Prerequisite Scanner
Versión 1.2

Guía del usuario

IBM

Prerequisite Scanner
Versión 1.2

Guía del usuario

IBM

Nota
Antes de utilizar esta información y el producto al que da soporte, lea la información del "Avisos" en la página 163.

Esta edición se aplica a la Versión 1.2 de IBM Prerequisite Scanner y a todos los siguientes releases y modificaciones hasta que se indique lo contrario en nuevas ediciones.

Contenido

Figuras

Tablas

Capítulo 1. Visión general de Prerequisite Scanner

IBM® Prerequisite Scanner es una herramienta de exploración que lleva a cabo la identificación, comprobación y verificación de los requisitos previos del software especificado antes que se lleve a cabo la implementación efectiva. Analiza el hardware y requisitos previos de software en base a los valores establecidos para las propiedades de requisitos previos.Scanner muestra los resultados del análisis en la interfaz de línea de comandos y también guarda los resultados en archivos de texto y opcionalmente XML. Graba también mensajes informativos, de seguimiento y de depuración en archivos de registro.

Prerequisite Scanner puede comprobar el sistema operativo de la máquina y comprobar si es la versión correcta del software especificado. Si alguno de los controles individuales de los requisitos previos falla, el análisis general fallará también.

Puede ejecutar Prerequisite Scanner después de una instalación o en cualquier momento para confirmar su entorno actual. Prerequisite Scanner no requiere que se ejecute el programa de instalación del software para el que desea comprobar los requisitos previos.

Puede ampliar Prerequisite Scanner para analizar los requisitos previos que no forman parte del conjunto básico de comprobaciones de requisitos previos suministrado con Scanner.

Prerequisite Scanner invoca los siguientes tipos de secuencias de comandos, en función de la plataforma:

- Windows: VBScript y por lotes
- UNIX: shell

Nota: No podrá ejecutar las secuencias de comandos de UNIX en sistemas Windows aunque haya instalado un entorno similar a UNIX en las máquinas Windows; por ejemplo, Cygwin.

Arquitectura de Prerequisite Scanner

IBM Prerequisite Scanner comprende los siguientes componentes principales: una secuencia de comandos para ejecutar en una interfaz de línea de comandos, un conjunto de propiedades para las comprobaciones de requisitos previos, archivos de configuración de propiedades de requisitos previos, recopiladores de requisitos previos y evaluadores de requisitos previos. Los resultados de la ejecución de Prerequisite Scanner están disponibles en varios formatos de salida.

Propiedades de requisitos previos

Las Propiedades de requisitos previos son los valores esperados para diferentes programas y requisitos de hardware, requeridos por los productos o soluciones que se van a instalar. Algunos ejemplos de propiedades de requisitos previos incluyen el espacio total disponible en la máquina, el conjunto de puertos que no están en uso en una máquina, y el conjunto actual de las aplicaciones instaladas.

Debido a que los valores de estas propiedades de requisitos previos pueden cambiar con diferentes productos, las propiedades y sus valores se representan

como pares de nombre y valor, con calificadores opcionales. Están incluidas en los archivos de configuración de propiedades de requisitos previos. Solo debe haber una propiedad de requisito previo por línea.

Las propiedades de requisitos previos tienen el siguiente formato:

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

donde:

- *prefix_identifier* es un identificador de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4. Algunas de las categorías predefinidas requieren este identificador de prefijo.
- *property_name* es el nombre de la propiedad de requisito previo.
- *suffix_identifier* es un identificador opcional de un subtipo de las propiedades de requisitos previos como se indica en la Tabla 4 en la página 7.
- *qualifier_name* es un atributo opcional de la propiedad de requisito previo. IBM Prerequisite Scanner lo utiliza para calificar la propiedad de requisito previo o el tipo de verificación para llevar a cabo en la propiedad de requisito previo.

  Nota: Puede tener múltiples calificadores, cada uno de ellos separado por una coma. El conjunto de calificadores debe estar entre corchetes `[]`.
- *qualifier_value* es el valor del atributo opcional. Cada calificador y su valor deben estar delimitados por dos puntos `:`.
- *property_value* es el valor de la propiedad de requisito previo y puede ser una cadena o un entero.

Una propiedad de requisito previo puede tener uno o muchos valores dependiendo del tipo de datos y calificador, como se indica a continuación:

- Un solo número entero; por ejemplo, `8080` para representar el valor de un número de puerto.
- Un rango o grupo de números enteros representados mediante caracteres especiales, como se indica en la Tabla 1.

Tabla 1. Caracteres especiales para representar tipos de rango

Carácter especial	Descripción
*	Identifica un marcador de posición para varios valores. Por ejemplo, `ports.*` puede representar un superconjunto de puertos para un producto de base de datos, `ports.DB` e IBM WebSphere Servidor de aplicaciones, `ports.WAS` .
+	Identifies that the actual version must at least match the value for expected version. For example, `os.versionNumber=5.0+`, means that the version must be 5.0 or later.
-	Identifica que la versión actual debe coincidir como mucho con el valor esperado de la versión. Por ejemplo, `os.versionNumber=5.0-`, significa que la versión debe ser 5.0 o anterior.
.*	Identifica que la versión real puede coincidir con cualquier valor de comodín para la versión esperada. Por ejemplo: `os.versionNumber=5.*, means that the version can be 5.0, 5.0.1 or 5.5.`

Restricción: En sistemas Windows, el comodín `*` solo se admite si se utiliza dentro de una expresión regular en la propiedad de requisito previo `OS Version`.

- Una cadena que puede representar cualquiera de los siguientes valores para los tipos de requisito previos
  - Un valor numérico con una unidad; por ejemplo, `8GB` o `10MB`
  - Una aplicación, sistema operativo, arquitectura, o paquete; por ejemplo, `IBM Lotus Symphony`, `RedHat Enterprise Linux 5.4`, `32-bit` o `ftp`

    Nota: Una cadena también podría comprender varios valores separados por una coma; por ejemplo, una lista de aplicaciones.
  - Cualquiera o valores representados por una de las siguientes combinaciones; por ejemplo, `True|False`, `Available|Unavailable` o `Enabled|Disabled`

Tabla 2 describe las propiedades de requisitos previos.

Tabla 2. Ejemplos de propiedades de requisitos previos

Propiedad de requisito previo	Explicación
`Disk=1GB`	La cantidad de espacio libre en disco, donde: • *property_name* es `Disk` • *property_value* es `1GB`
`user.isAdmin=True`	Si el usuario registrado pertenece a un grupo de administradores, donde: • *prefix_identifier* es `user`, para las propiedades de requisitos previos de usuario • *property_name* es `isAdmin` • *property_value* es `True`

`network.availablePorts.DB=60000-60005`
`network.availablePorts.WAS=8080`
`network.availablePorts.FTP=21`
	Comprueba si los puertos 60000-60005 están disponibles para el servidor de base de datos, si el puerto 8080 está disponible para WebSphere Application Server y si el puerto 21 está disponible para FTP, donde:
	• *prefix_identifier* es `network`, para las propiedades de requisitos previos generales
	• *property_name* es `availablePorts`
	• *suffix_identifier* are `DB` para puertos de base de datos disponibles, `WAS` para el puerto de WebSphere Application Server disponible y FTP para el puerto `FTP` disponible
	• *property_value* es `60000-60005` , `8080` o `21`

`os.dir.home=[dir:/home,type:permission]755+`	Comprueba si el directorio de inicio tiene permisos `drwxr-xr-x`, donde:
	• *prefix_identifier* es `os`, para las propiedades de requisitos previos de sistema operativo
	• *property_name* es `dir`
	• *suffix_identifier* es `home` para el directorio para comprobar
	• *qualifier_name* son `dir` y
	`type that qualify the prerequisite property and type of check`
	• *qualifier_value* son `home` y
	`permission, the values for the qualifiers`
	• *property_value* es `755+`, es decir, la representación de ocho dígitos de los permisos de acceso al directorio de inicio

Puede añadir o editar las propiedades de requisitos previos para cada producto para el que desee ejecutar Prerequisite Scanner. También puede crear propiedades

de requisitos previos personalizadas y utilizar recopiladores y evaluadores Prerequisite Scanner que sean necesarios para explorar y comparar las propiedades de requisitos previos.

Conceptos relacionados:

"Calificadores predefinidos de propiedades de requisitos previos" en la página 9
IBM Prerequisite Scanner proporciona un conjunto de calificadores básicos para algunas propiedades de requisitos previos de una categoría predefinida. Los calificadores representan atributos de la propiedad de requisito previo que Prerequisite Scanner utiliza para calificar la propiedad de requisito previo o el tipo de comprobación de que debe realizarse en esa propiedad.

Categorías predefinidas de propiedades de requisitos previos

IBM Prerequisite Scanner proporciona un conjunto de propiedades básicas de requisitos previos para las diferentes categorías de datos: común, software instalado, sistema operativo, usuario, conectividad, Internet Explorer, servidor de base de datos, variables de entorno y red, incluidas las propiedades específicas de plataforma para Windows y UNIX.

<*prefix_identifier*> es un identificador de una categoría predefinida de propiedades de requisitos previos.

Tabla 3 describe las categorías predefinidas de requisitos previos de software y hardware.

Tabla 3. Categorías de propiedades básicas de requisitos previos

Categoría de datos	**Descripción**	**Identificador de prefijo necesario**
Común	En esta categoría se comprueban requisitos previos comunes, como la velocidad del procesador, la memoria RAM, el disco y el espacio temporal. Este ejemplo muestra la propiedad de requisito previo para comprobar el sistema operativo: `OS Version=RedHat Enterprise Linux 5.4`	Ninguna
Software instalado	En esta categoría se comprueban los requisitos previos de software instalado, como los programas registrados en el registro de Windows y si cygwin y gskit están instalados. En este ejemplo se muestra la propiedad de requisito previo de la exploración del registro de sistema operativo para programas instalados con ubicaciones: `installedSoftware=`*`list_of_installed_programs`*	Ninguna
Usuario	En esta categoría se comprueban los requisitos previos de los usuarios; por ejemplo, si el usuario conectado tenía derechos de administrador o es el usuario root. En este ejemplo se muestra la propiedad de requisito previo para comprobar si el usuario conectado pertenece al grupo de administradores: `user.isAdmin=True`	`user`
Sistema operativo	En esta categoría se comprueban las propiedades del sistema operativo, como la versión, la arquitectura, la memoria total, la memoria disponible y la memoria física total. En este ejemplo se muestra si la propiedad de requisito previo para comprobar si el servicio de registro remoto se está ejecutando: `os.isServiceRunning.remoteRegistry=True`	`os`
Conectividad	En esta categoría se comprueban los requisitos previos de conectividad; por ejemplo si Telnet se está ejecutando y a qué direcciones IP y puertos se puede conectar Scanner.	Ninguna

Tabla 3. Categorías de propiedades básicas de requisitos previos (continuación)

Categoría de datos	Descripción	Identificador de prefijo necesario
Red	En esta categoría se comprueban los requisitos previos de red que pueden ser comunes a todas las plataformas; por ejemplo si hay puertos disponibles. En este ejemplo se muestra la propiedad de requisito previo para comprobar si el puerto `8080` está disponible para IBM WebSphere Application Server: `network.availablePorts.was=8080`	`network`
Red de Windows	En esta categoría se comprueban los requisitos previos de red de Windows; por ejemplo, si se han habilitado NetBIOS y DHCP en la máquina y las propiedades de hacer ping. En este ejemplo se muestra la propiedad de requisito previo para comprobar si al menos un adaptador con un dirección IP válida tiene habilitado NetBIOS como protocolo: `network.netBIOSEnabled=True`	`network`
Red de UNIX	En esta categoría se comprueban los requisitos previos de red de UNIX; por ejemplo, si se han habilitado NetBIOS y DHCP en la máquina y las propiedades de hacer ping. En este ejemplo se muestra la propiedad de requisito previo para comprobar si el host local responde al protocolo ping: `network.pingLocalhost=True`	`network`
Internet Explorer	En esta categoría se comprueban los requisitos previos de Microsoft Internet Explorer; por ejemplo, la versión. En este ejemplo se muestra la propiedad de requisito previo para comprobar si la versión de Internet Explorer es `7.0`: `internetExplorer.version=7.0`	`internetExplorer`
Servidor de bases de datos, DB2	En esta categoría se comprueban los requisitos previos de DB2; por ejemplo, la versión. En este ejemplo se muestra la propiedad de requisito previo para comprobar si la versión de DB2 es como mínimo `9.5`: `DB2 Version=9.5.*`	`DB2`
Servidor de base de datos, Oracle	En esta categoría se comprueban los requisitos previos de Oracle; por ejemplo, la versión. En este ejemplo se muestra la propiedad de requisito previo para comprobar si la versión de cliente Oracle es como mínimo `9.2.0.8`: `oracle.Client=9.2.0.8+`	`Oracle`
Variables de entorno	En esta categoría se comprueban los requisitos previos de las variables de entorno; por ejemplo si se ha establecido la variable de entorno. En este ejemplo se muestra la propiedad de requisito previo para comprobar si la ruta de clase contiene el archivo Derby JAR: `env.classpath.derbyJAR=False`	`env`
Autonomic Deployment Engine	En esta categoría se comprueban los requisitos previos de Autonomic Deployment Engine; por ejemplo, si se ha instalado Autonomic Deployment Engine o la unidad de instalación de Tivoli Integrated Portal. En este ejemplo se muestra la propiedad de requisito previo para comprobar si se ha instalado la unidad de instalación de Tivoli Integrated Portal Versión 2.1.1.0 o 2.1.1.1 en un sistema Windows: `de.installationUnit=regex{.*C37109911C8A11D98E1700061BDE7AEA.* |.*TIP 2.1.1.0.*|.*TIP 2.1.1.1.*}`	`de`
Servidor de base de datos, MS SQL	En esta categoría se comprueban los requisitos previos de MS SQL; por ejemplo, la versión. En este ejemplo se muestra la propiedad de requisito previo para comprobar si la versión de MS SQL Server es SQL Server 2008 R2 Developer Edition: `mssql.Server=10.50.1600.1`	`mssql`

Subtipos predefinidos de propiedades de requisitos previos

IBM Prerequisite Scanner proporciona un conjunto de subtipos básicos para algunas propiedades de requisitos previos de una categoría predefinida. Los subtipos categorizan aún más una propiedad de requisito previo, como la categorización por aplicación, programa de utilidad o subtipo de servicio.

Por ejemplo, puede tener una propiedad de requisito previo para los puertos de red disponibles. Puede categorizar aún más esa propiedad de requisito previo para comprobar los puertos disponibles de un servidor de bases de datos, servidor de aplicaciones o protocolo.

<*suffix_identifier*> es un identificador opcional de un subtipo del nombre de la propiedad de requisito previo.

Tabla 4 describe los subtipos predefinidos de las diferentes categorías de propiedades de requisitos previos, incluido el <*suffix_identifier*>.

Tabla 4. Subtipos predefinidos

Subtipo de propiedad de requisito previo	Identificador de sufijo	Plataforma	Descripción	Valores válidos del subtipo
Categoría de red independiente de plataforma				
`network.availablePorts.`*`app_type`*	*app_type*	Todas	Utilice esta convención de nomenclatura para comprobar si el puerto o rango de puertos no está escuchando o si está disponible para el tipo de aplicación de *app_type*.	Cadena para representar *app_type*; por ejemplo: • `DB2` comprueba los puertos del servidor de bases de datos DB2 • `WAS` comprueba los puertos de WebSphere Application Server • `ftp` comprueba el puerto FTP
`network.portsInUse.`*`app_type`*	*app_type*	Todas	Utilice este convenio de denominación para comprobar si el puerto o rango de puertos está escuchando o si está en uso para el tipo de aplicación *app_type*.	Cadena para representar *app_type*; por ejemplo: • `DB2` comprueba los puertos del servidor de bases de datos DB2 • `WAS` comprueba los puertos de WebSphere Application Server • `ftp` comprueba el puerto FTP
Categoría de sistema operativo				
`os.dir.`*`dir_name`*	*dir_name*	UNIX	Utilice este convenio de denominación para comprobar el sistema de archivos *dir_name*. El valor de la propiedad de requisito previo utiliza calificadores predefinidos.	Cadena para representar *dir_name*; por ejemplo: • `tmp` • `home`
`os.file.`*`script_name`*	*script_name*	UNIX	Utilice este convenio de denominación para comprobar si la secuencia de comandos *script_name* está disponible en la máquina.	Cadena para representar *script_name*; por ejemplo: • `bash` • `expect` • `gzip` • `tar`

Tabla 4. Subtipos predefinidos (continuación)

Subtipo de propiedad de requisito previo	Identificador de sufijo	Plataforma	Descripción	Valores válidos del subtipo
`os.isService Running.`*`service_name`*	*service_name*	Windows	Utilice este convenio de denominación para comprobar si el servicio *service_name* se está ejecutando en la máquina.	Cadena para representar *service_name*; por ejemplo: • `remoteRegistry` • `DNSClient` • `terminalServices`
`os.lib.`*`lib_name_version`*	*lib_name _version*	UNIX	Utilice este convenio de denominación para comprobar si la versión admitida de la biblioteca *lib_name_version* se encuentra instalada en la máquina.	Cadena para representar *lib_name_version*; por ejemplo, en negrita: • Biblioteca **`libstdc++.so.#`** de 32 bits • Biblioteca **`libstdc++.so.#`** de 64 bits • Biblioteca **`libXft.so.#`** de 32 bits • Biblioteca **`libXtst.so.#`** de 32 bits • Biblioteca **`libaio.so.#`** de 64 bits • Nivel de tiempo de ejecución XLC **`xlC.rte`** de 32 bits • Tiempo de ejecución XLC **`xlC.aix50.rte`** de 32 bits para AIX Versión 5.3 • Tiempo de ejecución XLC **`xlC.aix61.rte`** de 32 bits para AIX Versión 6.1 • Biblioteca AIX IOCP **`bos.iocp.rte`** • **`bos.loc.iso.en_us`**, el conjunto de archivos con código ISO para el sistema operativo base AIX `regex {`*`str`*`}`, expresión regular con el parámetro de entrada, *str*, que representa el patrón de búsqueda del nombre de la biblioteca; por ejemplo: `regex {.*libgcc.*}` Comprueba si existe una versión de la biblioteca en tiempo de ejecución del CCG de bajo nivel, `libgcc`, para ese sistema operativo.

