与平台无关的网络类别				
`network.availablePorts.`*`app_type`*	*app_type*	全部	使用此命名约定来检查端口或端口范围是未被侦听还是可用于 *app_type* 应用程序类型。	用于表示 *app_type* 的字符串，例如: • `DB2` 将检查 DB2 数据库服务器的端口 • `WAS` 将检查 WebSphere Application Server 的端口 • `ftp` 将检查 FTP 端口
`network.portsInUse.`*`app_type`*	*app_type*	全部	使用此命名约定来检查端口或端口范围是正被侦听还是正用于 *app_type* 应用程序类型。	用于表示 *app_type* 的字符串，例如: • `DB2` 将检查 DB2 数据库服务器的端口 • `WAS` 将检查 WebSphere Application Server 的端口 • `ftp` 将检查 FTP 端口
操作系统类别				
`os.dir.`*`dir_name`*	*dir_name*	UNIX	使用此命名约定来检查 *dir_name* 文件系统。此先决条件属性的值使用了预定义的限定符。	用于表示 *dir_name* 的字符串，例如: • `tmp` • `home`
`os.file.` *`script_name`*	*script_name*	UNIX	使用此命名约定来检查机器上是否存在可用的 *script_name* 脚本。	用于表示 *script_name* 的字符串，例如: • `bash` • `expect` • `gzip` • `tar`

`os.`
`isService`
`Running.`
`service_name`
	service_name	Windows	使用此命名约定来检查 *service_name* 服务是否正在机器上运行。	用于表示 *service_name* 的字符串，例如:
				• `remoteRegistry`
				• `DNSClient`
				• `terminalServices`

表 4. 预定义子类型 (续)

先决条件属性子类型	后缀标识	平台	描述	子类型的有效值
`os.lib.` *`lib_name_version`*	*lib_name* *_version*	UNIX	使用此命名约定来检查机器上是否安装了版本受支持的 *lib_name_version* 库。	例如，用于表示 *lib_name_version* 的字符串（显示为粗体）： • 32 位 **libstdc++.so.#** 库 • 64 位 **libstdc++.so.#** 库 • 32 位 **libXft.so.#** 库 • 32 位 **libXtst.so.#** 库 • 64 位 **libaio.so.#** 库 • 32 位 **xlC.rte** XLC 运行时级别 • AIX® V5.3 的 32 位 **xlC.aix50.rte** XLC 运行时 • AIX V6.1 的 32 位 **xlC.aix61.rte** XLC 运行时 • AIX IOCP **bos.iocp.rte** 库 • **bos.loc.iso.en_us**，AIX 基本操作系统的 ISO 代码文件集 `regex {`*`str`*`}`，这是具有输入参数 *str* 的正则表达式，用于表示库名搜索模式，例如: `regex {.*libgcc.*}` 检查是否存在该操作系统的 GCC 低级别运行时库 `libgcc` 版本。

表 4. 预定义子类型 (续)

先决条件属性子类型	后缀标识	平台	描述	子类型的有效值
`os.package.` *`package_name`*	*package_name*	UNIX	使用此命名约定来检查机器上是否安装了版本受支持的 *package_name* 软件包。	例如，用于表示 *package_name* 的字符串（显示为粗体）： • **`bash`** Shell • **`expect`**（对于 TCL 扩展包） • **`libgcc`**（对于 GCC 低级别运行时包） • **`openssh`**（对于开放式源代码安全 Shell） • **`openssl`**（对于开放式源代码 SSL/TLS 工具箱） • **`perl`**（对于 Perl 脚本包） • **`rpm`**（对于 RPM 或 RPM 构件包） • **`telnet`**（对于 Telnet 包） • **`wget`**（对于 GNU 文件检索包）
`os.space.` *`dir_name`*	*dir_name*	UNIX	使用此命名约定来检查所指定 *dir_name* 文件系统的可用磁盘空间量。此先决条件属性的值使用了预定义的限定符。	用于表示 *dir_name* 的字符串，例如: • `usr` • `home` • `tmp` • `var`

先决条件属性的预定义限定符

IBM Prerequisite Scanner 为预定义类别中的一些先决条件属性提供了一组基本限定符。限定符表示先决条件属性的属性，Prerequisite Scanner 使用这些属性来限定先决条件属性或者限定要对该先决条件属性执行的检查类型。

例如，您可能使用了针对文件系统的先决条件属性。您可以根据先决条件属性的文件系统名称和访问许可权属性来限定要对该属性执行的检查。另外，还可以根据文件系统路径和单位属性来限定检查可用磁盘空间量时使用的单位类型。

限定符支持进行定制以满足环境的需要，这使 Scanner 无需进行有关多维先决条件（例如缺省路径和访问许可权）的属性的隐式假定。您可以更改预定义限定符的值，但是无法向预定义先决条件属性的预定义现有限定符集合添加新的限定符。

限定符必须遵循以下格式:

`[`*`qualifier_name`*`:`*`qualifier_value`*`,` *`qualifier_name`*`:`*`qualifier_value`*`]`
`property_value`

其中:

- *qualifier_name* 是先决条件属性的可选属性，IBM Prerequisite Scanner 使用这些属性来限定先决条件属性或者限定要对该先决条件属性执行的检查类型。

- *qualifier_value* 是可选属性的值。

  限定符的值也可以是名称/值对，用于支持多个依赖于用户类型的有效值。例如，根据用户是否为 root 用户，主目录的路径有所不同。
- *property_value* 是先决条件属性的值，并且可以是字符串或整数。

每个限定符及其值都必须以冒号 **:** 定界。可以指定多个以逗号分隔的限定符。这组限定符必须括在 **[]** 方括号中。

表 5 概述了先决条件属性的不同类别的预定义限定符。某些先决条件属性还使用预定义的子类型对先决条件属性进行进一步分类。

要点: 不得将预定义限定符与其他预定义先决条件属性配合使用。

表 5. 预定义限定符

先决条件属性	平台	描述	有效的限定符和值
具有预定义子类型的操作系统类别			
os.dir.*dir_name*	UNIX	根据下列限定属性来检查 *dir_name* 文件系统: • dir 属性，用于确定要检查的文件系统 • type 属性，用于确定要检查的文件系统的属性，例如对该文件系统的访问许可权的 <octal_digits> 八进制数表示 例如，<*dir_name*> 可以表示: • tmp • home	具有以下限定符格式的字符串: [dir:*dir_name*, type:permission] *octal_digits*+ 例如，要检查主目录是否具有 drwxr-xr-x 许可权: os.dir.home=[dir:/home, type:permission]755+

表 5. 预定义限定符 (续)

先决条件属性	平台	描述	有效的限定符和值
os.space.*dir_name*	UNIX	用于根据下列一个或多个限定属性来检查所指定 *dir_name* 文件系统的可用磁盘空间量: • dir 属性，用于确定要检查的文件系统路径 • unit 属性，用于确定要使用的磁盘空间单位 dir 属性的值依赖于登录用户；因此，该值是一个用于表示用户类型（即 root 或非 root）和相关路径的名称/值对。 例如，*dir_name* 可以表示: • usr • home • tmp • var	具有以下限定符格式的字符串，用于指定 root 用户的文件系统: [dir:**root**=*dir_path*, unit:*unit_name*] *disk_space* 例如: os.space.usr= [dir:root=/usr/ibm/common/acsi, unit:GB]200 具有以下限定符格式的字符串，用于指定非 root 用户的文件系统: [dir:**non_root**=*dir_path*, unit:*unit_name*] disk_space 例如: os.space.home= [dir:non_root=USERHOME/.acsi_HOST, unit:MB]200 具有以下限定符格式（仅使用一个限定符）的字符串: [dir:*dir_path*] *disk_space* MB 例如: os.space.home=[dir:/home/sat]250MB
不具有预定义子类型的操作系统类别			
os.mountcheck	UNIX	用于根据下列限定属性来检查是否已装配文件系统: • drive 属性，用于确定哪个目录是已装配的文件系统 • nosuid 属性，用于确定是否设置了装配选项（如果已装配文件系统）	具有以下限定符格式的字符串: [drive:*dir_name*, *mount_option*: false|true] True|False 例如，要检查是否已装配 /home 目录以及是否未设置 nosuid 选项: os.mountcheck=[drive:/home, nosuid:false]True

表 5. 预定义限定符 (续)

先决条件属性	平台	描述	有效的限定符和值
os.SELinux	Linux	用于根据下列限定属性来检查 Security-Enhancement Linux 功能的实施状态: • source 属性，用于确定要用于相关操作系统的命令	• 具有以下限定符格式的字符串: [source:Command] Disabled|Enabled 例如，要检查功能部件在 Red Hat 或 SUSE 操作系统上是处于禁用状态还是处于许可状态: os.SELinux=[source: Command]Disabled • 不具有限定符的字符串，其中操作系统为一般 Linux 变体: os.SELinux=Disabled
os.ulimit	UNIX	使用此命名约定根据下列限定属性检查能否运行数目不受限的进程: • type 属性，用于确定要检查的附加限制，例如 filedescriptorlimit 检查进程可以打开的文件描述符数目的限制	具有以下限定符格式的字符串: [type:*limit_name*]*limit_value*, limited|unlimited 例如，要检查文件描述符限制是否大于 8192（进程数不受限）: os.ulimit= [type:filedescriptorlimit] 8192+,unlimited 要检查的限制的有效类型如下所示，其中 *limit_name* 表示限制类型: • ALL，用于检查所有限制 • corefilesizelimit • datasegmentlimit • filedescriptorlimit • filesizelimit • hardlimit • processlimit • maxmemorysizelimit • maxprocesseslimit • stacksizelimit • threadlimit
不具有预定义子类型的公共类别			

表 5. 预定义限定符 (续)

先决条件属性	平台	描述	有效的限定符和值
Disk	Windows	这是具有下列可选限定属性的可用磁盘空间量: • dir 属性，用于确定要检查的目录路径 • unit 属性，用于确定要使用的磁盘空间单位	具有以下限定符格式的字符串: [dir:*dir_path*, unit:*unit_name*] *disk_space* 例如: Disk= [dir:C:\Program Files\IBM\SQLLIB, unit:MB]1431 以 MB 或 GB 为单位的数字格式: <disk_space>MB|GB 例如: Disk=250MB

产品代码

IBM Prerequisite Scanner 在文件名和参数名中使用多字符代码来标识产品和组件以及确定要使用的配置文件类型。

product_code

这是用于在 Windows 或 UNIX 系统上表示产品代码的变量。产品代码用于标识产品、单个平台（例如 Windows、AIX、HP-UX、Linux 和 Solaris 以及该产品支持的操作系统版本（可选）。它们存储在 codename.cfg 文件中。任何支持多个平台的产品都有多个产品代码，并且每个产品代码都根据需要标识产品、平台和操作系统版本。

例如，COD、COK 和 COX 产品代码用于标识一些受支持的操作系统和 IBM Tivoli Provisioning Manager 版本:

```
COD=Tivoli Provisioning Manager for AIX 6.1
COK=Tivoli Provisioning Manager for HP-UX
COX=Tivoli Provisioning Manager for Windows 2008
```

运行 Prerequisite Scanner 时，可以传递产品代码以及可选的产品版本作为输入参数。Scanner 检查产品代码是否存在于 codename.cfg 文件中。在 UNIX 系统上，如果 Scanner 找不到产品代码，它将会退出。在 Windows 系统上，如果 Scanner 找不到产品代码，它不会退出。

然后，Scanner 使用输入参数在 *ips_root*/Windows|UNIX_Linux 目录中查找配置文件。文件名包含与输入参数相同的产品代码和产品版本。如果未传递可选的产品版本参数，那么 Scanner 将使用它在此目录中找到的最新版本配置文件。接着，Prerequisite Scanner 将开始执行扫描。

注: 仅限于 Windows 系统: 如果产品代码未存在于 codename.cfg 文件中，但名称中包含产品代码的配置文件存在，那么 Prerequisite Scanner 将在输出中显示没有为产品名定义的产品代码和版本号。

Prerequisite Scanner 配置文件

用于各个平台的 IBM Prerequisite Scanner 配置文件包含用于产品所支持的各个平台的先决条件属性及其期望值。Prerequisite Scanner 提供了一组可供您编辑的预定义配置文件。您必须针对所要支持的新产品和平台创建配置文件。

配置文件具有 .cfg 文件扩展名。请将其存储在 *ips_root*/<*OS*> 目录中，其中 <*OS*> 是操作系统类型的名称，例如 Windows 或 UNIX_Linux。

配置文件必须遵循下列规则:

- 文件扩展名必须是 .cfg
- 文件名的命名约定:

  product_code[_<*version*>].cfg

  其中:

  – *product_code*

    这是用于在 Windows 或 UNIX 系统上表示产品代码的变量。产品代码用于标识产品、单个平台（例如 Windows、AIX、HP-UX、Linux 和 Solaris 以及该产品支持的操作系统版本（可选）。它们存储在 codename.cfg 文件中。任何支持多个平台的产品都有多个产品代码，并且每个产品代码都根据需要标识产品、平台和操作系统版本。

  – <*version*> 是表示版本、发行版、修订版和级别的 8 位代码（此代码的每个部分各占两位）；例如，7.3.21 为 07032100。
- 各个节中的组先决条件属性必须遵循节标题的命名约定。
- 每个先决条件属性的标准格式都是具有可选限定符的名称/值对，并且每个属性各占一行:

  [<*prefix_identifier*>.]<*property_name*>[.<*suffix_identifier*>]=
  [[<*qualifier_name*>:<*qualifier_value*>]]<*property_value*>

没有节的配置文件示例

此示例检查先决条件属性，但是不对所需操作系统版本的不同先决条件属性进行区分。

```
os.space.var=[dir:root=/var/ibm/common/acsi,unit:MB]1.0
os.space.usr=[dir:root=/usr/ibm/common/acsi,unit:MB]200
os.space.home=[dir:non_root=USERHOME/.acsi_HOST,unit:MB]200
os.space.tmp=30MB
env.classpath.derbyJAR=False
network.pingSelf=True
network.pingLocalhost=True
network.availablePorts.Derby=4130
OS Version=RedHat Enterprise Linux 4.*,RedHat Enterprise Linux 5.*
os.package.compat-libstdc++-33=compat_libstdc++_33
os.package.libgcc=libgcc-3.4.3-9
```

相关概念:

『配置文件中的节』
您可以通过配置文件中的一组节对先决条件属性进行分组（每个节代表一个数据类型类别）。节在配置文件中是可选的。

配置文件中的节

您可以通过配置文件中的一组节对先决条件属性进行分组（每个节代表一个数据类型类别）。节在配置文件中是可选的。

节标题的命名约定为:

[*category_name*:*category_value*]

其中:

- *category_name* 是用于表示数据类型类别的多字符代码
- *category_value* 是用于表示类别的允许值的多字符代码

  注: 这些值可以使用第 2 页的表 1 中概述的特殊字符。

每个类别名称及其值都必须以冒号

: 定界并括在方括号 [] 中。

您可以通过组合节标题来使用多个数据类型类别，从而将先决条件属性限定为该组指定类别。

[*category_name*:*category_value*][*category_name*:*category_value*]

例如，要指定适用于运行 32 位 SUSE Linux Enterprise Server V11 Itanium 操作系统的机器的先决条件属性:

```
[OSType:SUSELinuxEnterpriseServer11][OSArch:64-bit][CPU:Itanium]
```

对于所有平台，您可以使用 | 逻辑 OR 符号来使用二者择一数据类型类别。例如，要将任何环境变量设置为 True，节标题的组合为:

- **UNIX 系统**

  ```
  [@TPAE_DB_FEATURE:True|@TPAE_DIR_FEATURE:True|@TPAE_J2EE_FEATURE:True]
  ```

- **Windows 系统**

  ```
  [@TPAE_DB_FEATURE:True]|[@TPAE_DIR_FEATURE:True]|[@TPAE_J2EE_FEATURE:True]
  ```

要点: | 逻辑 OR 符号的位置对于 Windows 和 UNIX 系统有所不同。对于 UNIX 系统，节标题集括在一组方括号 [] 中，各个节标题之间仅以此符号分隔。对于 Windows 系统，此符号对每个具有相关方括号 [] 的完整节标题进行定界。

仅限 Windows 系统: 可以使用 ! 逻辑 NOT 符号来排除数据类型类别。例如，要排除 Windows Server 2003 R2 变体，节标题的组合为: `[OSType:Windows Server 2003][!OSType:Windows Server 2003 R2]`

第 15 页的表 6 对支持的数据类型类别及相关的允许值作了概述。

表 6. 支持的数据类型类别和值

数据类型类别	描述	允许的值
OSType	操作系统类型	• UNIX 指示此类别中的所有属性是所有 UNIX 平台（包括 AIX、HP-UX、Linux 和 Solaris）的公共属性，例如: [OSType:UNIX] • AIX 指示此类别中的所有属性是所有 AIX 操作系统变体的公共属性，例如: [OSType:AIX] • HP-UX 指示此类别中的所有属性是所有 HP-UX 操作系统变体的公共属性，例如: [OSType:HP-UX] • LINUX 指示此类别中的所有属性是所有 Linux 操作系统变体的公共属性，例如: [OSType:LINUX] • RedHat 指示此类别中的所有属性是所有 RedHat Linux 操作系统变体的公共属性，例如: [OSType:RedHat] • RedHatEnterpriseLinuxServer 指示此类别中的所有属性是所有 RedHat Enterprise Linux Server 操作系统变体的公共属性，例如: [OSType:RedHatEnterpriseLinuxServer] • SUSE 指示此类别中的所有属性是所有 SUSE Linux 操作系统变体的公共属性，例如: [OSType:SUSE] • SUSELinuxEnterpriseServer 指示此类别中的所有属性是所有 SUSE Linux Enterprise Server 操作系统变体的公共属性，例如: [OSType:SUSELinuxEnterpriseServer] • Solaris 指示此类别中的所有属性是所有 Solaris 操作系统变体的公共属性，例如: [OSType:Solaris]

表 6. 支持的数据类型类别和值 (续)

数据类型类别	描述	允许的值

		• Windows
		
		指示此类别中的所有属性是所有 Windows 操作系统的公共属性，例如:
		
		`[OSType:Windows]`
		
		• Windows 2000 Workstation (V5.0.*)
		
		指示此类别中的所有属性是所有 Windows 2000 操作系统变体的公共属性，例如:
		
		`[OSType:Windows 2000]`
		
		• Windows XP Workstation (V5.1.*)
		
		指示此类别中的所有属性是所有 Windows XP Professional 32 位操作系统变体的公共属性，例如:
		
		`[OSType:Windows XP]`
		
		• Windows XP Workstation (V5.2.*)
		
		指示此类别中的所有属性是所有 Windows XP Professional 64 位操作系统变体的公共属性，例如:
		
		`[OSType:Windows XP]`
		
		• Windows Vista Workstation (V6.0.*)
		
		指示此类别中的所有属性是所有 Windows Vista 操作系统变体的公共属性，例如:
		
		`[OSType:Windows Vista]`
		
		• Windows 7 Workstation (V6.1.*)
		
		指示此类别中的所有属性是所有 Windows 7 操作系统变体的公共属性，例如:
		
		`[OSType:Windows 7]`
		
		• Windows 2000 Server (V5.0.*)
		
		指示此类别中的所有属性是所有 Windows 2000 Server 操作系统变体的公共属性，例如:
		
		`[OSType:Windows 2000]`
		
		• Windows Server 2003 (V5.2.*)
		
		指示此类别中的所有属性是所有 Windows Server 2003 操作系统变体的公共属性，例如:
		
		`[OSType:Windows Server 2003]`
		
		• Windows Server 2003 R2（V5.2.* 并且其他类型的操作系统描述为 R2）
		
		指示此类别中的所有属性仅是 Windows Server 2003 R2 操作系统变体的公共属性，例如:
		
		`[OSType:Windows Server 2003 R2]`

表 6. 支持的数据类型类别和值 (续)

数据类型类别	描述	允许的值

		• Windows Server 2008 (V6.0.*)
		指示此类别中的所有属性是所有 Windows Server 2008 操作系统变体的公共属性，例如:
		`[OSType:Windows Server 2008]`
		• Windows Server 2008 R2 (V6.1.*)
		指示此类别中的所有属性仅是 Windows Server 2008 R2 操作系统变体的公共属性，例如:
		`[OSType:Windows Server 2008 R2]`
		• *<OS_Name_Version>*
		指示此类别中的所有属性是该版本操作系统的公共属性，例如:
		`[OSType:RedHatEnterpriseLinuxServer4.2]`
		注: 允许使用特殊通配符 * 以指定多个版本。

OSArch	操作系统的体系结构	• 32 位，例如:
		`[OSArch:32-bit]`
		• 64 位，例如:
		`[OSArch:64-bit]`

CPU	通用处理器系列名称	Itanium，例如: `[CPU:Itanium]`
CPUArch	处理器的体系结构	64 位 PowerPC® 和 Power Architecture® 处理器的体系结构，即: • ppc4 • POWER4 • POWER5 • POWER6® • POWER7® 例如: `[CPUArch:ppc4]`
@*<EnvVar_Name>*	产品的环境变量	遵循该产品的规则，例如: `[@TPAE_DB_SERVER:True]`

使用了节的 Windows 配置文件示例

此示例使用节依次针对任何 Windows 机器以及运行特定 Windows 版本的机器的先决条件属性进行分类。

```
#Properties for all Windows operating systems, that is, Windows XP and above
[OSType:Windows]
os.versionNumber=5.1+
network.pingSelf=True
network.pingLocalhost=True
network.availablePorts.Derby=4130
env.CIT.homeExists=True
env.classpath.derbyJAR=False
# Disk space properties
commonPath=10MB
```

```
installPath=200MB
tempPath=30MB

[OSType:Windows Vista]
os.servicePack=2+
```

运行 Prerequisite Scanner 时，它将根据机器上安装的操作系统和版本来扫描并检查不同的先决条件属性。

例如，表 7 对各个包含根据此示例进行检查的先决条件属性的节作了概述。

表 7. 扫描的 Windows 配置文件节

平台或操作系统	包含先决条件属性的节
安装了 Windows XP 及更高版本的机器	[OSType:Windows]
仅安装了 Windows Vista 的机器	[OSType:Windows] [OSType:Windows Vista]

使用了节的 UNIX 配置文件示例

此示例包含特定产品的所有平台、单个平台和操作系统版本的相应先决条件属性。

```
# Properties common to all UNIX platforms
[OSType:UNIX]
os.space.var=[dir:root=/var/ibm/common/acsi,unit:MB]1.0
os.space.usr=[dir:root=/usr/ibm/common/acsi,unit:MB]200
os.space.home=[dir:non_root=USERHOME/.acsi_HOST,unit:MB]200
os.space.tmp=30MB
env.classpath.derbyJAR=False
network.pingSelf=True

# Properties common to all Linux platforms
[OSType:LINUX]
os.shell.default=bash
os.SELinux=[source:Command]Disabled
os.package.rpm=rpm

# Properties common to Linux platforms with the ppc64 CPU architecture
[OSType:LINUX][CPUArch:ppc64]
os.package.vacpp.rte=vacpp.rte-9.0.0-5+

# Properties common to all RedHat OS
[OSType:RedHat]
env.classpath.derbyJAR=False

# Properties common to all versions of Red Hat Enterprise
# Linux Server OS
[OSType: RedHatEnterpriseLinuxServer]
network.pingLocalhost=True

# Properties common to all Red Hat Enterprise Linux Server
# OS Version 6.x(6.1,6.2...)
[OSType: RedHatEnterpriseLinuxServer6.*]
os.package.compat-libstdc++-33=compat_libstdc++_33-3.2.3-68

[OSType:RedHatEnterpriseLinuxServer5.*]
os.package.compat-libstdc++-33=compat_libstdc++_33

# Properties common to all Red Hat Enterprise Linux Server
# Version 4.x(6.1,6.2...) OS and for Itanium  family CPU
[OSType:RedHatEnterpriseLinuxServer4.*][CPU:Itanium]
os.package.ia32el=ia32el-1.1-20

# Properties common to all Red Hat Enterprise Linux Server
```

```
# Version 4.x(6.1,6.2...) OS and for a 64-bit OS architecture
[OSType:RedHatEnterpriseLinuxServer4.*][OSArch:64-bit]
os.package.libgcc=libgcc-3.4.3-9

# Properties specific to RedHatEnterpriseLinuxServer5.2 OS
[OSType:RedHatEnterpriseLinuxServer5.2]
network.availablePorts.Derby=4130

# Properties specific to a 64 bit SUSE Linux Enterprise Server 11 OS
[OSType:SUSELinuxEnterpriseServer11][OSArch:64-bit]
os.package.libstdc++33-32bit=libstdc++33_32bit-3.3.3-11.9

# Properties specific to a 64 bit SUSE Linux Enterprise Server 11 OS
# and if the environment variable TPAE_DB_Server is set to 'True'
[OSType:SUSELinuxEnterpriseServer11][@TPAE_DB_Server:True]
os.package.libstdc++31-32bit=libstdc++31_32bit

# Properties specific to a 64 bit SUSE Linux Enterprise Server 11 OS
# and if the environment variables TPAE_DB_Server and TPAE_DIR_Server
# are set to 'True'
[OSType:SUSELinuxEnterpriseServer11][@TPAE_DB_Server:True]
[@TPAE_DIR_Server:True]
os.package.libstdc++34-32bit=libstdc++34_32bit

# Properties common to all AIX platforms
os.ulimit=[type:filesizelimit]unlimited
os.ulimit=[type:filedescriptorlimit]8192+,unlimited
os.FreePagingSpace=4GB+

# Properties specific to AIX 5.3.0.0 and
# if the environment variables TPAE_DB_FEATURE or TPAE_DIR_FEATURE
# are set to 'True'
[OSType:AIX5.3.0.0][@TPAE_DB_FEATURE:True|@TPAE_DIR_FEATURE:True]
os.lib.xlC.aix50.rte=xlC.aix50.rte.9.0.0.8+
```

运行 Prerequisite Scanner 时，它将根据机器上安装的操作系统和版本来扫描并检查不同的先决条件属性。

例如，第 18 页的表 7 对各个包含根据此示例进行检查的先决条件属性的节作了概述。

表 8. 扫描的 UNIX 配置文件节

操作系统和版本	包含先决条件属性的节
安装了 64 位 SUSE Linux Enterprise Server 11 的机器	[OSType:UNIX] [OSType:LINUX] [OSType:LINUX][CPUArch:ppc64] [OSType:SUSE Linux Enterprise Server 11] [OSArch:64-bit]
安装了 Red Hat Enterprise Linux Server 6.3 的机器	[OSType:UNIX] [OSType:LINUX] [OSType:RedHat] [OSType:RedHatEnterpriseLinuxServer] [OSType:RedHatEnterpriseLinuxServer6.*]
安装了 SUSE Linux Enterprise Server 11 且环境变量 @TPAE_DB_Server 设置为 true 的机器	[OSType:UNIX] [OSType:LINUX] [OSType:SUSELinuxEnterpriseServer11][@TPAE_DB_Server:True]
安装了 AIX 5.3.0.0 且环境变量 @TPAE_DB_FEATURE 或 @TPAE_DIR_FEATURE 设置为 True 的机器	[OSType:UNIX] [OSType:AIX] [OSType:AIX5.3.0.0][@TPAE_DB_FEATURE:True|@TPAE_DIR_FEATURE:True]

Prerequisite Scanner 收集器

IBM Prerequisite Scanner 收集器根据您为所要安装的产品设置的先决条件属性来收集关于当前环境的实际数据。收集器通过本机代码来获取数据。这些数据可以是公共数据，例如处理器速度和内存量、有关已安装的软件的数据、操作系统数据、用户数据、网络数据和连通性数据。另外，收集器也可扩展，因此您可以创建定制收集器以获取定制先决条件属性的实际值。

Prerequisite Scanner 使用下列语言的收集器，具体视您的平台而定:

- Windows: 具有 `.vbs` 扩展名的 VBScript
- UNIX: 具有 `.sh` 扩展名或不具有扩展名的 Shell

注: 即使在 Windows 机器上安装了类似于 UNIX 的环境（例如 Cygwin），也无法在 Windows 系统上运行 UNIX 脚本。

Windows 系统的收集器

Windows 系统的 VBScript 收集器在 Windows 脚本宿主环境中运行。它们使用组件对象模型来访问 Windows 环境的元素，例如 `FileSystemObject` 和 `TextStream`。

Prerequisite Scanner 运行 VBScript 收集器，以获取 Windows 环境的先决条件属性的实际值。每个收集器都可以获取一个或多个先决条件属性的数据。

对于 VBScript 收集器中的每个先决条件属性，收集器会将该先决条件属性的名称及其实际值写为标准输出。Prerequisite Scanner 将此标准输出写入临时文本文件（即 `localhost_hw.txt.`）。

您可以创建定制公共 VBScript 收集器，以收集适用于任何产品和产品版本的先决条件属性的数据。另外，也可以创建特定于产品的定制收集器，以收集适用于特定产品和产品版本的数据。

运行 Prerequisite Scanner 时，将通过在 *ips_root*`/Windows` 目录中搜索 *product_code*[_<*version*>]`.vbs` 文件按以下顺序运行收集器: 预定义的 VBScript 收集器; *ips_root*`/lib` 目录中的定制公共 VBScript 收集器; 以及特定于产品的定制 VBScript 收集器。

例如，`env.tcrhome.vbs` 文件是用于检查 Tivoli Common Reporting 的主目录环境变量的定制收集器。此文件存储在 *ips_root*`/lib` 目录中。

VBScript 收集器必须遵循下列规则:

- 定制公共 VBScript 收集器文件的命名约定

  它包含一个要提供给所有产品和产品版本（即，所有配置文件）使用的先决条件属性:

  prefix_identifier`.]property_name.vbs`

  其中:

  – *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应前缀标识。此前缀标识是某些预定义类别（例如 `env`）所必需。

  – *property_name* 是先决条件属性名，例如 `tcrhome`。

  请将此类 VBScript 收集器存储在 *ips_root*`/lib` 目录中。

- 特定于产品的定制 VBScript 收集器文件的命名约定

  它包含要提供给特定产品和产品版本（即，某个配置文件）使用的属性:

  product_code[_<*version*>].vbs

  其中:

  – *product_code*

    这是用于在 Windows 或 UNIX 系统上表示产品代码的变量。产品代码用于标识产品、单个平台（例如 Windows、AIX、HP-UX、Linux 和 Solaris 以及该产品支持的操作系统版本（可选）。它们存储在 codename.cfg 文件中。任何支持多个平台的产品都有多个产品代码，并且每个产品代码都根据需要标识产品、平台和操作系统版本。

  – <*version*> 是表示版本、发行版、修订版和级别的 8 位代码（此代码的每个部分各占两位）；例如，7.3.21 为 07032100。

  请将此类 VBScript 收集器存储在 *ips_root*/Windows 目录中。

- 各个先决条件属性的标准输出如下所示:

  WScript.Echo "*property_name*=" & <*var_for_value*>

  – *property_name* 表示在配置文件中编写的先决条件属性，例如

    env.tcrhome.