Tabla 4. Subtipos predefinidos (continuación)

Subtipo de propiedad de requisito previo	Identificador de sufijo	Plataforma	Descripción	Valores válidos del subtipo
`os.package.` *`package_name`*	*package_name*	UNIX	Utilice este convenio de denominación para comprobar si la versión admitida del paquete *package_name* se encuentra instalada en la máquina.	Cadena para representar *package_name*; por ejemplo, en negrita: • **bash**shell • **expect** para el paquete de extensión de TCL • **libgcc** para el paquete GCC tiempo de ejecución de bajo nivel • **openssh** para el Secure Shell de código abierto • **openssl** para el conjunto de herramientas de código abierto para SSL/TLS • **perl** para el paquete de extensión de Perl • **rpm** para los paquetes RPM o RPM Build • **telnet** para el paquete Telnet • **wget**para el paquete de recuperación de archivos GNU
`os.space.` *`dir_name`*	*dir_name*	UNIX	Utilice este convenio de denominación para comprobar el espacio en disco disponible para el sistema de archivos *dir_name*. El valor de la propiedad de requisito previo utiliza calificadores predefinidos.	Cadena para representar *dir_name*; por ejemplo: • `usr` • `home` • `tmp` • `var`

Calificadores predefinidos de propiedades de requisitos previos

IBM Prerequisite Scanner proporciona un conjunto de calificadores básicos para algunas propiedades de requisitos previos de una categoría predefinida. Los calificadores representan atributos de la propiedad de requisito previo que Prerequisite Scanner utiliza para calificar la propiedad de requisito previo o el tipo de comprobación de que debe realizarse en esa propiedad.

Por ejemplo, puede tener una propiedad de requisito previo para un sistema de archivos. Puede calificar qué comprobación se realizará para esa propiedades requisito previo en función de su nombre de sistema de archivos y atributos de permisos de acceso. También puede calificar qué tipo de unidades se utilizarán para comprobar el espacio en disco disponible en función de la ruta del sistema de archivos y los atributos de unidad.

Los calificadores admiten la personalización para satisfacer las necesidades de su entorno y evitar que Scanner tenga que hacer suposiciones implícitas sobre los atributos de los requisitos previos multidimensionales, como la ruta

predeterminada y los permisos de acceso. Puede cambiar los valores de los calificadores predefinidos, pero no podrá agregar calificadores nuevos al conjunto existente de calificadores predefinidos para una propiedad predefinida de requisito previo.

Los calificadores deben tener el siguiente formato:

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

donde:

- *qualifier_name* es un atributo opcional de la propiedad de requisito previo que IBM Prerequisite Scanner utiliza para calificar la propiedad de requisito previo o el tipo de comprobación de que debe realizarse en esa propiedad.
- *qualifier_value* es el valor del atributo opcional.

  El valor del calificador también puede ser un par de nombre y valor para admitir varios valores válidos en función del tipo de usuario. Por ejemplo, diferentes rutas del directorio de inicio, en función de si se trata de un usuario root o no.
- *property_value* es el valor de la propiedad de requisito previo y puede ser una cadena o un entero.

Cada calificador y su valor deben estar delimitados por dos puntos `:`.Puede tener múltiples calificadores, cada uno de ellos separado por una coma. El conjunto de calificadores debe estar entre corchetes `[]`.

Tabla 5 describe los calificadores predefinidos de diferentes categorías de propiedades de requisitos previos. Algunas propiedades de requisitos previos también utilizan subtipos predefinidos para categorizar aún más una propiedad de requisito previo.

Importante: No puede utilizar los calificadores predefinidos con otras propiedades de requisitos previos.

Tabla 5. Calificadores predefinidos

Propiedad de requisito previo	**Platafor.**	**Descripción**	**Calificadores y valores válidos**
Categoría de sistema operativo con subtipo predefinido			

`os.dir.dir_name`	UNIX	Comprueba el sistema de archivos *dir_name* en función de los siguientes atributo de cualificación:
		• Atributo `dir`, para determinar qué sistema de archivos se comprobará
		• Atributo `type`, para determinar qué atributo del sistema de archivos se comprobará; por ejemplo, la representación de ocho dígitos de `<octal_digits>` para los permisos de acceso a ese sistema de archivos
		<*dir_name*> puede representar por ejemplo:
		• `tmp`
		• `home`
			Cadena con el siguiente formato calificador:
			`[dir:dir_name, type:permission] octal_digits+`
			
			Por ejemplo, para comprobar si el directorio de inicio tiene permisos `drwxr-xr-x`:
			`os.dir.home=[dir:/home, type:permission]755+`

Tabla 5. Calificadores predefinidos (continuación)

Propiedad de requisito previo	Platafor.	Descripción	Calificadores y valores válidos

`os.space.`
`dir_name`
	UNIX	Comprueba el espacio en disco disponible para el sistema de archivos *dir_name* especificado, en función de uno o varios de los siguientes atributos de cualificación:
		• Atributo `dir`, para determinar qué ruta al sistema de archivos se comprobará
		• Atributo `unit`, para determinar qué unidades de espacio de disco se utilizarán
		El valor del atributo `dir` depende del usuario conectado; de este modo, el valor es un par de valor-nombre para representar el tipo de usuario, esto es, root o no root, y la ruta asociada.
		dir_name puede representar por ejemplo:
		• `usr`
		• `home`
		• `tmp`
		• `var`
			Cadena con el siguiente formato calificador para el sistema de archivos de un usuario root:
			`[dir:`**`root`**`=`*`dir_path`*`,`
			`unit:`*`unit_name`*`]`
			`disk_space`
			Por ejemplo:
			`os.space.usr=`
			`[dir:root=/usr/ibm/common/acsi,`
			`unit:GB]200`
			Cadena con el siguiente formato calificador para el sistema de archivos de un usuario no root:
			`[dir:`**`non_root`**`=`*`dir_path`*`,`
			`unit:`*`unit_name`*`]`
			`disk_space`
			Por ejemplo:
			`os.space.home=`
			`[dir:non_root=USERHOME/.acsi_HOST,`
			`unit:MB]200`
			Cadena con el siguiente formato calificador, que usa solo un calificador:
			`[dir:`*`dir_path`*`]`
			`disk_space` `MB`
			Por ejemplo:
			`os.space.home=[dir:/home/sat]250MB`

Categoría de sistema operativo sin subtipo predefinido			

`os.mountcheck`	UNIX	Comprueba si el sistema de archivos se monta en función de los siguientes atributos de cualificación:
		• Atributo `drive`, para determinar qué directorio es el sistema de archivos montado
		• Atributo `nosuid`, para determinar si la opción de montaje se establece si se monta el sistema de archivos
			Cadena con el siguiente formato calificador:
			`[drive:`*`dir_name`*`,`
			`mount_option``:`
			`false|true]`
			`True|False`
			Por ejemplo, para comprobar si el directorio `/home` está montado y no se ha establecido la opción `nosuid`:
			`os.mountcheck=[drive:/home,`
			`nosuid:false]True`

Tabla 5. Calificadores predefinidos (continuación)

Propiedad de requisito previo	Platafor.	Descripción	Calificadores y valores válidos

`os.SELinux`	Linux	Comprueba el estado de obligatoriedad de la característica de mejora de seguridad de Linux en función de los siguientes atributos de cualificación:
		• Atributo `source`, para determinar el comando que se utilizará para el sistema operativo correspondiente
			• Cadena con el siguiente formato calificador:
			`[source:Command] Disabled|Enabled`
			Por ejemplo, para comprobar si la función está desactivada o tiene un estado permisivo en el sistema operativo Red Hat o SUSE:
			`os.SELinux=[source: Command]Disabled`
			• Cadena sin un calificador, donde el sistema operativo es una variante genérica de Linux:
			`os.SELinux=Disabled`

`os.ulimit`	UNIX	Utilice este convenio de denominación para comprobar si se puede ejecutar un número ilimitado de procesos en función de los siguientes atributos de calificación:
		• Atributo `type`, para determinar el límite adicional que se comprobará; por ejemplo `filedescriptorlimit` comprueba el límite del número de descriptores de archivo que los procesos pueden abrir
			Cadena con el siguiente formato calificador:
			`[type:`*`limit_name`*`]`*`limit_value`*`, limited|unlimited`
			Por ejemplo, para comprobar si el límite del descriptor de fichero es mayor que 8192, con un número ilimitado de procesos:
			`os.ulimit= [type:filedescriptorlimit] 8192+,unlimited`
			Los tipos válidos de límites para comprobar, donde *limit_name* representa el tipo de límite, son los siguientes:
			• `ALL`, comprueba todos los límites
			• `corefilesizelimit`
			• `datasegmentlimit`
			• `filedescriptorlimit`
			• `filesizelimit`
			• `hardlimit`
			• `processlimit`
			• `maxmemorysizelimit`
			• `maxprocesseslimit`
			• `stacksizelimit`
			• `threadlimit`

Categoría común sin subtipo predefinido			

Tabla 5. Calificadores predefinidos (continuación)

Propiedad de requisito previo	Platafor.	Descripción	Calificadores y valores válidos

`Disk`	Windows	La cantidad de espacio en disco libre, con los siguientes atributos de calificación opcionales:
		• Atributo `dir`, para determinar qué ruta al directorio se comprobará
		• Atributo `unit`, para determinar qué unidades de espacio de disco se utilizarán
			Cadena con el siguiente formato calificador:
			`[dir:`*`dir_path`*`, unit:`*`unit_name`*`]`
			`disk_space`
			Por ejemplo:
			`Disk= [dir:C:\Program Files\IBM\SQLLIB, unit:MB]1431`
			Formato numérico en MB o GB:
			`<disk_space>MB|GB`
			Por ejemplo:
			`Disk=250MB`

Códigos de producto

IBM Prerequisite Scanner utiliza los códigos de varios caracteres en nombres de archivos y parámetros para identificar los productos y componentes y determinar qué tipo de archivo de configuración se utilizará.

product_code

Es la variable para representar un código de producto en sistemas Windows o UNIX. Los códigos de producto identifican el producto, una plataforma individual, como Windows, AIX, HP-UX, Linux y Solaris, y opcionalmente la versión del sistema operativo compatible con ese producto. Se guardan en el archivo `codename.cfg`. Los productos que admiten varias plataformas tienen varios códigos de producto. Cada uno de ellos identifica un producto, una plataforma y una versión del sistema operativo, según se requiera.

Por ejemplo, los códigos de producto de `COD` , `COK` y `COX` identifican algunos de los sistemas operativos y versiones de IBM Tivoli Provisioning Manager :

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

Cuando se ejecuta Prerequisite Scanner, se pasa el código de producto y, opcionalmente, la versión del producto como parámetros de entrada. Scanner comprueba si existe el código de producto en el archivo `codename.cfg`. En sistemas UNIX, si no se encuentra el código, Scanner se cierra. En sistemas Windows, si no se encuentra el código, no se cierra.

Scanner utiliza los parámetros de entrada para buscar el archivo de configuración en el directorio *ips_root*`/Windows|UNIX_Linux`. El nombre del archivo contiene el mismo código y versión del producto que los parámetros de entrada. Si no pasa el parámetro de versión del producto opcional, Scanner utiliza la última versión del archivo de configuración que encuentra en este directorio. Prerequisite Scanner comienza la exploración.

Nota: Sólo en sistemas Windows: si el código de producto no existe en el archivo `codename.cfg`, pero sí existe un archivo de configuración con el

código de producto en el nombre, Prerequisite Scanner muestra el código de producto y el número de versión en la salida e indica que el nombre de producto no se ha definido.

Archivos de configuración de Prerequisite Scanner

Los archivos de configuración de IBM Prerequisite Scanner para plataformas individuales contienen las propiedades de requisitos previos y sus valores esperados para cada plataforma que es compatible con el producto. Prerequisite Scanner ofrece un conjunto predefinido de archivos de configuración que se pueden editar. Debe crear archivos de configuración para que los nuevos productos y plataformas sean compatibles.

Los archivos tienen la extensión `.cfg`. Almacénelos en el directorio *ips_root*`/<OS>`, donde <*OS*> es el nombre del tipo de sistema operativo; por ejemplo, `Windows` o `UNIX_Linux`.

Los archivos de configuración deben cumplir las siguientes reglas:

- La extensión del archivo debe ser `.cfg`
- Convenio de denominación para el nombre del archivo:

  product_code`[_<`*version*`>].cfg`

  donde:

  – *product_code*

    Es la variable para representar un código de producto en sistemas Windows o UNIX. Los códigos de producto identifican el producto, una plataforma individual, como Windows, AIX, HP-UX, Linux y Solaris, y opcionalmente la versión del sistema operativo compatible con ese producto. Se guardan en el archivo `codename.cfg`. Los productos que admiten varias plataformas tienen varios códigos de producto. Cada uno de ellos identifica un producto, una plataforma y una versión del sistema operativo, según se requiera.

  – <*version*> es el código de 8 dígitos para representar la versión, release, modificación y nivel con dos dígitos para cada parte del código; por ejemplo, 7.3.21 es 07032100.
- Agrupe propiedades de requisitos previos en secciones que deben seguir un convenio de denominación para los títulos de sección.
- El formato estándar para cada propiedad de requisito previo es un par de nombre-valor con calificadores opciones y solo una propiedad en cada línea:

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

Ejemplo de un archivo de configuración sin secciones

En este ejemplo se comprueban propiedades de requisitos previos, pero no se diferencia entre distintas propiedades de requisitos previos para las versiones requeridas del sistema operativo.

```
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
```

Conceptos relacionados:

"Secciones de archivos de configuración"
Las propiedades de requisitos previos se pueden agrupar en un conjunto de secciones de los archivos de configuración, donde cada sección representa una categoría de tipo de datos. Las secciones son opcionales en los archivos de configuración.

Secciones de archivos de configuración

Las propiedades de requisitos previos se pueden agrupar en un conjunto de secciones de los archivos de configuración, donde cada sección representa una categoría de tipo de datos. Las secciones son opcionales en los archivos de configuración.

El convenio de denominación del título de sección es:

[*category_name*:*category_value*]

donde:

- *category_name* es el código con varios caracteres que representa la categoría del tipo de datos
- *category_value* es el código con varios caracteres que representa un valor permitido para la categoría

  Nota: Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.

Cada nombre de categoría y su valor debe estar delimitados por

```
: colon and enclosed by [] square brackets.
```

Puede tener varias categorías de tipos de datos mediante la combinación de títulos de secciones, lo que restringe las propiedades de requisitos previos solo para ese conjunto de categorías específicas.

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

Por ejemplo, para especificar propiedades de requisitos previos que se aplican a una máquina que ejecuta un sistema operativo Itanium SUSE Linux Enterprise Server versión 11 de 32 bits:

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

Para todas las plataformas, puede utilizar el símbolo OR lógico | para utilizar cualquiera de las dos categorías de tipo de datos. Por ejemplo, para establecer cualquiera de las variables de entorno en `True`, la combinación de títulos de sección sería:

- **Sistemas UNIX**

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

- **Sistemas Windows**

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

Importante: La posición del símbolo OR lógico | es diferente en sistemas Windows y UNIX. En sistemas UNIX, el conjunto de títulos de sección están incluidos en un

conjunto de corchetes [] y cada una de las secciones está separada por el símbolo. En sistemas Windows, el símbolo delimita cada título de sección completo con corchetes [] asociados.

Solo en sistemas Windows, puede utilizar el símbolo NOT lógico ! para excluir una categoría de tipo de datos. Por ejemplo, para excluir la variante Windows Server 2003 R2, la combinación de títulos de sección es: `[OSType:Windows Server 2003][!OSType:Windows Server 2003 R2]`

Tabla 6 en la página 17 describe las categorías de datos compatibles y los valores permitidos asociados.

Tabla 6. Categorías y valores de tipo de datos compatibles

Categoría de tipo de datos	Descripción	Valores permitidos

`OSType`	Tipo de sistema operativo	• UNIX
		Indica que todas las propiedades de esta categoría son comunes a todas las plataformas UNIX, incluidos AIX, HP-UX, Linux y Solaris; por ejemplo:
		`[OSType:UNIX]`
		• AIX
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo AIX; por ejemplo:
		`[OSType:AIX]`
		• HP-UX
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo HP-UX; por ejemplo:
		`[OSType:HP-UX]`
		• LINUX
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Linux; por ejemplo:
		`[OSType:LINUX]`
		• RedHat
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Linux; por ejemplo:
		`[OSType:RedHat]`
		• RedHatEnterpriseLinuxServer
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo RedHat Enterprise Linux Server; por ejemplo:
		`[OSType:RedHatEnterpriseLinuxServer]`
		• SUSE
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo SUSE Linux; por ejemplo:
		`[OSType:SUSE]`
		• SUSELinuxEnterpriseServer
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo SUSE Linux Enterprise Server;por ejemplo:
		`[OSType:SUSELinuxEnterpriseServer]`
		• Solaris
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Solaris; por ejemplo:
		`[OSType:Solaris]`

Tabla 6. Categorías y valores de tipo de datos compatibles (continuación)

Categoría de tipo de datos	Descripción	Valores permitidos

		• Windows
		Indica que todas las propiedades de esta categoría son comunes a todos los sistemas operativos Windows; por ejemplo:
		`[OSType:Windows]`
		• Windows 2000 Workstation (Versión 5.0.*)
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Windows 2000; por ejemplo:
		`[OSType:Windows 2000]`
		• Windows XP Workstation (Versión 5.1.*)
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Windows XP Professional de 32 bits; por ejemplo:
		`[OSType:Windows XP]`
		• Windows XP Workstation (Versión 5.2.*)
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Windows XP Professional de 64 bits; por ejemplo:
		`[OSType:Windows XP]`
		• Windows Vista Workstation (Versión 6.0.*)
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Windows Vista; por ejemplo:
		`[OSType:Windows Vista]`
		• Windows 7 Workstation (Versión 6.1.*)
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Windows 7; por ejemplo:
		`[OSType:Windows 7]`
		• Windows 2000 Server (Versión 5.0.*)
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Windows 2000 Server; por ejemplo:
		`[OSType:Windows 2000]`
		• Windows Server 2003 (Versión 5.2.*)
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Windows Server 2003; por ejemplo:
		`[OSType:Windows Server 2003]`
		• Windows Server 2003 R2 (Versión 5.2.* y otro tipo de descripción del sistema operativo es R2)
		Indica que todas las propiedades de esta categoría son comunes solo a la variante del sistema operativo Windows Server 2003 R2; por ejemplo:
		`[OSType:Windows Server 2003 R2]`

Tabla 6. Categorías y valores de tipo de datos compatibles (continuación)

Categoría de tipo de datos	Descripción	Valores permitidos

		• Windows Server 2008 (Versión 6.0.*)
		Indica que todas las propiedades de esta categoría son comunes a todas las variantes del sistema operativo Windows Server 2008; por ejemplo:
		`[OSType:Windows Server 2008]`
		• Windows Server 2008 R2 (Versión 6.1.*)
		Indica que todas las propiedades de esta categoría son comunes solo a la variante del sistema operativo Windows Server 2008 R2; por ejemplo:
		`[OSType:Windows Server 2008 R2]`
		• *<OS_Name_Version>*
		Indica que todas las propiedades de esta categoría son comunes a esa versión del sistema operativo; por ejemplo:
		`[OSType:RedHatEnterpriseLinuxServer4.2]`
		Nota: Se permite el carácter comodín especial, `*`, para especificar múltiples versiones.

`OSArch`	La arquitectura del sistema operativo	• 32 bits; por ejemplo:
		`[OSArch:32-bit]`
		• 64 bits; por ejemplo:
		`[OSArch:64-bit]`

`CPU`	Nombre de familia de procesadores genéricos	Itanium, por ejemplo: `[CPU:Itanium]`
`CPUArch`	Arquitectura del procesador	Arquitectura de procesadores PowerPC de 64 bits y Power Architecture, es decir: • ppc4 • POWER4 • POWER5 • POWER6 • POWER7 Por ejemplo: `[CPUArch:ppc4]`
`@<EnvVar_Name>`	Variable de entorno de un producto	Cumple las reglas de ese producto; por ejemplo: `[@TPAE_DB_SERVER:True]`

Ejemplo de archivo de configuración de Windows que utiliza secciones

En este ejemplo se utilizan secciones para categorizar propiedades de requisitos previos para cualquier máquina Windows y, a continuación, máquinas que ejecutan versiones específicas de Windows.

```
#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+
```

Cuando se ejecuta Prerequisite Scanner, explora y comprueba diferentes propiedades de requisitos previos en función del sistema operativo y la versión que está instalada en la máquina.

Por ejemplo, Tabla 7 describe las diferentes secciones que contienen las propiedades de requisitos previos que se comprueban según el ejemplo.

Tabla 7. Secciones de un archivo de configuración analizadas para Windows

Plataforma o sistema operativo	Secciones con requisitos previos
Máquina con Windows XP y superior	[OSType:Windows]
Máquina solo con Windows Vista	[OSType:Windows] [OSType:Windows Vista]

Ejemplo de archivo de configuración de UNIX que utiliza secciones

Este ejemplo contiene propiedades de requisitos previos para todas las plataformas, plataformas individuales y versiones de sistemas operativos de un determinado producto.

```
# 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+
```

Cuando se ejecuta Prerequisite Scanner, explora y comprueba diferentes propiedades de requisitos previos en función del sistema operativo y la versión que está instalada en la máquina.

Por ejemplo, Tabla 7 en la página 20 describe las diferentes secciones que contienen las propiedades de requisitos previos que se comprueban según el ejemplo.

Tabla 8. Secciones de un archivo de configuración analizadas para UNIX

Sistemas operativos y versiones	**Secciones con requisitos previos**
Máquina con SUSE Linux Enterprise Server 11 de 64 bits	[OSType:UNIX] [OSType:LINUX] [OSType:LINUX][CPUArch:ppc64] [OSType:SUSE Linux Enterprise Server 11] [OSArch:64-bit]
Máquina con Red Hat Enterprise Linux Server 6.3	[OSType:UNIX] [OSType:LINUX] [OSType:RedHat] [OSType:RedHatEnterpriseLinuxServer] [OSType:RedHatEnterpriseLinuxServer6.*]

Tabla 8. Secciones de un archivo de configuración analizadas para UNIX (continuación)

Sistemas operativos y versiones	Secciones con requisitos previos

Máquina con SUSE Linux Enterprise Server 11 y la variable de entorno `@TPAE_DB_Server` establecida en `true`	`[OSType:UNIX]`
	`[OSType:LINUX]`
	`[OSType:SUSELinuxEnterpriseServer11][@TPAE_DB_Server:True]`

Máquina con AIX 5.3.0.0 y las variables de entorno `@TPAE_DB_FEATURE` o `@TPAE_DIR_FEATURE` establecidas en `True`	`[OSType:UNIX]`
	`[OSType:AIX]`
	`[OSType:AIX5.3.0.0][@TPAE_DB_FEATURE:True|@TPAE_DIR_FEATURE:True]`

Recopiladores de Prerequisite Scanner

Los recopiladores de IBM Prerequisite Scanner recopilan los datos reales sobre el entorno actual en función de las propiedades de requisito previo establecidas para los productos que se van a instalar. Los recopiladores obtienen los datos a través de código nativo. Los datos pueden ser datos comunes, como la velocidad del procesador y la RAM, los datos de software instalados, datos del sistema operativo, datos del usuario, de red y de conectividad. Los recopiladores pueden también ampliarse para que pueda crear recopiladores personalizados para obtener valores reales de propiedades de requisito previo.

Prerequisite Scanner utiliza recopiladores en los siguientes idiomas, dependiendo de su plataforma:

- Windows: VBScript con extensión `.vbs`
- UNIX: Shell con `.sh` o sin extensión

Nota: No podrá ejecutar las secuencias de comandos de UNIX en sistemas Windows aunque haya instalado entornos similares a UNIX en las máquinas Windows; por ejemplo, Cygwin.

Recopiladores de sistemas Windows

Los recopiladores VBScript para los sistemas Windows se ejecutan en el entorno Script Host de Windows. Usan el Modelo de objetos componentes para acceder a los elementos del entorno Windows; por ejemplo, `FileSystemObject` y `TextStream`.

Prerequisite Scanner ejecuta los recopiladores VBScript para obtener los valores reales de las propiedades de requisitos previos del entorno Windows. Cada recopilador puede obtener datos para una o muchas propiedades de requisitos previos.

Para cada propiedad de requisito previo en un recopilador VBScript, el recopilador escribe el nombre de la propiedad de requisito previo y su valor real como salida estándar. Prerequisite Scanner graba esta salida estándar en un archivo de texto temporal, es decir, `localhost_hw.txt`.

Puede crear recopiladores VBScript comunes personalizados para recopilar datos de propiedades de requisitos previos aplicables a cualquier producto o versión del producto. También puede crear sus recopiladores personalizados específicos de producto para recopilar los datos aplicables a un producto y versión del producto específicos.

Cuando se ejecuta Prerequisite Scanner, ejecuta los recopiladores en el siguiente orden: recopiladores VBScript predefinidos; los recopiladores VBScript comunes personalizados en el directorio *ips_root*`/lib`; y los recopiladores VBScript específicos

de producto personalizados mediante la búsqueda del archivo *product_code*[_<*version*>].vbs en el directorio *ips_root*/Windows.

Por ejemplo, el archivo env.tcrhome.vbs es un recopilador personalizado que comprueba el entorno del directorio de inicio variable de Tivoli Common Reporting. Se almacena en el directorio *ips_root*/lib.

Los recopiladores VBScript deben cumplir las siguientes reglas:

- Convenio de denominación para el archivo del recopilador VBScript común personalizado

  Contiene una propiedad de requisito previo para poner a disposición de cualquier producto y versión del producto, es decir, todos los archivos de configuración:

  prefix_identifier.]property_name.vbs

  donde:

  – *prefix_identifier* es el identificador de prefijo de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4. Algunas de las categorías predefinidas requieren este identificador de prefijo; por ejemplo, env.
  – *property_name* es el nombre de la propiedad de requisito previo; por ejemplo, tcrhome.

  Guarde este tipo de recopilador VBScript en el directorio *ips_root*/lib.
- Convenio de denominación para el archivo del recopilador VBScript específico de producto personalizado

  Contiene propiedades para poner a disposición de un producto específico y versiones del producto, es decir, un fichero de configuración:

  product_code[_<*version*>].vbs

  donde:

  – *product_code*

    Es la variable para representar un código de producto en sistemas Windows o UNIX. Los códigos de producto identifican el producto, una plataforma individual, como Windows, AIX, HP-UX, Linux y Solaris, y opcionalmente la versión del sistema operativo compatible con ese producto. Se guardan en el archivo codename.cfg. Los productos que admiten varias plataformas tienen varios códigos de producto. Cada uno de ellos identifica un producto, una plataforma y una versión del sistema operativo, según se requiera.
  – <*version*> es el código de 8 dígitos para representar la versión, release, modificación y nivel con dos dígitos para cada parte del código; por ejemplo, 7.3.21 es 07032100.

  Guarde este tipo de recopilador VBScript en el directorio *ips_root*/Windows.
- La salida estándar de cada propiedad de requisito previo es la siguiente:

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

  – *property_name* que representa la propiedad de requisito previo como está grabada en el archivo de configuración; por ejemplo,

    env.tcrhome.
  – *var_for_value*, es decir, la variable VBScript del valor real que el recopilador obtiene para la propiedad de requisito previo.

  Por ejemplo, la siguiente salida estándar graba la propiedad de requisito previo de la variable de entorno de inicio de Tivoli Common Reporting y su valor real:

```
WScript.Echo "env.tcrhome=" & tcr_home
```

Recopiladores de sistemas UNIX

Los recopiladores de sistemas UNIX se ejecutan en el entorno de sistema principal Shell pertinente para AIX, HP-UX, Linux o Solaris. Usan los comandos y las opciones específicas de esa plataforma para acceder a elementos del entorno de sistema principal.

Cada recopilador de UNIX obtiene datos de una propiedad de requisito previo o una propiedad de requisito previo con subtipos predefinidos. El recopilador escribe el resultado de la comprobación de la propiedad de requisito previo como salida estándar.Prerequisite Scanner escribe esta salida estándar en un archivo de texto temporal.

Puede crear recopiladores UNIX personalizados para recopilar datos de propiedades de requisitos previos. Cada recopilador, predefinido o personalizado, se llama en el archivo *ips_root*/UNIX_Linux/packageTest.sh.

Cuando se ejecuta Prerequisite Scanner, ejecuta los recopiladores en el siguiente orden: recopiladores predefinidos con _plug en el nombre de archivo del directorio *ips_root*/lib; recopiladores predefinidos en el directorio *ips_root*/UNIX_Linux; y los recopiladores UNIX personalizados en el directorio *ips_root*/UNIX_Linux.

Por ejemplo, el archivo installedSoftware.TCR.version es un recopilador personalizado que obtiene la versión de Tivoli Common Reporting que está instalada en la máquina. Se almacena en el directorio *ips_root*/UNIX_Linux.

Los recopiladores de UNIX deben cumplir las siguientes reglas:

- Convenio de nomenclatura para el archivo del recopilador de UNIX personalizado sin ninguna extensión de archivo:

  [*prefix_identifier*.]*property_name*

  donde:

  – *prefix_identifier* es un identificador de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4. Algunas de las categorías predefinidas requieren este identificador de prefijo; por ejemplo, installedSoftware.

  – *property_name* es el nombre de la propiedad de requisito previo; por ejemplo, TCR.version.

  Almacene el recopilador en el directorio *ips_root*/UNIX_Linux. Asegúrese de que no tiene una extensión de archivo.

- Salida estándar de una propiedad de requisito previo que devuelve el valor real de la propiedad de requisito previo si es un entero o una cadena; por ejemplo, la versión de software o la cantidad de espacio en disco disponible para un sistema de archivos montado. También, puede devolver "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
  ```

- Código para llamar y ejecutar el recopilador en el script *ips_root*/UNIX_Linux/packageTest.sh.