  – *var_for_value*，即收集器针对先决条件属性获取的实际值的 VBScript 变量。

  例如，以下标准输出将写 Tivoli Common Reporting 主目录环境变量的先决条件属性及其实际值:

  WScript.Echo "env.tcrhome=" & tcr_home

UNIX 系统的收集器

UNIX 系统的收集器在 AIX、HP-UX、Linux 或 Solaris 的相应 Shell 主机环境中运行。它们使用特定于该平台的命令和选项来访问主机环境的元素。

每个 UNIX 收集器都获取先决条件属性的数据，或者获取具有预定义子类型的先决条件属性的数据。收集器将先决条件属性的检查结果写为标准输出。Prerequisite Scanner 将此标准输出写入临时文本文件。

您可以创建定制 UNIX 收集器，以收集定制先决条件属性的数据。在 *ips_root*/UNIX_Linux/packageTest.sh 文件中，对每个收集器（预定义收集器或定制收集器）进行了调用。

运行 Prerequisite Scanner 时，它将按以下顺序运行收集器: *ips_root*/lib 目录中文件名包含 _plug 的预定义收集器，*ips_root*/UNIX_Linux 目录中的预定义收集器，以及 *ips_root*/UNIX_Linux 目录中的定制 UNIX 收集器。

例如，installedSoftware.TCR.version 文件是一个定制收集器，用于获取机器上安装的 Tivoli Common Reporting 的版本。它存储在 *ips_root*/UNIX_Linux 目录中。

UNIX 收集器必须遵循下列规则:

- 不具有文件扩展名的定制 UNIX 收集器文件的命名约定:

  [*prefix_identifier*.]*property_name*

其中:

– *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应标识。此前缀标识是某些预定义类别（例如 installedSoftware）所必需。

– *property_name* 是先决条件属性的名称，例如 TCR.version。

请将收集器存储在 *ips_root*/UNIX_Linux 目录中。请确保它不具有文件扩展名。

- 返回了先决条件属性实际值（如果该属性是整数或字符串）的先决条件属性标准输出；例如，已装配的文件系统的软件版本或可用磁盘空间量。另外，也可以返回 "Unavailable"。

```
echo "True"|"False"
'If the scan checks for the existence of the prerequisite
'property
echo $res
'If the scan checks returns the value, for example, product version,
'of the prerequisite property
echo "Unavailable"
'If the scan returns no value for the prerequisite property
echo "Available"
'If the scan returns a valid check for the prerequisite property
```

- 以下是用于在 *ips_root*/UNIX_Linux/packageTest.sh 脚本中调用并运行收集器的代码。

```
res=`echo $line | grep installedSoftware.TCR.version`
if [ $res ]; then
ExpValue=`echo $res | cut -d "=" -f2`

 echo "\`wrlTrace "Starting" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
 echo "\`wrlTrace "Executing" "installedSoftware.TCR.version"\`" >>/tmp/prs.check
 echo "\`wrlDebug "Starting" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
 echo "\`wrlDebug "Expected" "ExpValue" \`" >>/tmp/prs.check


 echo "ss=\`./installedSoftware.TCR.version\`" >>/tmp/prs.check
 echo "\`wrlTrace "Finished" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
 echo "echo \"os.userLimits=\$ss\"" >>/tmp/prs.check
 echo "\`wrlDebug "Finished" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
 echo "\`wrlDebug "OutPutValueIs" \$ss\`"  >>/tmp/prs.check
 echo "\`wrlTrace "Done" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
fi
```

Prerequisite Scanner 评估程序

IBM Prerequisite Scanner 评估程序是一些脚本，用于对来自收集器的实际数据与配置文件中相同属性的预期数据进行比较。评估可以是：特定于平台；基于简单的运算符，例如小于、等于或大于；以及基于属性是否已安装、存在或启用。另外，它们还可以验证端口是在使用中还是可用，并可以验证机器的连通性状态。您可以创建或编辑评估程序。

Prerequisite Scanner 使用下列语言的评估程序，具体视您的平台而定:

- Windows: 扩展名为 .vbs 的 VBScript
- UNIX: 扩展名为 .sh 的 Shell

注: 即使在 Windows 机器上安装了类似于 UNIX 的环境（例如 Cygwin），也无法在 Windows 系统上运行 UNIX 脚本。

评估程序存储在 *ips_root*/*OS* 中，其中 *OS* 是操作系统的名称，例如 Windows 或 UNIX_Linux。

评估程序文件必须遵循下列规则:

- 文件名的命名约定:

  [*prefix_identifier*.]*property_name*[.*suffix_identifier*]_compare.vbs|sh

  其中:

  – *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应标识。此前缀标识是某些预定义类别所必需。
  – *property_name* 是先决条件属性的名称。
  – *suffix_identifier* 是第 6 页的表 4 中概述的先决条件属性子类型的相应可选标识。
- （可选）向脚本传递两个输入参数以进行复杂评估:
  – *expected_value*，即配置文件中设置的先决条件属性的期望值。
  – *actual_value*，即收集器在机器上发现的先决条件属性实际值。
- 标准输出如下所示:
  – "PASS"（如果先决条件属性的期望值等于或大于其实际值）。
  – "FAIL"（如果先决条件属性的期望值与其实际值不等）。

输出格式

IBM Prerequisite Scanner 生成以下屏幕和人类可读文件格式的输出: 发送到命令行界面、调试和跟踪日志文件以及文本和 XML 结果文件的输出。

Prerequisite Scanner 将扫描结果和日志文件保存到 *ips_output_dir* 目录中。您可以在运行 Scanner 时使用 **outputDir** 输入参数来设置此目录。如果未设置此参数，那么缺省输出位置为 *ips_root*。

注: Prerequisite Scanner 在其执行期间会创建临时文件，但是 Scanner 在完成其执行前会将这些文件删除。这些临时文件位于 *ips_output_dir*/temp 子目录中。除非 *ips_output_dir*/temp 子目录仅包含在 UNIX 系统上生成的调试文件和跟踪文件，否则 Scanner 还将删除该子目录。

如果您选择从 CD、DVD 或只读的网络驱动器运行 Prerequisite Scanner，那么也可以使用此参数来指定位置。

要点: 如果输出目录不存在，那么 Prerequisite Scanner 将创建该目录。您必须有权创建或写入 Prerequisite Scanner 用于保存文件的输出目录。

命令行界面输出

在设置了可选的 **detail** 参数的情况下运行 Prerequisite Scanner 脚本时，Prerequisite Scanner 将在命令行界面中显示详细的扫描结果。详细结果包含:

- Prerequisite Scanner 的版本
- 在其中运行 Scanner 的操作系统的版本
- 扫描类型和场景
  – 先决条件扫描: 场景: 先决条件扫描
- 对其运行先决条件或运行状况检查的产品或组件的名称
- 对于每个先决条件属性: 所检查的属性的名称、PASS 或 FAIL 结果、实际值和期望值
- 对于所有组件: 所检查的常规属性的名称、PASS 或 FAIL 结果、实际值和期望值

- 整体 PASS 或 FAIL 结果，以及导致整体扫描失败的单项检查的任何故障

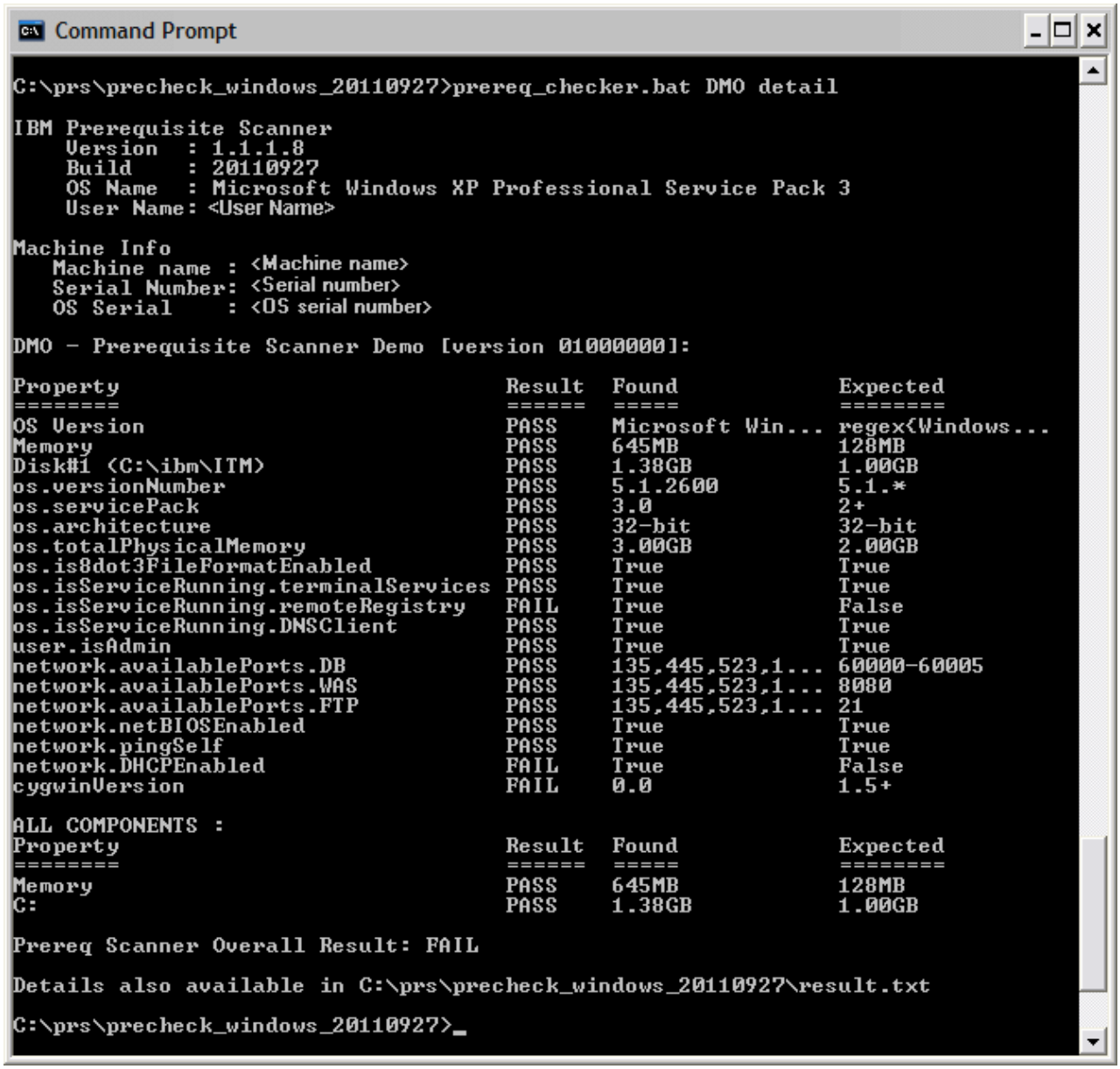

图 1. 在 Windows 系统上发送到命令行界面的输出

如果未设置 **detail** 参数，那么 Scanner 仅在屏幕上显示 PASS 或 FAIL 结果。

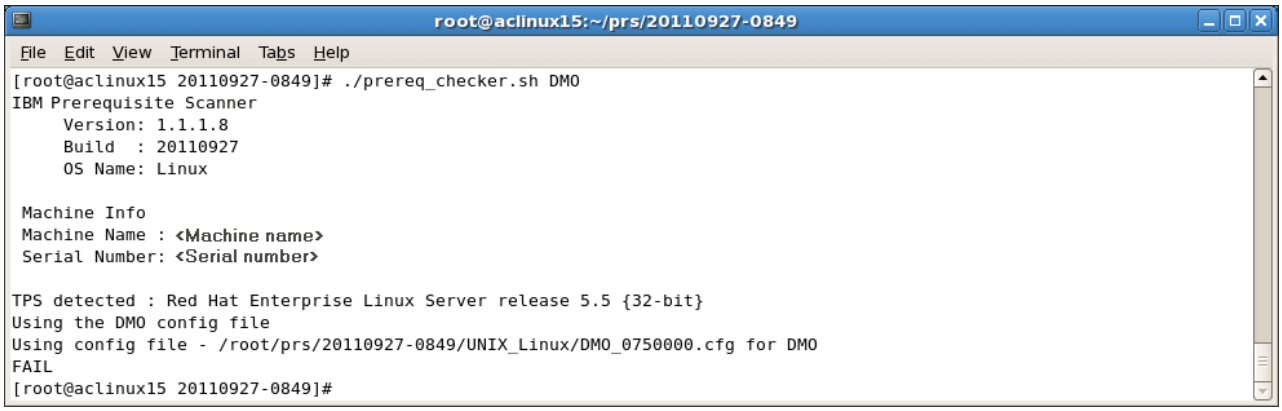

图 2. 在 UNIX 系统上发送到命令行界面的输出

Prerequisite Scanner 根据扫描结果以及它是否必须因为错误而退出来生成返回码。这些返回码将写入日志文件。例如，如果 Prerequisite Scanner 由于无法读取配置文件而无法运行扫描，那么它将生成返回码 2。

聚集内存和磁盘空间先决条件属性

通过指定多个产品代码作为输入参数，可以运行 Prerequisite Scanner 以同时检查一个或多个产品或组件的先决条件。如果在任何配置文件中指定了相关联的先决条件属性，那么 Prerequisite Scanner 将在输出的下列聚集节中聚焦内存和磁盘空间先决条件检查结果:

- UNIX 系统: `TOTAL ALL SPECIFIED COMPONENTS` 节
- Windows 系统: `ALL COMPONENTS` 节

常规先决条件属性如下所示:

- 目标环境中当前可用的总物理内存量，即:

  `Memory`
- 下列先决条件属性的文件系统磁盘空间量:

平台	先决条件属性
UNIX 和 Linux	• `os.space.home` • `os.space.opt` • `os.space.tmp` • `os.space.usr` • `os.space.var`
Windows	`Disk`

注: 如果尚未将 `opt`、`usr` 和 `var` 定义为目标计算机上的文件系统，那么 Prerequisite Scanner 不会在聚集的节中显示这些先决条件属性的期望值和返回值。

如果配置文件中不存在内存和磁盘空间先决条件属性，那么 Prerequisite Scanner 不会显示聚集的节。

Prerequisite Scanner 在扫描结果的聚集节中进行的磁盘空间值比较和显示处理与主节不同。请参阅第 31 页的『输出中的计量单位』。

Windows 系统上的调试日志文件输出

Prerequisite Scanner 在 *ips_output_dir*/precheck.log 文件中输出处理信息、警告和错误消息以及扫描结果。在设置了可选 **debug** 参数的情况下运行 Prerequisite Scanner 脚本时，Prerequisite Scanner 将在此文件中输出附加的调试消息。

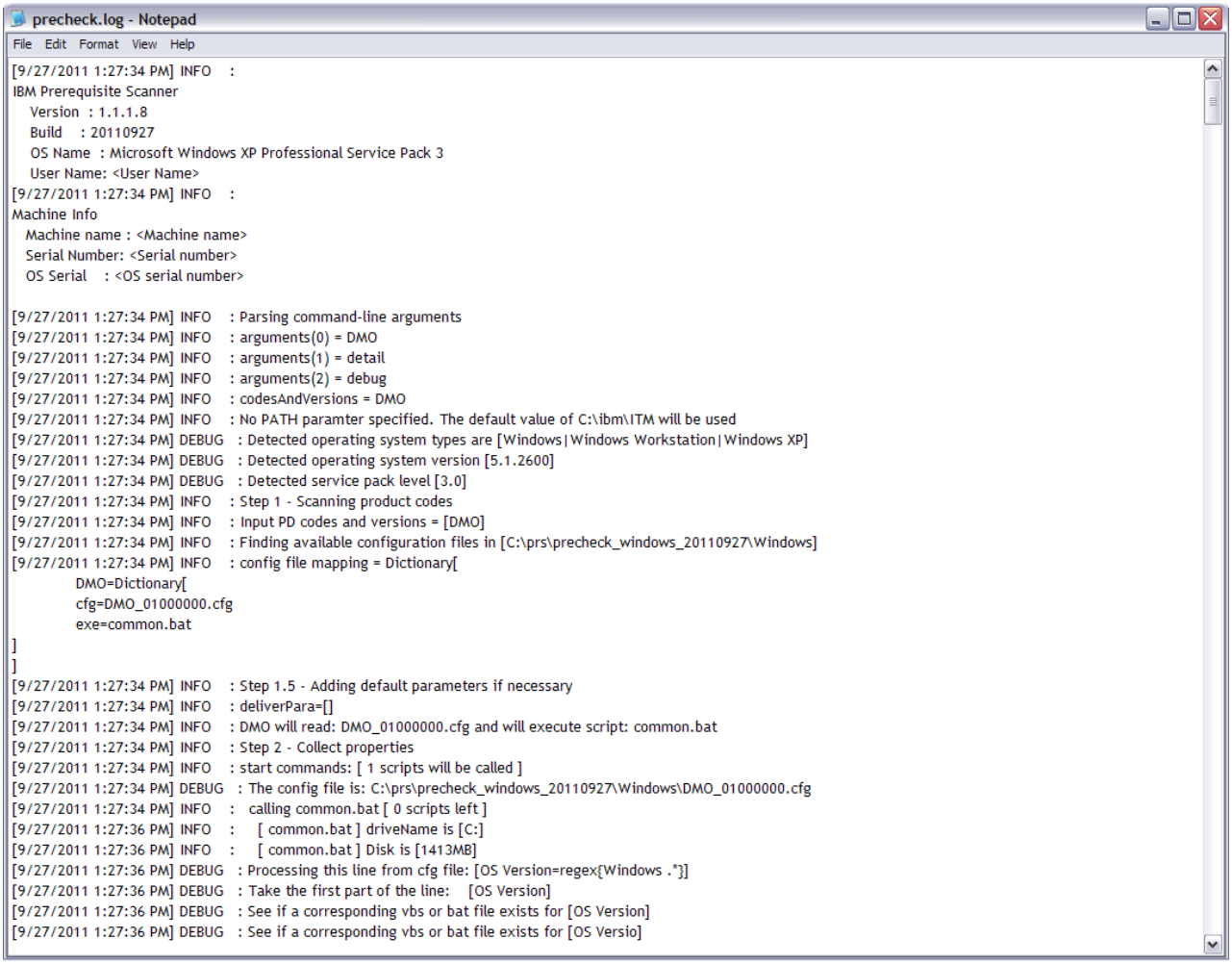

图 3. precheck.log 文件

UNIX 系统上的跟踪和调试日志文件输出

如果在设置了可选的 **debug** 参数的情况下运行 Prerequisite Scanner 脚本，那么 Prerequisite Scanner 会将详细的处理信息、警告消息和错误消息以及扫描结果输出到 *ips_output_dir*/temp/prs.debug 文件中。

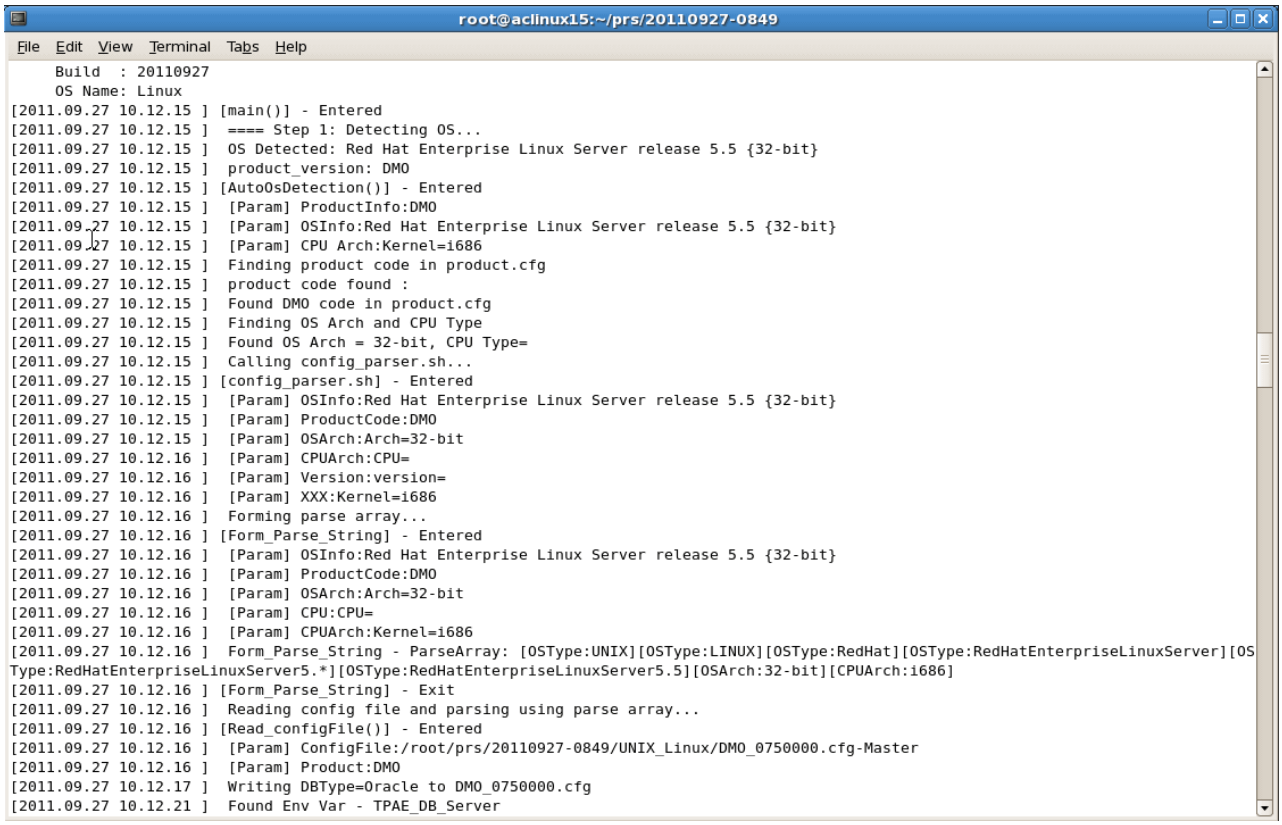

图 4. UNIX 系统上的 prs.debug 文件

如果在设置了可选的 **trace** 参数的情况下运行 Prerequisite Scanner 脚本，那么 Prerequisite Scanner 会将跟踪信息输出到 *ips_output_dir*/temp/prs.trc 文件中。

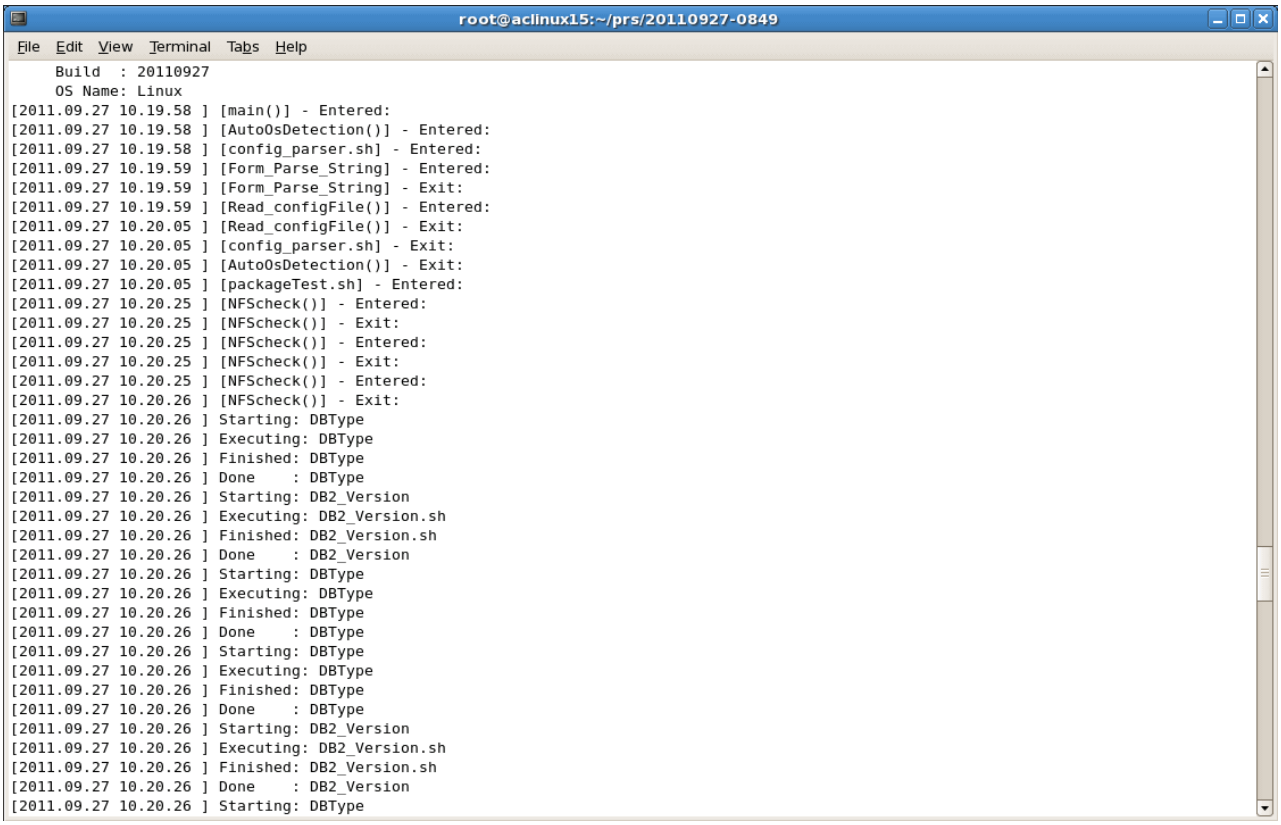

图 5. UNIX 系统上的 prs.trc 文件

文本文件输出

Prerequisite Scanner 在 *ips_output_dir*/result.txt 文件中输出详细的扫描结果。无论您是否设置了 **detail** 参数，结果都将保存到文本文件中。

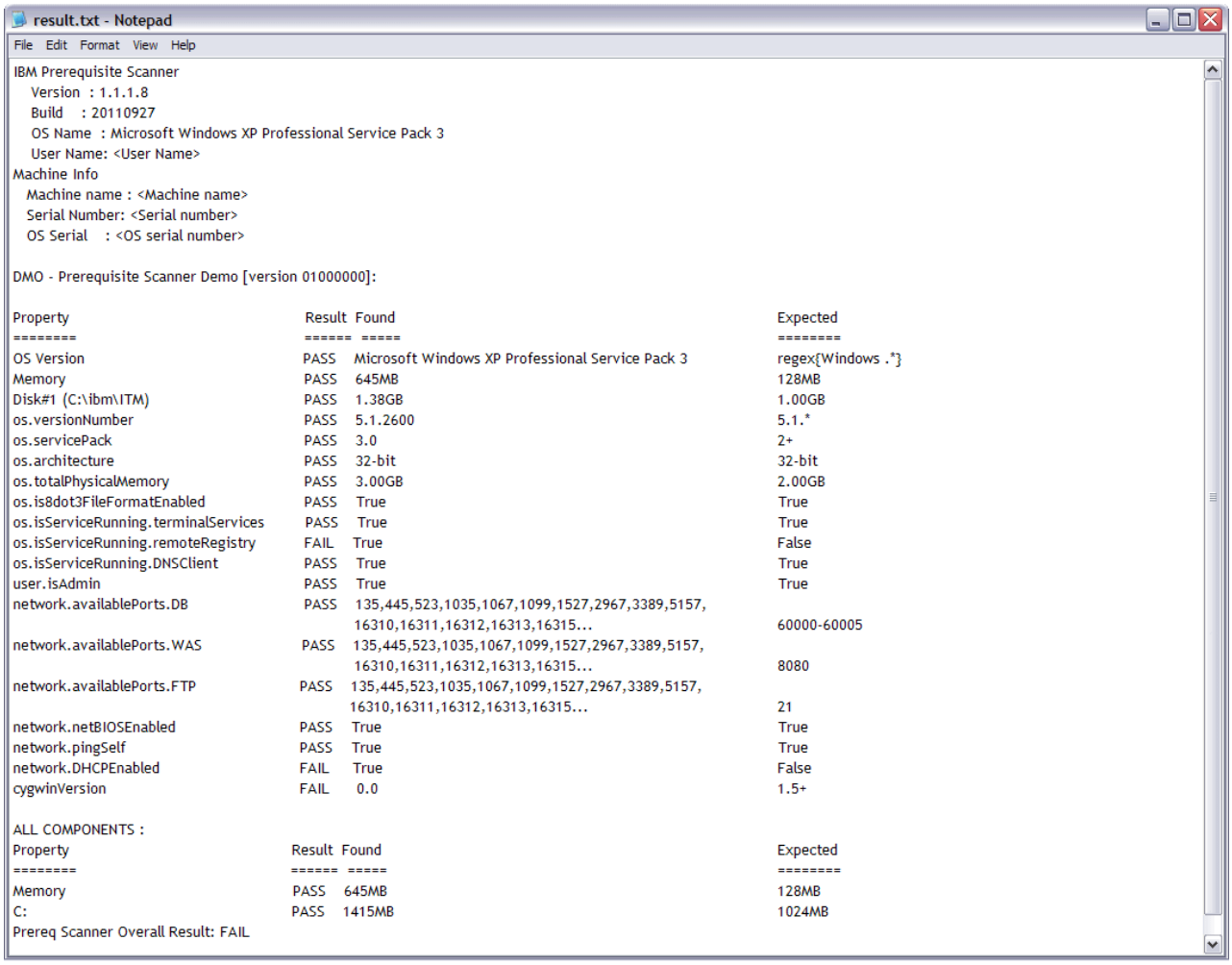

图 6. Windows 系统上的 result.txt 文件

```
[root@aclinux15 20110927-0849]# cat result.txt
IBM Prerequisite Scanner
     Version: 1.1.1.8
     Build  : 20110927
     OS Name: Linux

 Machine Info
 Machine Name : <Machine name>
 Serial Number: <Serial number>

DMO - Prerequisite Scanner Demo [0750000]:
Evaluation                              PASS/FAIL       Result                                  Expected Result
DBType                                  FAIL            Unknown                                 Oracle
DBType                                  FAIL            Unknown                                 DB2
DBType                                  FAIL            Unknown                                 regex{.*Oracle.*}
DBType                                  FAIL            Unknown                                 regex{.*DB2.*}
DBTypeDetails                           FAIL            Unknown                                 oracle
DBTypeDetails                           FAIL            Unknown                                 DB2
DBTypeDetails                           FAIL            Unknown                                 regex{.*Oracle.*}
DBTypeDetails                           FAIL            Unknown                                 regex{.*DB2.*}
OS Version                              PASS            "Red Hat Enterprise Linux Server release 5.5 (Tikanga)" "regex{R
ed Hat.*Tikanga.*}"
                                                                                                regex{AIX.*}
                                                                                                regex{Solaris.*}
os.lib.libstdc++                        PASS            /usr/lib/gcc/i386-redhat-linux/4.1.1/libstdc++.solibstdc++

os.lib.libgcc                           PASS            /usr/lib/gcc/i386-redhat-linux/3.4.6/libgcc_s.so[CheckPackage:Tr
ue]regex{libgcc.*}
os.lib.libXp                            PASS            /usr/lib/libXmu.so.6                    regex{libX.*}
os.space.var                            PASS            "38GB"                                   "[dir:root=/var/ibm/
common/acsi"
                                                                                                unit:MB]1.0
os.space.usr                            PASS            "38GB"                                   "[dir:root=/usr/ibm/
common/acsi"
                                                                                                unit:MB]200
os.space.tmp                            PASS            36GB                                    30MB
env.classpath.derbyJAR                  PASS            False                                   False
network.pingSelf                        PASS            True                                    True
env.classpath.derbyJAR                  PASS            False                                   False
```

图 7. UNIX 系统上的 result.txt 文件

XML 文件输出

如果您指定了可选的 **xmlResult** 输入参数，那么 Prerequisite Scanner 将在 *ips_output_dir*/result.xml 文件中输出详细的扫描结果。您可以使用此参数指示工具不仅将结果输出到纯文本测试结果文件，还输出到 XML 结果文件。无论您是否设置了 **detail** 参数，结果都将保存到 XML 文件中。

```
<PRSInfo>

<MachineInfo>
    <MachineName>my_machine_name</MachineName>
    <MachineSerialNumber>serial_number</MachineSerialNumber>
    <MachineOSSerial>os_serial_number</MachineOSSerial>
    <MachineOSName>Microsoft Windows XP Professional Service Pack 3</MachineOSName>
</MachineInfo>

<UserInfo>

<ProductInfo>
    <ProductElement>
        <ProductCode>DMO</ProductCode>
        <ProductName>Prerequisite Scanner Demo</ProductName>
        <ProductVersion>01000000</ProductVersion>
    </ProductElement>
</ProductInfo>

<DetailedResults>
    <DetailedProductResultsElement>
        <ProductCode>DMO</ProductCode>
        <ResultElement>
            <PropertyName>OS Version</PropertyName>
            <Result>FAIL</Result>
            <Found>Microsoft Windows XP Professional Service Pack 3</Found>
            <Expected>Windows 7 Ultimate</Expected>
        </ResultElement>
        <ResultElement>
            <PropertyName>Memory</PropertyName>
            <Result>PASS</Result>
            <Found>960MB</Found>
            <Expected>128MB</Expected>
        </ResultElement>
        <ResultElement>
            <PropertyName>Disk#1 (C:\ibm\ITM)</PropertyName>
            <Result>PASS</Result>
            <Found>22072MB</Found>
            <Expected>1GB</Expected>
        </ResultElement>
        <ResultElement>
            <PropertyName>os.versionNumber</PropertyName>
            <Result>FAIL</Result>
            <Found>5.1.2600</Found>
            <Expected>5.2.*</Expected>
        </ResultElement>
```

图 8. Windows 系统上的 result.XML 文件

开发者可以使用 Prerequisite Scanner Java™ Developer 工具箱来解析和读取 XML 文件。

输出中的计量单位

Prerequisite Scanner 在扫描结果的聚集节中进行的磁盘空间值比较和显示处理与主节不同。

在扫描结果的主节中，Prerequisite Scanner 按如下方式处理磁盘空间值的比较和显示:

平台	先决条件属性
UNIX 和 Linux	如果实际值大于 1024 MB，那么 Prerequisite Scanner 对该值进行转换并返回以 GB 为单位的值；否则，它将返回以 MB 为单位的值。
Windows	Prerequisite Scanner 使用配置文件中的期望值单位作为比较和显示单位。有需要时，它会将实际值转换为以此单位表示。

在扫描结果的聚集节中，Prerequisite Scanner 按如下方式处理磁盘空间值的比较和显示:

平台	先决条件属性
UNIX 和 Linux	如果实际值大于 1024 MB，那么 Prerequisite Scanner 对该值进行转换并返回以 GB 为单位的值；否则，它将返回以 MB 为单位的值。
Windows	Prerequisite Scanner 以 MB 单位为基础对磁盘空间量进行比较。对于每个包含 Disk 先决条件属性的配置文件，Prerequisite Scanner 将实际值和期望值转换为以 MB 为单位并执行比较。为了进行显示，如果聚集的实际值大于 1 GB，那么 Prerequisite Scanner 将返回以 GB 为单位且精度为 2 的实际值；否则，它将返回以 MB 为单位的值。

Prerequisite Scanner Java Developer 工具箱

Prerequisite Scanner Java Developer 工具箱是一组 API，这些 API 使开发者能够根据需要以编程方式解析和读取结果 XML 文件的内容；例如，解析扫描结果以便在安装程序中使用。

此工具箱提供了下列包:

- com.ibm.prs.common.exception

  包含 PRSApiException 类，这个类提供了针对 XML 查询 API 抛出异常的方法。
- com.ibm.prs.common.reports.api

  包含 PRSXmlResultReader 接口，此接口定义了针对 XML 结果文件的 XML 查询 API。
- com.ibm.prs.common.reports.api.impl

  包含 PRSXmlResultReaderImpl 类，这个类实现了 PRSXmlResultReader。

Prerequisite Scanner 可以根据 *ips_root*/PRSResults.xsd XML 模式文件来验证格式和结构。

工具箱的 Javadoc 在 *ips_root*/api/javadoc 目录中提供。

用于 XML 结果文件的 XML 模式文件

Prerequisite Scanner 提供了一个可针对其验证结果 XML 文件的 XML 模式文件。

XML 模式文件包含以下表示各个节的元素:

- PRSInfo，用于管理 Prerequisite Scanner 详细信息
- MachineInfo，用于管理有关对其运行扫描的目标环境的信息
- UserInfo，用于管理有关运行扫描的登录用户的信息
- ScenarioInfo，用于管理有关扫描类型和场景的信息
- ProductInfo，用于管理有关产品或组件及其配置文件的信息

- DetailedResults，用于管理产品或组件的每组先决条件属性的扫描结果（这些结果由 DetailedProductResultsElement 进行分组）
- AggregateResults，用于管理磁盘空间和内存的聚集扫描结果
- OverallResult，用于管理扫描的整体 PASS 或 FAIL 结果

XML 模式的名称和位置为：*ips_root*/PRSResults.xsd

作为开发者或部署者，您可以从查询 XML API 调用方法以验证结果 XML 文件。工具箱的 Javadoc 在 *ips_root*/api/javadoc 目录中提供。

扫描过程

运行 IBM Prerequisite Scanner 时，它将在扫描过程的每个阶段执行一组任务。用户可以打开命令行界面，然后运行 Prerequisite Scanner 脚本并指定一组包括产品代码在内的输入参数。