```
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
```

Evaluadores de Prerequisite Scanner

Los evaluadores de IBM Prerequisite Scanner son secuencias de comandos que comparan los datos reales de los recopiladores y los datos esperados de las mismas propiedades que están en los archivos de configuración. Las evaluaciones pueden ser: específicas de plataforma, basadas en operadores simples; como menor que, igual que, o mayor que; y basadas en el hecho de si una propiedad se ha instalado, está presente o se ha habilitado. También pueden verificar si los puertos están en uso o disponibles y el estado de conectividad de la máquina.Puede crear evaluadores o editarlos.

Prerequisite Scanner utiliza evaluadores en los siguientes idiomas, dependiendo de su plataforma:

- Windows: VBScript con la extensión .vbs
- UNIX: shell con la extensión .sh

Nota: No podrá ejecutar las secuencias de comandos de UNIX en sistemas Windows aunque haya instalado un entorno similar a UNIX en las máquinas Windows; por ejemplo, Cygwin.

Almacene los evaluadores en *ips_root*/*OS*, donde *OS* es el nombre del sistema operativo; por ejemplo, Windows o UNIX_Linux.

Los archivos de evaluador deben cumplir las siguientes reglas:

- Convenio de denominación para el nombre del archivo:
  [*prefix_identifier*.]*property_name*[.*suffix_identifier*]_compare.vbs|sh

  donde:
  - *prefix_identifier* es un identificador de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4. Algunas de las categorías predefinidas requieren este identificador de prefijo.
  - *property_name* es el nombre de la propiedad de requisito previo.
  - *suffix_identifier* es un identificador opcional de un subtipo de las propiedades de requisitos previos como se indica en la Tabla 4 en la página 7.
- Opcionalmente, pase dos parámetros de entrada a la secuencia de comandos para evaluaciones complejas:
  - *expected_value*, es decir, el valor esperado de la propiedad de requisito previo que se ha establecido en el archivo de configuración.
  - *actual_value*, es decir, el valor real que el recopilador detecta para la propiedad de requisito previo en la máquina.

- La salida estándar es la siguiente:
  - `"PASS"` cuando el valor esperado de la propiedad de requisito previo es igual o mayor que el valor real de la propiedad de requisito previo.
  - `"FAIL"` cuando el valor esperado de la propiedad de requisito previo no es igual que el valor real de la propiedad de requisito previo.

Formatos de salida

IBM Prerequisite Scanner genera una salida para la siguiente pantalla y formatos de archivo legibles para los humanos: salida en la interfaz de línea de comandos, archivos de registro de depuración y seguimiento y archivos XML y de texto para los resultados.

Prerequisite Scanner guarda los resultados de la exploración y los archivos de registro en el directorio *ips_output_dir*. Puede establecer este directorio con el parámetro de entrada **`outputDir`** cuando ejecute Scanner. Si no establece este parámetro, la ubicación de salida predeterminada es *ips_root* .

Nota: Prerequisite Scanner crea archivos temporales durante su ejecución, pero estos archivos se eliminan antes de que Scanner termine su ejecución. Estos archivos temporales se encuentran ubicados en el subdirectorio *ips_output_dir*`/temp`. Scanner también suprime el subdirectorio *ips_output_dir*`/temp`, a menos que el subdirectorio contenga los archivos de depuración y de rastreo que se han generado únicamente en sistemas UNIX.

Puede también utilizar el parámetro para especificar una ubicación si opta por ejecutar Prerequisite Scanner desde un CD, un DVD o una unidad de red de solo lectura.

Importante: Si el directorio de salida no existe, Prerequisite Scanner lo creará. Debe tener permisos de escritura para crear o grabar en el directorio de salida en el que Prerequisite Scanner guarda los archivos.

Salida de interfaz de línea de comandos

Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se establece el parámetro **`detail`**, Prerequisite Scanner muestra resultados detallados del análisis en la interfaz de línea de comandos. Los resultados detallados contienen:

- La versión de Prerequisite Scanner
- La versión del sistema operativo en el que se ejecutó Scanner
- El tipo de exploración y escenario
  - Exploración de requisito previo: `Escenario: exploración de requisito previo`
- El nombre de los productos o componentes para los que se ejecutaron las comprobaciones de los requisitos previos
- Para cada propiedad de requisito previo: el nombre de la propiedad comprobada, el resultado PASS o FAIL el valor real y el valor esperado
- Para todos los componentes: el nombre de la propiedad general, el resultado PASS o FAIL, el valor real y el valor esperado
- El resultado PASS o FAIL general, con cualquier anomalía de una comprobación individual que produzca como resultado un fallo del análisis general

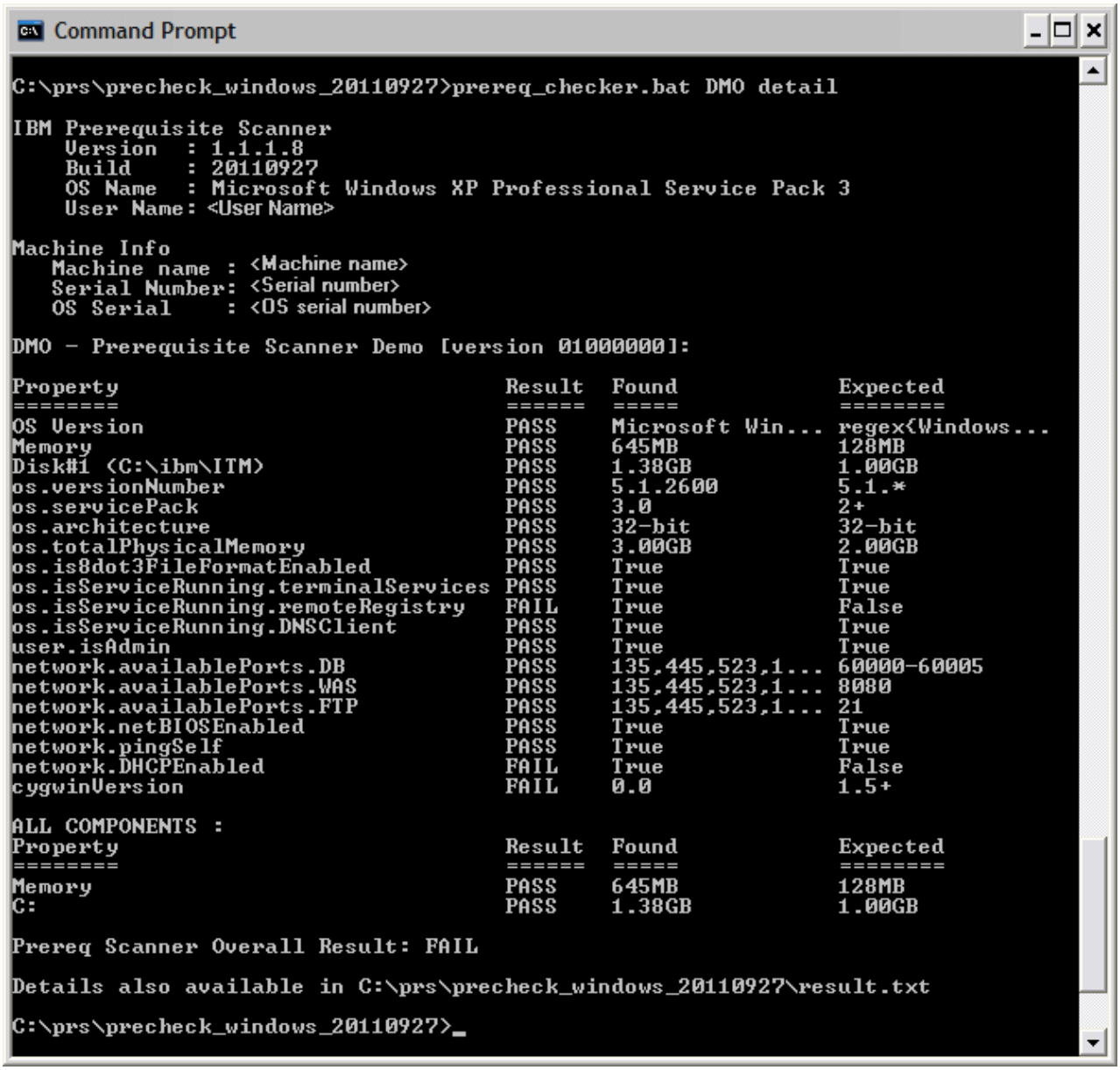


Figura 1. Salida en la interfaz de línea de comandos en sistemas Windows

Si no se establece el parámetro **detail**, Scanner muestra solo el resultado PASS o FAIL en la pantalla.

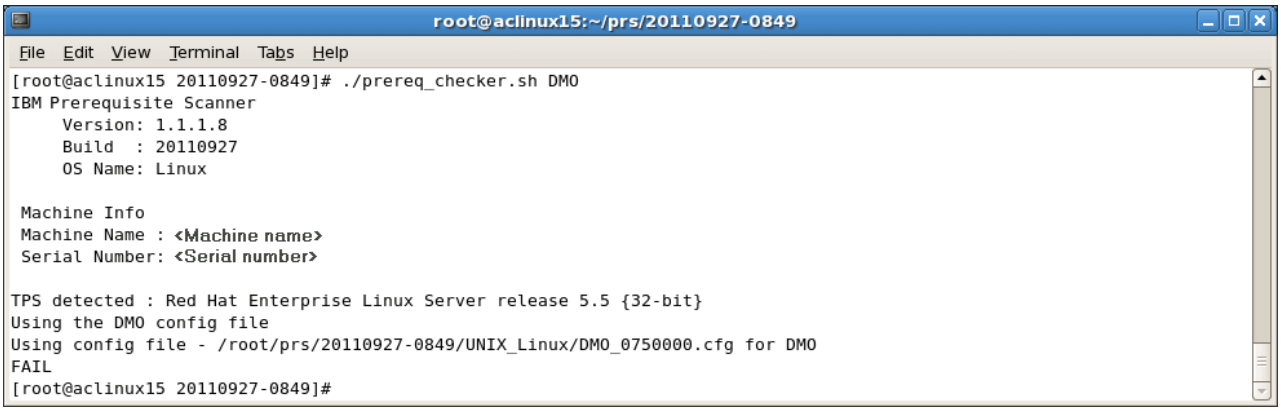


Figura 2. Salida en la interfaz de línea de comandos en sistemas UNIX

Prerequisite Scanner genera códigos de retorno en función de los resultados del análisis y si debe salir debido a errores. Estos códigos de retorno se graban en los archivos de registro. Por ejemplo, si Prerequisite Scanner no puede ejecutar el análisis porque no puede leer el archivo de configuración, genera el código de retorno 2.

Suma de las propiedades de requisitos previos de memoria y espacio en disco

Puede ejecutar Prerequisite Scanner para comprobar de forma simultánea los requisitos previos de uno o muchos productos o componentes cuando especifique varios códigos de producto como parámetros de entrada. Prerequisite Scanner agrega los resultados de las comprobaciones de requisitos previos de memoria y espacio en disco en las siguientes secciones agregadas de la salida, si se han especificado propiedades de requisitos previos en cualquier de los archivos de configuración:

- En sistemas UNIX, en la sección `TOTAL ALL SPECIFIED COMPONENTS`
- En sistemas Windows, en la sección `ALL COMPONENTS`

Las propiedades generales de requisitos previos:

- La cantidad total de memoria física que está actualmente disponible en el entorno de destino, que es:

  `Memory`
- Espacio en disco de los sistemas de archivos para las siguientes propiedades de requisitos previos:

Plataforma	Propiedades de requisitos previos
UNIX y Linux	• `os.space.home` • `os.space.opt` • `os.space.tmp` • `os.space.usr` • `os.space.var`
Windows	`Disk`

Nota: Si `opt`, `usr` y `var` no se han definido como sistemas de archivos en el equipo de destino, Prerequisite Scanner no muestra los valores esperados y los resultados devueltos para estas propiedades de requisitos previos en la sección intercalada.

Prerequisite Scanner no muestra la sección agregada, si no hay propiedades de requisitos previos de memoria ni de espacio en disco en los archivos de configuración.

Prerequisite Scanner se encarga de la comparación y visualización de los valores de espacio en disco en la sección agregada de los resultados del análisis de forma diferente a la sección principal. Consulte “Unidades de medida de salida” en la página 34.

Salida del archivo de registro de depuración en sistemas Windows

Prerequisite Scanner muestra información de procesamiento, mensajes de error y de aviso y los resultados del análisis en el archivo *ips_output_dir*/`precheck.log`. Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se define el parámetro opcional **debug**, Prerequisite Scanner muestra mensajes adicionales de depuración en este archivo.

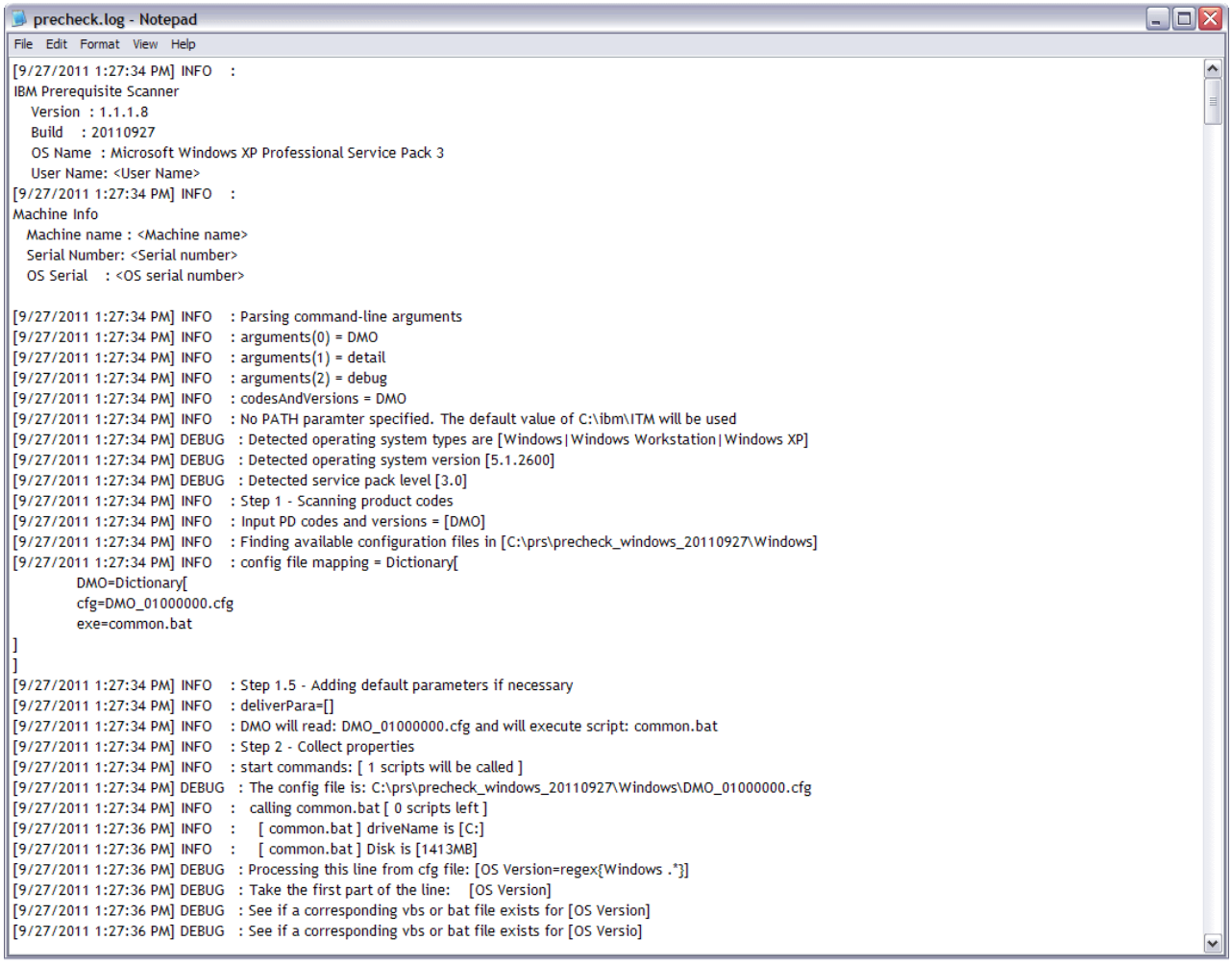


Figura 3. Archivo precheck.log

Salida del archivo de registro de depuración en sistemas UNIX

Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se establece el parámetro opcional **debug**, Prerequisite Scanner muestra información de

procesamiento, mensajes de error y de aviso y los resultados del análisis en el archivo *ips_output_dir*/temp/prs.debug.

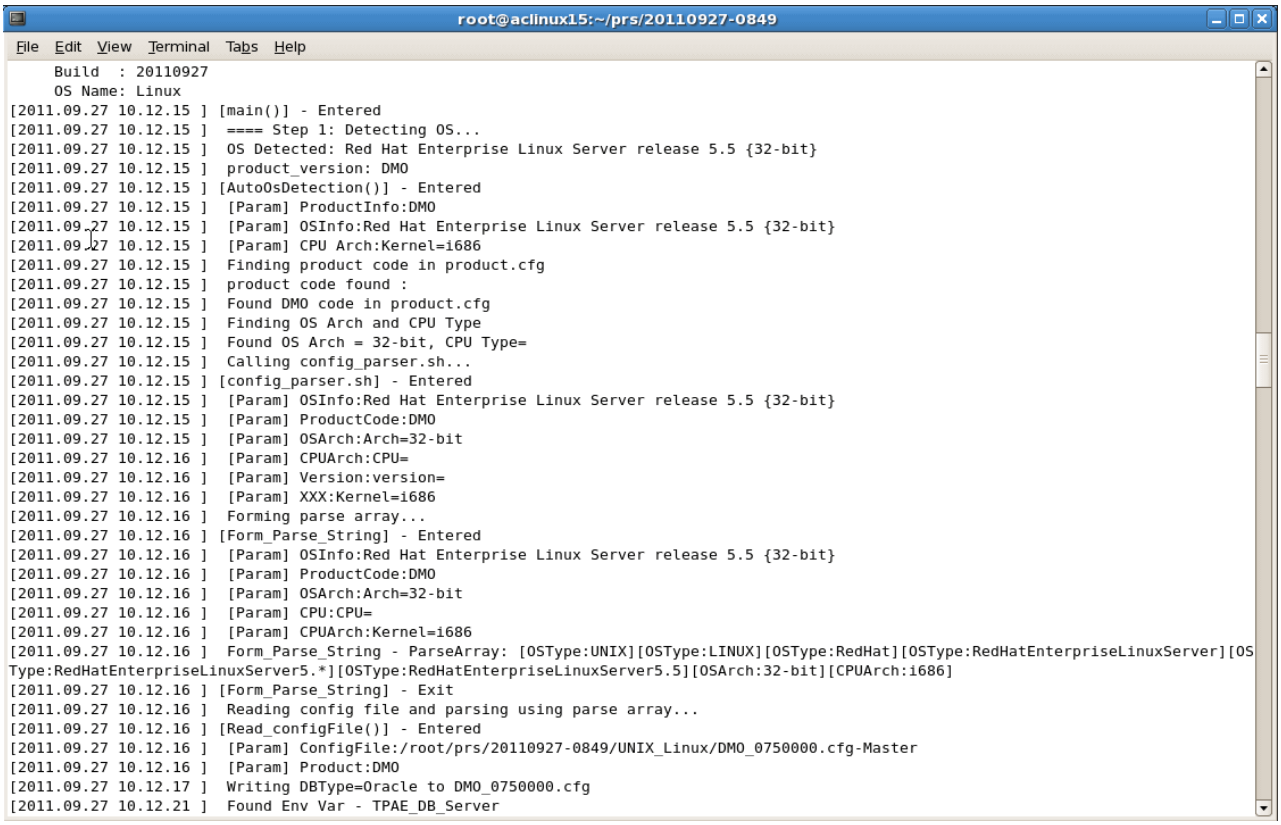


Figura 4. Archivo prs.debug en sistemas UNIX

Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se establece el parámetro opcional **trace**, Prerequisite Scanner muestra información de seguimiento en el archivo *ips_output_dir*/temp/prs.trc.

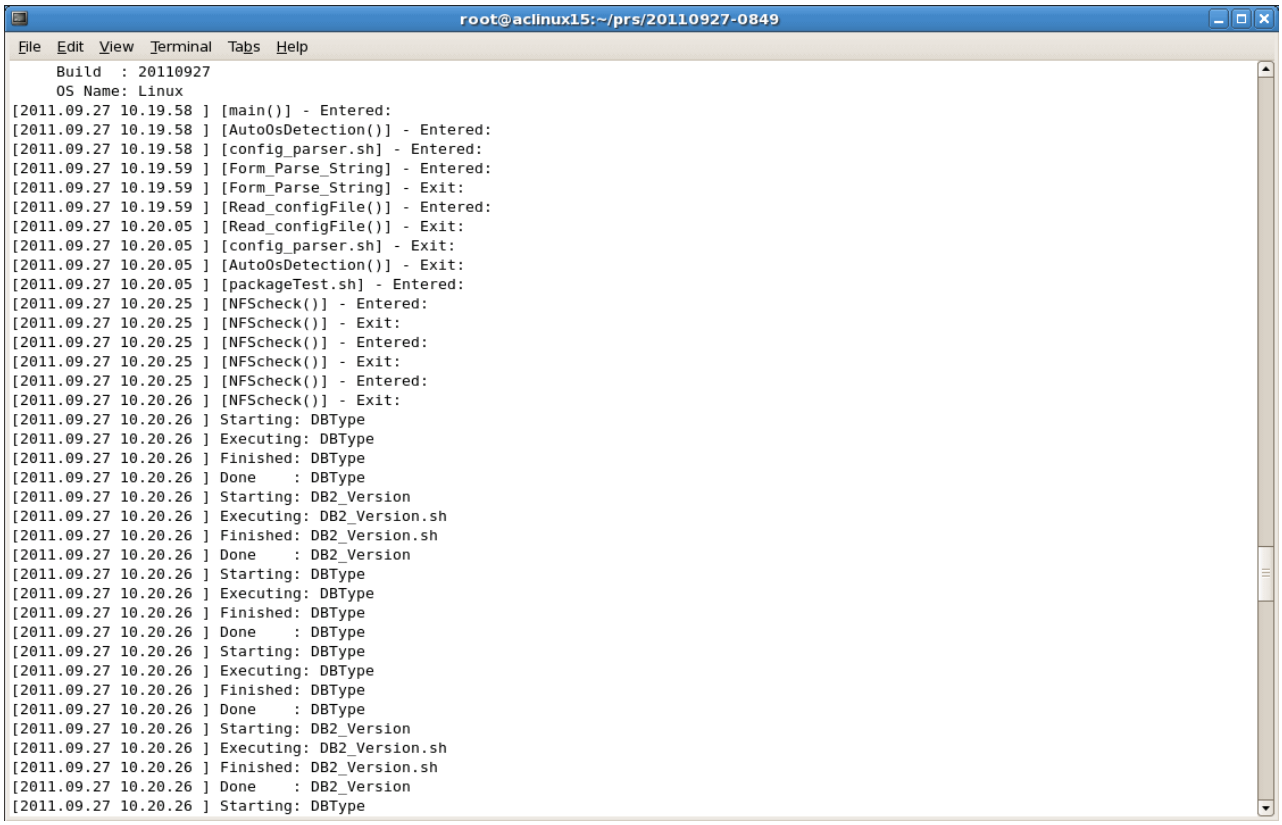
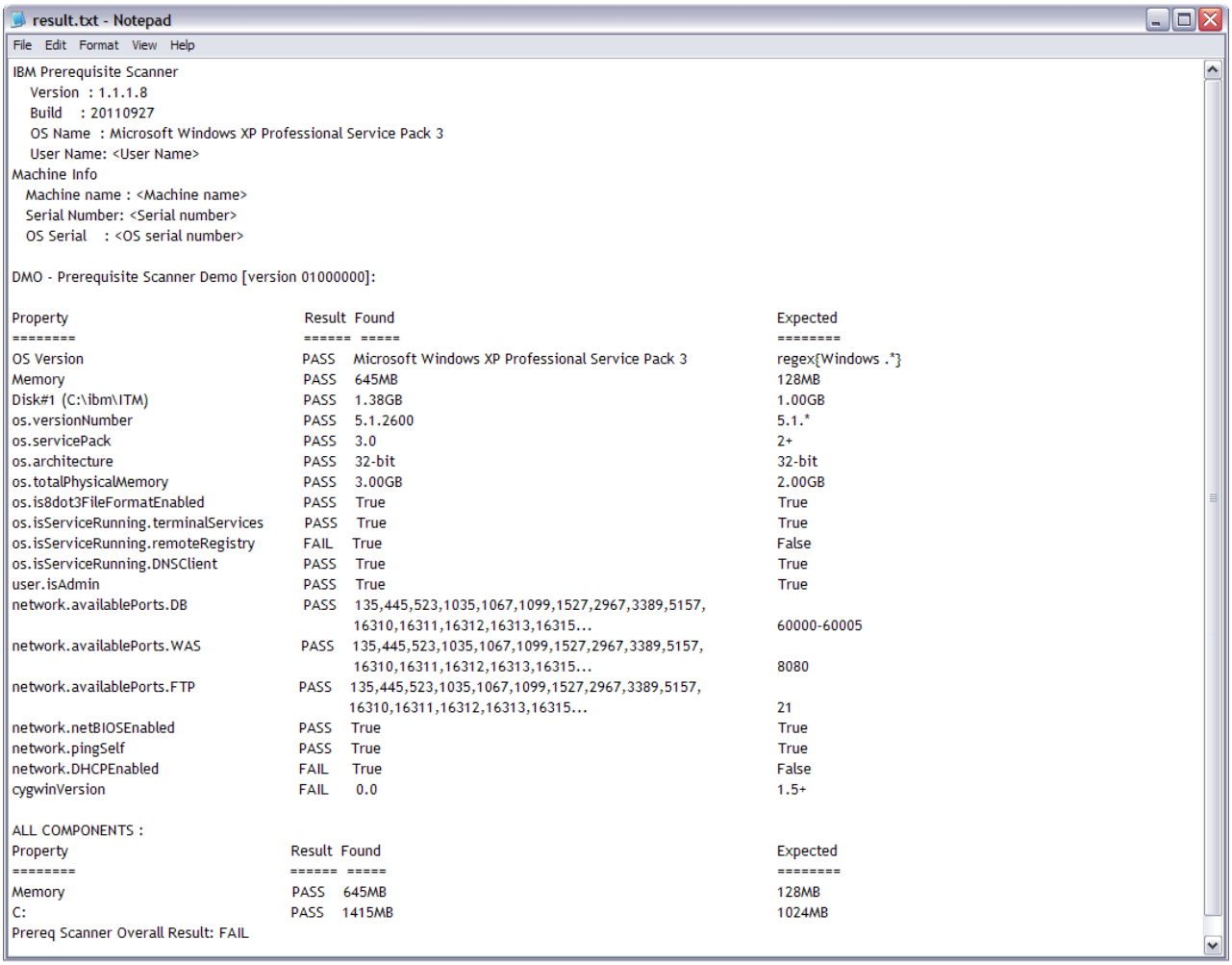


Figura 5. Archivo prs.trc en sistemas UNIX

Salida de archivo de texto

Prerequisite Scanner muestra resultados de análisis detallados en el archivo *ips_output_dir*/`result.txt`. Guarda los resultados en el archivo de texto independientemente de si se establece el parámetro **`detail`**.



result.txt - Notepad

File Edit Format View Help

```
IBM Prerequisite Scanner
   Version  : 1.1.1.8
   Build    : 20110927
   OS Name  : Microsoft Windows XP Professional Service Pack 3
   User Name: <User Name>
Machine Info
  Machine name : <Machine name>
  Serial Number: <Serial number>
  OS Serial    : <OS serial number>

DMO - Prerequisite Scanner Demo [version 01000000]:

Property                          Result  Found                                                    Expected
========                          ======  =====                                                    ========
OS Version                        PASS    Microsoft Windows XP Professional Service Pack 3         regex{Windows .*}
Memory                            PASS    645MB                                                    128MB
Disk#1 (C:\ibm\ITM)               PASS    1.38GB                                                   1.00GB
os.versionNumber                  PASS    5.1.2600                                                 5.1.*
os.servicePack                    PASS    3.0                                                      2+
os.architecture                   PASS    32-bit                                                   32-bit
os.totalPhysicalMemory            PASS    3.00GB                                                   2.00GB
os.is8dot3FileFormatEnabled       PASS    True                                                     True
os.isServiceRunning.terminalServices PASS  True                                                    True
os.isServiceRunning.remoteRegistry FAIL   True                                                     False
os.isServiceRunning.DNSClient     PASS    True                                                     True
user.isAdmin                      PASS    True                                                     True
network.availablePorts.DB         PASS    135,445,523,1035,1067,1099,1527,2967,3389,5157,
                                          16310,16311,16312,16313,16315...                         60000-60005
network.availablePorts.WAS        PASS    135,445,523,1035,1067,1099,1527,2967,3389,5157,
                                          16310,16311,16312,16313,16315...                         8080
network.availablePorts.FTP        PASS    135,445,523,1035,1067,1099,1527,2967,3389,5157,
                                          16310,16311,16312,16313,16315...                         21
network.netBIOSEnabled            PASS    True                                                     True
network.pingSelf                  PASS    True                                                     True
network.DHCPEnabled               FAIL    True                                                     False
cygwinVersion                     FAIL    0.0                                                      1.5+

ALL COMPONENTS :
Property                          Result  Found                                                    Expected
========                          ======  =====                                                    ========
Memory                            PASS    645MB                                                    128MB
C:                                PASS    1415MB                                                   1024MB
Prereq Scanner Overall Result: FAIL
```

Figura 6. Archivo result.txt en sistemas Windows

```
[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
```

Figura 7. Archivo result.txt en sistemas UNIX

Salida de archivo XML

Prerequisite Scanner muestra resultados de análisis detallados en el archivo *ips_output_dir*/`result.xml` cuando se especifica el parámetro de entrada opcional **`xmlResult`**. Puede utilizarlo para indicarle a la herramienta que muestre los resultados en el archivo de resultados XML, además de en el archivo de resultados de texto sin formato. Guarda los resultados en el archivo de XML independientemente de si se establece el parámetro **`detail`**.