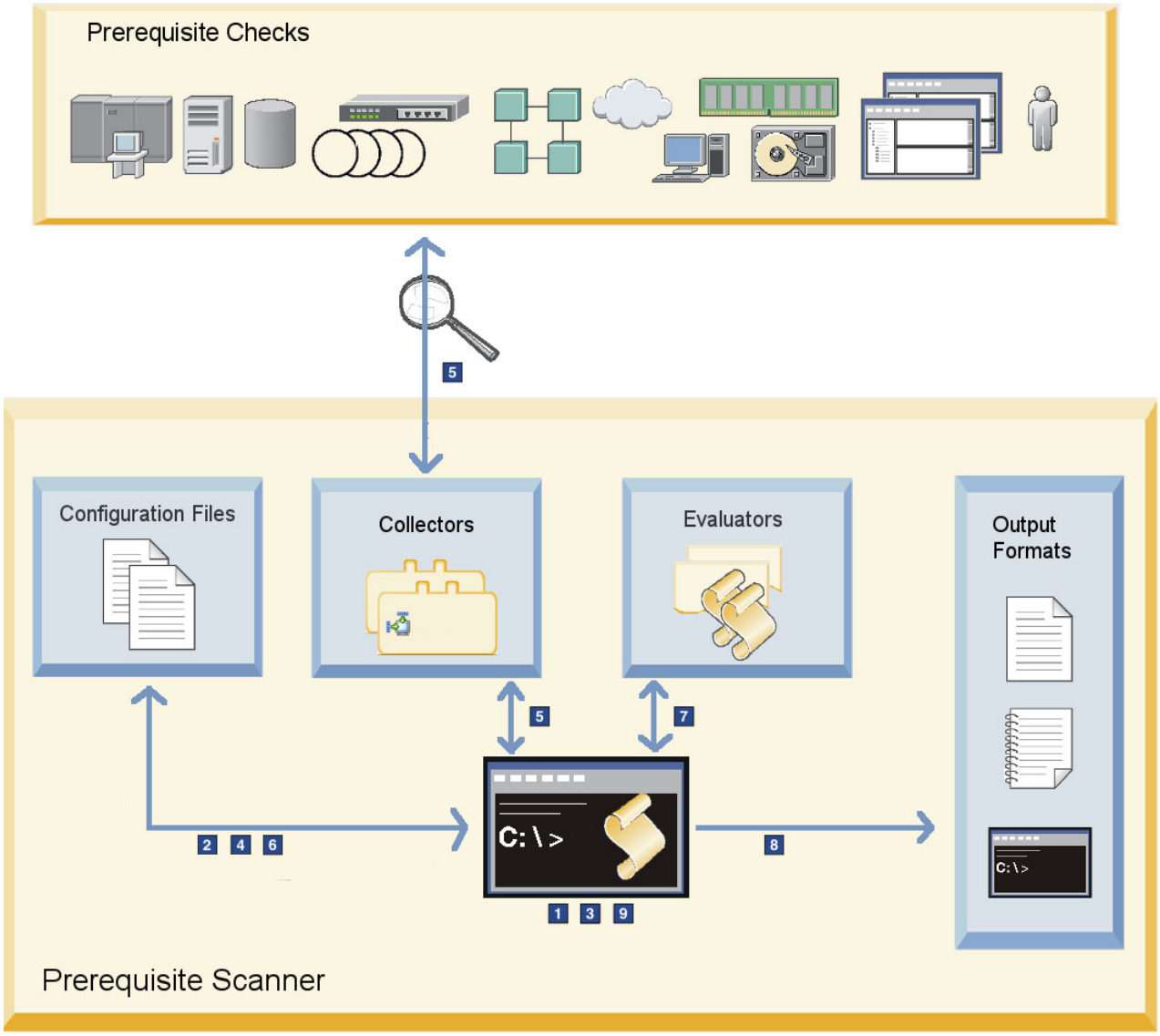

图 9. Prerequisite Scanner 体系结构和扫描过程

图 9 中的扫描过程总结如下:

1. Prerequisite Scanner 验证传递到 Scanner 的输入参数的格式。
2. Scanner 确定作为其中一个输入参数传递的产品代码是否为 `codename.cfg` 文件中的有效产品代码。
3. Scanner 将搜索与产品代码相关联的配置文件。如果未传递可选的产品版本参数，那么 Scanner 将使用 *ips_root*`/Windows|UNIX_Linux` 目录中的最新版本配置文件。
4. Scanner 验证机器的实际操作系统是否为受支持的操作系统。Scanner 根据配置文件（其文件名包含与输入参数相同的产品代码和产品版本）的节标题中的期望受支持操作系统来检查实际操作系统。
5. Scanner 通过使用 Prerequisite Scanner 收集器来收集实际先决条件属性以进行先决条件检查。
6. Scanner 将检查与产品代码和产品版本相关联的配置文件中的先决条件属性。

Scanner 根据配置文件（其文件名包含与输入参数相同的产品代码和产品版本）的操作系统版本先决条件属性或节标题中的期望受支持操作系统来检查实际操作系统。

7. Scanner 从配置文件中读取先决条件属性，并分析这些先决条件属性的实际值和期望值，以便执行先决条件检查。必要时，它将使用 Prerequisite Scanner 评估程序。
8. Scanner 将扫描结果输出到命令行界面、结果文本文件和 XML 文件以及人类可读日志文件中。
9. Scanner 将清空并除去临时文件和目录。

本发行版中的新增内容

IBM Prerequisite Scanner V1.2 提供了新的属性和增强功能。另外，还包含对缺陷的修订。

本修订版中的新增功能部件

能够解析并读取新的扫描结果 XML 文件。

Prerequisite Scanner Java Developer 工具箱是一组 API，这些 API 使开发者能够根据需要以编程方式解析和读取结果 XML 文件的内容；例如，解析扫描结果以便在安装程序中使用。请参阅第 32 页的『Prerequisite Scanner Java Developer 工具箱』。

本修订版中的新增配置文件

表 9 对 Prerequisite Scanner V1.2 随附的新配置文件和产品代码作了概述

表 9. 新增配置文件

产品或组件	产品代码	配置文件
Tivoli Composite Application Manager Agent for WebSphere MQ	KMQ	*ips_root*/Windows|UNIX_Linux/KMQ_07010000.cfg
Tivoli Composite Application Manager Agent for WebSphere Message Broker	KQI	*ips_root*/Windows|UNIX_Linux/KQI_07010000.cfg

本修订版中的新增先决条件属性

添加了 os.SeaMonkeyVersion 属性，用于检查机器上的 Mozilla SeaMonkey 的版本。请参阅第 89 页的『操作系统数据属性』。

添加了 env.var.set.*env_var_name* 属性，用于确定机器上是否设置了 *env_var_name* 所指定的环境变量。请参阅第 100 页的『环境变量数据属性』。

本修订版中的增强功能

能够将扫描结果写入 XML 文件。

ips_output_dir/result.xml 是 XML 格式的新扫描结果文件。在缺省情况下，工具仅将结果输出到纯文本结果文件中。请参阅第 23 页的『输出格式』。

在 Prerequisite Scanner V1.2 中，**xmlResult** 是 Prerequisite Scanner 脚本的新可选输入参数。您可以使用此参数指示工具不仅将结果输出到纯文本测试结果文件，还输出到 XML 结果文件。请参阅第 57 页的『prereq_checker』。

如果配置文件中不存在内存和磁盘空间先决条件属性，那么将除去结果中的聚集节。

如果配置文件中不存在内存和磁盘空间先决条件属性，那么 Prerequisite Scanner 将不会继续在结果文件中显示聚集的节。请参阅第 23 页的『输出格式』。

本修订版中不推荐使用的功能部件

无

本修订版中修正的缺陷

要获取本发行版中修正的缺陷的列表，请在解压缩 Prerequisite Scanner 软件包的内容后参阅 *ips_root* 目录中的 `Readme.html` 文件。

本修订版中的文档更改

《Prerequisite Scanner 用户指南》不再与 Prerequisite Scanner 的 Prerequisite Scanner 平台软件包捆绑在一起。您可以使用 IBM Prerequisite Scanner 信息中心。

第 2 章 安装 Prerequisite Scanner

IBM Prerequisite Scanner 没有安装程序。将压缩文件的内容解压缩时，核心文件将保存在包含下列子目录的根目录中: /api（存放用于支持查询 XML API 的 Prerequisite Scanner Java Developer 工具箱）； /lib（存放收集器和公共脚本）; /Windows（存放 Windows 系统上的评估程序和配置文件）; /UNIX_Linux（存放 UNIX 平台上的评估程序和配置文件）; 以及 /licenses（存放许可证文件）。

先决条件

IBM Prerequisite Scanner 可以在 Windows 系统、Windows XP 或更高版本（32 位或 64 位）上运行。另外，它还可以在 AIX、HP-UX、Linux 和 Solaris 操作系统的变体上运行。

确保已在目标环境中安装下列实用程序或者可以使用这些实用程序:

目标系统	先决条件
Windows	• 已启用 Telnet 客户机，以使预定义的“连通性”收集器中的连通性检查能够正常运行。 • 确保安装了 Microsoft .Net Framework 1.0 或更高版本，以便 Prerequisite Scanner 可以正常运行。

UNIX	• 已安装 Bash，以使 Prerequisite Scanner UNIX 收集器能够正常运行。
	• 对于非 root 用户，必须在 PATH 环境变量中设置 **mount**、**swapinfo** 和 **psrinfo** 命令的位置，以使这些命令可供 Prerequisite Scanner 使用。这些命令在 /usr/sbin 目录中；例如，请按如下所示设置 PATH 环境变量:
	`export PATH=$PATH:/usr/sbin/`
	• 确保对 **lscfg** 命令分配正确的访问许可权，包括任何由访问权标志（例如 setuid 位）设置的特定许可权。正确的访问许可权意味着 Prerequisite Scanner 可以运行命令并检索系统信息。此命令位于 /usr/sbin 目录中；例如，要为 **lscfg** 设置 setuid 位，请按如下所示运行 **chmod** 命令:
	`chmod 4777 /usr/sbin/lscfg`

Prerequisite Scanner 支持 Prerequisite Scanner 的运行所针对的指定产品或 IBM 解决方案的所有硬件和操作系统。

安装压缩文件

您可以将 IBM Prerequisite Scanner 压缩文件的内容解压缩。您必须对在其中进行压缩文件内容解压缩的根目录具有写许可权。

过程

1. 打开 Web 浏览器并输入指向 IBM Fix Central 的 URL。 确保登录到 IBM.com 或 IBM Support Portal。
2. 从**产品组**列表中选择 **Tivoli**。
3. 从**产品**列表中选择 IBM Prerequisite Scanner。
4. 从**已安装的版本**列表中，选择要下载的版本。
5. 从**平台**列表中，选择要在其上安装 Prerequisite Scanner 的平台。
6. 单击**继续**。此时将打开“查找补丁”页面。
7. 使用缺省选项**浏览修订**，然后单击**继续**。
8. 在“选择修订”页面上，选择软件包，然后单击**继续**。
9. 在“下载选项”页面上，选择下载选项，然后单击**现在下载**。
10. 将压缩文件的内容解压缩到 *ips_root* 指定的首选位置。

下一步做什么

请确保查阅产品的安装文档或技术说明，以了解任何其他必须在运行 Prerequisite Scanner 之前执行的步骤。例如，您可能需要设置环境变量，以便将目标计算机上安装的组件或功能部件以及因此需要检查的先决条件告知 Prerequisite Scanner。

卸载 Prerequisite Scanner

如果您希望安装更高的版本、将 IBM Prerequisite Scanner 移到另一环境或者它是您不再需要的版本，请将其除去。

过程

1. 打开 *ips_root* 目录。
2. 删除该目录及其内容。

第 3 章 扩展 Prerequisite Scanner

IBM Prerequisite Scanner 提供了一组可以用来运行工具并扫描先决条件的基本收集器、评估程序和配置。如果这组基本的文件、先决条件属性及值以及先决条件检查无法满足您的需求，您可以对 Prerequisite Scanner 进行扩展。

运行 Prerequisite Scanner 前的准备工作

在运行 IBM Prerequisite Scanner 之前，请确定预定义的先决条件属性、它们的期望值以及配置文件是否满足您的先决条件扫描需求。如果其中的任何一项无法满足您的需求，那么您可以执行一组先决条件任务以配置或扩展 Prerequisite Scanner。这组先决任务检查和任务取决于平台以及先决条件检查数目。

Windows 系统的必需检查和扩展任务

在运行 IBM Prerequisite Scanner 之前，您应执行一组检查和任务。这些检查确定您是可以编辑和使用现有配置文件还是必须扩展 Prerequisite Scanner。

表 10 提供了要执行的检查和任务的列表。

表 10. 使用配置文件前的检查和任务（Windows 系统）

	检查	任务
☐	检查 codename.cfg 文件中是否列示了产品、其支持的操作系统以及操作系统版本。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请将产品的产品代码、各个操作系统和可选操作系统版本添加到该文件中。有关更多信息，请参阅第 41 页的『添加产品代码』。
☐	检查是否存在与产品版本关联的产品代码的相应配置文件。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请创建一个配置文件并使其包含该操作系统及操作系统版本的先决条件属性。有关更多信息，请参阅第 41 页的『创建定制配置文件』。
☐	打开配置文件并检查它是否包含正确的先决条件属性。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请添加先决条件属性。有关更多信息，请参阅第 43 页的『添加先决条件属性』。
☐	检查先决条件属性是否具有期望值。	• 如果满足此条件，请运行 Prerequisite Scanner。有关更多信息，请参阅第 57 页的第 4 章，『运行 Prerequisite Scanner』。 • 如果未满足此条件，请编辑先决条件属性。有关更多信息，请参阅第 45 页的『编辑先决条件属性』。
☐	对于任何新的先决条件属性，请检查预定义的收集器能否收集这些先决条件属性的实际值。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请创建定制收集器。有关更多信息，请参阅第 45 页的『创建用于 Windows 系统的定制收集器』。

表 10. 使用配置文件前的检查和任务（Windows 系统）(续)

	检查	任务
☐	对于任何新的或经过编辑的先决条件属性，请检查预定义评估程序能否对该先决条件属性的期望值与实际值进行比较。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请创建定制评估程序。有关更多信息，请参阅第 52 页的『创建用于 Windows 系统的定制评估程序』。
☐	请确保所有文件都已保存在正确的目录中: • 配置文件、所有特定于产品的定制收集器和相关批处理文件以及所有定制评估程序文件位于 *ips_root*/Windows 目录中 • 定制公共收集器位于 *ips_root*/lib 目录中	运行 Prerequisite Scanner。有关更多信息，请参阅第 57 页的第 4 章，『运行 Prerequisite Scanner』。

UNIX 系统的必需检查和扩展任务

在运行 IBM Prerequisite Scanner 之前，您应执行一组先决条件检查和任务。这些检查确定您是可以编辑和使用现有配置文件还是必须扩展 Prerequisite Scanner。

表 11 提供了要执行的必需检查和任务的列表。

表 11. 使用配置文件前的检查和任务（UNIX 系统）

	检查	任务
☐	检查 codename.cfg 文件是否列示了产品。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请将产品代码添加到 codename.cfg 文件中。有关更多信息，请参阅第 41 页的『添加产品代码』。
☐	检查是否存在与产品关联的产品代码的相应配置文件。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请创建一个配置文件以包含该产品所支持的所有平台的先决条件属性。有关更多信息，请参阅第 41 页的『创建定制配置文件』。
☐	打开配置文件并检查它是否包含正确的先决条件属性。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请添加先决条件属性。有关更多信息，请参阅第 43 页的『添加先决条件属性』。
☐	检查先决条件属性是否具有期望值。	• 如果满足此条件，请运行 Prerequisite Scanner。有关更多信息，请参阅第 57 页的第 4 章，『运行 Prerequisite Scanner』。 • 如果未满足此条件，请编辑先决条件属性。有关更多信息，请参阅第 45 页的『编辑先决条件属性』。
☐	对于任何新的先决条件属性，请检查预定义的收集器能否收集这些先决条件属性的实际值。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请创建定制收集器。有关更多信息，请参阅第 49 页的『创建用于 UNIX 系统的定制收集器』。
☐	对于任何新的或经过编辑的先决条件属性，请检查评估程序能否对该先决条件属性的期望值与实际值进行比较。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请创建定制评估程序。有关更多信息，请参阅第 56 页的『创建用于 UNIX 系统的定制评估程序』。

表 11. 使用配置文件前的检查和任务（UNIX 系统）(续)

	检查	任务
☐	对于任何新的或经过编辑的先决条件属性，请检查 *ips_root*/UNIX_Linux/packageTest.sh 脚本是否包含用于调用并运行收集器的代码。	• 如果满足此条件，请执行下一项检查。 • 如果未满足此条件，请编辑主要软件包测试脚本。有关更多信息，请参阅第 50 页的『编辑用于 UNIX 系统的软件包测试脚本』。
☐	请确保所有文件都已保存在正确的目录中: • 配置文件、所有定制收集器文件和所有定制评估程序文件位于 *ips_root*/UNIX_Linux 目录中	运行 Prerequisite Scanner。有关更多信息，请参阅第 57 页的第 4 章，『运行 Prerequisite Scanner』。

添加产品代码

IBM Prerequisite Scanner 在 codename.cfg 文件中提供了一组预定义的产品版本代码。如果该文件未包含产品版本、其支持平台和操作系统版本的相应产品代码，那么您可以添加这些代码。

过程

1. 打开 *ips_root*/codename.cfg 文件。
2. 检查该文件是否已包含产品版本的名称/值对。
3. 如果不存在产品代码，请添加一个产品代码并确保使用如下所示的正确格式:

   product_code=code_value

   限制: IBM Tivoli Monitoring 和 Tivoli Composite Application Manager 具有预定义的产品代码，Prerequisite Scanner 将这些代码视为保留代码。除非这些代码引用与其相关联的 IBM Tivoli Monitoring 和 Tivoli Composite Application Manager 代理程序，否则不得用作 Prerequisite Scanner 产品代码。有关产品代码的更多信息，请参阅 ITM 6.X 产品代码技术说明。

   限制: 仅限于 UNIX: 在文件中输入产品代码的值时，请避免使用 for。这是一个保留字，可能会影响 Prerequisite Scanner 的运行方式。

   例如，要在所有 Windows 平台上添加 IBM Tivoli Monitoring for Energy Management 的产品代码，请在文件中添加下面这一行:

   ```
   MEA=IBM Tivoli Monitoring for Energy Management
   ```

创建定制配置文件

如果预定义的配置文件无法满足您的先决条件属性需求，那么您可以根据配置文件样本来创建定制配置文件。在创建定制配置文件之前，请确保了解要添加的先决条件属性及其期望值。

关于此任务

要点: 必须遵循用于管理定制配置文件的创建和编辑的命名约定和格式编排规则。如果未遵循这些规则，那么 Prerequisite Scanner 将无法成功地使用此文件来运行扫描。

过程

1. 必要时，请将产品的产品代码添加到 codename.cfg 文件中。
2. 通过使用文本编辑器，在 *ips_root*/OS 目录中创建配置文件。请确保文件名使用以下命名约定:

   product_code_version.cfg

   其中:

   - *product_code*

     这是用于在 Windows 或 UNIX 系统上表示产品代码的变量。产品代码用于标识产品、单个平台（例如 Windows、AIX、HP-UX、Linux 和 Solaris 以及该产品支持的操作系统版本（可选）。它们存储在 codename.cfg 文件中。任何支持多个平台的产品都有多个产品代码，并且每个产品代码都根据需要标识产品、平台和操作系统版本。

   - *version* 是表示版本、发行版、修订版和级别的 8 位代码（此代码的每个部分各占两位）；例如，7.3.21 为 07032100。
3. 查看第 81 页的附录 C, 『先决条件属性参考』中概述的基本先决条件属性，并确定要检查的先决条件属性。
4. 可选: 添加一节并确保节标题使用以下命名约定:

   - **预定义的单一数据类型类别**

     [*category_name*:*category_value*]

     例如，要为所有 Windows 平台的公共先决条件属性创建一节，请添加以下节标题:

     [OSType:Windows]

     例如，要为所有 RedHat Linux 操作系统变体的公共先决条件属性创建一节，请添加以下节标题:

     [OSType:RedHat]

   - **预定义的组合数据类型类别**

     [*category_name*:*category_value*]
     [*category_name*:*category_value*]

     例如，要为除 Windows Server 2003 R2 变体以外的 Windows Server 2003 变体的先决条件属性创建一节，请添加以下组合节标题:

     [OSType:Windows Server 2003][!OSType:Windows Server 2003 R2]

     例如，要为 SUSE Linux Enterprise Server 11 操作系统的先决条件属性以及环境变量 @TPAE_DB_SERVER 是否设置为 true 创建一节，请添加以下组合节标题:

     [OSType=SUSELinuxEnterpriseServer][@TPAE_DB_SERVER:true]

   其中:

   category_name 是用于表示第 15 页的表 6 中概述的数据类型类别的多字符代码

   category_value 是用于表示第 15 页的表 6 所概述类别的允许值的多字符代码
5. 可选: 对于每一节，请查看第 81 页的附录 C, 『先决条件属性参考』中概述的基本先决条件属性，并确定要检查的先决条件属性。

6. 对于每个要添加的先决条件属性，请输入名称/值对（有需要时，请指定可选的限定符）。请确保使用以下格式（每个先决条件属性各占一行）：

   ```
   [prefix_identifier.]property_name[.suffix_identifier]=
   [qualifier_name:qualifier_value]property_value
   ```

   其中:

   - *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应标识。此前缀标识是某些预定义类别所必需。
   - *property_name* 是先决条件属性的名称。
   - *suffix_identifier* 是第 6 页的表 4 中概述的先决条件属性子类型的相应可选标识。
   - *qualifier_name* 是先决条件属性的可选属性。IBM Prerequisite Scanner 使用此属性来限定先决条件属性或者对先决条件属性执行的检查类型（如第 8 页的『先决条件属性的预定义限定符』所述）。

     注: 可以指定多个以逗号分隔的限定符。这组限定符必须括在 [] 方括号中。
   - *qualifier_value* 是可选属性的值。每个限定符及其值都必须以冒号 : 定界。
   - *property_value* 是先决条件属性的值，并且可以是字符串或整数。

   例如，先决条件属性的用户预定义类别具有 user 前缀标识。用于检查登录用户是否属于管理员用户组的先决条件属性为: `user.isAdmin=True`
7. 如果某个先决条件属性在预定义的类别中不存在，请添加定制先决条件属性的名称、它的值以及可选的限定符。 然后，您必须创建下列文件以根据需要查找并比较定制先决条件属性: 用于收集先决条件属性实际值的定制收集器，以及标准比较函数无法对实际值与期望值进行比较时使用的定制评估程序。

添加先决条件属性

您可以将先决条件属性预定义类别中的基本先决条件属性添加到配置文件中。另外，还可以添加定制先决条件属性。

关于此任务

要点: 必须遵循用于管理对配置文件添加和编辑先决条件属性的格式编排规则。如果未遵循这些规则，那么 Prerequisite Scanner 将无法成功地对该先决条件属性运行扫描。

过程

1. 打开配置文件。
2. 查看第 81 页的附录 C,『先决条件属性参考』中概述的基本先决条件属性，并确定要检查的先决条件属性。
3. 对于每个要添加的先决条件属性，请输入名称/值对（有需要时，请指定可选的限定符）。

   例如，要添加公共预定义类别中的先决条件属性，请仅输入属性名称和期望值。请在文件中添加下列先决条件属性:

   ```
   Disk=1GB
   OS Version=regex{Windows 200[3-8]}
   ```

   例如，先决条件属性的网络预定义类别具有 network 前缀标识，并且，用于检查可用端口的先决条件属性名称为 availablePorts。可以按应用程序子类型（DB2 表示

DB2 数据库服务器，WAS 表示 WebSphere Application Server，FTP 表示 FTP 协议）对可用端口进行进一步分类。请在文件中添加下列先决条件属性:

```
network.availablePorts.DB2=5000-5005
network.availablePorts.WAS=9080
network.availablePorts.FTP=21
```

例如，先决条件属性的操作系统预定义类别具有 os 前缀标识，并且，用于检查文件系统的可用磁盘空间量的先决条件属性名称为 space。可以按文件系统子类型（usr 和 home）对检查进行进一步分类。可以对 dir 和 unit 限定符指定值。

请在文件中添加下列先决条件属性:

```
os.space.usr=[dir:root=/usr/ibm/common/acsi,unit:GB]2
os.space.home=[dir:non_root=USERHOME/.acsi_HOST,unit:MB]200
```

要点: 如第 9 页的表 5 所述，预定义的限定符只能与特定的预定义先决条件属性配合使用。

4. 如果某个先决条件属性在先决条件属性的预定义类别中不存在，请添加具有可选限定符的名称/值对作为定制先决条件属性和值。请确保使用以下格式（每个先决条件属性各占一行）。

   [*prefix_identifier*.]*property_name*[.*suffix_identifier*]=
   [[*qualifier_name*:*qualifier_value*]]*property_value*

   其中:

   - *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应标识。此前缀标识是某些预定义类别所必需。
   - *property_name* 是先决条件属性的名称。
   - *suffix_identifier* 是第 6 页的表 4 中概述的先决条件属性子类型的相应可选标识。
   - *qualifier_name* 是先决条件属性的可选属性。IBM Prerequisite Scanner 使用此属性来限定先决条件属性或者对先决条件属性执行的检查类型（如第 8 页的『先决条件属性的预定义限定符』所述）。

     注: 可以指定多个以逗号分隔的限定符。这组限定符必须括在 [] 方括号中。
   - *qualifier_value* 是可选属性的值。每个限定符及其值都必须以冒号 : 定界。
   - *property_value* 是先决条件属性的值，并且可以是字符串或整数。

   例如，env.tcrhome 是一个用于检查 Tivoli Common Reporting 的主目录环境变量的定制先决条件属性，并且期望值应该为 True:

   ```
   env.tcrhome=True
   ```

   env.path.jar 是用于检查是否在 PATH 环境变量中设置了 JRE 的定制先决条件属性，并且期望值应该为 False:

   ```
   env.path.jar=False
   ```

   注: 然后，您必须创建下列文件以根据需要查找并比较定制先决条件属性: 用于收集先决条件属性实际值的定制收集器，以及仅当标准比较函数无法对实际值与期望值进行比较时使用的定制评估程序。

编辑先决条件属性

您可以编辑先决条件属性、更改这些先决条件属性的期望值或者更改限定符的关联值。

开始之前

检查新值是否为先决条件属性所支持的有效值。例如，Disk 先决条件属性期望以 MB 或 GB 为单位的数字格式。如果要检查以百万兆 (TB) 为单位的可用磁盘空间量，您必须对比较 API 进行扩展，使其支持 TB 比较。另外，还必须对相关配置文件中的 Disk 先决条件属性进行编辑。

检查先决条件属性的预定义限定符和有效值（如第 8 页的『先决条件属性的预定义限定符』所述）。

过程

1. 打开配置文件。
2. 对于每个要编辑的先决条件属性，请输入新的期望值或者更改限定符的值。 例如，一位新的系统管理员是 root 用户，因此必须更改 user.userID 先决条件属性的值。请将值更改为新名称:

   ```
   user.userID=smithj
   ```

   例如，用于检查文件描述符限制的 os.ulimit 先决条件属性的 type 限定符的当前值为 filedescriptorlimit。您可能想检查另一限制，例如堆栈大小。请将先决条件属性的以下限定符的值由:

   ```
   os.ulimit=[type:filedescriptorlimit]8192+,unlimited
   ```

   更改为:

   ```
   os.ulimit=[type:stacksizelimit]512+,unlimited
   ```

   要点: 如第 9 页的表 5 所述，预定义的限定符只能与特定的预定义先决条件属性配合使用。

创建用于 Windows 系统的定制收集器

如果基本的收集器集合未收集所要安装的产品所需的先决条件属性的值，那么您可以创建定制收集器。您可以创建定制公共 VBScript 收集器，以收集适用于任何产品和产品版本的先决条件属性的数据。另外，也可以创建特定于产品的定制收集器，以收集适用于特定产品和产品版本的数据。虽然每一类定制 VBScript 收集器都通过相同的方法来收集数据，但创建、存储和执行方面的规则略有不同。

创建所有配置文件的公共定制 VBScript 收集器

创建公共定制 VBScript 收集器时，文件名必须包含先决条件属性的名称，且文件必须存储在 /lib 子目录中。此收集器包含用于获取先决条件属性的实际值的代码。另外，在有需要时，它还可以使用通用函数和子例程来获取该值。

开始之前

在创建收集器之前，请务必查看下列附录中的预定义函数及子例程集合。请确定能否使用其中的任何函数及子例程来获取实际值:

- 第 109 页的附录 E, 『Windows 系统的通用函数』
- 第 125 页的附录 G, 『Windows 系统的文件实用程序子例程』
- 第 123 页的附录 F, 『Windows 系统的日志记录实用程序子例程』
- 第 127 页的附录 H, 『Windows 系统的其他通用函数和子例程』

确定收集器是否必须检查该先决条件属性是否存在，以及必须进行此检查时还需收集的其他信息。无论属性是否存在，每次检查都必须返回一个值。例如:

- 检查产品主目录之类的环境变量（例如 Tivoli Common Reporting 的 TCR_HOME）是否存在。
- 检查环境变量是否包含 JAR 文件、二进制文件或路径，例如 PATH 环境变量中的 JRE 路径。
- 检查产品主目录之类的环境变量（例如 Tivoli Common Reporting 的 TCR_HOME）的实际值。
- 检查是否已安装某个产品。
- 检查已安装的产品版本。

过程

1. 创建 VBScript 文件。通过使用以下文件命名约定的变体，将此文件保存在 *ips_root*/lib 目录中:

   [*prefix_identifier*.]property_name.vbs

   其中:

   - *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应前缀标识。
   - *property_name* 是先决条件属性名，并且在收集器名称中使用。

   例如，mssqlVersion.vbs 包含用于在 Windows 机器上获取 MS SQL Server 先决条件属性的实际值的代码。
2. 通过使用 VBScript 编辑器，添加用于获取先决条件属性的值的代码。使用 VBScript COM 和函数来访问 Windows 环境的元素，并在 Windows 脚本宿主环境中运行。请确保此检查返回了如下标准输出:

   WScript.Echo "*property_name*=" & *var_for_value*

   - *property_name* 表示在配置文件中编写的先决条件属性，例如 env.tcrhome。
   - *var_for_value*，即收集器针对先决条件属性获取的实际值的 VBScript 变量。

   要检查 TCR_HOME 环境是否存在并返回实际值（先决条件属性名为 env.tcrhome）:

   ```
   set wshShell = WScript.CreateObject("WScript.Shell")
   tcr_home=WshShell.ExpandEnvironmentStrings("%TCR_HOME%")
   WScript.Echo "env.tcrhome=" &#38; tcr_home
   ```

   要检查是否在 PATH 变量中设置了 JRE（先决条件属性名为 env.path.jre）:

   ```
   Set wshShell = WScript.CreateObject("WScript.Shell")
   path = WshShell.ExpandEnvironmentStrings("%PATH%")
   Set objRegEx = new RegExp
   objRegEx.Pattern = "(^|([:;\\/]))(C:\Program Files\IBM\Java60\jre\bin)($|[:;])"
   objRegEx.IgnoreCase = True
   objRegEx.Global = True
   Set matches = objRegEx.Execute(path)
   WScript.Echo "env.path.jre=" &#38; (matches.Count > 0)
   ```

要检查已安装的 Tivoli Directory Integrator 版本（先决条件属性名为 installedSoftware.TDI.version）：

```
strComputer = "."
strKeyPath = "SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall"
regDisName = "DisplayName"
regDisVer = "DisplayVersion"

Set oReg = GetObject("winmgmts:{impersonationLevel=Impersonate}!\\" &#38;
          strComputer &#38; "\root\default:StdRegProv")

Set sftReg = new RegExp
sftReg.pattern = "Tivoli Directory Integrator"
sftReg.Global=False
oReg.EnumKey HKEY_LOCAL_MACHINE, strKeyPath, arrSubKeys
For Each subkey In arrSubKeys
    searchkey = strKeyPath &amp; "\" &#38; subkey
    oReg.GetStringValue HKEY_LOCAL_MACHINE, searchkey, regDisName, strName
    oReg.GetStringValue HKEY_LOCAL_MACHINE, searchkey, regDisVer, strVersion
    If Not IsNull(strName) Then
       Set matches = sftReg.Execute(strName)
       If matches.Count > 0 Then
           Wscript.Echo "installedSoftware.TDI.version=" &#38; strVersion
       End If
    End If
Next
```

3. 运行 VBScript 收集器以确保不存在运行时错误，必要时进行调试。
4. 仅当标准比较函数无法对实际值与期望值进行比较时，才需要创建定制评估程序。

创建特定于产品和产品版本的定制 VBScript 收集器

创建特定于产品的定制 VBScript 收集器时，文件名必须是与配置文件相同的产品代码，且文件必须存储在 /Windows 子目录中。此收集器可以包含用于收集一个或多个先决条件属性的实际值的代码。另外，在有需要时，它还可以使用通用函数和子例程来收集那些值。

开始之前

在创建收集器之前，请务必查看下列附录中的函数及子例程集合。请确定能否使用其中的任何函数及子例程来获取实际值:

- 第 109 页的附录 E, 『Windows 系统的通用函数』
- 第 125 页的附录 G, 『Windows 系统的文件实用程序子例程』
- 第 123 页的附录 F, 『Windows 系统的日志记录实用程序子例程』
- 第 127 页的附录 H, 『Windows 系统的其他通用函数和子例程』

确定收集器是否必须检查该先决条件属性是否存在，以及必须进行此检查时还需收集的其他信息。无论属性是否存在，每次检查都必须返回一个值。例如:

- 检查目录是否存在。
- 检查目录的可用磁盘空间量。
- 检查是否已安装某个产品。
- 检查已安装的产品版本。

过程

1. 创建 VBScript 文件。通过使用以下文件命名约定的变体，将此文件保存在 *ips_root*/Windows 目录中:

product_code[*_version*].vbs

其中:

- *product_code*

  这是用于在 Windows 或 UNIX 系统上表示产品代码的变量。产品代码用于标识产品、单个平台（例如 Windows、AIX、HP-UX、Linux 和 Solaris 以及该产品支持的操作系统版本（可选）。它们存储在 codename.cfg 文件中。任何支持多个平台的产品都有多个产品代码，并且每个产品代码都根据需要标识产品、平台和操作系统版本。

- *version* 是表示版本、发行版、修订版和级别的 8 位代码（此代码的每个部分各占两位）；例如，7.3.21 为 07032100。

2. 如果必须使用通用函数，请使用 VBScript 编辑器打开此文件并指定 common_function.vbs 的路径，如下所示:

   ```
   Include("..\lib\common_function.vbs")
   ```

3. 如果必须使用从 Prerequisite Scanner 传递的 PATH 和 -p 标志的值，请使用 Wscript.Arguments()，其中 Wscript.Arguments(0) 是 PATH 的值。Wscript.Arguments(1) 是 -p 标志及其值。

4. 添加代码，通过使用 VBScript COM 和函数来访问 Windows 环境的元素，获取先决条件属性的值。在 Windows 脚本宿主环境中运行。请确保此检查返回了如下标准输出:

   ```
   WScript.Echo "property_name=" &#38; var_for_value
   ```

   - *property_name* 表示在配置文件中编写的先决条件属性，例如 env.tcrhome。
   - *var_for_value*，即收集器针对先决条件属性获取的实际值的 VBScript 变量。