```
<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>
```

Figura 8. Archivo result.XML en sistemas Windows

Los desarrolladores pueden utilizar el kit de herramientas Prerequisite Scanner Java Developer para analizar y leer el archivo XML.

Unidades de medida de salida

Prerequisite Scanner se encarga de la comparación y visualización de los valores de espacio en disco en la sección agregada de los resultados del análisis de forma diferente a la sección principal.

En la sección principal de los resultados del análisis, Prerequisite Scanner se encarga de la comparación y visualización de los valores de espacio de disco del siguiente modo:

Plataforma	Propiedades de requisitos previos
UNIX y Linux	Si el valor real es mayor que 1024 MB, Prerequisite Scanner convierte y devuelve el valor en GB, de lo contrario, devuelve el valor en MB.

Plataforma	Propiedades de requisitos previos
Windows	Prerequisite Scanner utiliza la unidad del valor esperado en el archivo de configuración como unidad de comparación y visualización. Convierte el valor real a esta unidad si es necesario.

En la sección agregada de los resultados del análisis, Prerequisite Scanner se encarga de la comparación y visualización de los valores de espacio de disco del siguiente modo:

Plataforma	Propiedades de requisitos previos
UNIX y Linux	Si el valor real es mayor que 1024 MB, Prerequisite Scanner convierte y devuelve el valor en GB, de lo contrario, devuelve el valor en MB.
Windows	Prerequisite Scanner realiza la comparación del espacio en disco en base a unidades de MB. Para cada archivo de configuración que contiene la propiedad de requisito previo `Disk`, Prerequisite Scanner convierte los valores reales y esperados a MB y lleva a cabo la comparación. A efectos de visualización, si el valor real agregado es mayor que 1 GB, Prerequisite Scanner devuelve el valor real en GB con una precisión de 2; de lo contrario, devuelve el valor en MB.

Kit de herramientas Java Developer para Prerequisite Scanner

El kit de herramientas Java Developer para Prerequisite Scanner es un conjunto de APIs que le permite, como desarrollador, analizar y leer mediante programación los contenidos del archivo XML de resultados en función de sus necesidades; por ejemplo, analizar los resultados de la exploración para utilizar con su programa de instalación.

El kit de herramientas proporciona los siguientes paquetes:

- com.ibm.prs.common.exception

  Contiene la clase `PRSApiException` que proporciona los métodos para emitir excepciones para la API de consultas XML.
- com.ibm.prs.common.reports.api

  Contiene la interfaz `PRSXmlResultReader` que define la API de consultas XML para el archivo de resultados XML.
- com.ibm.prs.common.reports.api.impl

  Contiene la clase `PRSXmlResultReaderImpl` que implementa `PRSXmlResultReader`.

Prerequisite Scanner puede validar el formato y la estructura en relación con el archivo de esquema XML *ips_root*`/PRSResults.xsd`.

Javadoc esta disponible para el kit de herramientas en el directorio *ips_root*`/api/javadoc`.

Archivo de esquema XML para el archivo de resultado XML

Prerequisite Scanner proporciona un archivo de esquema XML frente al que se puede validar el archivo XML de resultado.

Los archivos de esquema XML contienen los siguientes elementos en representación de secciones:

- `PRSInfo` para gestionar detalles de Prerequisite Scanner
- `MachineInfo` para gestionar información sobre el entorno de destino en el que se ejecuta la exploración
- `UserInfo` para gestionar información sobre el usuario conectado que ejecuta la exploración
- `ScenarioInfo` para gestionar información sobre el tipo de exploración y escenario
- `ProductInfo` para gestionar información sobre el producto o el componente y el archivo de configuración correspondiente
- `DetailedResults` para gestionar los resultados de exploración de cada conjunto de propiedades de requisito previo para un producto o componente tal como se agrupan en `DetailedProductResultsElement`
- `AggregateResults` para gestionar los resultados de la exploración agregada de espacio de disco y memoria
- `OverallResult` para gestionar el resultado CORRECTO o INCORRECTO de la exploración

El nombre y la ubicación del esquema XML es:*ips_root*`/PRSResults.xsd`

Como desarrollador o desplegador, puede invocar métodos desde la API del XML de consulta para validar el archivo de resultado XML. El archivo Javadoc está disponible para el kit de herramientas en el directorio *ips_root*`/api/javadoc`.

Proceso de exploración

Cuando se ejecuta IBM Prerequisite Scanner, realiza un conjunto de tareas en cada etapa del proceso de exploración. El usuario abre una interfaz de línea de comandos y ejecuta la secuencia de comandos Prerequisite Scanner con el conjunto de parámetros de entrada que incluye un código de producto.

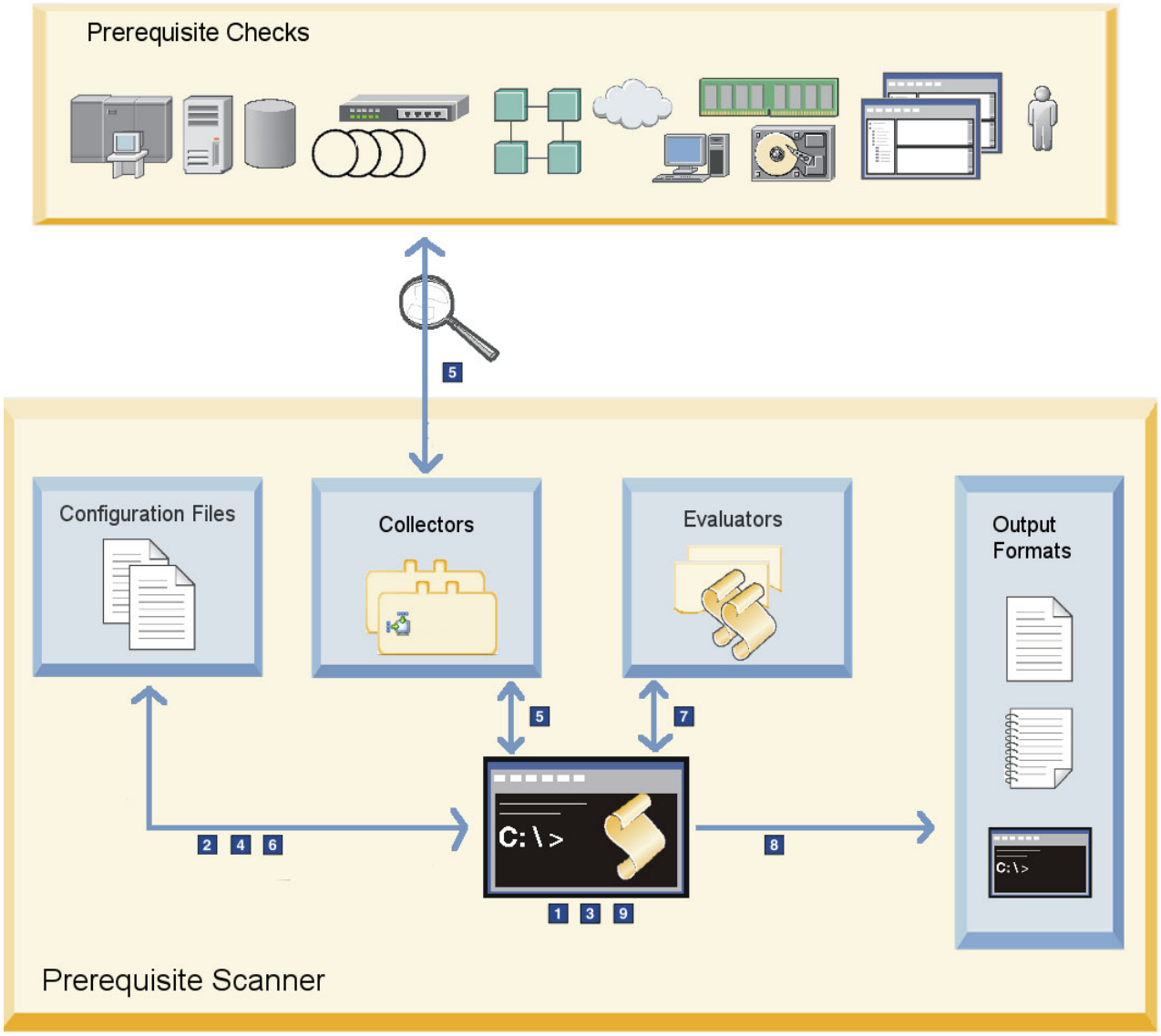


Figura 9. Proceso de exploración y arquitectura de Prerequisite Scanner

El proceso de exploración de Figura 9 se resumen del siguiente modo:

1. Prerequisite Scanner verifica el formato de los parámetros de entrada que se pasan a Scanner.
2. Scanner determina si el código de producto que se pasa como uno de los parámetros de entrada es un código de producto válido del archivo `codename.cfg` .
3. Scanner busca el archivo de configuración asociado con el código de producto. Si el parámetro de versión del producto opcional no se ha pasado, Scanner utiliza la última versión del archivo de configuración que encuentra en el directorio *ips_root*`/Windows|UNIX_Linux`.
4. Scanner verifica si el sistema operativo real de la máquina es un sistema operativo compatible. Scanner comprueba el sistema operativo real en relación con el sistema operativo compatible esperado en los títulos de sección del

fichero de configuración, cuyo nombre de archivo contiene el mismo código de producto y versión del producto que los parámetros de entrada.

5. Scanner recopila las propiedades de requisito previo reales de las comprobaciones de requisitos previos mediante los recopiladores de Prerequisite Scanner.
6. Scanner comprueba las propiedades de requisitos previos del archivo de configuración que está asociado con el código y versión del producto.

   Scanner comprueba el sistema operativo real en relación con el sistema operativo compatible esperado en la propiedad de requisito previo de la versión del SO o en los títulos de sección del archivo de configuración, cuyo nombre de archivo contiene el mismo código de producto y versión del producto que los parámetros de entrada.
7. Scanner lee las propiedades de requisitos previos del archivo de configuración y analiza los valores reales y esperados de las propiedades de requisitos previos para las comprobaciones de requisitos previos. Utiliza los evaluadores de Prerequisite Scanner en caso necesario.
8. Scanner muestra los resultados de la exploración en la interfaz de línea de comandos, los archivos de XML y de texto de resultados, así como los archivos de registro legibles para los humanos.
9. Scanner limpia y elimina los archivos temporales y directorios.

Novedades en este release

IBM Prerequisite Scanner Versión 1.2 proporciona nuevas propiedades y mejoras. También contiene correcciones de los defectos.

Nuevas características de esta modificación

Capacidad para analizar y leer el nuevo archivo XML de resultados de análisis.

El kit de herramientas Java Developer para Prerequisite Scanner es un conjunto de APIs que permite a los desarrolladores analizar y leer mediante programación los contenidos del archivo XML de resultados en función de sus necesidades; por ejemplo, analizar los resultados de la exploración para utilizar en un programa de instalación. Consulte “Kit de herramientas Java Developer para Prerequisite Scanner” en la página 35.

Nuevos archivos de configuración en esta modificación

Tabla 9 describe los nuevos archivos de configuración y códigos de producto enviados con Prerequisite Scanner Versión 1.2

Tabla 9. Archivos de configuración nuevos

Producto o componente	Código de producto	Archivo de configuración
Tivoli Composite Application Manager Agent for WebSphere MQ	KMQ	*ips_root*/Windows|UNIX_Linux/KMQ_07010000.cfg
Tivoli Composite Application Manager Agent para WebSphere Message Broker	KQI	*ips_root*/Windows|UNIX_Linux/KQI_07010000.cfg

Nuevas propiedades de requisitos previos en esta modificación

La propiedad `os.SeaMonkeyVersion` se ha añadido para buscar la versión de Mozilla SeaMonkey en el equipo. Consulte “Propiedad de los datos del sistema operativo” en la página 101.

La propiedad `env.var.set.env_var_name` se ha añadido si se ha establecido en el equipo la variable de entorno tal como especifica *nombre_var_ent*. Consulte “Propiedades de los datos de la variable de entorno” en la página 115.

Mejoras en esta modificación

Capacidad para escribir los resultados del análisis en un archivo XML.

ips_output_dir`/result.xml` es el nuevo archivo de resultados de análisis en formato XML. De forma predeterminada, la herramienta muestra los resultados únicamente en el archivo de resultados de texto sin formato. Consulte “Formatos de salida” en la página 26.

`xmlResult` es u nuevo parámetro de entrada opcional para la secuencia de comandos Prerequisite Scanner de Prerequisite Scanner Versión 1.2. Puede utilizarlo para indicarle a la herramienta que muestre los resultados en el archivo de resultados XML, además de en el archivo de resultados de texto sin formato. Consulte “prereq_checker” en la página 67.

Eliminación de la sección agregada en los resultados si no hay en los archivos de configuración propiedades de requisitos previos de memoria ni de espacio en disco.
Prerequisite Scanner ya no muestra las secciones agregadas en el archivo de resultados cuando en los archivos de configuración no existen propiedades de requisitos previos de memoria o de espacio en disco. Consulte “Formatos de salida” en la página 26.

Funciones obsoletas de esta modificación

Ninguna

Defectos arreglados en esta modificación

Para obtener una lista de defectos corregidos en este release, consulte el archivo `Readme.html` en el directorio *ips_root* cuando extraiga el contenido de los paquetes de software Prerequisite Scanner.

Cambios de documentación en esta modificación

La Guía del usuario de Prerequisite Scanner ya no se incluye con los paquetes de plataforma de Prerequisite Scanner para Prerequisite Scanner. Puede utilizar IBM Prerequisite Scanner Information Center.

Capítulo 2. Instalación de Prerequisite Scanner

No hay ningún programa de instalación para IBM Prerequisite Scanner. Al extraer el contenido del archivo comprimido, los archivos principales se encuentran en el directorio raíz, con los siguientes subdirectorios: `/api` para el kit de herramientas Prerequisite Scanner Java Developer que da soporte a la API XML de consultas, `/lib` para los recopiladores y secuencias de comandos comunes; `/Windows` para los evaluadores y archivos de configuración en Windows; `/UNIX_Linux` para los evaluadores y archivos de configuración en plataformas UNIX; y `/licenses` para los archivos de licencia.

Requisitos previos

IBM Prerequisite Scanner se puede ejecutar en sistemas Windows, Windows XP o posterior, de 32 bits o 64 bits. También se puede ejecutar en variantes de los sistemas operativos AIX, HP-UX, Linux y Solaris.

Asegúrese de que dispone de los siguientes programas de utilidad instalados o disponibles en los entornos de destino:

Sistema de destino	Requisitos previos
Windows	• El cliente Telnet debe estar activado, para que las comprobaciones de conectividad del recopilador de conectividad predefinido puedan funcionar correctamente. • Asegúrese de que Microsoft .Net Framework 1.0 o posterior esté instalado para que Prerequisite Scanner pueda funcionar correctamente.

Sistema de destino	Requisitos previos

UNIX	• Bash debe estar instalado para que los recopiladores Prerequisite Scanner UNIX puedan funcionar correctamente.
	• Para usuarios no root, la ubicación de los comandos **mount**, **swapinfo** y **psrinfo** debe establecerse en la variable de entorno PATH, para que los comandos estén disponibles en Prerequisite Scanner. Los comandos están en el directorio /usr/sbin; por ejemplo, establezca la variable de entorno PATH del siguiente modo:
	`export PATH=$PATH:/usr/sbin/`
	• Asegúrese de que se han asignado los permisos de acceso correctos al comando **lscfg**, incluidos los permisos específicos establecidos por los distintivos de derecho de acceso, como setuid bit. Tener los permisos de acceso adecuados significa que Prerequisite Scanner puede ejecutar el comando y recuperar la información del sistema. El comando se encuentra en el directorio /usr/sbin; por ejemplo, para establecer el bit setuid para **lscfg**, ejecute el comando **chmod** del siguiente modo:
	`chmod 4777 /usr/sbin/lscfg`

Prerequisite Scanner admite todo el hardware y sistemas operativos del producto especificado o la solución de IBM para la que ejecute Prerequisite Scanner.

Instalación del archivo comprimido

Puede extraer el contenido del archivo comprimido para IBM Prerequisite Scanner . Debe tener permisos de escritura para el directorio raíz en el que desee extraer el contenido del archivo comprimido.

Procedimiento

1. Abra el navegador web e introduzca la dirección URL de IBM Fix Central. Regístrese en IBM.com el Portal de soporte de IBM.
2. En la lista **Grupo de productos**, seleccione **Tivoli**.
3. En la lista **Seleccionar de Tivoli**, elija IBM Prerequisite Scanner.
4. En la lista **Versión instalada**, seleccione la versión que desee descargar.
5. En la lista **Plataforma**, seleccione la plataforma en la que desee instalar Prerequisite Scanner.
6. Pulse **Continuar**.Se abrirá la página Identificar arreglos.
7. Utilice la opción predeterminada, **Buscar arreglos** y pulse **Continuar**.
8. En la página Seleccionar arreglos, seleccione el paquete y pulse **Continuar**.
9. En la página Opción de descarga, seleccione la opción de descarga y pulse **Descargar ahora**.

10. Extraiga el contenido de los archivos comprimidos en la ubicación que prefiera según la especificación de *ips_root*.

Qué hacer a continuación

Asegúrese de comprobar la documentación sobre la instalación del producto o las notas técnicas para conocer los pasos adicionales que debe realizar antes de ejecutar Prerequisite Scanner. Por ejemplo, puede que necesite ajustar la variable de entorno que indica a Prerequisite Scanner qué componentes o características se están instalando en el equipo cliente y, por lo tanto, qué requisitos previos deben comprobarse.

Desinstalación de Prerequisite Scanner

Quite IBM Prerequisite Scanner si desea instalar una versión más reciente, moverlo a otro entorno o es una versión que ya no necesita.

Procedimiento

1. Abra el directorio *ips_root*.
2. Elimine el directorio y su contenido.

Capítulo 3. Ampliación de Prerequisite Scanner

IBM Prerequisite Scanner proporciona un conjunto básico de recopiladores, evaluadores y configuraciones que puede utilizar para ejecutar la herramienta y analizar requisitos previos. Si el conjunto básico de archivos, las propiedades de requisitos previos y valores, así como las comprobaciones de requisitos previos no se ajustan a sus necesidades, puede ampliar Prerequisite Scanner .

Antes de ejecutar Prerequisite Scanner

Antes de ejecutar IBM Prerequisite Scanner, determine si las propiedades de requisitos previos predefinidas, sus valores esperados y los archivos de configuración cumplen los requisitos de exploración de requisitos previos. Si alguno de ellos no se ajusta a sus necesidades, puede ejecutar una serie de tareas de requisitos previos para configurar o ampliar Prerequisite Scanner. El conjunto de comprobaciones de requisitos previos y las tareas dependen de la plataforma y el número de comprobaciones de requisitos previos.

Comprobaciones necesarias y tareas de ampliación para sistemas Windows

Debe realizar una serie de comprobaciones y tareas antes de ejecutar IBM Prerequisite Scanner. Estas comprobaciones determinan si puede editar y utilizar los archivos de configuración existentes o debe ampliar Prerequisite Scanner.

Tabla 10 proporciona una lista de las comprobaciones y tareas que deben realizarse.

Tabla 10. Comprobaciones y tareas anteriores al uso de un archivo de configuración para sistemas Windows

	Comprobación	**Tarea**
☐	Compruebe si el producto, sus sistemas operativos y versiones del sistema operativo compatibles están enumerados en el archivo `codename.cfg`.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, añada al archivo un código de producto para el producto, el sistema operativo individual y la versión del sistema operativo opcional. Para obtener más información, consulte el apartado “Adición de códigos de producto” en la página 48.
☐	Compruebe si existe un archivo de configuración para el código de producto asociado con la versión del producto.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, cree un archivo de configuración para que incluya las propiedades de requisito previo del sistema operativo y la versión del sistema operativo. Para obtener más información, consulte el apartado “Creación de archivos de configuración personalizados” en la página 48.
☐	Abra el archivo de configuración y compruebe si contiene las propiedades de requisitos previos correctas.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, añada propiedades de requisitos previos. Para obtener más información, consulte el apartado “Añadir propiedades de requisitos previos” en la página 50.

Tabla 10. Comprobaciones y tareas anteriores al uso de un archivo de configuración para sistemas Windows (continuación)

	Comprobación	Tarea
☐	Compruebe si las propiedades de requisitos previos tienen los valores esperados.	• Si es así, ejecute Prerequisite Scanner. Para obtener más información, consulte el apartado Capítulo 4, "Ejecución de Prerequisite Scanner", en la página 67. • En caso contrario, edite las propiedades de requisitos previos. Para obtener más información, consulte el apartado "Edición de las propiedades de requisitos previos" en la página 52.
☐	Para todas las propiedades nuevas de requisitos previos, compruebe si los recopiladores predefinidos pueden recopilar los valores reales de las propiedades de requisitos previos.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, cree recopiladores personalizados. Para obtener más información, consulte el apartado "Creación de recopiladores personalizados para sistemas Windows" en la página 53.
☐	Para todas las propiedades de requisitos previos nuevas o editadas, compruebe si los evaluadores predefinidos pueden comparar los valores reales y esperados de la propiedad de requisito previo.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, cree evaluadores personalizados. Para obtener más información, consulte el apartado "Creación de evaluadores personalizados para sistemas Windows" en la página 60.
☐	Asegúrese de que todos los archivos se han guardado en los directorios correctos: • Archivos de configuración, cualquier archivo de recopilador específico de producto personalizado y de lotes asociado y cualquier archivo de evaluador personalizado del directorio *ips_root*`/Windows_Linux` • Recopiladores comunes personalizados en el directorio *ips_root*`/lib`	Ejecute Prerequisite Scanner.Para obtener más información, consulte el apartado Capítulo 4, "Ejecución de Prerequisite Scanner", en la página 67.

Comprobaciones necesarias y tareas de ampliación para sistemas UNIX

Debe realizar una serie de comprobaciones de requisitos previos y tareas antes de ejecutar IBM Prerequisite Scanner. Estas comprobaciones determinan si puede editar y utilizar los archivos de configuración existentes o debe ampliar Prerequisite Scanner.

Tabla 11 proporciona una lista de las comprobaciones necesarias y tareas para realizar.

Tabla 11. Comprobaciones y tareas anteriores al uso de un archivo de configuración para sistemas UNIX

	Comprobación	Tarea
☐	Compruebe si el producto está incluido en la lista del archivo `codename.cfg`.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, añada un código de producto al archivo `codename.cfg`. Para obtener más información, consulte el apartado "Adición de códigos de producto" en la página 48.

Tabla 11. Comprobaciones y tareas anteriores al uso de un archivo de configuración para sistemas UNIX (continuación)

	Comprobación	Tarea
☐	Compruebe si existe un archivo de configuración para el código de producto asociado con el producto.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, cree un archivo de configuración para que incluya las propiedades de requisito previo de todas las plataformas compatibles del producto. Para obtener más información, consulte el apartado "Creación de archivos de configuración personalizados" en la página 48.
☐	Abra el archivo de configuración y compruebe si contiene las propiedades de requisitos previos correctas.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, añada propiedades de requisitos previos. Para obtener más información, consulte el apartado "Añadir propiedades de requisitos previos" en la página 50.
☐	Compruebe si las propiedades de requisitos previos tienen los valores esperados.	• Si es así, ejecute Prerequisite Scanner. Para obtener más información, consulte el apartado Capítulo 4, "Ejecución de Prerequisite Scanner", en la página 67. • En caso contrario, edite las propiedades de requisitos previos. Para obtener más información, consulte el apartado "Edición de las propiedades de requisitos previos" en la página 52.
☐	Para todas las propiedades nuevas de requisitos previos, compruebe si los recopiladores predefinidos pueden recopilar los valores reales de las propiedades de requisitos previos.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, cree recopiladores personalizados. Para obtener más información, consulte el apartado "Creación de recopiladores personalizados para sistemas UNIX" en la página 57.
☐	Para todas las propiedades de requisitos previos nuevas o editadas, compruebe si los evaluadores pueden comparar los valores reales y esperados de la propiedad de requisito previo.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, cree evaluadores personalizados. Para obtener más información, consulte el apartado "Creación de evaluadores personalizados para sistemas UNIX" en la página 64.
☐	Para todas las propiedades de requisitos previos nuevas o editadas, compruebe si el código para llamar y ejecutar los recopiladores se encuentra en la secuencia de comandos *ips_root*/UNIX_Linux/packageTest.sh.	• Si es así, lleve a cabo la siguiente comprobación. • En caso contrario, edite la secuencia de comandos de prueba de paquete maestra. Para obtener más información, consulte el apartado "Edición de la secuencia de comandos de prueba de paquete para sistemas UNIX" en la página 59.
☐	Asegúrese de que todos los archivos se han guardado en los directorios correctos: • Archivos de configuración, archivos de recopiladores personalizados y cualquier archivo de evaluador personalizado del directorio *ips_root*/UNIX_Linux	Ejecute Prerequisite Scanner.Para obtener más información, consulte el apartado Capítulo 4, "Ejecución de Prerequisite Scanner", en la página 67.

Adición de códigos de producto

IBM Prerequisite Scanner proporciona un conjunto de códigos predefinidos de versión del producto en el archivo `codename.cfg`. Puede añadir códigos de producto si el archivo no los contiene para la versión de producto, sus plataformas compatibles y las versiones de los sistemas operativos.

Procedimiento

1. Abra el archivo *ips_root*/`codename.cfg`.
2. Compruebe si el archivo ya contiene pares de nombre y valor para las versiones del producto.
3. Si el código de producto no existe, añada uno y asegúrese de utilizar el formato correcto, del siguiente modo:

   product_code=code_value

   Restricción: IBM Tivoli Monitoring y Tivoli Composite Application Manager tienen códigos predefinidos de productos que Prerequisite Scanner considera como reservados. Estos códigos no deben utilizarse como códigos de producto de Prerequisite Scanner a menos que se refieran a sus agentes asociados IBM Tivoli Monitoring y Tivoli Composite Application Manager. Para obtener más información sobre los códigos de producto, consulte el apartado ITM 6.X Product Codes Technote.

   Restricción: Sólo en UNIX: al introducir el valor para el código de producto en el archivo, evite el uso de `for`. Es una palabra reservada y puede afectar a la ejecución de Prerequisite Scanner.

   Por ejemplo, para añadir un código de producto de IBM Tivoli Monitoring for Energy Management a todas las plataformas de Windows, añada la siguiente línea al archivo:

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

Creación de archivos de configuración personalizados

Puede crear archivos de configuración personalizados a partir del archivo de configuración de ejemplo, si los archivos de configuración predefinidos no satisfacen sus requisitos para las propiedades de requisitos previos. Antes de crear el archivo de configuración personalizado, asegúrese de conocer las propiedades de requisitos previos que desee agregar y sus valores esperados.

Acerca de esta tarea

Importante: Debe cumplir los convenios de denominación y reglas de formato que rigen la creación y edición de un archivo de configuración personalizado. Si no lo hace, Prerequisite Scanner no podrá ejecutar con éxito un análisis utilizando este archivo.

Procedimiento

1. Si es necesario, añada códigos de producto del producto al archivo `codename.cfg`.
2. Cree el archivo de configuración mediante un editor de texto en el directorio *ips_root*/*OS*. Asegúrese de que utiliza el convenio de denominación para el nombre de archivo:

   product_code_version`.cfg`

donde:

- *product_code*

  Es la variable para representar un código de producto en sistemas Windows o UNIX. Los códigos de producto identifican el producto, una plataforma individual, como Windows, AIX, HP-UX, Linux y Solaris, y opcionalmente la versión del sistema operativo compatible con ese producto. Se guardan en el archivo `codename.cfg`. Los productos que admiten varias plataformas tienen varios códigos de producto. Cada uno de ellos identifica un producto, una plataforma y una versión del sistema operativo, según se requiera.
- *version* es el código de 8 dígitos para representar la versión, release, modificación y nivel con dos dígitos para cada parte del código; por ejemplo, 7.3.21 es 07032100.

3. Revise las propiedades de requisitos previos básicos descritas en Apéndice C, "Referencia de propiedades de requisitos previos", en la página 91 y determine qué propiedades de requisitos previos desea comprobar.
4. Opcional: Añada una sección y asegúrese de que utiliza el siguiente convenio de denominación para el título de la sección:
   - **Categoría de datos de tipo de datos predefinida y única**

     [*category_name*:*category_value*]

     Por ejemplo, para crear una sección para las propiedades de requisitos previos comunes a todas las plataformas Windows, añada el siguiente título de sección:

     ```
     [OSType:Windows]
     ```

     Por ejemplo, para crear una sección para las propiedades de requisitos previos comunes a todas las variantes de SO RedHat Linux, añada el siguiente título de sección:

     ```
     [OSType:RedHat]
     ```
   - **Categorías de tipo de datos predefinidas y combinadas**

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

     Por ejemplo, para crear una sección para las propiedades de requisitos previos para las variantes de Windows Server 2003, excluida la variante Windows Server 2003 R2, añada el siguiente título de sección:

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

     Por ejemplo, cree una sección para las propiedades de requisitos previos para el SO SUSE Linux Enterprise Server 11 y si la variable de entorno `@TPAE_DB_SERVER` se ha establecido como `true`. Añada el siguiente título de sección de combinación:

     ```
     [OSType=SUSELinuxEnterpriseServer][@TPAE_DB_SERVER:true]
     ```

   donde:

   category_name es el código con varios caracteres que representa la categoría del tipo de datos como se indica en la Tabla 6 en la página 17

   category_value es el código con varios caracteres que representa un valor permitido para la categoría como se indica en la Tabla 6 en la página 17
5. Opcional: Para cada sección, revise las propiedades de requisitos previos básicos descritas en Apéndice C, "Referencia de propiedades de requisitos previos", en la página 91 y determine qué propiedades de requisitos previos desea comprobar.

6. Para cada propiedad de requisito previo que desee añadir, especifique un par de valor-nombre con calificadres opcionales si se necesita.Asegúrese de que utiliza el siguiente formato, con solo una propiedad de requisito previo por línea:

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

   donde:

   - *prefix_identifier* es un identificador de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4. Algunas de las categorías predefinidas requieren este identificador de prefijo.
   - *property_name* es el nombre de la propiedad de requisito previo.
   - *suffix_identifier* es un identificador opcional de un subtipo de las propiedades de requisitos previos como se indica en la Tabla 4 en la página 7.
   - *qualifier_name* es un atributo opcional de la propiedad de requisito previo. IBM Prerequisite Scanner lo utiliza para calificar la propiedad de requisito previo o el tipo de verificación para llevar a cabo en la propiedad de requisito previo, como se indica en "Calificadores predefinidos de propiedades de requisitos previos" en la página 9.

     Nota: Puede tener múltiples calificadores, cada uno de ellos separado por una coma. El conjunto de calificadores debe estar entre corchetes `[]`.
   - *qualifier_value* es el valor del atributo opcional. Cada calificador y su valor deben estar delimitados por dos puntos `:`.
   - *property_value* es el valor de la propiedad de requisito previo y puede ser una cadena o un entero.

   Por ejemplo, la categoría predefinida de usuario de las propiedades de requisitos previos tiene el identificador de prefijo `user`. La propiedad de requisito previo para comprobar si el usuario que ha iniciado sesión pertenece al grupo de usuarios de administrador es:`user.isAdmin=True`
7. Si una propiedad de requisito previo no existe en las categorías predefinidas de las propiedades de requisitos previos, añada el nombre de la propiedad de requisito previo, su valor y calificadores opcionales. A continuación, debe crear los siguientes archivos para comprobar y comparar la propiedad de requisito previo personalizada según sea necesario:un recopilador personalizado para recopilar el valor real de la propiedad de requisito previo y un evaluador personalizado si las funciones de comparación estándar no pueden comparar los valores reales y previstos.

Añadir propiedades de requisitos previos

Puede agregar las propiedades básicas de requisitos previos procedentes de las categorías predefinidas de las propiedades de requisitos previos a los archivos de configuración. O también puede agregar propiedades personalizadas de requisitos previos.

Acerca de esta tarea

Importante: Debe cumplir las reglas de formato que rigen la adición y edición de las propiedades de requisitos previos en un archivo de configuración. Si no lo hace, Prerequisite Scanner no podrá ejecutar con éxito un análisis de esa propiedad de requisito previo.

Procedimiento

1. Abra el archivo de configuración.
2. Revise las propiedades de requisitos previos básicos descritas en Apéndice C, "Referencia de propiedades de requisitos previos", en la página 91 y determine qué propiedades de requisitos previos desea comprobar.
3. Para cada propiedad de requisito previo que desee añadir, especifique un par de valor-nombre con calificadres opcionales si se necesita.

   Por ejemplo, para añadir propiedades de requisito previo procedentes de la categoría predefinida común, especifique solo el nombre de la propiedad y el valor previsto. Añada las siguientes propiedades de requisito previo al archivo:

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

   Por ejemplo, la categoría predefinida de red de las propiedades de requisitos previos tiene el identificador de prefijo `network` y el nombre de propiedad de requisito previo para comprobar los puertos disponibles es `availablePorts`. Puede categorizar aún más los puertos disponibles por subtipos de aplicación, `DB2` para servidor de bases de datos de DB2, `WAS` para WebSphere Application Server y `FTP` para protocolo FTP. Añada las siguientes propiedades de requisitos previos al archivo:

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

   Por ejemplo, la categoría predefinida de sistema operativo de las propiedades de requisitos previos tiene el identificador de prefijo `os` y el nombre de propiedad de requisito previo para comprobar el espacio de disco disponible es `space`. Puede categorizar aún más la comprobación por subtipos de sistema de archivos, `usr` y `home`. Puede especificar los valores para los calificadores `dir` y `unit`.

   Añada las siguientes propiedades de requisitos previos al archivo:

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

   Importante: Solo puede utilizar los calificadores predefinidos con determinadas propiedades de requisitos previos predefinidas, como se indica en la Tabla 5 en la página 10.
4. Si una propiedad de requisito previo no existe en las categorías predefinidas de las propiedades de requisitos previos, agregue el par nombre-valor con el calificador optativo del valor y la propiedad de requisito previo personalizada. Asegúrese de que utiliza el siguiente formato, con solo una propiedad de requisito previo por línea.

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

   donde:

   - *prefix_identifier* es un identificador de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4. Algunas de las categorías predefinidas requieren este identificador de prefijo.
   - *property_name* es el nombre de la propiedad de requisito previo.
   - *suffix_identifier* es un identificador opcional de un subtipo de las propiedades de requisitos previos como se indica en la Tabla 4 en la página 7.
   - *qualifier_name* es un atributo opcional de la propiedad de requisito previo. IBM Prerequisite Scanner lo utiliza para calificar la propiedad de requisito previo o el tipo de verificación para llevar a cabo en la propiedad de

requisito previo, como se indica en “Calificadores predefinidos de propiedades de requisitos previos” en la página 9.

Nota: Puede tener múltiples calificadores, cada uno de ellos separado por una coma. El conjunto de calificadores debe estar entre corchetes `[]`.

- *qualifier_value* es el valor del atributo opcional. Cada calificador y su valor deben estar delimitados por dos puntos `:`.
- *property_value* es el valor de la propiedad de requisito previo y puede ser una cadena o un entero.

Por ejemplo, `env.tcrhome` es una propiedad de requisito previo personalizada que comprueba la variable de entorno del directorio de inicio de Tivoli Common Reporting y el valor esperado debe ser `True`:

```
env.tcrhome=True
```

`env.path.jar` es una propiedad de requisito previo personalizada que comprueba si el JRE se ha configurado en la variable de entorno `PATH` y el valor esperado debe ser `False`:

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

Nota: A continuación, debe crear los siguientes archivos para comprobar y comparar la propiedad de requisito previo personalizada según sea necesario:un recopilador personalizado para recopilar el valor real de la propiedad de requisito previo y un evaluador personalizado sólo si las funciones de comparación estándar no pueden comparar los valores reales y previstos.

Edición de las propiedades de requisitos previos

Puede editar las propiedades de requisitos previos, cambiar los valores esperados para esas propiedades de requisitos previos o cambiar los valores asociados de los calificadores.

Antes de empezar

Compruebe si el nuevo valor es un valor válido, compatible con la propiedad de requisito previo. Por ejemplo, la propiedad de requisito previo `Disk` espera un formato numérico, con la unidad MB o GB. Si desea comprobar el espacio en disco disponible en terabytes (TB), debe ampliar la API de comparación para que admita comparaciones de TB. También debe modificar la propiedad de requisito previo `Disk` en los archivos de configuración correspondientes.

Compruebe los calificadores predefinidos y los valores válidos de la propiedad de requisito previo, como se describe en “Calificadores predefinidos de propiedades de requisitos previos” en la página 9.

Procedimiento

1. Abra el archivo de configuración.
2. Para cada propiedad requisito previo que desee editar, introduzca el nuevo valor esperado o cambie el valor del calificador. Por ejemplo, un nuevo administrador del sistema es el usuario root, por lo tanto, el valor de la propiedad de requisito previo `user.userID` debe cambiar. Cambie el valor por el nuevo nombre:

   ```
   user.userID=smithj
   ```

Por ejemplo, el calificador `type` de la propiedad de requisito previo `os.ulimit` tiene actualmente un valor de `filedescriptorlimit` para comprobar el límite de los descriptores de archivos. Es posible que desee comprobar otro límite, como el tamaño de pila, por ejemplo. Cambie el siguiente valor del calificador de la propiedad de requisito previo de:

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

a:

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

Importante: Solo puede utilizar los calificadores predefinidos con determinadas propiedades de requisitos previos predefinidas, como se indica en la Tabla 5 en la página 10.

Creación de recopiladores personalizados para sistemas Windows

Puede crear conectores personalizados si los recopiladores de conjuntos básicos no recopilan los valores de las propiedades de requisitos previos necesarias para instalar el producto. Puede crear recopiladores VBScript comunes personalizados para recopilar datos de propiedades de requisitos previos aplicables a cualquier producto o versión del producto. También puede crear sus recopiladores personalizados específicos de producto para recopilar los datos aplicables a un producto y versión del producto específicos. Si bien cada tipo de recopilador VBScript personalizado recopila datos utilizando los mismos métodos, las reglas de creación, almacenamiento y ejecución son ligeramente distintas.

Creación de recopiladores VBScript personalizado comunes a todos los archivos de configuración

Cuando cree recopiladores VBScript comunes personalizados, el nombre de archivo debe contener el nombre de la propiedad de requisito previo y estar guardarse en el subdirectorio `/lib`. El recopilador contiene código para obtener el valor real de una propiedad de requisito previo. También puede utilizar las funciones comunes y subrutinas para obtener el valor si es necesario.

Antes de empezar

Asegúrese de revisar el conjunto de funciones predefinidas y subrutinas de los siguientes apéndices antes de crear los recopiladores. Determine si puede utilizar alguno de ellos para obtener los valores reales:

- Apéndice E, “Funciones comunes para los sistemas Windows”, en la página 123
- Apéndice G, “Subrutinas de programa de utilidad de archivado para sistemas Windows”, en la página 139
- Apéndice F, “Subrutinas de programa de utilidad de registro para sistemas Windows”, en la página 137
- Apéndice H, “Otras funciones comunes y subrutinas para sistemas Windows”, en la página 141

Determine si el recopilador debe comprobar que la propiedad de requisito previo existe y en caso afirmativo, qué otra información debe recopilarse. Todas las comprobaciones deben devolver un valor, tanto si existe uno como si no. Por ejemplo:

- Compruebe si existe una variable de entorno, como el directorio de inicio de un producto; por ejemplo, `TCR_HOME` para Tivoli Common Reporting.

- Compruebe si la variable de entorno contiene un archivo JAR, binario, o ruta, como por ejemplo la ruta del JRE en la variable de entorno PATH.
- Compruebe el valor real de una variable de entorno, como el directorio de inicio de un producto; por ejemplo, TCR_HOME para Tivoli Common Reporting.
- Compruebe si se ha instalado un producto.
- Compruebe qué versión del producto está instalada.

Procedimiento

1. Cree un archivo VBScript. Guarde el archivo en el directorio *ips_root*/lib, con una variante de la siguiente convención de nomenclatura:

   [*prefix_identifier*.]property_name.vbs

   donde:

   - *prefix_identifier* es el identificador de prefijo de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4.
   - *property_name* es el nombre de la propiedad de requisito previo y se utiliza en el nombre del recopilador.

   Por ejemplo, mssqlVersion.vbs contiene el código para obtener el valor real de la propiedad de requisito previo del servidor MS SQL en la máquina Windows.
2. Con un editor de VBScript, agregue el código para obtener el valor de la propiedad de requisito previo. Utilice VBScript COM y funciones para acceder a los elementos del entorno Windows y ejecutar el entorno Windows Script Host. Asegúrese de que la comprobación devuelve la siguiente salida estándar:

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

   - *property_name* que representa la propiedad de requisito previo como está grabada en el archivo de configuración; por ejemplo, env.tcrhome.
   - *var_for_value*, es decir, la variable VBScript del valor real que el recopilador obtiene para la propiedad de requisito previo.

   Para comprobar si el entorno existe TCR_HOME y devolver el valor real, donde el nombre de propiedad de requisito previo es env.tcrhome:

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

   Para comprobar si se ha establecido el JRE en la variable PATH, donde el nombre de propiedad de requisito previo es 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)
   ```

   Para comprobar la versión instalada de Tivoli Directory Integrator, donde el nombre de propiedad de requisito previo es 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. Ejecute el recopilador VBScript para asegurarse de que no hay errores de ejecución y depurar si es necesario.
4. Cree un evaluador personalizado solo si las funciones de comparación estándar no pueden comparar los valores reales y esperados.

Creación de recopiladores VBScript personalizados específicos de un producto y una versión del producto

Cuando cree recopiladores VBScript específicos de producto personalizados, el nombre de archivo debe ser el mismo código de producto que el archivo de configuración y guardarse en el subdirectorio `/Windows`. El recopilador puede contener código para recopilar valores reales para una o muchas propiedades de requisitos previos. También puede utilizar las funciones comunes y subrutinas para recopilar esos valores si es necesario.