   检查产品安装目录的可用磁盘空间量。例如，要使用第 128 页的『getValue()』子例程来检查 Tivoli Monitoring for Energy Management Reporting and Optimization（先决条件属性为 InstallDir）:

   ```
   Set wshShell = WScript.CreateObject("WScript.Shell")
   'Check the disk space for the installation path that is passed as
    the value for the PATH argument
   installPath = Wscript.Arguments(0)
   sInstallPath= "InstallDir="
   Wscript.Echo "installation path    : " & installPath
   set fso = CreateObject("Scripting.FileSystemObject")

   getValue fso, sInstallPath, installPath

   'Common sub routine
   Sub getValue(fso, sKey, drvPath)
       Wscript.Echo "getValue(" & skey & "," & drvPath & ")"
      If fso.driveExists(fso.getDriveName(drvPath))  then
          Set disk = fso.GetDrive(fso.getDriveName(drvPath))
            'Value returned is in bytes.  Convert to MB
            cSize = CLng((disk.FreeSpace/1024)/1024) & "MB"
            WScript.Echo sKey & cSize
       Else
        Wscript.Echo " Disk for " & sKey & " -> " & drvPath & " does NOT exist"
      End If
   End Sub
   ```

5. 创建用于调用 VBScript 收集器的批处理文件。此批处理文件必须与配置文件同名并具有 .bat 扩展名，即名为 *product_code*[*_version*].bat，如下所示:

```
@echo off

set CMD_LINE_ARGS=
:setArgs
if ""%1""=="""" goto doneSetArgs
set CMD_LINE_ARGS=%CMD_LINE_ARGS% %1
shift
goto setArgs
:doneSetArgs

cscript.exe //nologo collector_file_name.vbs %CMD_LINE_ARGS%
```

6. 运行 VBScript 收集器以确保不存在运行时错误，必要时进行调试。
7. 仅当标准比较函数无法对实际值与期望值进行比较时，才需要创建定制评估程序。

创建用于 UNIX 系统的定制收集器

如果基本的收集器集合未收集所要安装的产品所需的先决条件属性的值，那么您可以创建定制收集器。创建定制收集器时，文件名必须与先决条件属性相同（尽管其名称不包含子类型）。此收集器存储在 /UNIX_Linux 子目录中。此收集器可以包含用于获取一个或多个先决条件属性的实际值的代码。另外，在有需要时，它还可以使用通用函数来获取这些值。

开始之前

在创建收集器之前，请务必查看下列附录中的函数集。请确定能否使用其中的任何函数来获取实际值:

- 第 131 页的附录 I, 『UNIX 系统的通用函数』
- 第 139 页的附录 J, 『UNIX 系统的其他函数』
- 第 147 页的附录 K, 『UNIX 系统的日志记录实用程序函数』

确定收集器是否必须检查该先决条件属性是否存在，以及必须进行此检查时还需收集的其他信息。无论属性是否存在，每次检查都必须返回一个值。例如:

- 检查是否已安装某产品，例如，随 RPM 一起安装的软件包。
- 检查已安装的产品版本。
- 检查已装配的文件系统是否有可用的磁盘空间。

如果要使用子类型 *suffix_identifier*，并且要按应用程序、实用程序或服务子类型对先决条件属性进行进一步分类，那么可以创建公共收集器。请将 *suffix_identifier* 子类型的区分符（即 *differentiator_suffix_identifier*）传递到它的收集器。例如，os.package 是用于检查软件包是否存在的公共收集器。要检查 openssh 是否存在，请在 packageTest.sh 脚本文件中调用 os.package 收集器时传递软件包的名称，如下所示:

```
./os.package openssh
```

其中，openssh 是软件包的名称，即 *suffix_identifier* 子类型和 *differentiator_suffix_identifier* 区分符。

过程

1. 创建 Shell 脚本文件。通过使用以下文件命名约定的变体但不使用文件扩展名，将此文件保存在 *ips_root*/Unix_Linux 目录中:

   [*prefix_identifier*.]*property_name*

其中:

- *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应标识。此前缀标识是某些预定义类别（例如 env）所必需。
- *property_name* 是先决条件属性的名称，例如 path.jre。

2. 如果必须使用通用函数，请使用编辑器打开此文件并指定 common_function.sh 的路径，如下所示:

```
. ../lib/common_function.sh
```

3. 添加代码，通过使用特定于该平台的命令和选项访问主机环境的元素，获取先决条件属性的值。例如，定制 env.path.jar 先决条件属性需要检查是否在 PATH 变量中设置了 JRE。以下代码运行 env 命令，搜索 PATH 变量的输出，然后在它的值中搜索 JRE 路径。

```
envJRE=`env | grep "PATH" | grep -w "/opt/IBM/Java60/jre/bin"`
```

4. 请确保此检查返回了标准输出:

```
echo "True"|"False" 'If the scan checks for the existence of the prerequisite
     property
echo $res 'If the scan checks returns the value, for example, product version,
          'of the prerequisite property
echo "Unavailable" 'If the scan returns no value for the prerequisite property
echo "Available" 'If the scan returns a valid check for the prerequisite property
```

在此示例中，根据 $envJRE 变量的值不同，此检查将返回 True 或 False:

```
if [ $envJRE ]; then
 echo "True"
else
 echo "False"
fi
```

5. 运行定制收集器以确保不存在运行时错误，必要时进行调试。
6. 编辑 *ips_root*/UNIX_Linux/packageTest.sh 脚本，以调用并运行定制收集器。
7. 仅当定制收集器返回除布尔值以外的值时，才需创建定制评估程序。

编辑用于 UNIX 系统的软件包测试脚本

您可以更新 packageTest.sh 脚本文件，以便在 UNIX 系统上调用定制收集器。

开始之前

请确保您了解与预定义的先决条件属性相关联的收集器的名称（如 第 103 页的附录 D, 『UNIX 系统的预定义收集器』所述）。如果按应用程序、实用程序或服务子类型对先决条件属性进行进一步分类，请将 *suffix_identifier* 子类型的区分符（即 *differentiator_suffix_identifier*）传递到它的收集器中。

例如，os.package 是用于检查软件包是否存在的公共收集器。要检查 openssh 是否存在，请在 packageTest.sh 脚本文件中调用 os.package 收集器时传递软件包的名称，如下所示:

```
./os.package openssh
```

其中，openssh 是软件包的名称，即 *suffix_identifier* 子类型和 *differentiator_suffix_identifier* 区分符。

过程

1. 使用编辑器打开 *ips_root*/UNIX_Linux/packageTest.sh 脚本。
2. 添加用于从配置文件中读取定制先决条件属性并对该属性的值进行解析的代码。

```
res=`echo $line | grep [prefix_identifier.]property_name[.suffix_identifier]`
if [ $res ]; then
ExpValue=`echo $res | cut -d "=" -f2`
```

例如，要读取定制 env.path.jar 先决条件属性并检查是否在 PATH 变量中设置了 JRE:

```
res=`echo $line | grep env.path.jar`
if [ $res ]; then
ExpValue=`echo $res | cut -d "=" -f2`
```

在此示例中:

```
 echo "\`wrlTrace "Starting" "env.path.jar"\`"  >>/tmp/prs.check
 echo "\`wrlTrace "Executing" "env.path.jar"\`"  >>/tmp/prs.check
 echo "\`wrlDebug "Starting" "env.path.jar"\`" >>/tmp/prs.check
 echo "\`wrlDebug "Expected" "ExpValue" \`" >>/tmp/prs.check
```

3. 在调用定制收集器之前，调用日志记录函数以记录跟踪数据和调试数据。

```
 echo "\`wrlTrace "Starting" "[prefix_identifier.]property_name
[.suffix_identifier]"\`"  >>/tmp/prs.check
 echo "\`wrlTrace "Executing" "[prefix_identifier.]property_name
[.suffix_identifier]"\`"  >>/tmp/prs.check
 echo "\`wrlDebug "Starting" "[prefix_identifier.]property_name
[.suffix_identifier]"\`" >>/tmp/prs.check
 echo "\`wrlDebug "Expected" "ExpValue" \`" >>/tmp/prs.check
```

4. 调用定制收集器。

注: 如果定制收集器具有子类型（即文件名包含 [*suffix_identifier*]），并需要根据子类型进行额外的检查，请将子类型的 [*differentiator_suffix_identifier*] 区分符传递到定制收集器。

```
 echo "ss=\`./[prefix_identifier.]property_name[.suffix_identifier]
 [differentiator_suffix_identifier]\`" >>/tmp/prs.check
```

在此示例中:

```
 echo "ss=\`./env.path.jar\`" >>/tmp/prs.check
```

注: os.file.*script_name* 先决条件属性的 *script_name* 子类型的区分符示例是传递到 os.filepath 收集器的脚本路径:

```
echo "ss=\`./os.filepath /usr/bin/expect\`" >>/tmp/prs.check #os.file.expect
echo "ss=\`./os.filepath /usr/bin/tar\`" >>/tmp/prs.check #os.file.tar
echo "ss=\`./os.filepath /usr/bin/gzip\`" >>/tmp/prs.check #os.file.gzip
```

5. 在退出定制收集器之后，调用日志记录函数以记录跟踪数据和调试数据。

```
 echo "\`wrlTrace "Finished" "[prefix_identifier.]property_name
[.suffix_identifier]"\`" >/tmp/prs.check
 echo "echo \"[prefix_identifier.]property_name
[.suffix_identifier]=\$ss\"" >>/tmp/prs.check
 echo "\`wrlDebug "Finished" "[prefix_identifier.]property_name
[.suffix_identifier]"\`"  >>/tmp/prs.check
 echo "\`wrlDebug "OutPutValueIs" \$ss\`" >/tmp/prs.check
 echo "\`wrlTrace "Done" "[prefix_identifier.]property_name
[.suffix_identifier]"\`"  >>/tmp/prs.check
fi
```

在此示例中:

```
 echo "ss=\`./env.path.jar\`" >>/tmp/prs.check
 echo "\`wrlTrace "Finished" "env.path.jar"\`"  >>/tmp/prs.check
 echo "echo \"env.path.jar=\$ss\"" >>/tmp/prs.check
 echo "\`wrlDebug "Finished" "env.path.jar"\`"  >>/tmp/prs.check
 echo "\`wrlDebug "OutPutValueIs" \$ss\`"  >>/tmp/prs.check
 echo "\`wrlTrace "Done" "env.path.jar"\`"  >>/tmp/prs.check
fi
```

6. 对每个定制先决条件属性重复步骤 2 到 5。

创建用于 Windows 系统的定制评估程序

如果基本评估程序未使用正确的评估条件对先决条件属性的期望值和实际值进行比较，那么您可以创建 VBScript 评估程序。创建定制评估程序时，文件名必须以 _compare 结尾，并且文件必须存储在 /Windows 子目录中。有需要时，定制评估程序可以使用通用函数和子例程对值进行比较。

开始之前

在创建评估程序之前，请务必查看下列附录中的函数及子例程集合。请确定能否使用其中的任何函数及子例程来比较值:

- 第 109 页的附录 E, 『Windows 系统的通用函数』
- 第 125 页的附录 G, 『Windows 系统的文件实用程序子例程』
- 第 123 页的附录 F, 『Windows 系统的日志记录实用程序子例程』
- 第 127 页的附录 H, 『Windows 系统的其他通用函数和子例程』

注: 通用函数第 119 页的『passOrFail()』可以对下列数据类型的实际值和期望值进行比较: 常规数字、以 MB 或 GB 为单位的大小、以 MHz 或 GHz 为单位的处理器速度、布尔值或字符串。仅当无法使用 passOrFail 函数时，才应创建定制评估程序。

过程

1. 创建 VBScript 文件。通过使用以下文件命名约定的变体，将此文件保存在 *ips_root*/Windows 目录中:

   [*prefix_identifier*.]*property_name*[.*suffix_identifier*]_compare.vbs

   其中:

   - *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应标识。此前缀标识是某些预定义类别所必需。
   - *property_name* 是先决条件属性的名称。
   - *suffix_identifier* 是第 6 页的表 4 中概述的先决条件属性子类型的相应可选标识。
2. 添加代码，对使用 VBScript COM 和相关函数作为自变量传递到评估程序的实际值和期望值进行比较。请确保此比较返回如下所示的标准输出:
   - "PASS"（如果先决条件属性的期望值等于或大于其实际值）
   - "FAIL"（如果先决条件属性的期望值与其实际值不等）
3. 运行定制评估程序以确保不存在运行时错误，必要时进行调试。

示例

此定制评估程序将检查 Tivoli Directory Integrator 版本的实际值和期望值。它使用了通用函数第 129 页的『versionCompare()』。

```
wscript.echo "expect: " &#38; wscript.arguments(0)
wscript.echo "real value: " &#38; wscript.arguments(1)
wscript.echo tdiVersionCompare(wscript.arguments(0), wscript.arguments(1))

function tdiVersionCompare(expect, real)
    if len(real) = 0 then
        tdiVersionCompare = "FAIL"
        exit function
    end if

    expect = Trim(expect)
    real = Trim(real)

    Dim expectedVersion
    'if (StrComp(Right(expect,1),"+")=0 or StrComp(Right(expect,1),"-")=0) Then
    if (Right(expect,1)="+" or Right(expect,1)="-") Then
        expectedVersion = Left(expect,len(expect)-1)
    else
        expectedVersion = expect
    end if

    Dim cmp
    cmp = versionCompare(expectedVersion,real)

    if (StrComp(Right(expect,1),"+")=0) Then
        ' Version must be at least expected value
        if (cmp=0 or cmp=-1) Then
            tdiVersionCompare = "PASS"
        else
            tdiVersionCompare = "FAIL"
        end if
    elseif (StrComp(Right(expect,1),"-")=0) Then
            ' Version must be less than or equal to expected value
            if (cmp=0 or cmp=1) Then
                tdiVersionCompare = "PASS"
            else
                tdiVersionCompare = "FAIL"
            end if
    elseif cmp=0 then
        tdiVersionCompare = "PASS"
    else
        tdiVersionCompare = "FAIL"
    end if
end function

' Generic function for comparing 2 version strings
'
' Parameters
'       ver1 The first version string
'       ver2 The second version string
'
' ver1 and ver2 are expected to be dot-separated version strings
'(e.g. 1.0.0.4, 2.3, 3.40.26.7800, 2.3.a)Version strings can have any
' number of parts. When comparing versions with different numbers of
' parts, missing parts of the shorter version string will be treated
' as if there was a zero there. If any non-numeric characters are
' included in a version part, those corresponding parts will be compared
' asstrings and not parsed into numeric form
'
' Returns
'       1 version1 > version2
'      -1 version1 &#60; version2
'       0 version1 = version2
'
' Special cases:
' RESULT    version 1    version 2
'   0         empty        empty
```

```
'   1      validString     empty
'  -1         empty     validString
'
' NOTE: This function should eventually move to common_functions.vbs

function versionCompare(ver1, ver2)
    WScript.echo "Comparing [" &#38; ver1 &#38; "] to [" &#38; ver2 &#38; "]"

    Const UNASSIGNED = "*UNASSIGNED*"
    Dim v1Default, v2Default

    ' Handle special cases:
    if (IsEmpty(ver1) and IsEmpty(ver2)) Then
        versionCompare = 0
        exit function
    end if
    if (IsEmpty(ver1) and not IsEmpty(ver2)) Then
        versionCompare = -1
        exit function
    end if
    if (not IsEmpty(ver1) and IsEmpty(ver2)) Then
        versionCompare = 1
        exit function
    end if

    Dim ver1Parts, ver2Parts

    ' Versions are not empty.  Break into parts and compare numbers
    ver1Parts = Split(ver1,".")
    ver2Parts = Split(ver2,".")

    Dim v1Size, v2Size
    v1Size = ubound(ver1Parts)
    v2Size = ubound(ver2Parts)

    ' If last version part is "*", treat all missing parts as "*"
    '(so 2.* matches 2.1.3, for example)
    if (v1Size > v2Size) Then
        Redim Preserve ver2Parts(v1Size)
        if (ver2Parts(v2Size)="*") Then
            for i = v2Size to v1Size
                ver2Parts(i) = "*"
            next
        end if
    elseif (v2Size > v1Size) Then
        Redim Preserve ver1Parts(v2Size)
        if (ver1Parts(v1Size)="*") Then
            for i = v1Size to v2Size
                ver1Parts(i) = "*"
            next
        end if
    end if

    Dim i
    i = 0

    Do While (i&#60;=ubound(ver1Parts) or i&#60;=ubound(ver2Parts))
        Dim v1, v2, v1Str, v2Str

        v1Str = UNASSIGNED
        v2Str = UNASSIGNED

        if (i&#60;=ubound(ver1Parts)) Then
            on error resume next
            v1 = Int(ver1Parts(i))
            if not Err=0 Then
                v1Str = ver1Parts(i)
```

```
                if (i&#60;=ubound(ver2Parts)) Then
                    v2Str = ver2Parts(i)
                else
                    v2Str = "0"
                end if
            end if
        else
            v1 = 0
        end if

        if (i&#60;=ubound(ver2Parts)) Then
            on error resume next
            v2 = Int(ver2Parts(i))
            if not Err=0 Then
                if (i&#60;=ubound(ver1Parts)) Then
                    v1Str = ver1Parts(i)
                else
                    v1Str = "0"
                end if
                v2Str = ver2Parts(i)
            end if
        else
            v2 = 0
        end if

        if (not v1Str=UNASSIGNED or not v2Str=UNASSIGNED) Then
            if (IsEmpty(v1Str)) Then
                v1Str = "0"
            end if
            if (IsEmpty(v2Str)) Then
                v2Str = "0"
            End if

            'WScript.echo "Comparing as strings: " &#38; v1Str &#38; " : " &#38; v2Str
            ' Compare as Strings if either part could not be converted to a number
            if (not v1Str="*" and not v2Str="*") Then
                if (not v1Str=v2Str) Then
                    versionCompare = StrComp(v1Str,v2Str)
                    exit function
                end if
            end if
        else
            'WScript.echo "Comparing as numbers: " &#38; v1 &#38; " : " &#38; v2

            if (v1 > v2) Then
                versionCompare = 1
                exit function
            end if
            if (v2 > v1) Then
                versionCompare = -1
                exit function
            end if
        end if

        i = i + 1
    Loop

    ' If we got here, versions must be equal
    versionCompare = 0

end function
```

创建用于 UNIX 系统的定制评估程序

如果定制收集器未返回布尔值（即 True 或 False），那么您可以创建定制评估程序。创建定制评估程序时，文件名必须以 _compare 结尾，并且文件必须存储在 /UNIX_Linux 子目录中。有需要时，定制评估程序可以使用通用函数对值进行比较。

开始之前

在创建定制评估程序之前，请务必查看下列附录中的函数集。请确定能否使用其中的任何函数对实际值和期望值进行比较:

- 第 131 页的附录 I, 『UNIX 系统的通用函数』
- 第 139 页的附录 J, 『UNIX 系统的其他函数』
- 第 147 页的附录 K, 『UNIX 系统的日志记录实用程序函数』

您可以使用 /Unix_Linux 子目录中的两个脚本文件（即 ._compare.sh 和 _compare.sh）作为起点。

要点: 如果定制收集器返回 True 或 False，请不要创建定制评估程序。对于所有返回布尔值的收集器，IBM Prerequisite Scanner 都使用预定义的评估程序。

过程

1. 创建 Shell 文件。通过使用以下文件命名约定的变体，将此文件保存在 *ips_root*/UNIX_Linux 目录中:

   [*prefix_identifier*.]*property_name*[.*suffix_identifier*]_compare.sh

   其中:

   - *prefix_identifier* 是第 4 页的表 3 中概述的先决条件属性预定义类别的相应标识。此前缀标识是某些预定义类别所必需。
   - *property_name* 是先决条件属性的名称。
   - *suffix_identifier* 是第 6 页的表 4 中概述的先决条件属性子类型的相应可选标识。
2. 添加代码，对作为自变量和相关函数传递到评估程序的实际值和期望值进行比较。请确保此比较返回如下所示的标准输出:
   - "PASS"（如果先决条件属性的期望值等于或大于其实际值）
   - "FAIL"（如果先决条件属性的期望值与其实际值不等）
3. 运行定制评估程序以确保不存在运行时错误，必要时进行调试。

第 4 章 运行 Prerequisite Scanner

您可以使用命令行界面来运行 IBM Prerequisite Scanner。Prerequisite Scanner 脚本 prereq_checker 使用一组必需参数和可选参数，并使用一个命令标志作为附加的可选参数。

表 12 对 Prerequisite Scanner 脚本语法中使用的特殊字符作了说明。

表 12. Prerequisite Scanner 脚本的特殊字符图注

特殊字符	描述
<>	用于标识占位符名称。
[]	用于标识可选参数。未用方括号括起的参数是必需参数。
...	表示可以对一个参数指定多个值。
|	指示互斥参数。请指定分隔符左边或右边的参数，但不得同时指定这两者。
{}	用于将一组由 | 分隔的互斥参数括起来。

prereq_checker

prereq_checker 脚本运行 IBM Prerequisite Scanner，并根据您运行该脚本时指定的一组参数来检查先决条件。

语法

```
prereq_checker.bat|sh
  "Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."
  [detail]
  [outputDir="ips_output_dir"]
  [xmlResult]
  [PATH="product_root"]
  [-p Product_Code.instance.parameter=value,...]
  [debug]
  [trace]
```

prereq_checker 脚本具有一个必需参数和多个可选参数。

第 58 页的『"Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."』
必需参数

第 58 页的『[detail]』
可选参数

第 61 页的『[outputDir="ips_output_dir"]』
可选参数

第 61 页的『[xmlResult]』
可选参数

第 61 页的『[PATH="product_root"]』

第 61 页的『[-p Product_Code.instance.parameter=value,...]』
可选标志

第 62 页的『[debug]』
可选参数

第 62 页的『[trace]』
可选参数

"Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."

您必须至少设置一个 **Product_Code** 参数，以标识要对其运行先决条件检查的产品或组件以及相关联的配置文件。**Product_Code** 是 *ips_root*/codename.cfg 文件中设置的产品代码。

例如，KMS 是 product.cfg 文件中设置的 Tivoli Enterprise Monitoring Server 产品代码。要运行 Scanner，请输入以下脚本及产品代码:

```
./prereq_checker.sh KMS
```

如果设置了没有相应配置文件的 **Product_Code** 参数，那么 Prerequisite Scanner 将忽略该参数，而不会出错。日志文件将包含一条消息，指出找不到配置文件。

相关联 **Product_Code** 参数的 **Product_Version** 参数指示产品的版本。这是表示版本、发行版、修订版和级别的 8 位代码（此代码的每个部分各占两位）；例如，7.3.21 为 07032100。**Product_Version** 是可选参数。如果未设置此参数，那么 Prerequisite Scanner 将检查最新的可用版本。

您可以设置一个或多个具有可选 **Product_Version** 参数的 **Product_Code** 参数并使用逗号对各个参数进行分隔。

要点: 如果设置了多个具有可选 **Product_Version** 参数的 **Product_Code** 参数，请将这些参数括在引号内。如果未这样做，那么 Scanner 将失败。

以下示例检查 Tivoli Monitoring Operating System Agent for Windows 的最新版本以及 Tivoli Monitoring Agent for DB2 V6.2.1 的相应先决条件。

```
prereq_checker.bat "KNT,KUD 06210000"
```

[detail]

此可选参数指示是否在命令行界面中显示详细的扫描结果。

要点: 请勿将此参数括在引号内。

设置 **detail** 参数后，详细结果将包含以下内容:

- Prerequisite Scanner 的版本
- 在其中运行 Scanner 的操作系统的版本
- 对其运行先决条件检查的产品或组件的名称
- 对于每个先决条件属性: 所检查的先决条件属性的名称、PASS 或 FAIL 结果、实际值和期望值

- 对于所有组件: 所检查的常规先决条件属性的名称、PASS 或 FAIL 结果、实际值和期望值
- 整体 PASS 或 FAIL 结果

Prerequisite Scanner 还将这些结果保存到 *ips_output_dir*/result.txt 文件中。无论您是否设置了 **detail** 参数，结果都将保存到文本文件中。

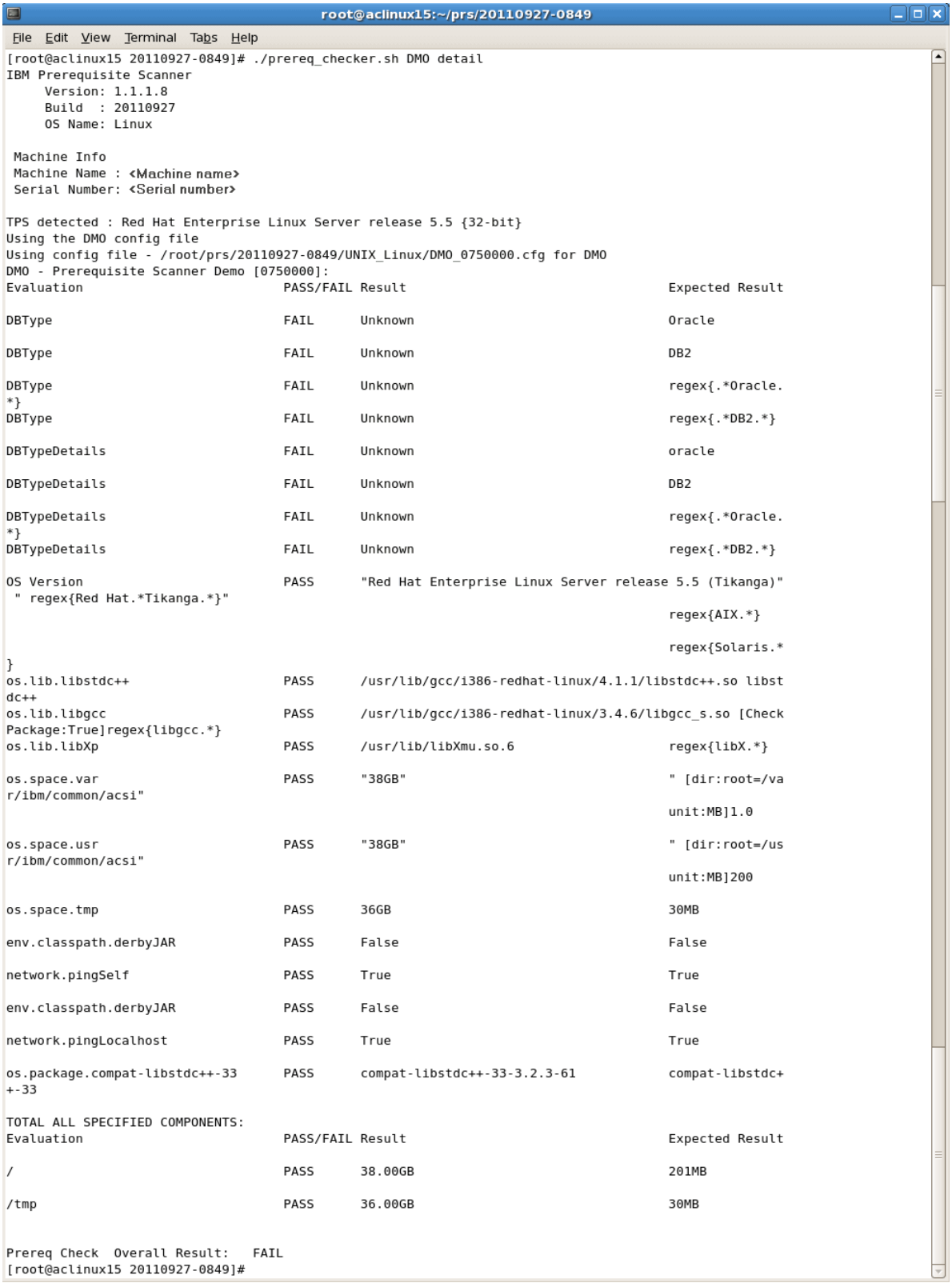

图 10. 在 UNIX 系统上，在设置了 detail 参数的情况下运行脚本

如果未设置 **detail** 参数，那么 Scanner 仅在命令行界面中显示 PASS 或 FAIL 结果。

图 11. 在 Windows 系统上，在未设置 detail 参数的情况下运行脚本

[outputDir="*ips_output_dir*"]

此可选参数表明您希望为 Prerequisite Scanner 的扫描结果和日志文件设置输出目录。

如果在设置了可选的 **outputDir** 参数的情况下运行 Prerequisite Scanner 脚本，那么 Prerequisite Scanner 会将结果文本、XML 和日志文件输出到该参数的值所指定的目录中。在整个文档中，将此值称为 *ips_output_dir*。

如果未设置此参数，那么缺省输出位置为 *ips_root*。

如果您选择从 CD、DVD 或只读的网络驱动器运行 Prerequisite Scanner，那么必须使用此参数来指定位置。您必须有权写入 *ips_output_dir*，否则 Prerequisite Scanner 将失败。

要点：如果输出目录不存在，那么 Prerequisite Scanner 将创建该目录。您必须有权创建或写入 Prerequisite Scanner 用于保存文件的输出目录。

[xmlResult]

此可选参数表明您不仅希望将结果输出到纯文本测试结果文件中，还要将其输出到 XML 结果文件中。

如果在设置了可选的 **xmlResult** 参数的情况下运行 Prerequisite Scanner 脚本，那么 Prerequisite Scanner 会将结果输出到 *ips_output_dir*/result.xml 文件中。

如果未设置此参数，那么结果将仅输出到纯文本文件中。

[PATH="*product_root*"]

此可选参数指示产品的安装目录。

要点：在 Windows 上，请勿设置仅包含盘符的路径，例如 C:。请确保设置有效的路径。

如果未设置 **path** 参数，那么 Scanner 将检查 IBM Tivoli 产品的缺省安装目录：

- 在 **UNIX 系统上**：/opt/ibm/itm
- 在 **Windows 系统上**：C:\IBM\itm

[-p *Product_Code.instance.parameter=value,...*]

可选的 **-p** 标志指示必须将随后的一组参数传递到脚本文件以进行附加的先决条件检查。**<Product_Code>** 是产品代码。只有每组 *instance.parameter=value* 才会传递到脚本中。您可以传递多组以逗号分隔的参数。

参数所传递到的脚本由下列选项确定：

- 如果指定了 **Product_Code** 前缀，那么参数将传递到具有相关联 **Product_Code** 的脚本中
- 如果未指定 **Product_Code** 前缀，那么参数将传递到公共收集器中。

示例 1-p KUD.inst1.DB2_INST_OWNER=db2inst1, KUD.inst2.DB2_INST_OWNER=db2inst2 具有参数的此标志将 db2inst1.DB2_INST_OWNER=db2inst1 和 db2inst2.DB2_INST_OWNER=db2inst2 传递到 KUD.**Product_Version**.bat 脚本文件中。

示例 2

```
-p SERVER=IP.PIPE://mymachine:1918
```

具有参数的此标志将 SERVER=IP.PIPE://mymachine:1918 传递到公共收集器以检查端口。

注: 此脚本接受 **-p** 中的参数作为 tacmd createNode。
您可以在 *ips_root*/lib/common_configuration 中设置 SERVER、PROTOCOL、PORT、BACKUP 和 BSERVER 参数。Prerequisite Scanner 将通过命令行界面传递的参数视为优先于 common_configuration 文件中设置的那些参数。

[debug]

此可选参数表明您希望在开启调试的情况下运行 Prerequisite Scanner。

如果在设置了可选的 **debug** 参数的情况下运行 Prerequisite Scanner 脚本，那么 Prerequisite Scanner 会将详细的处理信息、警告消息和错误消息以及扫描结果输出到日志文件中。在 UNIX 系统上，此文件为 *ips_output_dir*/prs.debug；而在 Windows 系统上，此文件为 *ips_output_dir*/precheck.log。

要点: 在缺省情况下，Scanner 调试处于关闭状态。

[trace]

（仅限于 UNIX 系统）此可选参数表明您希望在开启跟踪日志记录的情况下运行 Prerequisite Scanner。

如果在设置了可选的 **trace** 参数的情况下运行 Prerequisite Scanner 脚本，那么 Prerequisite Scanner 会将跟踪信息输出到 *ips_output_dir*/prs.trc 文件中。

要点: 在缺省情况下，Scanner 跟踪日志记录处于关闭状态。

从命令行运行 Prerequisite Scanner

您可以从命令行界面运行 IBM Prerequisite Scanner 并提供脚本的相关输入参数。

开始之前

请确保查阅产品的安装文档或技术说明，以了解任何其他必须在运行 Prerequisite Scanner 之前执行的步骤。例如，您可能需要设置环境变量，以便将目标计算机上安装的组件或功能部件以及因此需要检查的先决条件告知 Prerequisite Scanner。

过程

1. 打开命令行界面，然后打开 *ips_root* 目录。
2. 运行 Prerequisite Scanner 脚本文件 **prereq_checker**，如下所示:

UNIX

```
./prereq_checker.sh
"Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."
[detail]
[outputDir="ips_output_dir"]
[xmlResult]
[PATH="product_root"]
[-p Product_Code.instance.parameter=value,...]
```

以下示例使用配置文件及相关产品代码 ADE 来运行 Prerequisite Scanner for Autonomic Deployment Engine:

```
./prereq_checker.sh
ADE 072000
detail
PATH=/opt/ibm/tivoli
```

Windows

```
prereq_checker.bat
"Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."
[detail]
[outputDir="ips_output_dir"]
[xmlResult]
[PATH="product_root"]
[-p Product_Code.instance.parameter=value,...]
```

以下示例使用产品代码 COX 和 COY 来运行 Prerequisite Scanner for Tivoli Provisioning Manager for Windows 2003 和 2008。

```
prereq_checker.bat
"COX, COY 07200000"
detail
PATH="D:\ibm\tivoli"
-p SERVER=IP.PIPE://mytems:1234
```

以下示例使用产品代码 KZE 来运行 Prerequisite Scanner for Tivoli zEnterprise® Monitoring Agent。另外，它还使用可选的 **outputDir** 参数将结果文件和日志文件的位置设置为 *ips_output_dir*。

要点: 如果您选择从 CD、DVD 或只读的网络驱动器运行 Prerequisite Scanner，那么必须使用 **outputDir** 参数来指定位置。您必须有权写入 *ips_output_dir*，否则 Prerequisite Scanner 将失败。

```
Windows
prereq_checker.bat
"KZE 06230000"
outputDir="%TEMP%\ips"
UNIX
./prereq_checker.sh
"KZE 06230000"
outputDir="/tmp/ips"
```

Scanner 将 result.txt 文件和 precheck.log 文件输出到下列位置:

- 在 Windows 系统上: D:\temp\ips，其中 TEMP 是临时文件夹的环境变量。
- 在 UNIX 系统上: /tmp/ips

要点: 如果输出目录不存在，那么 Prerequisite Scanner 将创建该目录。您必须有权创建或写入 Prerequisite Scanner 用于保存文件的输出目录。

公共目录位置

公共目录有相应的路径名变量。

IBM Prerequisite Scanner 安装目录

ips_root 描述 Prerequisite Scanner 的安装位置。您可以在安装期间指定此位置。

Prerequisite Scanner 输出目录

ips_output_dir 描述 Prerequisite Scanner 的扫描结果和日志文件的保存位置。您可以在运行 Scanner 时使用 **outputDir** 输入参数来指定此位置。如果未设置此参数，那么缺省输出位置为 *ips_root*。

注: Prerequisite Scanner 在其执行期间会创建临时文件，但是 Scanner 在完成其执行前会将这些文件删除。这些临时文件位于 *ips_output_dir*/temp 子目录中。除非 *ips_output_dir*/temp 子目录仅包含在 UNIX 系统上生成的调试文件和跟踪文件，否则 Scanner 还将删除该子目录。

第 5 章 Prerequisite Scanner 故障诊断

在创建定制先决条件检查时，您可以使用日志文件和日志记录功能对 IBM Prerequisite Scanner 中的问题进行故障诊断。

Prerequisite Scanner 根据扫描结果以及它是否必须因为错误而退出来生成返回码。这些返回码将写入日志文件。例如，如果 Prerequisite Scanner 由于无法读取配置文件而无法运行扫描，那么它将生成返回码 2。

在 Windows 系统上进行故障诊断

缺省情况下，运行 IBM Prerequisite Scanner 时，它将创建一个日志文件。此文件包含有关 Scanner 按顺序执行的各个步骤和函数的详细信息。另外，此文件还包含时间戳记（包括各个函数和步骤的开始时间及结束时间）。您可以调试和查看日志文件，以确定错误发生位置和时间。

Prerequisite Scanner 在 *ips_output_dir*/precheck.log 文件中输出处理信息、警告和错误消息以及扫描结果。在设置了可选 **debug** 参数的情况下运行 Prerequisite Scanner 脚本时，Prerequisite Scanner 将在此文件中输出附加的调试消息。

第 66 页的图 12 显示设置了可选参数 **debug** 时的日志文件示例，第 67 页的图 13 显示未设置该参数时的日志文件。

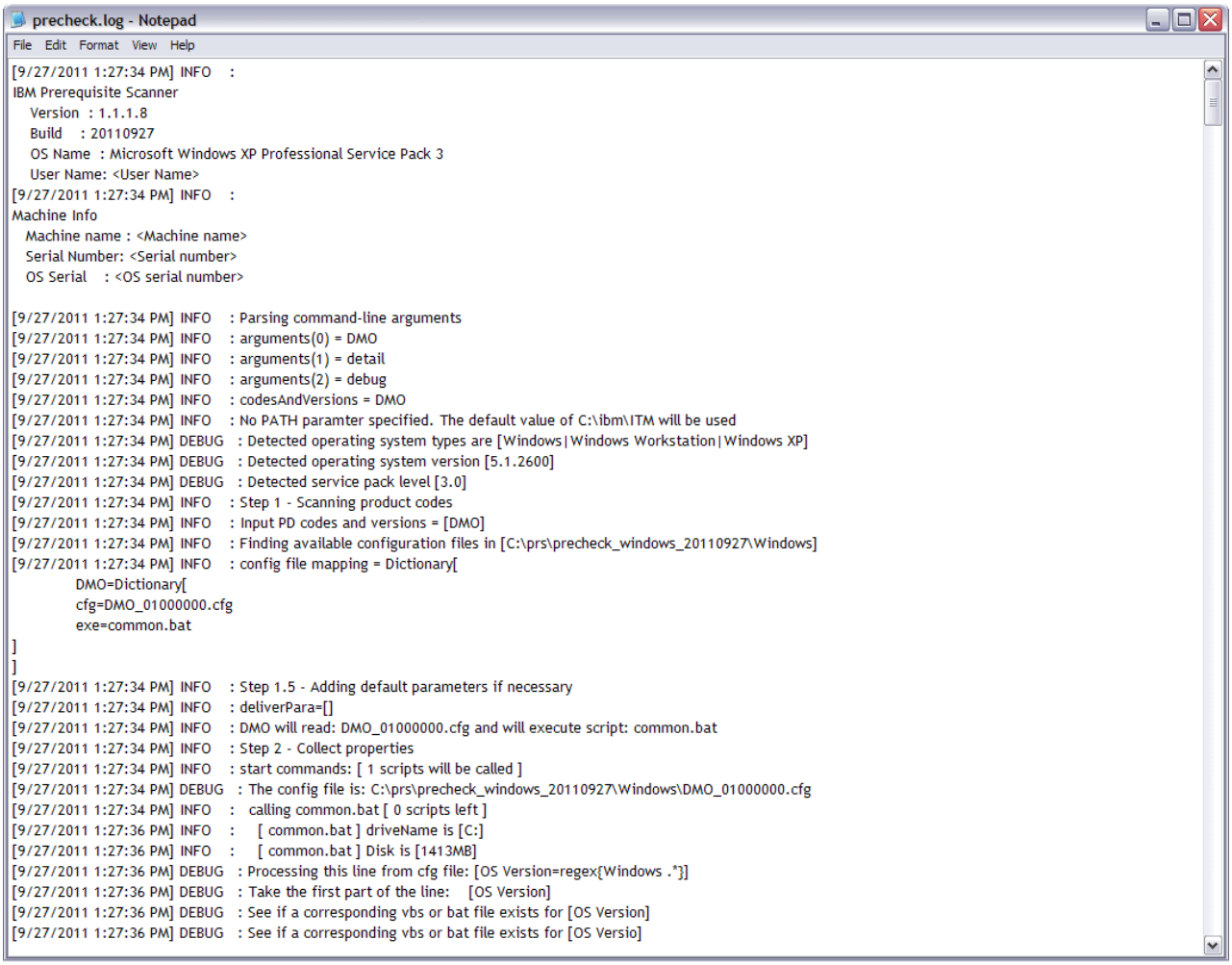

图 12. 包含调试数据的 precheck.log 文件

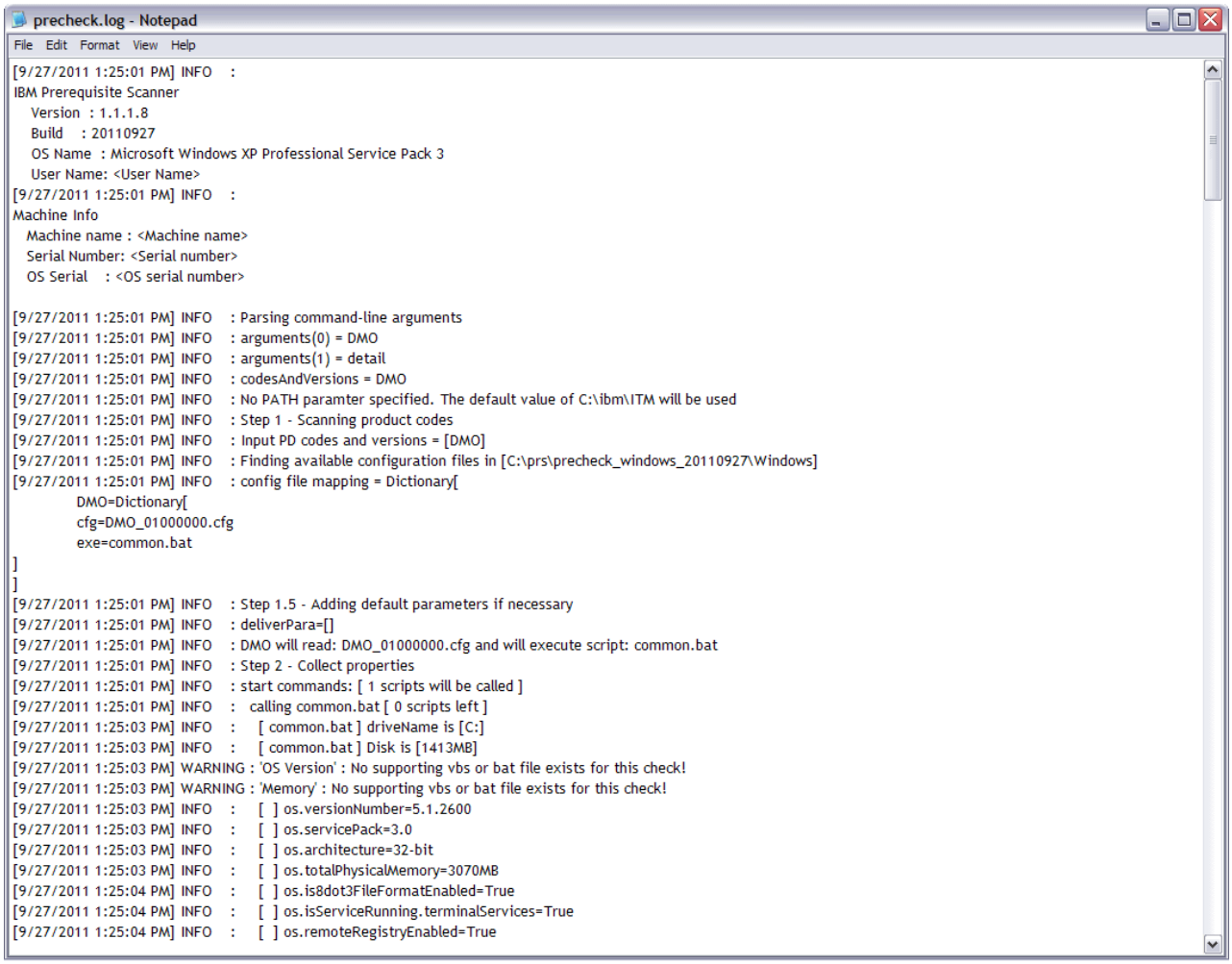

图 13. 未包含调试数据的 precheck.log 文件

在 UNIX 系统上进行故障诊断

在 UNIX 系统上，缺省情况下不会将消息写入日志文件。您可以使用 **debug** 和 **trace** 输入参数来启用调试或跟踪功能。Scanner 将调试和跟踪数据写入不同的日志文件，并使用时间戳记对步骤或函数的开始时间和结束时间进行标记。您可以使用这两个文件来关联特定的问题、函数或先决条件检查并对其进行故障诊断。

调试日志文件

如果在设置了可选的 **debug** 参数的情况下运行 Prerequisite Scanner 脚本，那么 Prerequisite Scanner 会将详细的处理信息、警告消息和错误消息以及扫描结果输出到 *ips_output_dir*/temp/prs.debug 文件中。此文件包含有关 Scanner 按顺序执行的各个步骤和函数的详细信息。另外，此文件还包含时间戳记（包括各个函数和步骤的开始时间及结束时间）。*ips_output_dir*/temp 子目录还包含向最终 *ips_output_dir*/result.txt 文件提供输入的中间 result1.txt 和 result2.txt 文件。您可以使用这些中间文件来确定特定先决条件检查的结果所存在的问题。

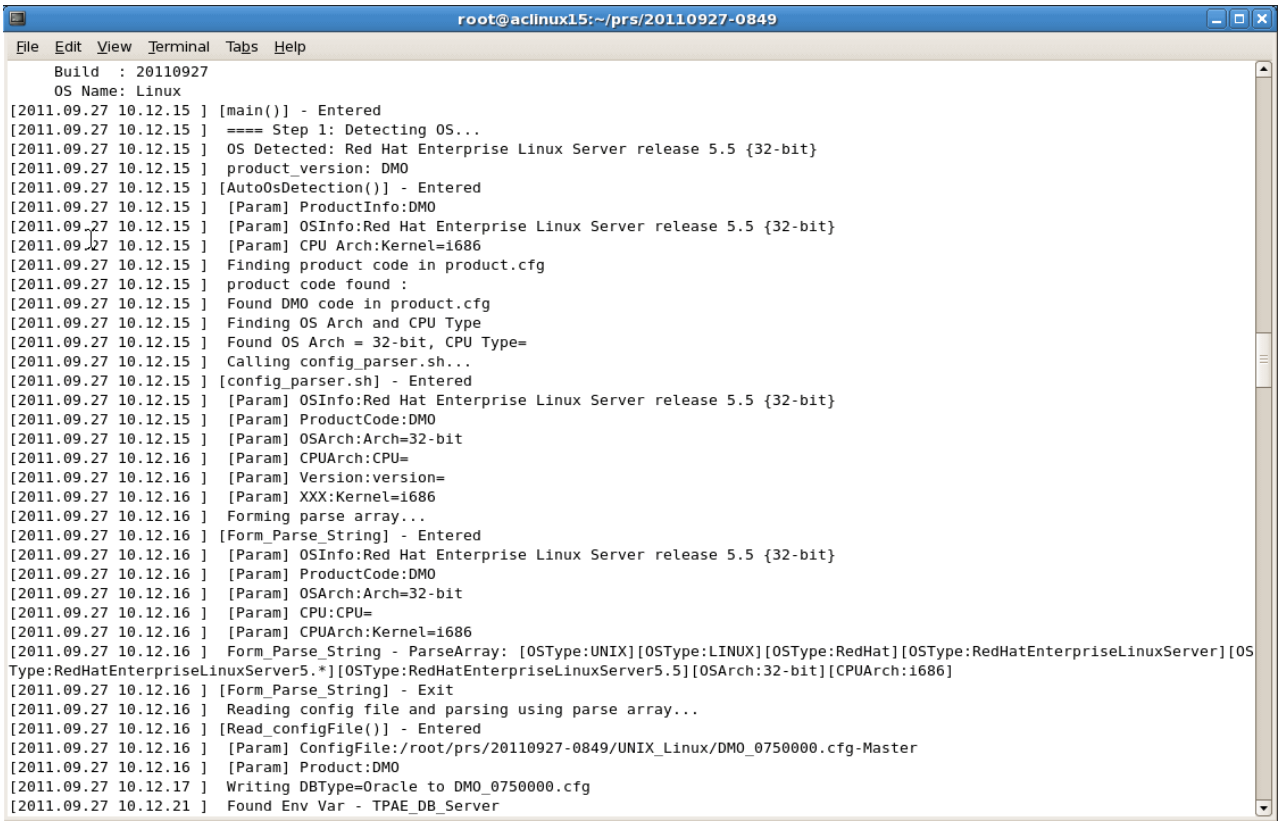

图 14. UNIX 系统上的 prs.debug 文件

跟踪日志文件

如果在设置了可选的 **trace** 参数的情况下运行 Prerequisite Scanner 脚本，那么 Prerequisite Scanner 会将跟踪信息输出到 *ips_output_dir*/temp/prs.trc 文件中。此文件包含有关 Scanner 按顺序执行的各个函数的信息。另外，此文件还包含时间戳记（包括各个函数的开始时间及结束时间）。

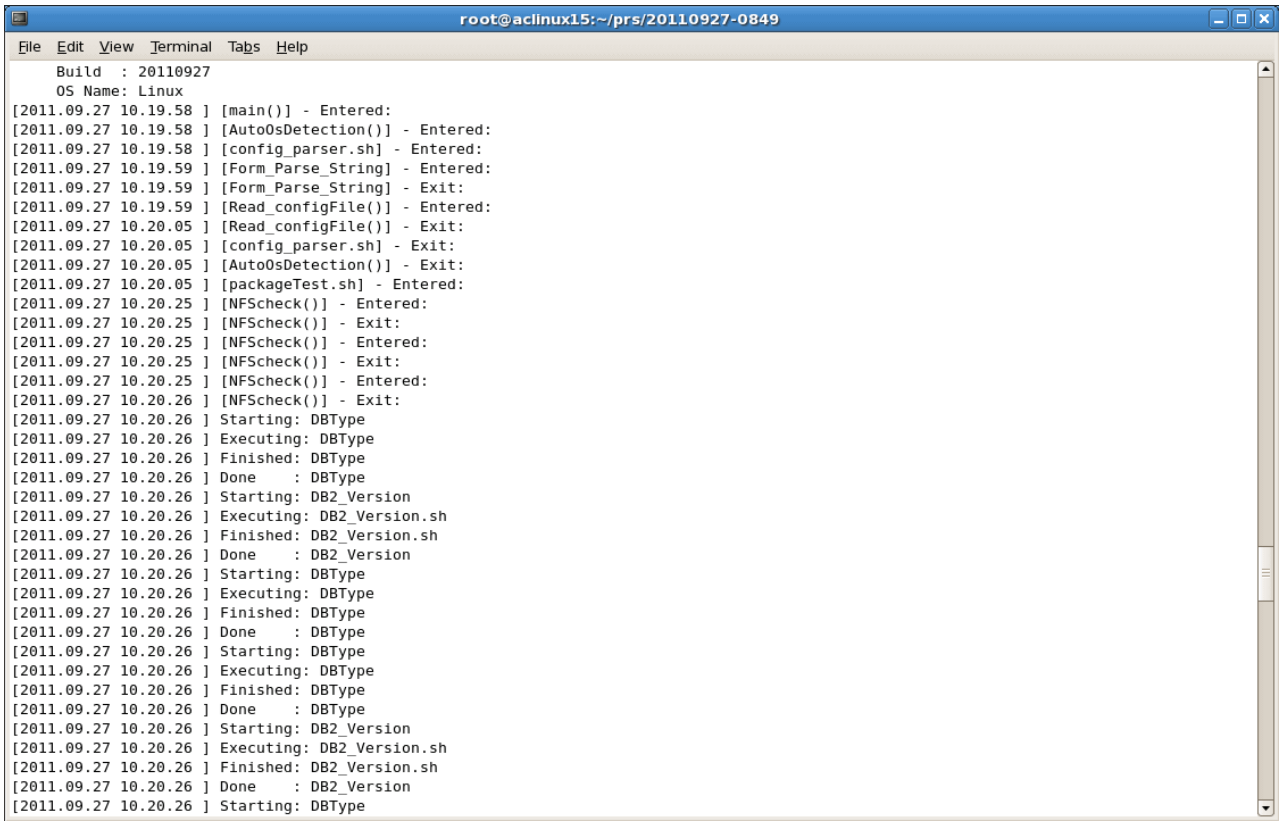

图 15. UNIX 系统上的 prs.trc 文件

执行问题

您可以使用执行问题核对表对运行 Prerequisite Scanner 时可能遇到的错误进行故障诊断。

请在指定了可选输入参数 **debug** 和 **trace** 的情况下运行 Prerequisite Scanner 脚本，以帮助调试问题。

表 13. 执行问题核对表

检查	项目
☐	如果在命令行中设置了可选参数 **outputDir** 并且输出目录不存在，那么 Prerequisite Scanner 将创建该目录。您必须有权创建或写入 Prerequisite Scanner 用于保存文件的输出目录。如果您不具有写许可权，那么以下错误消息将写至命令行界面: `ERROR: Cannot create files in output directory ips_output_dir. Exit.`
☐	在运行 Prerequisite Scanner 之前，请确保用于运行 Prerequisite Scanner 并且要将结果输出目录保存到的磁盘未满；否则，以下错误消息将写至命令行界面: `ERROR: Cannot create files in output directory ips_output_dir. Exit.`
☐	如果 Prerequisite Scanner 生成了返回码 2，那么表明可能发生了脚本使用错误或收集器错误。请查看与此错误代码相关联的原因。如果发生了脚本使用错误，请使用正确的语法重新运行 Prerequisite Scanner。

相关概念:

在 UNIX 系统上，缺省情况下不会将消息写入日志文件。您可以使用 **debug** 和 **trace** 输入参数来启用调试或跟踪功能。Scanner 将调试和跟踪数据写入不同的日志文件，并使用时间戳记对步骤或函数的开始时间和结束时间进行标记。您可以使用这两个文件来关联特定的问题、函数或先决条件检查并对其进行故障诊断。

Prerequisite Scanner 根据扫描结果以及它是否必须因为错误而退出来生成返回码。这些返回码将写入日志文件。

`prereq_checker` 脚本运行 IBM Prerequisite Scanner，并根据您运行该脚本时指定的一组参数来检查先决条件。

返回码

Prerequisite Scanner 根据扫描结果以及它是否必须因为错误而退出来生成返回码。这些返回码将写入日志文件。

Prerequisite Scanner 根据一组预定义的结果来生成返回码，如下所示:

返回码	描述
`0`	如果 Prerequisite Scanner 成功运行且所有扫描结果均为 `PASS`，那么将返回 `0`。
`1`	如果 Prerequisite Scanner 成功运行，但是一项或多项先决条件检查返回了 `FAIL`，那么将返回 `1`。
`2`	如果 Prerequisite Scanner 未成功运行，并且由于下列各类错误而必须退出，那么将返回 `2`。 • 脚本使用错误 • 收集器错误 • 其他错误

脚本使用错误

Prerequisite Scanner 可能由于运行脚本时发生下列任何使用错误而退出:

- **Product_Code** 输入参数无效；例如，找不到该参数或者该参数的格式不受支持。
- **Product_Code** 和 **Product_Version** 输入参数的模式无效；例如，在引号内不仅提供了代码和版本，或者未将模式括在引号内。
- **Product_Version** 输入参数无效；例如，产品版本并非全是数字字符。
- 未在命令行界面中提供输入参数。
- 在命令行界面中输入的语法不正确；例如，输入了不受支持的命令行参数。
- 未提供必需的 **Product_Code** 输入参数。

收集器错误

Prerequisite Scanner 可能由于下列任何收集器错误而退出:

- 在 *ips_output_dir*`/temp` 目录中找不到收集器的临时结果文件。
- 未正确执行收集器的脚本文件。

其他错误

Prerequisite Scanner 可能由于用户对 *ips_output_dir* 输出目录不具有写许可权而退出。

相关概念:

IBM Prerequisite Scanner 生成以下屏幕和人类可读文件格式的输出: 发送到命令行界面、调试和跟踪日志文件以及文本和 XML 结果文件的输出。

附录 A. 产品代码参考

IBM Prerequisite Scanner 使用多字符代码 *product_code* 来标识产品、各个受支持的平台以及操作系统版本。*ips_root*/codename.cfg 文件包含用于表示产品的产品代码、其支持的平台以及操作系统版本的名称/值对。

表 14 对当前的预定义产品代码集作了概述。

限制: IBM Tivoli Monitoring 和 Tivoli Composite Application Manager 具有预定义的产品代码，Prerequisite Scanner 将这些代码视为保留代码。除非这些代码引用与其相关联的 IBM Tivoli Monitoring 和 Tivoli Composite Application Manager 代理程序，否则不得用作 Prerequisite Scanner 产品代码。有关产品代码的更多信息，请参阅 ITM 6.X 产品代码技术说明。

限制: 仅限于 UNIX: 在文件中输入产品代码的值时，请避免使用 for。这是一个保留字，可能会影响 Prerequisite Scanner 的运行方式。

表 14. 预定义的产品代码

预定义的产品代码	平台	产品版本、平台和操作系统
ADE	全部	Autonomic Deployment Engine
BSM	全部	Tivoli Business Service Manager
CDB	全部	Tivoli Composite Application Manager (ITCAM) for Applications: DB2
COA	UNIX	Tivoli Provisioning Manager for UNIX
COB	AIX	Tivoli Provisioning Manager for AIX
COC	AIX	Tivoli Provisioning Manager for AIX V5.3.0.0（64 位）
COD	AIX	Tivoli Provisioning Manager for AIX 6.1
COE	Linux	Tivoli Provisioning Manager for Linux
COF	Linux	Tivoli Provisioning Manager for Red Hat Linux
COG	Linux	Tivoli Provisioning Manager V7.2 for Red Hat Enterprise Linux 5 x86（64 位）
COH	Linux	Tivoli Provisioning Manager for Red Hat Enterprise Linux 5 System z®（64 位）
COI	Linux	Tivoli Provisioning Manager for SUSE 10
COJ	Solaris	Tivoli Provisioning Manager V7.2 for Solaris
COK	HP-UX	Tivoli Provisioning Manager V7.2 for HP-UX
COL	Linux	Tivoli Provisioning Manager V7.2 for SUSE zSeries® 10
COM	Linux	Tivoli Provisioning Manager V7.2 for SUSE 11
CON	Linux	Tivoli Provisioning Manager V7.2 for SUSE zSeries 11
COX	Windows	Tivoli Provisioning Manager V7.2 for Windows 2008
COY	Windows	Tivoli Provisioning Manager V7.2 for Windows 2003
COZ	Windows	Tivoli Provisioning Manager V7.2 for Windows

表 14. 预定义的产品代码 (续)

预定义的产品代码	平台	产品版本、平台和操作系统
DMO	全部	Prerequisite Scanner 演示
GYM	UNIX	IBM Tivoli Netcool® Performance Manager
KCJ	Windows	Tivoli Enterprise Portal Client
	UNIX	Tivoli Enterprise Portal Client for UNIX
KCQ	Windows	Tivoli Enterprise Portal Server
	UNIX	Tivoli Enterprise Portal Server for UNIX
KHD	全部	Warehouse Proxy Agent
KHE	UNIX	Warehouse Proxy Agent for UNIX
KIS	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Internet Service Monitoring
KLZ	UNIX	Tivoli Monitoring Operating System Agent for Linux
KM6	Windows	IBM Tivoli Composite Application Manager Agent for WebSphere MQ File Transfer Edition
KMQ	全部	Tivoli Composite Application Manager Agent for WebSphere MQ
KMS	Windows	Tivoli Enterprise Monitoring Server
	UNIX	Tivoli Enterprise Monitoring Server for UNIX
KNT	Windows	Tivoli Monitoring Operating System Agent for Windows
	UNIX	Windows OS Monitoring Agent for UNIX
KOR	Windows	Tivoli Monitoring Agent for Oracle
KQI	全部	Tivoli Composite Application Manager Agent for WebSphere Message Broker
KSY	Windows	Summarization and Pruning Agent
	UNIX	Summarization and Pruning Agent for UNIX
KUD	Windows	Tivoli Monitoring Agent for DB2
	UNIX	Tivoli Monitoring Agent for DB2
KTO	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Reporter
KTU	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Collector
KT3	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Application Management Console
KT4	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Client Response Time
KT5	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Web Response Time
KT6	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Robotic Response Time
KZE	全部	Tivoli zEnterprise Monitoring Agent
LCM	Windows	Tivoli License Compliance Manager
	UNIX	Tivoli License Compliance Manager for UNIX

表 14. 预定义的产品代码 (续)

预定义的产品代码	平台	产品版本、平台和操作系统
NCI	全部	Tivoli Netcool/Impact
NOC	全部	Tivoli Netcool/OMNIbus 服务器组件和桌面组件
NOD	全部	Tivoli Netcool/OMNIbus 桌面组件
NOS	全部	Tivoli Netcool/OMNIbus 服务器组件
PAE	全部	Tivoli Process Automation Engine
TAD	Windows	Tivoli Asset Discovery for Distributed
	UNIX	Tivoli Asset Discovery for Distributed for UNIX
TCR	全部	Tivoli Common Reporting
TPM	全部	Tivoli Provisioning Manager

附录 B. 配置文件参考

IBM Prerequisite Scanner 提供了一组可供您编辑的预定义配置文件。这些文件位于 *ips_root*/UNIX_Linux 或 *ips_root*/Windows 中。文件扩展名为 .cfg。

表 15 列示了当前支持的预定义配置文件。

表 15. 预定义配置文件

配置文件	平台	产品版本、平台和操作系统
ADE_01040000.cfg	全部	Autonomic Deployment Engine V1.4
BSM_04210000.cfg	全部	Tivoli Business Service Manager V4.2.1
BSM_06100000.cfg	全部	Tivoli Business Service Manager V6.1
CDB_06220000.cfg	全部	Tivoli Composite Application Manager (ITCAM) for Applications: DB2 V6.2.2
COA_07200000.cfg	UNIX	Tivoli Provisioning Manager V7.2 for UNIX
COB_07200000.cfg	AIX	Tivoli Provisioning Manager V7.2 for AIX
COC_07200000.cfg	AIX	Tivoli Provisioning Manager V7.2 for AIX V5.3.0.0（64 位）
COD_07200000.cfg	AIX	Tivoli Provisioning Manager V7.2 for AIX 6.1
COE_07200000.cfg	Linux	Tivoli Provisioning Manager V7.2 for Linux
COF_07200000.cfg	Linux	Tivoli Provisioning Manager V7.2 for Red Hat Linux
COG_07200000.cfg	Linux	Tivoli Provisioning Manager V7.2 for Red Hat Enterprise Linux 5 x86（64 位）
COH_07200000.cfg	Linux	Tivoli Provisioning Manager V7.2 for Red Hat Enterprise Linux 5 System z（64 位）
COI_07200000.cfg	Linux	Tivoli Provisioning Manager V7.2 for SUSE 10
COJ_07200000.cfg	Solaris	Tivoli Provisioning Manager V7.2 for Solaris
COK_07200000.cfg	HP-UX	Tivoli Provisioning Manager V7.2 for HP-UX
COL_07200000.cfg	Linux	Tivoli Provisioning Manager V7.2 for SUSE zSeries 10
COM_07200000.cfg	Linux	Tivoli Provisioning Manager V7.2 for SUSE 11
CON_07200000.cfg	Linux	Tivoli Provisioning Manager V7.2 for SUSE zSeries 11
COX_07200000.cfg	Windows	Tivoli Provisioning Manager V7.2 for Windows 2008
COY_07200000.cfg	Windows	Tivoli Provisioning Manager V7.2 for Windows 2003
COZ_07200000.cfg	Windows	Tivoli Provisioning Manager V7.2 for Windows
DMO_00000000.cfg	全部	Prerequisite Scanner 演示
DMO_01000000.cfg	全部	Prerequisite Scanner V1.0 演示
GYM_01030200.cfg	UNIX	IBM Tivoli Netcool Performance Manager V1.3.2
KCJ_06200000.cfg	Windows	Tivoli Enterprise Portal Client V6.2
KCJ_06210000.cfg	UNIX	Tivoli Enterprise Portal Client V6.2.1
KCJ_06220000.cfg	全部	Tivoli Enterprise Portal Client V6.2.2
KCQ_06200000.cfg	Windows	Tivoli Enterprise Portal Server V6.2
KCQ_06210000.cfg	UNIX	Tivoli Enterprise Portal Server V6.2.2
KCQ_06220000.cfg	全部	Tivoli Enterprise Portal Server V6.2.2
KHD_06200000.cfg	Windows	Warehouse Proxy Agent V6.2

表 15. 预定义配置文件 (续)

配置文件	平台	产品版本、平台和操作系统
KHD_06210000.cfg	全部	Warehouse Proxy Agent V6.2.1
KHD_06220000.cfg	全部	Warehouse Proxy Agent V6.2.2
KHE_06220000.cfg	UNIX	Warehouse Proxy Agent V6.2.2
KIS_07200000.cfg	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Internet Service Monitoring V7.2
KIS_07300000.cfg	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Internet Service Monitoring V7.3
KLZ_06210000.cfg	UNIX	Tivoli Monitoring Operating System Agent for Linux V6.2.1
KLZ_06220000.cfg	UNIX	Tivoli Monitoring Operating System Agent for Linux V6.2.2
KM6_070100000.cfg	Windows	Tivoli Composite Application Manager Agent for WebSphere MQ File Transfer Edition V7.1
KMQ_070100000.cfg	全部	Tivoli Composite Application Manager Agent for WebSphere MQ V7.1
KMS_06200000.cfg	Windows	Tivoli Enterprise Monitoring Server V6.2
KMS_06210000.cfg	全部	Tivoli Enterprise Monitoring Server V6.2.1
KMS_06220000.cfg	全部	Tivoli Enterprise Monitoring Server V6.2.2
KNT_06200000.cfg	Windows	Tivoli Monitoring Operating System Agent for Windows V6.2
KNT_06210000.cfg	Windows	Tivoli Monitoring Operating System Agent for Windows V6.2.1
KNT_06220000.cfg	Windows	Tivoli Monitoring Operating System Agent for Windows V6.2.2
KOR_06220000.cfg	Windows	Tivoli Monitoring Agent for Oracle V6.2.2
KQI_07010000.cfg	全部	Tivoli Composite Application Manager Agent for WebSphere Message Broker V7.1
KSY_06200000.cfg	Windows	Summarization and Pruning Agent V6.2
KSY_06210000.cfg	全部	Summarization and Pruning Agent V6.2.1
KSY_06220000.cfg	全部	Summarization and Pruning Agent V6.2.2
KTO_07200000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Reporter V7.2
KTO_07200200.cfg	Windows	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Reporter V7.2.2
KTO_07300000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Reporter V7.3
KTU_07200000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Collector V7.2
KTU_07200200.cfg	Windows	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Collector V7.2.2
KTU_07300000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Collector V7.3
KT3_07300000.cfg	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Application Management Console V7.3
KT4_07300000.cfg	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Client Response Time V7.3
KT5_07300000.cfg	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Web Response Time V7.3

表 15. 预定义配置文件 (续)

配置文件	平台	产品版本、平台和操作系统
KT6_07300000.cfg	全部	Tivoli Composite Application Manager (ITCAM) for Transactions: Robotic Response Time V7.3
KUD_06100000.cfg	Windows	Tivoli Monitoring Agent for DB2 V6.1
KUD_06200000.cfg	全部	Tivoli Monitoring Agent for DB2 V6.2
KUD_06210000.cfg	全部	Tivoli Monitoring Agent for DB2 V6.2.1
KUD_06220000.cfg	全部	Tivoli Monitoring Agent for DB2 V6.2.2
KZE_06020300.cfg	全部	Tivoli zEnterprise Monitoring Agent V6.2.3
LCM_01000000.cfg	全部	Tivoli License Compliance Manager V1.0
LCM_02300000.cfg	全部	Tivoli License Compliance Manager V2.3
NCI_06100000.cfg	全部	Tivoli Netcool/Impact V6.1
NOC_07310000.cfg	全部	Tivoli Netcool/OMNIbus 服务器组件和桌面组件 V7.3.1
NOD_07310000.cfg	全部	Tivoli Netcool/OMNIbus 桌面组件 V7.3.1
NOS_07310000.cfg	全部	Tivoli Netcool/OMNIbus 服务器组件 V7.3.1
PAE_07500000.cfg	全部	Tivoli Process Automation Engine
TAD_07200000.cfg	全部	Tivoli Asset Discovery for Distributed V7.2
TAD_07220000.cfg	全部	Tivoli Asset Discovery for Distributed V7.2.2
TCR_02010100.cfg	全部	Tivoli Common Reporting
TPM_07210000.cfg	全部	Tivoli Provisioning Manager V7.2.1

附录 C. 先决条件属性参考

本参考资料概述硬件和软件先决条件的各个预定义类别的基本先决条件属性。

表 16 对硬件和软件先决条件的预定义类别作了概述。

表 16. 先决条件属性的预定义类别

数据类别	描述	必需的前缀标识	参考
公共	公共数据属性用于检查公共先决条件，例如处理器速度、内存量、磁盘空间量和临时空间量。	无	第 82 页的『公共数据属性』
Autonomic Deployment Engine	Autonomic Deployment Engine 数据属性用于检查 Autonomic Deployment Engine 先决条件，例如安装单元。	de	第 85 页的『Autonomic Deployment Engine 数据属性』
已安装的软件	“已安装的软件”数据属性用于检查已安装的软件先决条件（例如在 Windows 注册表中注册的程序）以及是否安装了 cygwin 和 gskit。	无	第 99 页的『“已安装的软件”数据属性』
用户	用户数据属性用于检查用户先决条件，例如登录用户是否具有管理权以及是否为 root 用户。	user	第 99 页的『用户数据属性』
操作系统	操作系统数据属性用于检查操作系统先决条件，例如版本、体系结构、内存总量、可用内存量和物理内存总量。	os	第 89 页的『操作系统数据属性』
连通性	连通性数据属性用于检查连通性先决条件，例如 Telnet 是否处于运行状态以及 Scanner 所能够连接到的 IP 地址和端口。	无	第 86 页的『连通性数据属性』
网络	网络数据属性用于检查所有平台的公共网络先决条件，例如是否有可用的端口。	network	第 88 页的『网络数据属性』
Windows 网络	Windows 网络数据属性用于检查网络先决条件，例如是否在机器上启用了 NetBIOS 和 DHCP 以及 Ping 属性。	network	第 99 页的『Windows 网络数据属性』
UNIX 网络	UNIX 网络数据属性用于检查网络先决条件，例如是否在机器上启用了 NetBIOS 和 DHCP 以及 Ping 属性。	network	第 100 页的『UNIX 网络数据属性』
Internet Explorer	Microsoft Internet Explorer 数据属性用于检查 Internet Explorer 先决条件，例如版本。	internetExplorer	第 87 页的『Internet Explorer 数据属性』
数据库服务器: DB2	DB2 数据属性用于检查 DB2 先决条件，例如版本。	DB2	第 86 页的『DB2 数据属性』
数据库服务器: MS SQL	MS SQL Server 数据属性用于检查 MS SQL Server 先决条件，例如版本。	mssql	第 87 页的『MS SQL Server 数据属性』
数据库服务器: Oracle	Oracle 数据属性用于检查 Oracle 先决条件，例如版本。	Oracle	第 89 页的『Oracle 数据属性』
环境变量	环境变量用于检查环境变量先决条件，例如是否设置了环境变量。	env	第 100 页的『环境变量数据属性』

公共数据属性

公共数据属性用于检查公共先决条件，例如 CPU 速度、内存量、磁盘空间量和临时空间量。对于 Windows 系统，它使用 IBM Prerequisite Scanner 主脚本。对于 UNIX 系统，它使用 Prerequisite Scanner 主脚本和公共收集器 *ips_root*/Unix_Linux/common.sh。

表 17 对公共数据先决条件属性作了概述。此类先决条件属性不需要前缀标识。

表 17. 公共数据先决条件属性

先决条件属性	平台	描述	有效值
CPU Name	全部	CPU 的名称，仅用于在结果中显示	不适用
CpuArchitecture	UNIX	操作系统的体系结构	包含多个以逗号分隔的受支持值的字符串，例如: x86_64,s390x,ppc64,AMD64
DBType	全部	用于检查机器上安装的数据库服务器的类型。 仅适用于 UNIX 系统上的 Oracle: 收集器期望 $HOME/.profile 文件中设置了 ORACLE_BASE 和 ORACLE_HOME 环境变量，例如: export ORACLE_BASE=/home/oracle/app/oracle/product/11.2.0/ export ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1 其中 $HOME 必须是 Oracle 用户的主目录 /home/oracle。	值可以是下列任意类型: • 用于表示任何数据库服务器类型的字符串，例如: any • 用于表示数据库服务器类型的字符串，例如: Oracle • regex{*str*}，这是具有输入参数 *str* 的正则表达式，用于表示数据库服务器类型的搜索模式，例如: regex{.*MSSQL.*|DB2.*} 用于检查数据库服务器类型是 Windows 系统上的 MS SQL 还是 DB2。 • 用于表示无数据库服务器类型的字符串，例如: unknown

表 17. 公共数据先决条件属性 (续)

先决条件属性	平台	描述	有效值
DBTypeDetails	全部	机器上安装的数据库服务器的类型。 仅适用于 UNIX 系统上的 Oracle: 收集器期望 $HOME/.profile 文件中设置了 ORACLE_BASE 和 ORACLE_HOME 环境变量，例如: export ORACLE_BASE=/home/oracle/app/oracle/product/11.2.0/ export ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1 其中 $HOME 必须是 Oracle 用户的主目录 /home/oracle。 先决条件属性将有关数据库服务器类型的详细信息（即数据库服务器类型、安装位置和版本）写入 result.txt 文件。各种数据库服务器类型的详细信息以分号分隔。	值可以是下列任意类型: • 用于表示任何数据库服务器类型的字符串，例如: any • 用于表示一种数据库服务器类型的字符串，例如: DB2 • regex{*str*}，这是具有输入参数 *str* 的正则表达式，用于表示数据库服务器类型的搜索模式，例如: regex{.*MSSQL.*|DB2.*} 用于检查数据库服务器类型是 Windows 系统上的 MS SQL 还是 DB2。
Disk	Windows	这是具有下列可选限定属性的可用磁盘空间量: • dir 属性，用于确定要检查的目录路径 • unit 属性，用于确定要使用的磁盘空间单位	值可以是下列任意类型: • 具有以下限定符格式的字符串: [dir:*dir_path*, unit:*unit_name*] *disk_space* 例如: Disk=[dir:C:\Program Files\IBM\SQLLIB, unit:MB]1431 • 以 MB 或 GB 为单位的数字格式: *disk_space*MB|GB 例如: Disk=250MB
Disk	UNIX	这是可用磁盘空间量	以 GB 或 MB 为单位的数字格式，例如: 2GB
intel.cpu	全部	这是 Intel 处理器的 CPU 速度	以 GHz 为单位的数字格式，并且在 Windows 上仅以 MHz 为单位，例如: 2GHz
Memory	全部	这是机器上的当前可用物理内存总量。 **提示**: 通过使用操作系统类别中的预定义先决条件属性，分别检查可用物理内存量和虚拟内存量。	以 GB 或 MB 为单位的数字格式，例如: 300MB

表 17. 公共数据先决条件属性 (续)

先决条件属性	平台	描述	有效值
OS Version	全部	这是机器上运行的操作系统的全名和版本；另外，您也可以使用正则表达式来传递用于表示多个操作系统变体的字符串。**提示**：请将此先决条件属性与 os.servicePack 和 os.architecture 配合使用，以检查当前的 Service Pack 和系统体系结构。	值可以是下列任意类型: • 可以表示多个版本的字符串（各个版本之间以逗号分隔），例如: RedHat Enterprise Linux 6.*, SuSE Linux Enterprise Server 11, SuSE Linux Enterprise Server 10, SuSE Linux Enterprise Server 9, AIX V6.1,AIX V5.3 **限制**: 在 Windows 系统上，* 通配符仅在正则表达式中受支持。 • regex{*str*}，这是具有输入参数 *str* 的正则表达式，用于表示版本的搜索模式，例如: regex{Windows 200[3-8]} 检查实际操作系统是否与任何从 Windows 2003 到 Windows 2008 的版本匹配。 regex{Red Hat*.*} 检查实际操作系统是否与 Red Hat Linux 的某个变体匹配。 **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
numCPU	全部	这是计算机上的核心或独立处理器的数量。如果此工具在扫描计算机后未找到任何核心或者找到均不是核心的处理器，那么它将返回“找不到”结果。	数字，例如 4
risc.cpu	UNIX	这是 RISC 处理器的 CPU 速度	以 GHz 为单位的数字格式，例如: 1.4GHz
Temp	UNIX	这是所指定 *Temp* 文件系统的可用磁盘空间量	以 GB 或 MB 为单位的数字格式，例如: 300MB

相关概念:

根据计算机上是否已在运行 Tivoli Monitoring 或 Tivoli Composite Application Manager 代理程序，Prerequisite Scanner 以不同的方式检查 Memory 先决条件属性。

相关参考:

操作系统数据属性用于检查操作系统先决条件，例如版本、体系结构、内存总量、可用内存量和物理内存总量。仅限于 Windows 系统: 它使用 *ips_root*/lib 目录中文件名包含 os 前缀标识的操作系统 VBScript 收集器。仅限于 UNIX 系统: 它使用 *ips_root*/UNIX_Linux 目录中文件名包含 os 前缀的 UNIX 操作系统收集器。

内存先决条件属性和 Tivoli Monitoring 代理程序的系统行为

根据计算机上是否已在运行 Tivoli Monitoring 或 Tivoli Composite Application Manager 代理程序，Prerequisite Scanner 以不同的方式检查 Memory 先决条件属性。

如果已安装代理程序，那么 Prerequisite Scanner 将根据新配置文件与现有配置文件（如果现有配置文件仍在计算机上）的期望值之差使用 Memory 先决条件属性的期望值；否则，它将按缺省行为处理期望值。

您运行 Prerequisite Scanner 以便针对正在升级或重新安装的 Tivoli Monitoring 代理程序检查先决条件时，它将首先检查该代理程序是否已在计算机上运行。如果该代理程序正在运行，那么 Prerequisite Scanner 将搜索与正在运行的代理程序的现有版本相关联的配置文件。根据搜索结果不同，行为如下:

- 如果找不到配置文件，那么 Prerequisite Scanner 将假定先前未对目标环境执行扫描；因此，Prerequisite Scanner 将使用新配置文件中指定的 Memory 先决条件属性的期望值，这与缺省行为一致。Prerequisite Scanner 将该期望值写入结果输出。
- 如果找到配置文件，那么 Prerequisite Scanner 将现有版本的 Memory 先决条件属性的期望值与新版本配置文件中的期望值进行比较。如果这两个值有差别，并且新值大于现有期望值，那么 Prerequisite Scanner 将差值设置为期望值。Prerequisite Scanner 会将期望值差值写入结果输出。例如，代理程序 V1 的配置文件指定 1 GB 作为期望值。代理程序 V2 的新配置文件指定 1.5 GB 作为期望值；因此，Prerequisite Scanner 将使用并写入 0.5 GB 作为期望值差值。

Autonomic Deployment Engine 数据属性

Autonomic Deployment Engine 数据属性用于检查 Autonomic Deployment Engine 先决条件，例如安装单元。仅限于 Windows 系统: 它使用 *ips_root*/lib/ 目录中文件名包含 de 前缀的 Autonomic Deployment Engine 收集器。仅限于 UNIX 系统: 它使用 *ips_root*/UNIX_Linux 目录中文件名包含 de 前缀的 UNIX Autonomic Deployment Engine 收集器。

表 18 对先决条件属性作了概述。此类先决条件属性需要 de 前缀标识。

表 18. Autonomic Deployment Engine 数据属性

先决条件属性	平台	描述	有效值
de.installed	全部	用于检查是否已安装	布尔值，例如: true|false

表 18. Autonomic Deployment Engine 数据属性 (续)

先决条件属性	平台	描述	有效值
de.installationUnit	全部	通过使用 **listIU** - v 命令，检查是否安装了指定的安装单元	值可以是下列任意类型: • 用于表示单一安装单元的字符串，例如 Tivoli Integrated Portal 的安装单元: C37109911C8A11D98E1700061BDE7AEA, B24209911C8A11D98E1700061BDE7AEA • 用于表示多个安装单元的字符串，例如: 5FFE79F918DF3BA0D67511FD3F7C358E • regex {str}，这是包含输入参数 str 的正则表达式，用于表示安装单元、版本和安装路径的搜索模式；例如要检查安装单元、WebSphere Application Server 版本以及 Tivoli Integrated Portal 的安装路径，搜索模式如下所示: • regex{.*C00DA95AFD9B7E0397153CD944B5A255. *6.1.0.2100.*SIU eWAS. *C:\\IBM\\tivoli\\tip.*} **注**: 另外，还可以在安装路径中使用环境变量；例如，通过将路径替换为 TIPHOME 环境变量，搜索模式为: regex{.*C00DA95AFD9B7E0397153CD944B5A255. *6.1.0.2100.*SIU eWAS. *%TIPHOME%.*} • 用于表示多项检查的多个 regex {str} 参数；例如: regex{.*C37109911C8A11D98E1700061BDE7AEA.*}, regex{.*B24209911C8A11D98E1700061BDE7AEA.*}

连通性数据属性

连通性数据属性用于检查连通性先决条件，例如 Telnet 是否处于运行状态以及 Scanner 所能够连接到的 IP 地址和端口。对于 Windows 系统，它使用连通性收集器 *ips_root*/lib/connectivity_plug.vbs。对于 UNIX 系统，它使用 IBM Prerequisite Scanner 主脚本和连通性收集器 *prs_root*/Unix_Linux/connectivity_plug.sh。输出将仅传递到调试日志文件中。

DB2 数据属性

DB2 数据属性用于检查 DB2 先决条件，例如版本。仅限于 Windows 系统: 它使用 DB2 收集器 *ips_root*/lib/db2_version_plug.bat。仅限于 UNIX 系统: 它使用 *ips_root*/UNIX_Linux 目录中文件名包含 db2 前缀的 UNIX DB2 收集器。

第 87 页的表 19 对 DB2 先决条件属性作了概述。此类先决条件属性需要 DB2 前缀标识。

表 19. DB2 数据属性

先决条件属性	平台	描述	有效值
DB2 Version	全部	这是机器上当前安装的 DB2 版本。	字符串，例如: v9.5.100.179FP4
db2.home.space	UNIX	这是 DB2 主目录的可用磁盘空间量	以 GB 为单位的数字格式，例如: 8GB

MS SQL Server 数据属性

MS SQL Server 数据属性用于检查 MS SQL Server 先决条件，例如版本和位置。仅限于 Windows 系统: 它使用 *ips_root*/Windows 目录中文件名包含 mssql 前缀的 MS SQL Server 收集器。

表 20 对 MS SQL Server 先决条件属性作了概述。此类先决条件属性需要 mssql 前缀标识。

表 20. MS SQL Server 数据属性

先决条件属性	平台	描述	有效值
mssql.Client	Windows	用于检查机器上当前安装的 MS SQL 客户机版本。	期望的字符串值可以是多个以逗号分隔的版本，例如: 10.50.1600.1 **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
mssql.Server	Windows	用于检查机器上当前安装的 MS SQL Server 版本。	期望的字符串值可以是多个以逗号分隔的版本，例如: 10.50.1600.1 **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
mssql.Server.Location	Windows	用于检查 MS SQL 数据库服务器的主目录	字符串，例如: any

Internet Explorer 数据属性

Microsoft Internet Explorer 数据属性用于检查 Internet Explorer 先决条件，例如版本。它使用 Internet Explorer 收集器 *ips_root*/lib/internetExplorer_plug.vbs。

表 21 对 Internet Explorer 先决条件属性作了概述。此类先决条件属性需要 internetExplorer 前缀标识。

表 21. Internet Explorer 数据属性

先决条件属性	描述	有效值
internetExplorer. version	这是机器上安装的 Internet Explorer 版本	数字格式，例如 7.0+ **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。

网络数据属性

网络数据属性用于检查所有平台的公共网络先决条件，例如是否有可用的端口。它使用 *ips_root*/lib 目录中文件名包含 network 前缀标识的网络收集器。

表 22 对所有平台之间的公共网络先决条件属性作了概述。此类先决条件属性需要 network 前缀标识。

表 22. 网络数据属性

先决条件属性	平台	描述	有效值
network.availablePorts.*app_type*	全部	请使用此命名约定来检查端口或范围端口是否可用于 *app_type* 应用程序类型。您可以检查哪些端口未被侦听，例如: • network.availablePorts.DB2 用于检查 DB2 数据库服务器的端口，其中 *app_type* 为 DB2 • network.availablePorts.WAS 用于检查 WebSphere Application Server 的端口，其中 *app_type* 为 WAS	正整数，例如: • network.availablePorts.DB2 = 50000-50005 • network.availablePorts.WAS = 8080 **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
network.portsInUse.*app_type*	全部	请使用此命名约定来检查端口或范围端口是否正用于 *app_type* 应用程序类型。您可以检查哪些端口正被侦听，例如: • network.availablePorts.DB2 用于检查 DB2 数据库服务器的端口，其中 *app_type* 为 DB2 • network.availablePorts.WAS 用于检查 WebSphere Application Server 的端口，其中 *app_type* 为 WAS	正整数，例如: • network.portsInUse.DB2 = 50900-50905 • network.portsInUse.WAS = 8080 **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
network.validate HostsFile	Windows	用于检查是否 hosts 文件中列示的所有主机都具有以下格式: *IP_Address Host_Name Short_Name* 其中: • *IP_Address* 是计算机的 IP，例如 127.0.0.1 • *Host_Name* 是计算机的标准主机名，例如 localhost.localdomain • *Short_Name* 是计算机的短名称，例如 localhost	布尔值，例如 True

Oracle 数据属性

Oracle 数据属性用于检查 Oracle 先决条件，例如版本。仅限于 Windows 系统：它使用 Oracle 收集器。仅限于 UNIX 系统：它使用 *ips_root*/UNIX_Linux 目录中文件名包含 oracle 前缀的 UNIX Oracle 收集器。仅限于 Window 系统：它使用 *ips_root*/lib 目录中文件名包含 oracle 前缀的 Windows Oracle 收集器。

表 23 对 Oracle 先决条件属性作了概述。此类先决条件属性需要 oracle 前缀标识。

表 23. Oracle 数据属性

先决条件属性	平台	描述	有效值
ORACLE Version	Windows	用于检查机器上当前安装的 Oracle 版本。	期望的字符串值可以是多个以逗号分隔的版本，例如: 9.2, 10.1, 10.2
oracle.Client	全部	用于检查机器上当前安装的 Oracle 客户机版本。	期望的字符串值可以是多个以逗号分隔的版本，例如: 9.2.0.8+ **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
oracle.Client.Location	全部	用于检查 Oracle 客户机的主目录	字符串，例如: /opt/oracle/products/10.1.0/client_1
oracle.Server	全部	用于检查机器上当前安装的 Oracle 服务器版本。	期望的字符串值可以是多个以逗号分隔的版本，例如: 10.2.0.4g,11g R1 **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
oracle.Server.Location	全部	用于检查 Oracle 数据库服务器的主目录	字符串，例如: /opt/oracle/product/10.1.0/Db_1

操作系统数据属性

操作系统数据属性用于检查操作系统先决条件，例如版本、体系结构、内存总量、可用内存量和物理内存总量。仅限于 Windows 系统：它使用 *ips_root*/lib 目录中文件名包含 os 前缀标识的操作系统 VBScript 收集器。仅限于 UNIX 系统：它使用 *ips_root*/UNIX_Linux 目录中文件名包含 os 前缀的 UNIX 操作系统收集器。

表 24 对操作系统先决条件属性作了概述。此类先决条件属性需要 os 前缀标识。

表 24. 操作系统数据属性

先决条件属性	平台	描述	有效值
os.architecture	全部	用于检查系统体系结构	32-bit|64-bit
os.automount	UNIX	用于检查自动装配功能是否正常工作	布尔值，例如: True
os.autoUpdateEnabled	Windows	用于检查是否自动启用 Windows Update；如果启用，那么将返回 True	布尔值，例如: True

表 24. 操作系统数据属性 (续)

先决条件属性	平台	描述	有效值
os.availableMemory	Windows	用于检查当前可用但未被操作系统使用的虚拟内存量	以 MB 为单位的数字格式，例如: 900MB
os.dir.*dir_name*	UNIX	根据下列限定属性来检查 *dir_name* 文件系统: • dir 属性，用于确定要检查的文件系统 • type 属性，用于确定要检查的文件系统的属性，例如对该文件系统的访问许可权的 *octal_digits* 八进制数表示 例如，*dir_name* 可以表示: • tmp • home	具有以下限定符格式的字符串: [dir:*dir_name*, type:permission] *octal_digits*+ 例如，要检查主目录是否具有 drwxr-xr-x 许可权: os.dir.home=[dir:/home, type:permission]755+
os.diskquota		用于检查登录用户的磁盘使用配额；返回以千字节为单位的配额值或者 Unlimited	值可以是下列任意类型: • 用于表示千字节数的值，例如 414000 • 用于表示不受限磁盘限额的字符串，例如 Unlimited
os.expectLink	UNIX	用于检查 TCL 的 Expect 扩展在机器上是否可用；如果其状态为可用，那么将返回 Available **注**: os.file.expect 先决条件属性用于检查是否在机器上安装了 Expect 扩展。	Available|Unavailable
os.file.*script_name*	UNIX	用于检查机器上是否有可用的 *script_name* 脚本。例如，*script_name* 可以表示: • bash • expect • gzip • tar	布尔值，例如: True
os.Firefox	UNIX	用于检查是否在机器上安装了 Mozilla Firefox；如果已安装，那么将返回 Available	Available|Unavailable
os.FreePagingSpace	UNIX	用于检查可用页面高速缓存的总大小	以 MB 或 GB 为单位的数字格式，例如: 4GB+ **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
os.ftpusers	UNIX	用于检查 root 用户是否列示在 ftpusers 文件（此文件用于确定无权进行 FTP 登录的用户）中；如果未列示该用户，那么返回 Available	Available|Unavailable

表 24. 操作系统数据属性 (续)

先决条件属性	平台	描述	有效值
os.gnu.tar	UNIX	用于检查 GNU tar 实用程序在机器上是否可用；如果已安装，那么将返回 Available	Available|Unavailable
os.hostformat	UNIX	用于检查 /etc/host 中的条目格式是否正确	布尔值，例如: True
os.iodevicestatus	AIX	用于检查异步 I/O (aio0)，即用于提高 I/O 操作性能的内核进程的状态；如果其状态为可用，那么将返回 Available	Available|Unavailable
os.is8dot3File FormatEnabled	Windows	用于检查是否自动应用了 8.3 文件名格式；如果已应用此格式，那么将返回 True	布尔值，例如: True
os.localhost InHostsFile	全部	检查 hosts 文件中是否包含将本地主机映射至 127.0.0.1 IP 地址的条目，例如: 127.0.0.1 localhost	布尔值，例如: True
os.isService Running.*service_name*	Windows	使用此命名约定来检查 *service_name* 是否正在机器上运行。例如，*service_name* 可以表示: • remoteRegistry（表示远程注册服务） • DNSClient（表示 DNS 客户机服务） • terminalServices（表示远程桌面服务或终端服务）	布尔值，例如: True
os.kernelMode	AIX	用于检查 CPU 体系结构是否支持内核方式或不受限方式	32-bit|64-bit
os.kernelParameters	Linux	用于检查内核参数是否可供操作系统使用	Available|Unavailable
os.kernelversion	Linux	用于检查 Linux 操作系统的内核发行版	数字格式，例如 2.6
os.largeFile	UNIX	用于检查大型文件支持	布尔值，例如: True
os.ldLibPath	UNIX	检查 LD_LIBRARY_PATH 环境变量是否存在并以冒号结尾，即 os.ldLibPath=[endsWith=:]	Available|Unavailable
os.level	AIX	用于检查 AIX 操作系统级别是否高于 10（对于 AIX V5.3），或者是否高于级别 3（对于 AIX V6.1）	布尔值，例如: True

表 24. 操作系统数据属性 (续)

先决条件属性	平台	描述	有效值
os.lib.*lib_name_version*	UNIX	用于检查机器上是否安装了 *lib_name* 库的受支持版本。例如，用于表示 *lib_name_version* 的字符串或正则表达式（显示为粗体）： • 32 位 **libstdc++.so.#** 库 • 64 位 **libstdc++.so.#** 库 • 32 位 **libXft.so.#** 库 • 32 位 **libXtst.so.#** 库 • 64 位 **libaio.so.#** 库 • 32 位 **xlC.rte** XLC 运行时级别 • AIX V5.3 的 32 位 **xlC.aix50.rte** XLC 运行时 • AIX V6.1 的 32 位 **xlC.aix61.rte** XLC 运行时 • AIX IOCP **bos.iocp.rte** 库 • **bos.loc.iso.en_us**，AIX 基本操作系统的 ISO 代码文件集	值可以是下列任意类型: 字符串，例如: • /usr/lib/libstdc++.so.# 作为 32 位 libstdc++.so.# 库的值 • /usr/lib64/libaio.so.# 作为 64 位 libaio.so.# 库的值 • AIX V5.3 的 xlC.aix50.rte.9.0.0.8+ as the value for the 32-bitxlC.aix50.rte XLC 运行时 • bos.loc.iso.en_us（表示 ISO 代码文件集） regex {*str*}，这是具有输入参数 *str* 的正则表达式，用于表示库名搜索模式，例如: regex{.*libgcc.*} 检查是否存在该操作系统的 GCC 低级别运行时库 libgcc 版本。 **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
os.loginVariable	UNIX	检查是否在 PATH 和 SUPATH 变量中设置了 root 用户的缺省路径；如果已进行设置，那么返回 Available	Available|Unavailable
os.maximoDirectory	UNIX	用于检查 /export/home/maximo 目录是否可用	Available|Unavailable
os.maximoDirOwner	UNIX	用于检查 /export/home/maximo 目录的所有者	maximo
os.maximumProcesses	UNIX	用于检查可以为每个用户运行的最大进程数	数字，例如 2048
os.MozillaVersion	UNIX	用于检查机器上不同于 os.Firefox 先决条件属性的特定 Mozilla Firefox 版本	数字格式，例如 3.0+ **注**: 该值可使用在 第 2 页的表 1 中概括出的特殊字符。

表 24. 操作系统数据属性 (续)

先决条件属性	平台	描述	有效值
os.mountcheck	UNIX	确定文件系统是否已基于下列条件属性装配: • drive 属性，用于确定安装了文件系统的目录 • nosuid 属性，用于确定是否设置了装配选项（如果已装配文件系统）	具有以下限定符格式的字符串: [drive:*dir_name*, *mount_option*: false|true] True|False 例如，要检查是否已装配 /home 目录以及是否未设置 nosuid 选项: os.mountcheck=[drive:/home, nosuid:false]True
os.package.*package_name*	UNIX	用于检查机器上是否安装了 *package_name* 软件包的受支持版本。例如，用于表示 *package_name* 的字符串（显示为粗体）: • **bash** Shell • **expect**（对于 TCL 扩展包） • **libgcc**（对于 GCC 低级别运行时包） • **openssh**（对于开放式源代码安全 Shell） • **openssl**（对于开放式源代码 SSL/TLS 工具箱） • **perl**（对于 Perl 脚本包） • **rpm**（对于 RPM 或 RPM 构件包） • **telnet**（对于 Telnet 包） • **wget**（对于 GNU 文件检索包）	字符串，例如: • bash-3.2+ for bash shell • expect-1.2.0（对于 Expect） • libgcc-3.4.3-9（对于 libgcc） • openssh-5.0.0.5301-（对于 openssh） • openssl-4.2.0-（对于 OpenSSL） • perl-5.8.2（对于 Perl） • rpm • telnet • wget **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
os.pagesize	UNIX	用于检查系统的页大小。	以 KB 为单位的数字格式，例如: 4KB **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。
os.RAMSize	全部	用于检查系统的内存量	以 GB 为单位的数字格式，例如 8GB
os.SeaMonkeyVersion	UNIX	用于在 PATH 环境变量中设置了特定版本的 Mozilla SeaMonkey 的路径时，在计算机上查找此特定版本的 Mozilla SeaMonkey	数字格式，例如 2.10 **注**: 这些值可以使用第 2 页的表 1 中概述的特殊字符。

表 24. 操作系统数据属性 (续)

先决条件属性	平台	描述	有效值
os.SELinux	Linux	用于根据下列限定属性来检查 Security-Enhancement Linux 功能的实施状态: • source 属性，用于确定要用于相关操作系统的命令	值可以是下列任意类型: • 具有以下限定符格式的字符串: [source:Command] Disabled|Enabled 例如，要检查功能部件在 Red Hat 或 SUSE 操作系统上是处于禁用状态还是处于许可状态: os.SELinux=[source: Command]Disabled • 不具有限定符的字符串，其中操作系统为一般 Linux 变体: os.SELinux=Disabled
os.servicePack	全部	用于检查已安装 Service Pack 的当前版本	数字格式，仅具有 *majorVersion.minorVersion* 或 *majorVersion* 版本 例如，要检查是否安装了 Service Pack 2 或更高版本，2+ **注**: 该值可使用在 第 2 页的表 1 中概括的特殊字符。
os.shell.default	UNIX	检查是否已安装了缺省 shell 脚本	标识 shell 脚本的字符串，例如，bash

os.space.*dir_name* 先决条件属性

Prerequisite Scanner 具有 os.space.*dir_name* 属性的三个变体:

- os.space.*dir_name*，用于检查指定的文件系统是否有足够的可用磁盘空间，而与登录用户是否始终为 root 用户或非 root 用户无关。

  如果要检查文件系统的指定路径，而这与登录用户是否始终为 root 用户或非 root 用户无关，请使用此先决条件属性变体。
  注: 不得对同一文件系统使用该变体两次（但单一配置文件中的不同用户类型除外）；请改为使用另外两种变体的组合。
- os.space.*dir_name*_nonroot，用于检查非 root 用户的指定文件系统是否有足够的可用磁盘空间。

  如果您以非 root 用户身份登录，并且希望显式地检查文件系统的指定路径，请使用此先决条件属性变体。
  注: 非 root 用户与目标系统上安装产品的用户应该是同一用户。
- os.space.*dir_name*_root，用于检查 root 用户的指定文件系统是否有足够的可用磁盘空间。

  如果您以 root 用户身份登录，并且希望显式地检查文件系统的指定路径，请使用此先决条件属性变体。

可以在同一配置文件中指定 os.space.*dir_name*_nonroot 和 os.space.*dir_name*_root 变体。对于不适用的变体，Prerequisite Scanner 将在实际结果单元中输出 NOT_REQ_CHECK_ID。例如，如果登录用户为 root 用户，那么 Prerequisite Scanner 将对 os.space.*dir_name*_nonroot 变体输出 NOT_REQ_CHECK_ID。

表 24. 操作系统数据属性 (续)

先决条件属性	平台	描述	有效值
os.space.*dir_name*	UNIX	用于根据下列一个或多个限定属性来检查所指定 <*dir_name*> 文件系统的可用磁盘空间量: • dir 属性，用于确定要检查的文件系统路径 • unit 属性，用于确定要使用的磁盘空间单位 dir 属性的值依赖于登录用户；因此，该值是一个用于表示用户类型（即 root 或非 root）和相关路径的名称/值对。 例如，*dir_name* 可以表示: • home • opt • tmp • usr • var **注:** 不得对同一文件系统使用该变体两次（但单一配置文件中的不同用户类型除外）。请使用 os.space.*dir_name*_nonroot 变体与 os.space.*dir_name*_root 变体的组合。	具有以下限定符格式的字符串，用于指定 root 用户的文件系统: [dir:**root**=<*dir_path*>, unit:<*unit_name*>] <*disk_space*> 例如: os.space.usr= [dir:root=/usr/ibm/common/ acsi, unit:GB]200 具有以下限定符格式的字符串，用于指定非 root 用户的文件系统: [dir:**non_root**=<*dir_path*>, unit:<*unit_name*>] <disk_space> 例如: os.space.home= [dir:non_root=USERHOME/ .acsi_HOST, unit:MB]200 具有以下限定符格式（仅使用一个限定符）的字符串: [dir:<*dir_path*>] <*disk_space*> MB 例如: os.space.home= [dir:/home/sat]250MB 以 MB 或 GB 为单位的数字格式，例如: os.space.opt=11GB

表 24. 操作系统数据属性 (续)

先决条件属性	平台	描述	有效值
os.space.*dir_name*_nonroot	UNIX	用于根据下列一个或多个限定属性来检查非 root 用户的 *dir_name* 文件系统的可用磁盘空间量: • dir 属性，用于确定要检查的文件系统路径 • unit 属性，用于确定要使用的磁盘空间单位 例如，*dir_name* 可以表示: • home • opt • tmp • usr • var	具有以下限定符格式的字符串，用于指定非 root 用户的文件系统: [dir:**non_root**=*dir_path*, unit:*unit_name*] disk_space 例如: os.space.home_nonroot= [dir:non_root= USERHOME/.acsi_HOST, unit:MB]200 仅具有非 root 用户的文件系统的 dir 限定属性的字符串: [dir:**non_root**=*dir_path*] *disk_space*GB|MB 例如: os.space.opt_nonroot= [dir:non_root=/opt/IBM/ITM] 1024MB
os.space.*dir_name*_root	UNIX	用于根据下列一个或多个限定属性来检查 root 用户的 *dir_name* 文件系统的可用磁盘空间量: • dir 属性，用于确定要检查的文件系统路径 • unit 属性，用于确定要使用的磁盘空间单位 例如，*dir_name* 可以表示: • home • opt • tmp • usr • var	具有以下限定符格式的字符串，用于指定 root 用户的文件系统: [dir:**root**=*dir_path*, unit:*unit_name*] *disk_space* 例如: os.space.usr_root= [dir:root= /usr/ibm/common/acsi, unit:GB]200 仅具有 root 用户的文件系统的 dir 限定属性的字符串: [dir:**root**=*dir_path*] *disk_space*GB|MB 例如: os.space.opt_root= [dir:root=/opt/IBM/ITM] 1024MB
os.sshdConfig	UNIX	用于检查是否为 SSH 守护程序会话配置了许可式 root 用户登录	Available|Unavailable

表 24. 操作系统数据属性 (续)

先决条件属性	平台	描述	有效值
os.swapSize	UNIX	用于检查交换空间是否必须大于 RAM 大小或交换空间总量	值可以是下列任意类型: • 布尔值，例如: True • 以 MB 或 GB 为单位的数字格式，例如: 2GB
os.tmpdir	UNIX	检查为 /tmp 文件系统指定的访问许可权，包括任何由访问权标志设置的特定许可权，例如八进制数中的 sticky、setuid 或 setgid 位。	用于表示访问许可权的 octal_digits 八进制数的数字 例如，要检查是否启用了具有 drwxrwxrwt 许可权和粘性位许可权的临时目录: 1777 再如，要检查临时目录是否具有除粘性位以外的 drwxrwxrwx 许可权: 777
os.totalMemory	Windows	这是操作系统可以访问的虚拟内存总量	以 MB 或 GB 为单位的数字格式，例如: 4GB
os.totalPhysicalMemory	Windows	这是操作系统可以访问的物理内存总量，但这并不是目标计算机上的实际物理内存量	以 MB 或 GB 为单位的数字格式，例如: 2030MB

表 24. 操作系统数据属性 (续)

先决条件属性	平台	描述	有效值
os.ulimit	UNIX	根据下列限定属性检查是否可以运行数目不受限的进程: • type 属性，用于确定要检查的附加限制，例如 filedescriptorlimit 检查进程可以打开的文件描述符数目的限制 另外，它还检查是否在 /etc/security/limits.conf 文件中对指定的域设置了下列限制: root - stack unlimited ctginst1 - stack unlimited root - nofile 8192 tioadmin - nofile 32767	值可以是下列任意类型: • 具有以下限定符格式的字符串: [type:*limit_name*] *limit_value*, limited|unlimited 例如，要检查文件描述符限制是否大于 8192（进程数不受限）: os.ulimit=[type: filedescriptorlimit] 8192+, unlimited 要检查的限制的有效类型如下所示，其中 *limit_name* 表示限制类型: – ALL，用于检查所有限制 – corefilesizelimit – datasegmentlimit – filedescriptorlimit – filesizelimit – hardlimit – processlimit – maxmemorysizelimit – maxprocesseslimit – stacksizelimit – threadlimit • Available|Unavailable 用于指定是否在 /etc/security/limits.conf 文件中对相关的域设置了限制。
os.umask	UNIX	用于检查文件方式创建掩码的许可权	用于表示访问许可权的 *octal_digits* 八进制数的数字。例如，要检查新文件是否仅可供所有者写入，请将八进制数设置为 0022
os.userLimits	UNIX	检查最大堆栈大小是否不受限；如果不受限，那么将返回 Available	Available|Unavailable
os.versionNumber	Windows	检查机器上安装的操作系统的当前版本	数字格式，例如 5.0+ **注**: 该值可使用在 第 2 页的表 1 中概括的特殊字符。
os.windowManager	UNIX	检查 GNOME 或 KDE 作为是否可作为图形桌面使用	Available|Unavailable

“已安装的软件”数据属性

“已安装的软件”数据属性用于检查已安装的软件先决条件（例如在 Windows 注册表中注册的程序）以及是否安装了 cygwin 和 gskit。仅限于 Windows 系统：它使用 *ips_root*/lib 目录中文件名包含 installedSoftware、cygwin 或 gskit 前缀标识的“已安装的软件”收集器。

表 25 对公共数据先决条件属性作了概述。此类先决条件属性不需要前缀标识。

表 25. “已安装的软件”数据属性

先决条件属性	平台	描述	有效值
installedSoftware	Windows	用于扫描操作系统注册表以查找已安装的程序及其位置	字符串，各个应用程序之间以逗号分隔。
cygwinVersion	Windows	用于检查机器上安装的 cygwin 版本；如果未安装任何版本，那么将返回 0.0	正整数，例如 1.5 **注**：这些值可以使用第 2 页的表 1 中概述的特殊字符。
gskit7Version	Windows	检查机器上是否安装了 gskit V7；如果未安装 V7，那么将返回 0.0	正整数，例如 7.0
gskit8Version	Windows	检查机器上是否安装了 gskit V8；如果未安装 V8，那么将返回 0.0	正整数，例如 8.0

用户数据属性

用户数据属性用于检查用户先决条件，例如登录用户是否具有管理权以及是否为 root 用户。仅限于 Windows 系统：它使用 *ips_root*/lib 目录中文件名包含 user 前缀标识的用户收集器。仅限于 UNIX 系统：它使用 *ips_root*/lib/packageTest.sh 中的用户收集器。

表 26 对用户先决条件属性作了概述。此类先决条件属性需要 user 前缀标识。

表 26. 用户数据属性

先决条件属性	平台	描述	有效值
user.userID	Windows	这是当前登录用户的标识	字符串，例如 smithj
user.isAdmin	全部	用于检查登录用户是否为管理员组中的成员	布尔值，例如 True

Windows 网络数据属性

Windows 网络数据属性用于检查网络先决条件，例如是否在机器上启用了 NetBIOS 和 DHCP 以及 Ping 属性。它使用 *ips_root*/lib 目录中文件名包含 network 前缀标识的 Windows 网络收集器。.

第 100 页的表 27 对所有 Windows 平台之间的公共网络先决条件属性作了概述。此类先决条件属性需要 network 前缀标识。

表 27. Windows 网络数据属性

先决条件属性	描述	有效值
network.DHCPEnabled	检查是否至少有一个具有有效 IP 地址的适配器已通过 DHCP 获取 IP 地址；如果至少存在一个这样的适配器，那么将返回 True。	布尔值，例如 False
network.netBIOSEnabled	检查是否至少有一个具有有效 IP 地址的适配器启用了 NetBIOS 作为协议；如果至少存在一个这样的适配器，那么将返回 True。	布尔值，例如 True
network.pingLocalhost	用于检查本地主机是否对 Ping 协议作出了响应；如果是，那么将返回 True。	布尔值，例如 True
network.pingSelf	用于检查是否已使用 DCHP 解析了本地计算机名称，并且是否可对其执行 Ping 操作；如果可以对其执行 Ping 操作，那么将返回 True。	布尔值，例如 True
network.ValidateHostsFile	用于检查 C:\WINDOWS\system32\drivers\etc\hosts 中的条目格式是否正确；如果格式有效，那么将返回 True。	布尔值，例如 True

UNIX 网络数据属性

UNIX 网络数据属性用于检查网络先决条件，例如是否在机器上启用了 NetBIOS 和 DHCP 以及 Ping 属性。它使用 *ips_root*/UNIX_Linux 目录中的网络收集器。

表 28 对所有 UNIX 平台之间的公共网络先决条件属性作了概述。此类先决条件属性需要 network 前缀标识。

表 28. UNIX 网络数据属性

先决条件属性	描述	有效值
network.DHCPEnabled	检查是否至少有一个具有有效 IP 地址的适配器已通过 DHCP 获取 IP 地址	布尔值，例如 False
network.dns	用于检查主机的 DNS 条目是否正确	布尔值，例如 True
network.fqdn	用于检查是否对主机设置了标准域名	布尔值，例如 True
network.pingLocalhost	用于检查本地主机是否对 Ping 协议作出了响应	布尔值，例如 True
network.pingSelf	用于检查是否已使用 DCHP 解析了本地计算机名称，并且是否可对其执行 Ping 操作	布尔值，例如 True

环境变量数据属性

环境变量数据属性检查所有平台之间的公共环境变量先决条件，例如是否设置了环境变量以及环境变量的值。仅限于 Windows 系统：它使用 *ips_root*/lib 目录中文件名包含 env 前缀标识的环境变量收集器。仅限于 UNIX 系统：它使用 *ips_root*/UNIX_Linux 目录中文件名包含 env 前缀标识的 UNIX 环境变量收集器。

第 101 页的表 29 对所有平台之间的公共环境变量先决条件属性作了概述。此类先决条件属性需要 env 前缀标识。

表 29. 环境变量数据属性

先决条件属性	平台	描述	有效值
env.var.set.*env_var_name*	UNIX	使用此命名约定来检查计算机上是否设置了指定的 *env_var_name* 环境变量，例如: • env.var.set.HOME 检查是否设置了主目录的环境变量，其中 *env_var_name* 是 HOME 环境变量的名称 • env.var.set.JAVA_HOME 检查是否设置了 Java 主目录的环境变量，其中 *env_var_name* 是 JAVA_HOME 环境变量的名称	布尔值，例如 True
env.var.set.*env_var_name* [type:*env_var_type*]	Windows	使用此命名约定来检查是否为指定的 *env_var_type* 环境变量类型设置了指定的 *env_var_name* 环境变量，例如: • env.var.set.HOME 检查是否设置了主目录的环境变量，其中 *env_var_name* 是 HOME 环境变量的名称 • env.var.set.JAVA_HOME[type:User] 检查是否为登录用户设置了 Java 主目录的环境变量，其中 *env_var_name* 是 JAVA_HOME 环境变量的名称，而 *env_var_type* 是环境变量的 User 类型 *env_var_type* 环境变量类型是可选的，它表示 Windows 操作系统所支持的环境变量的类型，如下所示: • Process • System • User • Volatile 如果未指定类型，那么缺省类型为 Process。	布尔值，例如 True
env.classpath.derbyJAR	全部	检查类路径环境变量中是否包含 Derby JAR 文件的路径	布尔值，例如 True
env.CIT.homeExists	Windows	检查 HOMEDRIVE 和 HOMEPATH 环境变量是否都存在	布尔值，例如 True

附录 D. UNIX 系统的预定义收集器

在 *ips_root*/lib 目录中，提供了用于在 UNIX 系统上执行先决条件属性检查的各个收集器。在创建定制收集器之前，您可以查看这些收集器及其输入参数。

表 30 对 UNIX 系统的预定义收集器作了概述。

表 30. UNIX 收集器

收集器	针对先决条件属性	输入
DB2_Version	DB2 Version	无
DBType	DBType	无
DBTypeDetails	DBTypeDetails	无
env.classpath.derbyJAR	env.classpath.derbyJAR	无
env.var.set	env.var.set*env_var_name* *env_var_name* 是要检查的环境变量的名称	*$env_var_name*
network.DHCPEnabled	network.DHCPEnabled	无
network.dns	network.dns	无
network.fqdn	network.fqdn	无
network.pingSelf	network.pingSelf	无
network.port	network.availablePorts.* network.portsInUse.*	$ports
oracle.Client	oracle.Client	无
oracle.Client.Location	oracle.Client.Location	无
oracle.Server	oracle.Server	无
oracle.Server.Location	oracle.Server.Location	无
os.architecture	os.architecture	32 bit|64 bit
os.automount	os.automount	无
os.cmd	os.lookup	nslookup
os.cmd	os.tar|os.gnu.tar	tar|gtar
os.dir	os.dir.*dir_name* 例如，*dir_name* 可以表示: • tmp • home	格式如下的字符串: [dir:*dir_name*, type:permission] *octal_digits*+ 例如，要检查 *dir_name* 目录（也就是主目录）是否具有 drwxr-xr-x 许可权: [dir:/home, type:permission] 755+
os.diskquota	os.diskquota	无
os.expectLink	os.expectLink	无

表 30. UNIX 收集器 (续)

收集器	针对先决条件属性	输入
os.filepath	os.file.*script_name* 例如，*script_name* 可以表示: • bash • expect • gzip • tar	脚本文件的路径，其中路径可以是: • /bin/bash • /usr/bin/expect • /usr/bin/gzip • /usr/bin/tar
os.Firefox	os.Firefox	无
os.FreePagingSpace	os.FreePagingSpace	无
os.ftpusers	os.ftpusers	无
os.hostformat	os.hostformat	无
os.iodevicestatus	os.iodevicestatus	无
os.kernelMode	os.kernelMode	无
os.kernelParameters	os.kernelParameters	无
os.kernelversion	os.kernelversion	无
os.largeFile	os.largeFile	无
os.ldLibPath	os.ldLibPath	产品
os.level	os.level	无
os.lib	os.lib.*lib_name_version* 例如，用于表示 *lib_name_version* 的字符串或正则表达式（显示为粗体）: • 32 位 **libstdc++.so.#** 库 • 64 位 **libstdc++.so.#** 库 • 32 位 **libXft.so.#** 库 • 32 位 **libXtst.so.#** 库 • 64 位 **libaio.so.#** 库 • 32 位 **xlC.rte** XLC 运行时级别 • AIX V5.3 的 32 位 **xlC.aix50.rte** XLC 运行时 • AIX V6.1 的 32 位 **xlC.aix61.rte** XLC 运行时 • AIX IOCP **bos.iocp.rte** 库 • **bos.loc.iso.en_us**，AIX 基本操作系统的 ISO 代码文件集 • **regex{*str*}**，这是具有输入参数 *str* 的正则表达式，用于表示库名搜索模式，例如 .*libgcc.*	*path_to_library* 或 *lib_name_version* 例如，要检查 os.lib.libstdc++.so 先决条件属性的实际值，输入参数为 /usr/lib/libstdc++.so.5 和 libstdc++.so: os.lib /usr/lib/libstdc++.so.5 libstdc++.so 例如，要检查机器上是否存在某个版本的 GCC 低级别运行时库 libgcc，输入参数为: regex{.*libgcc.*}
os.loginVariable	os.loginVariable	无
os.maximoDirectory	os.maximoDirectory	无

表 30. UNIX 收集器 (续)

收集器	针对先决条件属性	输入
os.maximoDirOwner	os.maximoDirOwner	无
os.maximumProcesses	os.maximumProcesses	无
os.MozillaVersion	os.MozillaVersion	无
os.mountcheck	os.mountcheck	具有以下限定符格式的字符串: [drive:*dir_name*, *mount_option*: false|true] True|False 例如，要检查是否已装配 /home 目录以及是否未设置 nosuid 选项: os.mountcheck=[drive:/home, nosuid:false]True
os.package	os.package.*package_name* 例如，用于表示 *package_name* 的字符串（显示为粗体）: • **bash** Shell • **expect**（对于 TCL 扩展包） • **libgcc**（对于 GCC 低级别运行时包） • **openssh**（对于开放式源代码安全 Shell） • **openssl**（对于开放式源代码 SSL/TLS 工具箱） • **perl**（对于 Perl 脚本包） • **rpm**（对于 RPM 或 RPM 构件包） • **telnet**（对于 Telnet 包） • **wget**（对于 GNU 文件检索包）	例如 *package_name*，其中 *package_name* 为 rpm: os.package rpm
os.pagesize	os.pagesize	无
os.RAMSize	os.RAMSize	无
os.SELinux	os.SELinux	• 具有以下限定符格式的字符串: [source:Command] Disabled|Enabled 例如，要检查功能部件在 Red Hat 或 SUSE 操作系统上是处于禁用状态还是处于许可状态: os.SELinux=[source: Command]Disabled • 如果未指定限定符，那么不会将值传递到收集器。
os.servicePack	os.servicePack	Service Pack 值

表 30. UNIX 收集器 (续)

收集器	针对先决条件属性	输入
os.shell.default	os.shell.default	这是先决条件属性的期望值，例如 bash
os.space	os.space.*dir_name* 例如，*dir_name* 可以表示: • usr • home • tmp • var	具有以下限定符格式的字符串，用于指定 root 用户的文件系统: [dir:**root**=*dir_path*, unit:*unit_name*] *disk_space* 例如: os.space.usr= [dir:root=/usr/ibm/common/acsi, unit:GB]200 具有以下限定符格式的字符串，用于指定非 root 用户的文件系统: [dir:**non_root**=*dir_path*, unit:*unit_name*] disk_space 例如: os.space.home= [dir:non_root=USERHOME/ .acsi_HOST, unit:MB]200 具有以下限定符格式（仅使用一个限定符）的字符串: [dir:*dir_path*] *disk_space* MB 例如: os.space.home= [dir:/home/sat]250MB
os.sshdConfig	os.sshdConfig	无
os.swapSize	os.swapSize	无
os.tmpdir	os.tmpdir	无

表 30. UNIX 收集器 (续)

收集器	针对先决条件属性	输入
os.ulimit	os.ulimit	具有以下限定符格式的字符串: [type:*limit_name*] *limit_value*, limited|unlimited 例如，要检查文件描述符限制是否大于 8192（进程数不受限）: os.ulimit= [type:filedescriptorlimit] 8192+,unlimited 要检查的限制的有效类型如下所示，其中 *limit_name* 表示限制类型: • ALL，用于检查所有限制 • corefilesizelimit • datasegmentlimit • filedescriptorlimit • filesizelimit • hardlimit • processlimit • maxmemorysizelimit • maxprocesseslimit • stacksizelimit • threadlimit
os.umask	os.umask	无
os.userLimits	os.userLimits	无
os.windowManager	os.windowManager	无

附录 E. Windows 系统的通用函数

Prerequisite Scanner 在 /lib/common_function.vbs 文件中提供了一组用于在 Windows 系统上运行检查的通用函数。

表 31. common_function.vbs 中的函数

函数	描述
第 110 页的 『allFiles()』	将指定目录中的文件名读取到一个数组中。
第 110 页的 『arrayToString()』	创建数组的字符串表示。
第 111 页的 『bigthan()』	计算先决条件属性的期望值与实际值之差（如果该先决条件属性是以 MB 或 GB 为单位的大小）。
第 111 页的 『changeMG()』	对于磁盘空间或内存先决条件属性，将输入参数转换为以 MB 或 GB 为单位。
第 112 页的 『checkItemToString()』	创建 CheckItem 对象的字符串表示。
第 112 页的 『dictionaryToString()』	创建脚本编制字典对象的字符串表示。
第 113 页的 『exeCommand()』	执行指定的命令并返回该命令的执行结果。
第 113 页的 『filterCommand()』	执行指定的命令，并返回命令结果中与指定模式匹配的行。
第 114 页的 『filterFile()』	读取文件内容并对其进行过滤，然后将过滤结果写入一个脚本编制字典对象。
第 114 页的 『findNewest()』	查找最新的配置文件。
第 115 页的 『findSuitableFile()』	查找产品和版本的相关配置文件。
第 115 页的 『fmt()』	通过添加指定数目的字符对一个字符串进行修改，这些字符来自另一个字符串；对于后一个字符串，使用空格字符对其进行填充（如果其长度过短）或者将其截断（如果其长度过长）。
第 116 页的 『formatForDisplay()』	对输入参数编排格式以使其可读。
第 116 页的 『formatSizeForDisplay()』	接受输入参数，并将输入参数的小数部分追加或修剪为两个小数位，例如将 123MB 转换为 123.00MB，或者将 12.123MBs 转换为 12.12MBs。
第 117 页的 『getDecimalSeparator()』	确定要用于当前语言环境的十进制分隔符。
第 117 页的 『getFirstMatch()』	获取搜索字符串在数组中的第一个匹配项。
第 118 页的 『isMatch()』	检查字符串是否包含搜索模式。
第 118 页的 『notInLatter()』	对第一个数组进行过滤，以确定内容是否包含在第二个数组中。根据 in_or_not 输入参数的值不同，此函数将返回第一个数组中的那些与第二个数组匹配或不匹配的内容。

表 31. common_function.vbs 中的函数（续）

函数	描述
第 119 页的『passOrFail()』	对先决条件属性的期望值和实际值进行比较，并确定该先决条件属性是否通过检查。输入参数可以是一般的数字、以 MB 或 GB 为单位的大小、以 MHz 或 GHz 为单位的 CPU 速度、布尔值或字符串。
第 119 页的『ppread()』	将文件内容读取到脚本编制字典对象中，从而按指定的分隔符输入参数对文件中的每一行进行进一步分割（如果该分隔符存在于该行中）。
第 120 页的『readFile()』	将文件的每一行读取到数组的下标条目中。
第 120 页的『unitMGTOG()』	对数组内容进行累加以获取 MB 总计。
第 121 页的『varToString()』	创建变量的字符串表示。要检查的变量可以是字符串、数字、脚本编制字典对象、数组或 `CheckItem` 对象。

allFiles()

将指定目录中的文件名读取到一个数组中。

用途

此函数用于获取目录输入参数中的文件列表，并将它们添加到数组中。然后，返回该数组。

语法

```
allFiles(filepath)
```

输入参数

String *filepath*
文件所在目录的路径。

返回值

Array *fileNames*
返回一个数组，该数组包含指定目录中的文件名。

arrayToString()

创建数组的字符串表示。

用途

此函数使用作为输入参数传递的数组，并返回该数组的内容的字符串表示。

语法

```
arrayToString(arr)
```

输入参数

Array *arr*
包含数组。

返回值

String *result*
返回数组的字符串表示，各个项之间以逗号分隔。

bigthan()

计算先决条件属性的期望值与实际值之差（如果该先决条件属性是以 MB 或 GB 为单位的大小）。

用途

此函数先调用 『changeMG()』 函数，以便在有需要时将先决条件属性的期望值和实际值更改为以 MB 为单位。然后，此函数检查是否该函数的任何一个返回值为 Null，如果任何一个返回值为 Null，那么此函数的返回值为 0MB，并且此函数退出。此函数检查值是以 MB 还是 GB 为单位，并在有需要时将值转换为以 MB 为单位。接着，此函数计算格式化的最终值之差并返回结果。

语法

```
bigthan(expect,real)
```

输入参数

String *expect*
这是先决条件属性的期望值。

String *real*
这是先决条件属性的实际值。

返回值

String *bigthan*
返回以 MB 计的差值，或者返回 0MB（如果没有差别）。

changeMG()

除非输入参数包含 MB 或 GB，否则对输入参数进行格式编排，以便从中除去任何其他数字分组字符，并返回格式编排后的参数。如果输入参数包含 MB 或 GB，那么此函数将输入参数分别转换为以 GB 或 MB 为单位。

用途

此函数调用第 117 页的『getDecimalSeparator()』函数以确定当前语言环境的十进制分隔符，然后从数字输入参数中除去该语言环境的任何其他数字分组字符。接着，调用 第 117 页的『getFirstMatch()』 函数以确定值是以 MB 还是 GB 为单位，并将值相应转换为以 GB 或 MB 为单位。

语法

```
changeMG(tochange)
```

输入参数

String *tochange*
包含要在必要时进行格式编排和转换的值。

返回值

String *changeMG*
返回进行了格式编排且不带数字分组字符的数字，或者返回以 MB 或 GB 为单位的数字。

checkItemToString()

创建 CheckItem 对象的字符串表示。

用途

此函数使用作为输入参数传递的 CheckItem 对象并返回一个字符串表示，该字符串表示由该 CheckItem 对象实例的不同属性的值组成。

语法

```
checkItemToString(var)
```

输入参数

CheckItem *var*
包含 CheckItem 对象的实例。

返回值

String *result*
返回 CheckItem 对象的属性的字符串表示，如下所示:

```
result = "CheckItem[pdCode[" & chkItem.pdCode & "],pdName[" & chkItem.pdName & _
         "],itype[" & chkItem.itype & "],recommended[" & chkItem.recommended & _
         "],realValue[" & chkItem.realValue & "],passOrFail[" &
     chkItem.passOrFail & "]]"
```

dictionaryToString()

创建脚本编制字典对象的字符串表示。

用途

此函数使用作为输入参数传递的字典对象，并返回该字典对象的内容的字符串表示。

语法

```
dictionaryToString(dic)
```

输入参数

Dictionary *dic*
包含字典对象。

返回值

String *result*
返回字典对象的字符串表示，其中，每个键与项之间以等号分隔。

exeCommand()

执行指定的命令并返回该命令的执行结果。

用途

此函数执行命令输入参数。如果发生任何错误，那么将调用 logWarning 子例程以显示错误；否则，将返回该命令的执行结果。

语法

```
exeCommand(cmd)
```

输入参数

String *cmd*
要执行的命令的名称。

返回值

String *result*
返回一个字符串，该字符串包含该命令的执行结果。

filterCommand()

执行指定的命令，并返回命令结果中与指定模式匹配的行。

用途

此函数执行命令输入参数。它解析该命令的执行结果，并检查结果中是否有任何行与行模式输入参数匹配。如果存在匹配项，那么它将调用 第 117 页的『 getFirstMatch() 』函数，以确定信息行输入参数与命令结果之间是否也存在匹配项。如果存在，那么它将使用 Join 函数以返回来自 getFirstMatch() 函数的字典对象的内容。

语法

```
filterCommand(cmd, line_patt, after_line, info_patt)
```

输入参数

String *cmd*
要执行的命令的名称。

String *line_patt*
这是要在命令执行结果中搜索的行模式。

Number *after_line*
这是搜索信息模式停止前的行数。

String *info_patt*
这是要在命令结果的每一行中搜索的信息模式。

返回值

String *filterCommand*
将字典对象的内容作为单一字符串返回。

filterFile()

读取文件内容并对其进行过滤，然后将过滤结果写入一个脚本编制字典对象。

用途

此函数读取文件的每一行，并将每一行随搜索模式一起传递给 第 117 页的『getFirstMatch()』 函数。如果该函数返回了匹配项，并且该行尚未存在于字典对象中，那么该行将写入到字典对象中。此函数将执行循环直至到达文件末尾，然后返回该字典对象。

语法

```
filterFile(fileName, patt)
```

输入参数

String *fileName*
这是要进行过滤的文件。

String *patt*
这是要在该文件的每一行中搜索的模式。

返回值

Dictionary *dic.keys*
返回 *dic* 字典对象，此对象包含文件中经过过滤的行。

findNewest()

在数组中查找最新的配置文件。

用途

此函数对整个数组执行循环，并确定该数组中的哪个文件是最新的配置文件。然后，返回该文件的名称。

语法

```
findNewest(arr)
```

输入参数

Array *arr*
包含要检查的配置文件集。

返回值

String *result*
返回最新配置文件的名称。

findSuitableFile()

查找产品和版本的相关配置文件。

用途

此函数调用第 117 页的『getFirstMatch()』函数，以获取 第 110 页的『allFiles()』 函数所返回的文件列表中的那些将扩展名输入参数用作文件扩展名的文件集。接着，它再次调用 第 117 页的『getFirstMatch()』 函数，以返回文件名包含产品代码输入参数的文件集。然后，调用同一函数以获取文件名包含版本输入参数的文件集。如果这些函数找到该版本的一个或多个文件匹配项，那么此函数将调用 第 114 页的『findNewest()』函数以获取该文件的最新版本并返回该文件名；否则，此函数将返回 `common.bat` 文件，或者先使用 `logScreen` 和 `logWarning` 子例程，然后再返回产品代码的配置文件的最新版本。

语法

```
findSuitableFile(pd,version,suf,filepath)
```

输入参数

String *pd*

这是与所要查找的文件相关联的产品代码，此代码在产品代码文件 *ips_root*`/codename.cfg` 中指定。

String *version*

这是与所要查找的文件相关联的产品版本。<*version*> 是表示版本、发行版、修订版和级别的 8 位代码（此代码的每个部分各占两位）；例如，7.3.21 为 07032100。

String *suf*

这是要查找的文件类型的扩展名，例如 `cfg` 或 `bat`。

String *filepath*

要查找的文件所在目录的路径。

返回值

String *findSuitableFile*

根据所调用的函数的结果，返回下列其中一个文件名:

- *pd_version*`.cfg`，这是相关联产品代码和版本的最新版本文件。
- `common.bat`（如果文件扩展名输入参数的值为 `bat`）。
- *pd*`.cfg`，这是产品的通用配置文件的最新版本（如果找不到包含版本输入参数的文件）。

fmt()

通过添加指定数目的字符对一个字符串进行修改，这些字符来自另一个字符串；对于后一个字符串，使用空格字符对其进行填充（如果其长度过短）或者将其截断（如果其长度过长）。

用途

此函数用于在类型为字符串的 *s* 输入参数中搜索 `%#s` 表达式。`%#s` 表达式确定 *args* 输入参数中要添加到第一个字符串中该表达式所在位置的指定字符数。如果指定的 # 大于

args 输入参数的长度，那么将填充差值数目的空格字符。如果指定的 # 小于 *args* 输入参数的长度，那么长度将截去差值。如果指定的 # 为 0，那么整个 *args* 输入参数将添加到第一个字符串中的相应位置。

语法

```
fmt(s, args)
```

输入参数

String *s*
包含一个字符串，此字符串将由其中的 %#s 表达式中的指定字符数修改。

Array *args*
包含一组用于修改 *s* 输入参数的字符。

返回值

String *result*
返回修改后的字符串。

示例

```
fmt("Hello %5s!",array("Neo")) 将返回 "Hello Neo  !"（填充附加的空格字符）
fmt("Hello %5s!",array("Mr. Anderson")) 将返回 "Hello Mr. A!"（截断为只添加 "Mr. A"）
fmt("Hello %0s!",array("Mr. Anderson")) 将返回 "Hello Mr. Anderson!"
```

formatForDisplay()

对输入参数编排格式以使其可读。

用途

此函数调用『formatSizeForDisplay()』函数，以便对输入参数进行格式编排。

语法

```
formatForDisplay(val)
```

输入参数

Variable *val*
要进行格式编排的变量。

返回值

String *vartoString*
返回 『formatSizeForDisplay()』 函数的调用结果。

formatSizeForDisplay()

接受输入参数，并将输入参数的小数部分追加或修剪为两个小数位，例如将 123MB 转换为 123.00MB，或者将 12.123MBs 转换为 12.12MBs。

用途

此函数对输入参数中的字符进行计数，检查该输入参数是数字还是字符串，并将输入参数分割为整数和小数部分。根据小数部分的不同，此函数将其追加或修剪为两个小数位。然后，返回结果。

语法

```
formatSizeForDisplay(size)
```

输入参数

Integer *size*
要四舍五入到小数点后两位的值。

返回值

Integer *val*
返回四舍五入到小数点后两位的值。

getDecimalSeparator()

确定要用于当前语言环境的十进制分隔符。

用途

此函数创建一个小数，然后使用 `Mid()` 函数来确定在该小数中使用的十进制分隔符。

语法

```
getDecimalSeparator()
```

输入参数

无

返回值

Character *sep*
返回语言环境的相应十进制分隔符，例如 ，或 .。

getFirstMatch()

获取搜索字符串在数组中的第一个匹配项。

用途

此函数使用正则表达式在作为输入参数传递的数组中搜索同样作为输入参数传递的模式。在数组中找到该模式的第一个匹配项后，此函数将该数组中的值添加到脚本编制字典对象。

语法

```
getFirstMatch(patt, arr)
```

输入参数

String *patt*
包含要搜索的模式。

Array *arr*
包含要在其中进行搜索模式搜索的数组。

返回值

Dictionary *keys*
返回脚本编制字典对象的键。

isMatch()

检查字符串是否包含搜索模式。

用途

此函数调用第 117 页的『 getFirstMatch() 』函数，并将模式和字符串（包含在数组中）作为输入参数传递到此函数。此函数调用 ubound 函数，以检查 getFirstMatch() 函数返回的值是否大于或等于 0。如果是，那么表明存在匹配项，否则不存在匹配项。

语法

```
isMatch(patt,str)
```

输入参数

String *patt*
包含要搜索的模式。

String *str*
包含要在其中进行搜索模式搜索的字符串。

返回值

Boolean *True*|*False*
如果存在匹配项，那么返回 True，否则返回 False。

notInLatter()

对第一个数组进行过滤，以确定内容是否包含在第二个数组中。根据 in_or_not 输入参数的值不同，此函数将返回第一个数组中的那些与第二个数组匹配或不匹配的内容。

用途

语法

```
notInLatter(arr1, arr2, in_or_not)
```

输入参数

Array *arr1*
从某个位置复制

Array *arr2*
复制到其他位置

String *in_or_out*
包含 "in" 或 "not"，具体取决于此函数是否应该返回第一个数组经过过滤后的内容，以便只返回与第二个数组匹配的内容 ("in") 或者与第二个数组不匹配的内容 ("not")。

返回值

Dictionary *keys*
返回一个脚本编制字典对象的键，此对象包含第一个数组经过过滤后的内容（仅保留与第二个数组匹配的内容 (in_or_not = "in") 或者与第二个数组不匹配的内容 (in_or_not = "not")）。

passOrFail()

对先决条件属性的期望值和实际值进行比较，并确定该先决条件属性是否通过检查。输入参数可以是一般的数字、以 MB 或 GB 为单位的大小、以 MHz 或 GHz 为单位的 CPU 速度、布尔值或字符串。

用途

此函数先调用 第 111 页的『changeMG()』 函数以进行格式编排，并在必要时对期望值和实际值进行转换。此函数将检查是否任意一个值为 0，如果是，那么将返回 ″FAIL″ 并退出。如果各个值均不为 0，那么此函数将检查这些值是布尔值、数字、以 MB 或 GB 为单位的大小、以 MHz（仅限于 Windows）或 GHz 为单位的 CPU 速度还是字符串。然后，它对这些值进行比较并返回结果。

语法

```
passOrFail(expect,real)
```

输入参数

String *expect*
这是先决条件属性的期望值。

String *real*
这是先决条件属性的实际值。

返回值

String *passOrFail*
根据期望值是等于还是大于实际值，返回 "PASS" 或 "FAIL"。

ppread()

将文件内容读取到脚本编制字典对象中，从而按指定的分隔符输入参数对文件中的每一行进行进一步分割（如果该分隔符存在于该行中）。

用途

此函数读取文件的每一行，除去所有前导空格和尾部空格，并检查该行是否包含分隔符。如果该行包含分隔符，那么此函数将按分隔符来分割该行，从而将每个部分作为一个项添加到字典对象中；否则，将修剪后的行添加到字典对象中的一个项。此函数将返回一个数组，该数组包含该字典对象作为第一个下标。

语法

```
ppread(fileName, sep)
```

输入参数

String *fileName*
这是要读入到字典对象中的文件的名称。

Character *sep*
这是一个字符，它表示用于对该文件中的行进行分割的分隔符。

返回值

Array *array(dic)*
返回一个数组，其中，字典对象 (dic) 是它的第一个下标。

示例

示例待提供。

readFile()

将文件的每一行读取到数组的下标条目中。

用途

此函数用于打开文件并将该文件的每一行读取到数组的下标条目中。然后，返回该数组。

语法

```
readFile(fileName)
```

输入参数

String *fileName*
这是要读入到数组中的文件的名称。

返回值

Array *fileContents*
返回包含文件内容的数组。

unitMGTOG()

对数组内容进行累加以获取 MB 总计。

用途

此函数将数组中每个下标的值转换为以 MB 为单位，并将它们进行累加。

语法

```
unitMGTOG(arr)
```

输入参数

Array *arr*
包含数组。

返回值

String *unitMGTOG*
返回数组内容的总计（以 MB 为单位），并对总计追加 "MB"。

varToString()

创建变量的字符串表示。要检查的变量可以是字符串、数字、脚本编制字典对象、数组或 CheckItem 对象。

用途

此函数检查变量的数据或对象类型，并调用相关函数以创建该数据或对象类型的字符串表示。

表 32. 针对每种变量类型调用的函数。

变量类型	调用的函数
数组	第 110 页的『arrayToString()』
CheckItem 对象	第 112 页的『checkItemToString()』
脚本编制字典对象	第 112 页的『dictionaryToString()』

语法

```
varToString(var)
```

输入参数

Variable *var*
支持的变量包括：字符串、数字、脚本编制字典对象、数组或 CheckItem 对象

返回值

String *vartoString*
返回该变量的字符串表示（包括根据需要从调用的任何函数返回的值）。

附录 F. Windows 系统的日志记录实用程序子例程

IBM Prerequisite Scanner 在 preq.vbs 文件中提供了一组用于在屏幕上显示消息或者将消息写入日志文件的通用日志记录子例程。

表 33 对日志记录实用程序作了描述。

表 33. 日志记录实用程序子例程

子例程	描述	输入参数
deleteLogFile	删除日志文件（如果该文件存在）。	无
log(level, msg)	通过使用第 115 页的『fmt()』函数，将消息写入日志文件。此日志还包含当前日期和时间。	• level，这是用于设置消息类型（例如参考或警告）的字符串 • msg，这是用于表示所要记录的消息的字符串
logDebug(msg)	通过传递 "DEBUG" 作为级别输入参数，调用 log() 函数。	msg，这是用于表示所要记录的消息的字符串
logError(msg)	通过传递 "ERROR" 作为级别输入参数，调用 log() 函数。	msg，这是用于表示所要记录的消息的字符串
logInfo(msg)	通过传递 "INFO" 作为级别输入参数，调用 log() 函数。	msg，这是用于表示所要记录的消息的字符串
logScreen(msg)	将消息写至屏幕。	msg，这是用于表示要写至屏幕的消息的字符串
logScreenWith Replacement (msg, replaceStr)	通过传递消息代码和字符串作为输入参数，将消息写至屏幕。	• msg，这是用于表示要写至屏幕的消息字符串的消息代码 • replaceStr，这是用于替换消息代码值中的 *%variable* 的字符串
logScreenWith MultiReplacements (msg, replaceStrArray)	通过传递消息代码和字符串数组作为输入参数，将消息写至屏幕。	• msg，这是用于表示要写至屏幕的消息字符串的消息代码 • replaceStrArray，这是一个字符串数组，其中的每个下标都将替换消息代码值中的 *%variable*
logWarning(msg)	通过传递 "WARNING" 作为级别输入参数，调用 log() 函数。	msg，这是用于表示所要记录的消息的字符串

附录 G. Windows 系统的文件实用程序子例程

Prerequisite Scanner 在 /lib/common_function.vbs 文件中提供了一组用于处理文件的通用文件子例程。另外，还提供了一组用于处理文件的函数。

表 34 对文件实用程序作了描述。

表 34. 文件实用程序子例程

子例程	描述	输入参数
appendToFile(text, fileName)	将文本追加到指定文件的末尾。	• text，这是包含要对文件追加的文本的字符串 • filename，这是表示要修改的文件的名称的字符串
writeToFile(text, fileName)	将文本写入指定的文件（必要时覆盖现有内容）。	• text，这是包含要写入文件的文本的字符串。 • filename，这是表示要修改的文件的名称的字符串

表 35 对用于处理文件的文件函数作了概述。

表 35. 文件实用程序函数

函数	描述
第 110 页的『allFiles()』	将指定目录中的文件名读取到一个数组中。
第 114 页的『filterFile()』	读取文件内容并对其进行过滤，然后将过滤结果写入一个脚本编制字典对象。
第 114 页的『findNewest()』	查找最新的配置文件。
第 115 页的『findSuitableFile()』	查找产品和版本的相关配置文件。
第 119 页的『ppread()』	将文件内容读取到脚本编制字典对象中，从而按指定的分隔符输入参数对文件中的每一行进行进一步分割（如果该分隔符存在于该行中）。
第 120 页的『readFile()』	将文件的每一行读取到数组的下标条目中。

附录 H. Windows 系统的其他通用函数和子例程

Prerequisite Scanner 提供了另一组在各个文件中使用的通用函数和子例程。

对其他通用函数和子例程作了概述。

表 36. Windows 系统的其他通用函数和子例程

函数或子例程	描述
『ffirstMatch()』	获取搜索字符串在数组中的第一个匹配项。
第 128 页的『getValue()』	获取所指定目录的可用磁盘空间量。
第 128 页的『removeSpecialCharacters()』	除去商标或其他特殊字符以便于进行比较。
第 129 页的『versionCompare()』	解析用于表示先决条件属性的实际值和期望值的输入参数，并对其进行比较，以确定先决条件属性是否通过了先决条件检查。此函数期望使用以点分隔的版本字符串作为输入参数，例如 1.0.0.4、2.3、3.40.26.7800 或 2.3.*。

ffirstMatch()

获取搜索字符串在数组中的第一个匹配项。

用途

此函数使用正则表达式在作为输入参数传递的数组中搜索同样作为输入参数传递的模式。在数组中找到该模式的第一个匹配项后，此函数将该数组中的值添加到脚本编制字典对象。

父函数

表 37. 调用了 ffirstMatch() 的父函数

父函数和脚本	描述
DB2_Version_compare.vbs 中的 ud620db2level (expect, real)	对 DB2 版本先决条件属性的实际值和期望值进行比较。
OS_Version_compare.vbs 中的 oslevelcompare (expect, real)	对操作系统版本先决条件属性的实际值和期望值进行比较。

语法

```
ffirstmatch(patt,arr)
```

输入参数

String *patt*
包含要搜索的模式。

Array *arr*
包含要在其中进行搜索模式搜索的数组。

返回值

Dictionary *keys*
返回脚本编制字典对象的键。

getValue()

获取所指定目录的可用磁盘空间量。

用途

这个子例程使用文件系统对象的实例针对路径输入参数调用 getDriveName() 函数，然后使用 freeSpace 属性来获取可用磁盘空间量（接着，此空间量转换为以 MB 为单位）。先决条件属性输入参数及其值将写入脚本文件的相关联临时文本文件。

脚本

表 38. 使用了 getValue() 的脚本

脚本	描述
DEZ_01040000.vbs	此脚本用于收集先决条件属性并使其仅可供 DEZ 01040000 配置文件使用
LCM_TAD_common.vbs	此脚本用于收集先决条件属性并使其仅可供 LCM 02300000 和 TAD 07200000 配置文件使用
TAD722_impl.vbs	此脚本用于收集先决条件属性并使其仅可供 TAD 07220000 配置文件使用

语法

```
getValue fso, sKey, drvPath
```

输入参数

File system object *fso*
这是文件系统对象的实例。

String *sKey*
包含一个具有先决条件属性名称和等号的字符串。

String *drvPath*
包含要获取其可用磁盘空间量的路径。

返回值

无

removeSpecialCharacters()

除去商标或其他特殊字符以便于进行比较。此函数位于 /lib/common.vbs 文件中。

用途

此函数调用 Replace() 函数，以便将商标、版权和注册符号替换为 ""。

语法

```
removeSpecialCharacters(s)
```

输入参数

String *s*
包含必须从中除去字符的字符串

返回值

String *s*
返回不包含特殊字符的字符串。

versionCompare()

解析用于表示先决条件属性的实际值和期望值的输入参数，并对其进行比较，以确定先决条件属性是否通过了先决条件检查。此函数期望使用以点分隔的版本字符串作为输入参数，例如 1.0.0.4、2.3、3.40.26.7800 或 2.3.*。

用途

此函数首先处理一个或两个输入参数为空的特殊情况，并返回表示这些情况的返回码。它通过点分隔符将每个版本分割成多个部分。如果版本的最后一部分是 * 通配符，那么此函数将该版本的所有缺失部分视为通配符，例如，2.* 与 2.1 或 2.3.* 匹配。然后，此函数对每个版本的各个组成部分的列表执行循环，并对其进行比较。接着，它根据期望值是小于、等于还是大于实际值来返回相应的返回码。

父函数

表 39. 调用了 versionCompare 的父函数

父函数和脚本	描述
cygwinVersion_compare.vbs	对 cygwin 版本先决条件属性的实际值和期望值进行比较。
gskit7Version_compare.vbs	对 gskit V7 先决条件属性的实际值和期望值进行比较。
gkit8Version_compare.vbs	对 gskit V8 先决条件属性的实际值和期望值进行比较。
internetExplorer.version_compare.vbs	对 Internet Explorer 版本先决条件属性的实际值和期望值进行比较。
os.servicePack_compare.vbs	对操作系统服务包先决条件属性的实际值和期望值进行比较。
os.versionNumber_compare.vbs	对操作系统版本先决条件属性的实际值和期望值进行比较。

语法

```
versionCompare(ver1,ver2)
```

输入参数

String *ver1*
包含先决条件属性的期望版本。

String *ver2*
包含先决条件属性的实际版本。

返回值

Integer *0*
如果两个输入参数相等，那么将返回 0 返回码。父函数将返回 "PASS"。

特殊情况：如果两个输入参数均为空，那么将返回 0 返回码并退出。

Integer *-1*
如果第一个输入参数小于第二个输入参数，那么将返回 -1 返回码。父函数将返回 "FAIL"。

特殊情况：如果第一个输入参数为空，那么将返回 -1 返回码并退出。

Integer *1*
如果第一个输入参数大于第二个输入参数，那么将返回 1 返回码。父函数将返回 "PASS"。

特殊情况：如果第二个输入参数为空，那么将返回 1 返回码并退出。

附录 I. UNIX 系统的通用函数

Prerequisite Scanner 在 /lib/common_function.sh 文件中提供了一组用于在基于 UNIX 的系统上运行检查的通用函数。

表 40. common_function.sh 中的函数

函数	描述
第 132 页的『AddMG()』	检查输入参数是以 MB 还是 GB 为单位，并对各个参数进行累加。
『changeMG()』	对于磁盘空间或内存先决条件属性，将输入参数转换为以 MB 或 GB 为单位。
第 132 页的『compare()』	解析用于表示先决条件属性的实际值和期望值的输入参数，并对这些参数进行比较，以确定第一个值（实际值）是否小于第二个值（期望值）。
第 133 页的『cutdown()』	解析用于表示先决条件属性的实际值和期望值的输入参数，并对这些参数进行比较，以确定第一个值（实际值）是否小于第二个值（期望值）。然后，如果第一个值不小于第二个值，那么打印这两个值之差。
第 135 页的『findOSInfo()』	查找系统的操作系统版本、操作系统发行版级别和版本以及硬件实现数据。
第 134 页的『mes4path()』	查找每个已装配的文件系统的可用磁盘空间量。
第 134 页的『mes4Path1()』	查找每个已装配的文件系统的可用磁盘空间量（仅适用于 Solaris 系统）。
第 136 页的『NFScheck()』	检查基于 UNIX 的系统上的装配的 NFS 状态。
第 135 页的『telnetNFS()』	检查能否使用 Telnet 通过通过缺省端口 2049 访问装配的文件系统的 IP 地址。

changeMG()

对于磁盘空间或内存先决条件属性，将输入参数转换为以 MB 或 GB 为单位。

用途

此函数先检查是否接收到输入参数。如果接收到输入参数，那么它确定值是以 MB 还是 GB 为单位，然后将值分别转换为 GB 或 MB。

语法

```
changeMG val
```

输入参数

String *$val*
包含磁盘空间量值或内存量值（以 MB 或 GB 为单位）。

返回值

Integer *1*
如果此函数未接收到输入参数，那么将返回 1。

String *printf "%.0fM%s",mm[1]*1024,mm[2];*
返回以 MB 为单位的值。

String *printf "%.2fG%s",mm[1],mm[2];*
返回以 GB 为单位的值。

AddMG()

检查输入参数是以 MB 还是 GB 为单位，并对各个参数进行累加。

用途

此函数先检查是否接收到输入参数。如果接收到输入参数，那么确定值是以 MB 还是 GB 为单位，然后对值进行累加。

语法

```
AddMG val1 val2
```

输入参数

String *$val1*
包含要添加到另一个输入参数的磁盘空间量值或内存量值（以 MB 或 GB 为单位）。

String *$val2*
包含要添加到另一个输入参数的磁盘空间量值或内存量值（以 MB 或 GB 为单位）。

返回值

Integer *1*
如果此函数未接收到两个输入参数，那么将返回 1。

String *val*
返回以 MB 或 GB 为单位的累加值。

compare()

解析用于表示先决条件属性的实际值和期望值的输入参数，并对这些参数进行比较，以确定第一个值（实际值）是否小于第二个值（期望值）。

用途

此函数先检查是否接收到两个输入参数。如果接收到两个输入参数并且这两者并非均为 false，那么它将确定这些值是以 MB 还是 GB 为单位，然后对这两个值进行比较以检查第一个值是否小于第二个值。如果是，那么将返回 false 值；否则将返回 pass 值。

语法

```
compare real expected
```

输入参数

String *$real*
包含先决条件属性的实际值。

String *$expected*
包含先决条件属性的期望值。

返回值

Integer *1*
如果此函数未接收到两个输入参数，那么将返回 1。

String *"FAIL|PASS"*
如果实际值小于期望值，那么将返回字符串 "FAIL"；否则将返回字符串 "PASS"。

cutdown()

解析用于表示先决条件属性的实际值和期望值的输入参数，并对这些参数进行比较，以确定第一个值（实际值）是否小于第二个值（期望值）。然后，如果第一个值不小于第二个值，那么打印这两个值之差。

用途

此函数先检查是否接收到两个输入参数。如果接收到两个输入参数，那么它将确定值是以 MB 还是 GB 为单位，然后将它们转换为 MB（如果原先以 GB 为单位）。然后，此函数对这两个值进行比较以检查第一个值是否小于第二个值。如果是，那么将返回“0MB”值，否则将返回这两个值之差（以 MB 为单位）。

语法

```
cutdown real expected
```

输入参数

String *$real*
包含先决条件属性的实际值。

String *$expected*
包含先决条件属性的期望值。

返回值

Integer *1*
如果此函数未接收到两个输入参数，那么将返回 1。

String *"FAIL|PASS"*
任意一个值以 MB 或 GB 为单位时，如果实际值小于期望值，那么将返回字符串 "FAIL"；否则将返回字符串 "PASS"。

String *"OMB|Real-ExpectedMB"*
如果实际值小于期望值，那么将返回字符串 "OMB"，否则将返回两个转换后的值之差（以 MB 为单位）的字符串表示。

mes4path()

查找每个已装配的文件系统的可用磁盘空间量。

用途

此函数使用路径作为输入，并调用 **uname** 命令以确定操作系统，然后调用 NFScheck 函数以确定该系统是否已启动并确定装配。接着，它调用 df 命令以确定系统上每个装配的可用磁盘空间量。然后，返回可用磁盘空间量的值。

语法

```
mes4Path path
```

输入参数

String *$path*
要检查可用磁盘空间量的系统的路径。

返回值

Integer *1*
如果此函数未接收到输入参数，那么将返回 1 返回码。

Integer *2*
如果输入参数不是路径，那么将返回 2 返回码。

String *$NF*
返回各个装配的可用磁盘空间量。

String *"$path Server NotAvailable Responding for $path"*
返回一条消息，以指出该路径的服务器不可用。

mes4Path1()

查找每个已装配的文件系统的可用磁盘空间量（仅适用于 Solaris 系统）。

用途

此函数使用路径作为输入，并调用 **uname** 命令以确定操作系统是否为 Solaris。接着，它调用 df 命令以确定系统上每个装配的可用磁盘空间量。然后，返回可用磁盘空间量的值。

语法

```
mes4Path1 path
```

输入参数

String *$path*
要检查可用磁盘空间量的系统的路径。

返回值

Integer *1*
如果此函数未接收到输入参数，那么将返回 1 返回码。

Integer *2*
如果输入参数不是路径，那么将返回 2 返回码。

String *$NF*
返回各个装配的可用磁盘空间量。

findOSInfo()

查找系统的操作系统版本、操作系统发行版级别和版本以及硬件实现数据。

用途

此函数运行 uname 命令，并解析命令输出以确定系统的操作系统版本、操作系统发行版级别和版本以及硬件实现数据。

语法

```
findOSInfo
```

输入参数

无

返回值

String *$oo*
uname 的输出（不含基本系统信息）。

String *$kk*
操作系统版本。

String *$hh*
表示为 I（对于 i386 硬件）或 Z（对于 s390 硬件）的硬件实现。

String *$rr*
操作系统发行版级别。

String *$vv*
操作系统发行版级别版本。

telnetNFS()

检查能否使用 Telnet 通过通过缺省端口 2049 访问装配的文件系统的 IP 地址。

用途

此函数使用 IP 作为输入，并调用 **telnet** 命令以测试缺省 Telnet 端口 2049 上进行的远程连接是否成功。它将尝试进行 10 次远程连接。如果 **telnet** 命令失败，那么此函数将返回 "FALSE" 值，否则将返回 "PASS" 值。

语法

```
telnetNFS ipaddr
```

输入参数

String *$ipaddr*

这是要检查能否对其执行 Telnet 的 IP 地址。

返回值

String *"FALSE|TRUE"*

返回 Telnet 检查结果。如果检查成功，那么此函数将返回 "TRUE"，否则将返回 "FALSE"。

NFScheck()

检查基于 UNIX 的系统上的装配的 NFS 状态。

用途

此函数使用路径作为输入，并调用 mount 命令以获取已装配的文件系统的列表。它将调用 **uname** 命令以确定操作系统。接着，此函数调用 **ping** 命令对每个已装配的系统执行 Ping 操作，如果可以执行 Ping 操作，那么将调用 **telnetNFS** 函数以检查能否执行远程连接。如果 Ping 操作或 Telnet 操作失败，那么此函数将返回 "FALSE" 值，否则将返回 "PASS" 值。

语法

```
NFScheck path
```

输入参数

String *$path*

使用目录的有效路径作为其输入。

返回值

Boolean value *TRUE* **或** *FALSE*

如果 NFS 检查成功（即，如果成功地对相关 IP 地址执行了 Ping 操作，或者可以使用 Telnet 来连接到各个文件系统的相关 IP 地址），那么将返回 TRUE，否则返回 FALSE。

示例

此用法示例来自于 **mes4Path()** 函数:

```
# check if it's a path
 path=`echo "$1" | sed -n '/^\//p'`
 if [ -z "$path" ];then
  return 2;
 else
```

```
     nfs_check_status=`NFScheck $path`
     if [ "$nfs_check_status" = "TRUE" ]; then
    case `uname` in
...
```

附录 J. UNIX 系统的其他函数

Prerequisite Scanner 在各个文件中提供了一组通用函数。

表 41 对多个文件中的这组函数作了概述。

表 41. 多个文件中的通用函数

函数	描述
第 140 页的『formatSizeDisplay()』	接受输入参数，并将输入参数的小数部分追加或修剪为两个小数位，例如将 123MB 转换为 123.00MB，或者将 12.123MBs 转换为 12.12MBs。
第 140 页的『versionCompare()』	解析用于表示先决条件属性的实际值和期望值的输入参数，并对版本的各个部分进行比较，以确定先决条件属性是否通过了先决条件检查。

表 42 对 `UNIX-Linux/TAD722_impl.sh` 文件中用于对 Tivoli License Compliance Manager 和 Tivoli Asset Discovery for Distributed 运行检查的这组函数作了概述。

表 42. TAD722_impl.sh 中的通用函数

函数	描述
第 142 页的『checkSunOS()』	检查 Solaris 操作系统版本是用于 SPARC 还是 X86 平台。
第 141 页的『checkHpux()』	检查 HP-UX 操作系统版本是用于 IA64 还是 PARISC 平台。
第 142 页的『checkLinux()』	检查 Linux 操作系统版本是用于 System p®、System z 还是 x86 平台。
第 143 页的『getSystemId()』	调用不同的操作系统函数，以检查相关操作系统的平台。
第 142 页的『getValue()』	获取指定文件中某个键的值（如果该键存在）。
第 143 页的『setValue()』	设置指定文件中某个键的值（如果该先决条件键存在）。
第 143 页的『copyValue()』	根据产品和操作系统来获取并设置先决条件属性（键）的值。
第 144 页的『parseDirParameter()』	根据扫描程序的 -p 标志的参数列表解析参数，并将该参数的值放入列表。
第 144 页的『getClosestExistingParentDir()』	获取最接近的父目录或自身。
第 144 页的『printDirSize()』	检查已装配的文件系统的 NFS 状态，然后获取该文件系统或其父目录的磁盘空间量。

formatSizeDisplay()

接受输入参数，并将输入参数的小数部分追加或修剪为两个小数位，例如将 123 MB 转换为 123.00 MB，或者将 12.123 MB 转换为 12.12 MB。

用途

此函数对输入参数中的字符进行计数，检查该输入参数是数字还是字符串，并将输入部分分割为整数和小数部分。根据小数部分的不同，此函数将其追加或修剪为两个小数位。然后，返回结果。

父脚本

下列脚本包含此函数:

- `./Unix-Linux/common.sh`
- `LCM_TAD_common.sh`

语法

```
formatSizeDisplay val
```

输入参数

Integer *$val*
要四舍五入到小数点后两位的值。

返回值

Integer *val*
返回四舍五入到小数点后两位的值。

versionCompare()

解析用于表示先决条件属性的实际值和期望值的输入参数，并对版本的各个部分进行比较，以确定第一个值（实际值）是否大于第二个值（期望值）。

用途

此函数先检查是否接收到两个版本作为输入参数。它使用 awk 来解析各个版本并将其分割为各个部分，其中“.”是用于将值分割为各个部分的定界符。然后，它执行一个循环，将第一个版本的各个部分与第二个版本的相同部分进行比较，并确定它们是否相等。

父函数

表 43. 调用了 versionCompare 的父函数

父函数和脚本	描述
`db2.home_compare.sh`	对 DB2 HOME 先决条件属性的磁盘空间实际值和期望值进行比较。
`oracle.Client_compare.sh`	对 Oracle 客户机先决条件属性的实际值和期望值进行比较。
`os.locale_compare.sh`	对操作系统语言环境先决条件属性的实际值和期望值进行比较。

表 43. 调用了 versionCompare 的父函数 (续)

父函数和脚本	描述
os.MozillaVersion_compare.sh	对 Mozilla Firefox 先决条件属性的实际值和期望值进行比较。
os.package.perl_compare.sh	对 Perl 软件包先决条件属性的实际值和期望值进行比较。调用自身。
os.RAMSize_compare.sh	对 RAM 必备属性的实际值和期望值进行比较。
os.space_compare.sh	对可用磁盘空间量先决条件属性的实际值和期望值进行比较。
OS_Version_compare.sh	对操作系统版本先决条件属性的实际值和期望值进行比较。

语法

```
versionCompare real expected
```

输入参数

String *$real*
包含先决条件属性的实际值。

String *$expected*
包含先决条件属性的期望值。

返回值

Integer *0*
如果实际值与期望值相等，那么返回 0 返回码。父函数将返回“PASS”。

特殊情况：如果函数接收到空的输入参数，那么将返回 0 返回码并退出。

Integer *-1*
如果实际值小于期望值，那么将返回 -1 返回码。父函数将返回 "FAIL"。

如果函数接收到的第二个输入参数为空，那么将返回 -1 返回码并退出。

Integer *1*
如果实际值大于期望值，那么将返回 1 返回码。父函数将返回 "PASS"。

如果函数接收到的第一个输入参数为空，那么将返回 1 返回码并退出。

checkHpux()

检查 HP-UX 操作系统版本是用于 IA64 还是 PARISC 平台。

用途

此函数使用 **uname** 命令的 -m 标志来确定 HP-UX 操作系统是用于 IA64 还是 PARISC 平台。

语法

```
checkHpux
```

返回值

String *HPUXIA64*|*HPUXPARISC*

如果 `-m` 标志为 `"ia64"`，那么此函数将返回 `"HPUXIA64"`，否则将返回 `"HPUXPARISC"`。

checkLinux()

检查 Linux 操作系统版本是用于 System p、System z 还是 x86 平台。

用途

此函数使用 **uname** 命令的 `-m` 标志来确定 Linux 操作系统是用于 System p、System z 还是 x86 平台。

语法

```
checkLinux
```

输入参数

返回值

String *LINUXPSERIES*|*LINUXZSERIES*|*LINUXX86*

如果 `-m` 标志为 `"ppc64"` 或 `"ppc"`，那么此函数将返回 `"LINUXPSERIES"`。如果值为 `"s390x"` 或 `"s390"`，那么此函数将返回 `"LINUXZSERIES"`，否则将返回 `"LINUXX86"`。

checkSunOS()

检查 Solaris 操作系统版本是用于 SPARC 还是 X86 平台。

用途

此函数使用 **uname** 命令的 `-p` 标志来确定 Solaris 操作系统是用于 SPARC 还是 X86 平台。

语法

```
checkSunOS
```

输入参数

返回值

String *SOLARISSPARC*|*SOLARISX86*

如果 `-p` 标志为 `"sparc"`，那么此函数将返回 `"SOLARISSPARC"`，否则将返回 `"SOLARISX86"`。

getValue()

获取指定文件中某个键的值（如果该键存在）。

用途

语法

```
getValue key file
```

输入参数

String *$key*
包含要设置的键。

String *$file*
包含该键所在文件的名称。

setValue()

设置指定文件中某个键的值（如果该先决条件键存在）。

语法

```
setValue key value file
```

输入参数

String *$key*
包含要设置的先决条件属性。

String *$value*
包含该先决条件属性的值。

String *$file*
包含该先决条件属性所在文件的名称。

copyValue()

根据产品和操作系统来获取并设置先决条件属性（键）的值。

用途

此函数调用 **getValue()** 函数，以获取产品和操作系统的所指定先决条件属性的值。接着，它将调用 **setValue()** 函数，以便在 Prerequisite Scanner 文件中设置先决条件属性的值。

语法

```
copyValue key file
```

输入参数

String *$key*
包含要获取并设置的键。

String *$file*
包含该键所在文件的名称。

返回值

getSystemId()

调用不同的操作系统函数，以检查相关操作系统的平台。

用途

此函数调用不同的操作系统函数，以确定相关操作系统的平台。

语法

```
getSystemId
```

输入参数

返回值

String *AIX|Linux*
如果产品为 Tivoli License Compliance Manager 且操作系统为 AIX 或 Linux，那么此函数将返回 "AIX" 或 "Linux"；如果产品为 Tivoli Asset Discovery for Distributed 且操作系统为 AIX，那么将返回 ″AIX″。

getClosestExistingParentDir()

获取最接近的父目录或自身。

用途

语法

```
getClosestExistingParentDir dirpath
```

输入参数

String *$dirpath*
包含要获取其父目录或自身的路径。

返回值

String *dirpath*
返回父目录或自身。

parseDirParameter()

根据扫描程序的 -p 标志的参数列表解析参数，并将该参数的值放入列表。

用途

语法

输入参数

String

返回值

printDirSize()

检查已装配的文件系统的 NFS 状态，然后获取该文件系统或其父目录的磁盘空间量。

用途

此函数首先调用 **NFScheck** 函数以确定目录的 NFS 状态。如果状态为 true，那么它将调用 **getClosestExistingParentDir** 函数以返回该目录或其父目录，然后使用 **df** 命令来获取可用磁盘空间量。最后，调用 **formatSizeDisplay** 函数将值四舍五入到特定小数位。

语法

```
printDirSize dirpath
```

输入参数

String *$dirpath*
包含要获取其可用磁盘空间量的目录的路径。

返回值

Integer *dsize*
返回两个小数位的可用磁盘空间量。

String *"NFS_NOT_AVAILABLE"*
此返回值表明装配的文件系统不可用。

附录 K. UNIX 系统的日志记录实用程序函数

Prerequisite Scanner 在 /lib/common_function.sh 文件中提供了一组通用日志记录函数，用于将调试和跟踪数据写入日志文件。

表 44 对日志记录实用程序作了描述。

表 44. UNIX 系统上的日志记录实用程序函数

函数	描述	输入参数

wrlTrace *log_str1 log_str2*	将 *log_str1* 和 *log_str2* 字符串随时间戳记一起写入跟踪文件	跟踪字符串 *log_str1* 和 *log_str2*，用于表示要执行并记录到跟踪文件中的操作和收集器。例如:
		`wrlTrace Starting os.lib`
		`wrlTrace Executing os.lib`
		`wrlDebug Starting os.lib`
		`wrlDebug Expected libXp `
		ss=`./os.lib libXp libXp`
		`wrlTrace Finished os.lib`
		echo "os.lib.libXp=$ss"
		`wrlDebug Finished os.lib`
		`wrlDebug OutPutValueIs $ss`
		`wrlTrace Done os.lib`

wrlTraceFuncStart *fn_name*	将 *fn_name* 函数传递到 wrlTrace()	跟踪字符串 *fn_name*，用于表示刚刚调用的函数。例如: `wrlTraceFuncStart "$1"`
wrlTraceFuncExit *fn_name*	将 *fn_name* 函数传递到 wrlTrace()	跟踪字符串 *fn_name*，用于表示刚刚完成的函数。例如: `wrlTraceFuncExit "$1"`

wrlDebug *log_str1 log_str2*	将 *log_str1* 和 *log_str2* 字符串传递到 wrlDebugGeneric()	调试字符串 *log_str1* 和 *log_str2*，用于表示要执行并记录到调试文件中的操作和收集器。例如:
		`wrlTrace Starting os.lib`
		`wrlTrace Executing os.lib`
		`wrlDebug Starting os.lib`
		`wrlDebug Expected libXp `
		ss=`./os.lib libXp libXp`
		`wrlTrace Finished os.lib`
		echo "os.lib.libXp=$ss"
		`wrlDebug Finished os.lib`
		`wrlDebug OutPutValueIs $ss`
		`wrlTrace Done os.lib`

wrlDebugFuncStart *fn_name*	将 *fn_name* 函数传递到 wrlDebug()	调试字符串 *fn_name*，用于表示刚刚调用的函数。例如: `wrlDebugFuncStart "$1"`
wrlDebugFuncExit *fn_name*	将 *fn_name* 函数传递到 wrlDebug()	调试字符串 *fn_name*，用于表示刚刚完成的函数。例如: `wrlDebugFuncExit "$1"`

表 44. UNIX 系统上的日志记录实用程序函数 (续)

函数	描述	输入参数
`wrlDebugFuncReturn result_value`	将函数返回的 *result_value* 结果写入日志文件	调试字符串 *result_value*，用于表示函数返回的值。例如: `` `wrlDebugFuncReturn "$versionCompare"` ``
`wrlDebugFuncParam param1 param2`	将 *param1* 和 *param2* 参数传递到 `wrlDebugFunc()`	调试字符串 *param1* 和 *param2*，用于表示被调用函数的已解析节标题、已解析限定符或者已解析输入自变量。例如: `` `wrlDebugFuncParam "OSArch" "$3"` ``
`wrlDebugGeneric formatspec log_str1 log_str2`	将 *log_str1* 和 *log_str2* 字符串写入由 *formatspec* 字符串自变量编排格式的调试文件	• 字符串 *log_str1* 和 *log_str2*，用于表示要在调试文件中的某一行中记录的特定数据 • *formatspec*，这是要放在时间戳记之后、左对齐的日志字符串和换行符之前的字符串自变量 例如: `` `wrlDebugGeneric "" "$1" "$2"` ``
`wrlDebugFunc str`	将跳进字符和 *str* 输入参数传递到 `wrlDebugGeneric()`	字符串 *str*，用于表示要记录的数据，即执行的检查或操作的状态。例如: `` `wrlDebugFunc "Reading config file and parsing using parse array..." ` ``
`wrlLogFuncStart str`	将 *str* 输入参数传递到 `wrlTraceFuncStart()` 和 `wrlDebugFuncStart()`	字符串 *str*，用于表示要记录的数据，即调用的函数的名称。例如: `` `wrlLogFuncStart "main()"` ``
`wrlLogFuncExit str`	将 *str* 输入参数传递到 `wrlTraceFuncExit()` 和 `wrlDebugFuncExit()`	字符串 *str*，用于表示要记录的数据，即退出的函数的名称。例如: `` `wrlLogFuncExit "main()"` ``

声明

本信息是为在美国提供的产品和服务编写的。IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

有关双字节 (DBCS) 信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区:

International Business Machines Corporation“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。

某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本出版物中描述的产品进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名字都是虚构的，若现实生活中实际业务企业使用的名字和地址与此相似，纯属巧合。

版权许可:

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口 (API) 进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。用户如果是为了按照 IBM 应用程序编程接口开发、使用、经销或分发应用程序，那么可以任何形式复制、修改和分发这些样本程序，而无须向 IBM 付费。

如果您正在查看本信息的软拷贝，图片和彩色图例可能无法显示。

商标

IBM、IBM 徽标和 ibm.com® 是 International Business Machines Corp. 在全球许多行政管辖地区的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。Web 页面“版权和商标信息”(www.ibm.com/legal/copytrade.shtml) 提供了 IBM 商标的最新列表。

Adobe、Acrobat、PostScript 和所有基于 Adobe 的商标是 Adobe Systems Incorporated 在美国和/或其他国家或地区的注册商标或商标。

Cell Broadband Engine 和 Cell/B.E. 是 Sony Computer Entertainment, Inc. 在美国和/或其他国家或地区的商标，经特许使用。

Intel、Intel 徽标、Intel Inside、Intel Inside 徽标、Intel Centrino、Intel Centrino 徽标、Celeron、Intel Xeon、Intel SpeedStep、Itanium 和 Pentium 是 Intel Corporation 或其子公司在美国和其他国家或地区的商标或注册商标。

IT Infrastructure Library 是 Central Computer and Telecommunications Agency（它现在是 Office of Government Commerce 的一部分）的注册商标。

ITIL 是一个注册商标，是 Office of Government Commerce 的共同体注册商标，并且已在 U.S. Patent and Trademark Office 进行注册。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的商标。

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

Java 和所有基于 Java 的商标和徽标是 Sun Microsystems, Inc. 在美国和/或其他国家或地区的商标或注册商标。

其他公司、产品和服务名称可能是其他公司的商标或服务标记。

支持信息和反馈

如果您的 IBM 软件出现问题，您肯定希望迅速加以解决。IBM 为您提供了不同的方式来获取所需的支持，例如通过在线方式或使用 IBM Support Assistant 来获取支持。另外，您还可以提供反馈或提交产品改进请求。

在线 下列站点提供了故障诊断信息:

- 访问 IBM Support Portal 上的 IBM Prerequisite Scanner 页面。
- 访问 Service Management Connect 上的 Prerequisite Scanner 主题。您可以随意对这些主题进行补充。

使用下列站点来提供反馈、提交请求或讨论 Prerequisite Scanner:

- 访问 Prerequisite Scanner at Service Management Connect 上的 Prerequisite Scanner 主题。您可以随意对这些主题进行补充。
- 使用 Service Management Connect 上的 Integrated Service Management Message Board。
- 在 Tivoli RFE Community 提交或查看针对 Prerequisite Scanner 的产品改进请求。

IBM Support Assistant

IBM Support Assistant (ISA) 是一个免费的本地软件可维护性工作台，可以帮助您解决 IBM 软件产品问题。ISA 使您能够快速访问与支持相关的信息和可维护性工具，以便确定问题。要安装 ISA 软件，请访问 http://www.ibm.com/software/support/isa。

索引

[M]

[N]

[P]

[R]

[S]

[T]

[W]

[X]

[Y]

[Z]

C

D

I

M

O

P

S

T

U

V

W

X

[特别字符]

Printed in China