Antes de empezar

Asegúrese de revisar el conjunto de funciones y subrutinas de los siguientes apéndices antes de crear los recopiladores. Determine si puede utilizar alguno de ellos para obtener los valores reales:

- Apéndice E, “Funciones comunes para los sistemas Windows”, en la página 123
- Apéndice G, “Subrutinas de programa de utilidad de archivado para sistemas Windows”, en la página 139
- Apéndice F, “Subrutinas de programa de utilidad de registro para sistemas Windows”, en la página 137
- Apéndice H, “Otras funciones comunes y subrutinas para sistemas Windows”, en la página 141

Determine si el recopilador debe comprobar que la propiedad de requisito previo existe y en caso afirmativo, qué otra información debe recopilarse. Todas las comprobaciones deben devolver un valor, si existe uno. Por ejemplo:

- Compruebe si el directorio existe.
- Compruebe el espacio de disco disponible para un directorio.
- Compruebe si se ha instalado un producto.
- Compruebe qué versión del producto está instalada.

Procedimiento

1. Cree un archivo VBScript. Guarde el archivo en el directorio *ips_root*`/Windows`, con una variante de la siguiente convención de nomenclatura:

product_code[*_version*].vbs

donde:

- *product_code*

  Es la variable para representar un código de producto en sistemas Windows o UNIX. Los códigos de producto identifican el producto, una plataforma individual, como Windows, AIX, HP-UX, Linux y Solaris, y opcionalmente la versión del sistema operativo compatible con ese producto. Se guardan en el archivo `codename.cfg`. Los productos que admiten varias plataformas tienen varios códigos de producto. Cada uno de ellos identifica un producto, una plataforma y una versión del sistema operativo, según se requiera.
- *version* es el código de 8 dígitos para representar la versión, release, modificación y nivel con dos dígitos para cada parte del código; por ejemplo, 7.3.21 es 07032100.

2. Con un editor VBScript, abra el archivo e incluya la ruta de acceso a `common_function.vbs` si tiene que utilizar las funciones más comunes, del siguiente modo:

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

3. Si debe utilizar los valores de los distintivos `PATH` y `-p` pasados desde Prerequisite Scanner, utilice `Wscript.Arguments()` donde `Wscript.Arguments(0)` es el valor de `PATH`. `Wscript.Arguments(1)` es el distintivo `-p` y sus valores.
4. Añada el código para obtener el valor de la propiedad de requisito previo mediante el uso de VBScript COM y funciones para acceder a los elementos del entorno Windows. Realice la ejecución en el entorno Windows Script Host. Asegúrese de que la comprobación devuelve la siguiente salida estándar:

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

   - *property_name* que representa la propiedad de requisito previo como está grabada en el archivo de configuración; por ejemplo, `env.tcrhome`.
   - *var_for_value*, es decir, la variable VBScript del valor real que el recopilador obtiene para la propiedad de requisito previo.

   Para comprobar el espacio de disco disponible para el directorio de instalación de un producto. Por ejemplo, para comprobar Tivoli Monitoring for Energy Management Reporting and Optimization utilizando la subrutina “getValue()” en la página 142, donde la propiedad de requisito previo es `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. Cree un archivo por lotes para llamar al recopilador VBScript. El archivo por lotes debe tener el mismo nombre que el archivo de configuración y una extensión `.bat`, *product_code*[*_version*]`.bat`, del siguiente modo:

```
@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. Ejecute el recopilador VBScript para asegurarse de que no hay errores de ejecución y depurar si es necesario.
7. Cree un evaluador personalizado solo si las funciones de comparación estándar no pueden comparar los valores reales y esperados.

Creación de recopiladores personalizados para sistemas UNIX

Puede crear conectores personalizados si los recopiladores de conjuntos básicos no recopilan los valores de las propiedades de requisitos previos necesarias para instalar el producto. Cuando cree recopiladores personalizados, el nombre de archivo debe ser el mismo que la propiedad de requisito previo aunque si el subtipo en su nombre. El recopilador se guarda en el subdirectorio `/UNIX_Linux`. El recopilador puede contener código para obtener valores reales para una o muchas propiedades de requisitos previos. También pueden utilizar las funciones comunes para obtener esos valores si es necesario.

Antes de empezar

Asegúrese de revisar el conjunto de funciones de los siguientes apéndices antes de crear los recopiladores. Determine si puede utilizar alguno de ellos para obtener los valores reales:

- Apéndice I, “Funciones comunes de los sistemas UNIX”, en la página 145
- Apéndice J, “Otras funciones de los sistemas UNIX”, en la página 153
- Apéndice K, “Funciones de utilidad de registro para sistemas UNIX”, en la página 161

Determine si el recopilador debe comprobar que la propiedad de requisito previo existe y en caso afirmativo, qué otra información debe recopilarse. Todas las comprobaciones deben devolver un valor, si existe uno. Por ejemplo:

- Compruebe si se ha instalado un producto; por ejemplo, un paquete instalado con RPM.
- Compruebe qué versión del producto está instalada.
- Compruebe si hay espacio en disco disponible para un sistema de archivos montado

Si desea utilizar subtipos, *suffix_identifier*, y categorizar aún más una propiedad de requisito previo por aplicación, programa de utilidad o subtipo, puede crear un recopilador común. Pase el diferenciador del subtipo *suffix_identifier*, es decir, *differentiator_suffix_identifier* a su recopilador. Por ejemplo, `os.package` es el recopilador común para comprobar la existencia de paquetes. Para comprobar la

existencia de openssh, pase el nombre del paquete cuando se invoque al recopilador os.package en el archivo de secuencia de comandos packageTest.sh , del siguiente modo:

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

Donde openssh es el nombre del paquete, es decir, el subtipo *suffix_identifier* y el diferenciador *differentiator_suffix_identifier*.

Procedimiento

1. Cree un archivo de secuencia de comandos shell. Guarde el archivo en el directorio *ips_root* / Unix_Linux , con una variante de la siguiente convención de nomenclatura, pero sin una extensión de archivo:

   [*prefix_identifier*.]*property_name*

   donde:

   - *prefix_identifier* es un identificador de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4. Algunas de las categorías predefinidas requieren este identificador de prefijo; por ejemplo, env.
   - *property_name* es el nombre de la propiedad de requisito previo; por ejemplo, path.jre.
2. Con un editor, abra el archivo e incluya la ruta de acceso a common_function.sh si tiene que utilizar las funciones más comunes, del siguiente modo:

   ```
   . ../lib/common_function.sh
   ```
3. Añada el código para obtener el valor de la propiedad de requisito previo mediante el uso de comandos y opciones específicas de esa plataforma para acceder a los elementos del entorno de sistema principal. Por ejemplo, la propiedad de requisito previo personalizada env.path.jar necesita comprobar si se ha establecido el JRE en la variable PATH. El siguiente código ejecuta el comando env, busca la salida de la variable PATH y después busca el valor de la ruta del JRE.

   ```
   envJRE=`env | grep "PATH" | grep -w "/opt/IBM/Java60/jre/bin"`
   ```
4. Asegúrese de que la comprobación devuelve la salida estándar:

   ```
   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
   ```

   En el ejemplo, en función del valor de la variable $envJRE, la comprobación devuelve True o False:

   ```
   if [ $envJRE ]; then
    echo "True"
   else
    echo "False"
   fi
   ```
5. Ejecute el recopilador personalizado para asegurarse de que no hay errores de ejecución y depurar si es necesario.
6. Edite la secuencia de comandos *ips_root*/UNIX_Linux/packageTest.sh para llamar al recopilador personalizado y ejecutarlo.
7. Cree un evaluador personalizado solo si el recopilador personalizado devuelve valores que no son booleanos.

Edición de la secuencia de comandos de prueba de paquete para sistemas UNIX

Puede actualizar el archivo de secuencia de comandos `packageTest.sh` para llamar a los recopiladores personalizados en sistemas UNIX.

Antes de empezar

Asegúrese de que conoce los nombres de los recopiladores asociados con las propiedades de requisitos previos predefinidas, como se indica en Apéndice D, "Recopiladores predefinidos para sistemas UNIX", en la página 117. Si la propiedad de requisito previo se categoriza aún más por la aplicación, el programa de utilidad o el subtipo de servicio, pase el diferenciador del subtipo *suffix_identifier*, es decir, *differentiator_suffix_identifier* a su recopilador.

Por ejemplo, `os.package` es el recopilador común para comprobar la existencia de paquetes. Para comprobar la existencia de `openssh`, pase el nombre del paquete cuando se invoque al recopilador `os.package` en el archivo de secuencia de comandos `packageTest.sh`, del siguiente modo:

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

Donde `openssh` es el nombre del paquete, es decir, el subtipo *suffix_identifier* y el diferenciador *differentiator_suffix_identifier*.

Procedimiento

1. Con un editor, abra la secuencia de comandos *ips_root*`/UNIX_Linux/packageTest.sh`.
2. Añada el código para leer la propiedad de requisito previo personalizada del archivo de configuración y analice su valor.

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

   Por ejemplo, para leer la propiedad de requisito previo personalizada `env.path.jar` y comprobar si se ha establecido el JRE en la variable `PATH`.

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

   En el ejemplo:

   ```
    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. Llame a las funciones de registro de datos de seguimiento y depuración antes que al recopilador personalizado.

   ```
    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. Llame al recopilador personalizado.

Nota: Si el recopilador personalizado tiene subtipos, es decir, [*suffix_identifier*] en el nombre del archivo y necesita controles adicionales en función del subtipo, pase el diferenciador del subtipo [*differentiator_suffix_identifier*] al recopilador personalizado.

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

En el ejemplo:

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

Nota: Los ejemplos de diferenciadores del subtipo *script_name* para las propiedades de requisitos previos `os.file.script_name` son las rutas a las secuencias de comandos que se pasan al recopilador `os.filepath`:

```
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. Llame a las funciones de registro de datos de seguimiento y depuración al salir del recopilador personalizado.

```
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
```

En el ejemplo:

```
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. Repita los pasos 2 a 5 con cada propiedad de requisito previo personalizada.

Creación de evaluadores personalizados para sistemas Windows

Puede crear evaluadores VBScript si los evaluadores básicos no comparan los valores esperados y reales de las propiedades de requisitos previos utilizando los criterios de evaluación correctos. Cuando cree evaluadores personalizados, el nombre de archivo debe terminar en `_compare` y almacenarse en el subdirectorio `/Windows`. El evaluador personalizado puede utilizar las funciones comunes y subrutinas para comparar los valores si resulta necesario.

Antes de empezar

Asegúrese de revisar el conjunto de funciones y subrutinas de los siguientes apéndices antes de crear el evaluador. Determine si puede utilizar alguno de ellos para comparar los valores:

- Apéndice E, "Funciones comunes para los sistemas Windows", en la página 123
- Apéndice G, "Subrutinas de programa de utilidad de archivado para sistemas Windows", en la página 139
- Apéndice F, "Subrutinas de programa de utilidad de registro para sistemas Windows", en la página 137

- Apéndice H, “Otras funciones comunes y subrutinas para sistemas Windows”, en la página 141

Nota: La función común, “passOrFail()” en la página 133, puede comparar los valores reales y esperados de los siguientes tipos de datos: un número genérico, el tamaño en MB o GB, la velocidad de procesador en MHz o GHz, un valor booleano o una cadena. Cree un evaluador personalizado si la función `passOrFail` no se puede utilizar.

Procedimiento

1. Cree un archivo VBScript. Guarde el archivo en el directorio *ips_root*`/Windows`, con una variante de la siguiente convención de nomenclatura:

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

   donde:

   - *prefix_identifier* es un identificador de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4. Algunas de las categorías predefinidas requieren este identificador de prefijo.
   - *property_name* es el nombre de la propiedad de requisito previo.
   - *suffix_identifier* es un identificador opcional de un subtipo de las propiedades de requisitos previos como se indica en la Tabla 4 en la página 7.
2. Añada el código para comparar los valores reales y esperados que se pasan al evaluador como argumentos utilizando VBScript COM y funciones asociadas. Asegúrese de que la comparación devuelve la siguiente salida estándar:
   - `"PASS"` cuando el valor esperado de la propiedad de requisito previo es igual o mayor que el valor real de la propiedad de requisito previo
   - `"FAIL"` cuando el valor esperado de la propiedad de requisito previo no es igual que el valor real de la propiedad de requisito previo
3. Ejecute el evaluador personalizado para asegurarse de que no hay errores de ejecución y depurar si es necesario.

Ejemplo

Este evaluador personalizado comprueba los valores reales y esperados de la versión de Tivoli Directory Integrator. Utiliza la función común, “versionCompare()” en la página 143.

```
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
```

Creación de evaluadores personalizados para sistemas UNIX

Puede crear evaluadores personalizados si el recopilador personalizado no devuelve valores booleanos, esto es, `True` o `False`. Cuando cree evaluadores personalizados, el nombre de archivo debe terminar en `_compare` y almacenarse en el subdirectorio `/UNIX_Linux`. El evaluador personalizado puede utilizar las funciones comunes para comparar los valores si resulta necesario.

Antes de empezar

Asegúrese de revisar el conjunto de funciones de los siguientes apéndices antes de crear los evaluadores personalizados. Determine si puede utilizar alguno de ellos para comparar los valores reales y esperados:

- Apéndice I, “Funciones comunes de los sistemas UNIX”, en la página 145
- Apéndice J, “Otras funciones de los sistemas UNIX”, en la página 153
- Apéndice K, “Funciones de utilidad de registro para sistemas UNIX”, en la página 161

Hay dos archivos de secuencia de comandos que puede utilizar como punto de partida, esto es, `._compare.sh` y `_compare.sh` en el subdirectorio `/Unix_Linux`.

Importante: No cree evaluadores personalizados si los recopiladores personalizados devuelven `True` o `False`. IBM Prerequisite Scanner utiliza evaluadores predefinidos para cualquier recopilador que devuelve valores booleanos.

Procedimiento

1. Cree un archivo shell. Guarde el archivo en el directorio *ips_root*`/UNIX_Linux`, con una variante de la siguiente convención de nomenclatura:

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

   donde:

   - *prefix_identifier* es un identificador de una categoría predefinida de las propiedades de requisitos previos como se indica en la Tabla 3 en la página 4. Algunas de las categorías predefinidas requieren este identificador de prefijo.
   - *property_name* es el nombre de la propiedad de requisito previo.
   - *suffix_identifier* es un identificador opcional de un subtipo de las propiedades de requisitos previos como se indica en la Tabla 4 en la página 7.
2. Añada el código para comparar los valores reales y esperados que se pasan al evaluador como argumentos y funciones asociadas. Asegúrese de que la comparación devuelve la siguiente salida estándar:
   - `"PASS"` cuando el valor esperado de la propiedad de requisito previo es igual o mayor que el valor real de la propiedad de requisito previo
   - `"FAIL"` cuando el valor esperado de la propiedad de requisito previo no es igual que el valor real de la propiedad de requisito previo
3. Ejecute el evaluador personalizado para asegurarse de que no hay errores de ejecución y depurar si es necesario.

Capítulo 4. Ejecución de Prerequisite Scanner

Puede utilizar una interfaz de línea de comandos para ejecutar IBM Prerequisite Scanner. La secuencia de comandos Prerequisite Scanner, `prereq_checker`, adopta un conjunto de parámetros necesarios y opcionales y un distintivo de comando con los parámetros opcionales.

Tabla 12 explica los caracteres especiales que se utilizan en la sintaxis de la secuencia de comandos Prerequisite Scanner.

Tabla 12. Leyenda de caracteres especiales para la secuencia de comandos Prerequisite Scanner

Carácter especial	Descripción
<>	Identifica un nombre de marcador de posición.
[]	Identifica un parámetro opcional. Son obligatorios parámetros no incluidos entre corchetes.
...	Indica que puede especificar varios valores para un parámetro.
|	Indica los parámetros que se excluyen mutuamente. Especifica el parámetro a la izquierda o a la derecha del separador, pero no ambos.
{}	Incluye un conjunto de parámetros mutuamente excluyentes separados por | .

prereq_checker

La secuencia de comandos `prereq_checker` ejecuta IBM Prerequisite Scanner y comprueba los requisitos previos en función del conjunto de parámetros que se especifique al ejecutar la secuencia.

Sintaxis

```
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]
```

La secuencia de comandos `prereq_checker` tiene un parámetro necesario y varios parámetros opcionales.

""Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."" en la página 68
Parámetro obligatorio

"[detail]" en la página 68
Parámetro opcional

"[outputDir="ips_output_dir"]" en la página 71
Parámetro opcional

"[xmlResult]" en la página 71
Parámetro opcional

"[PATH="product_root"]" en la página 71

"[-p Product_Code.instance.parameter=value,...]" en la página 72
Distintivo opcional

"[debug]" en la página 72
Parámetro opcional

"[trace]" en la página 72
Parámetro opcional

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

Debe establecer como mínimo un parámetro **Product_Code** para identificar el producto o el componente para el que se ejecutará la comprobación de requisitos previos y el archivo de configuración asociado **Product_Code** es el código de producto que se establece en el archivo *ips_root*/codename.cfg.

Por ejemplo, KMS es el código de producto para Tivoli Enterprise Monitoring Server en el archivo product.cfg. Para ejecutar Scanner, introduzca la siguiente secuencia de comandos con el código de producto:

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

Si establece un parámetro **Product_Code** que no tiene el archivo de configuración correspondiente, Prerequisite Scanner lo ignora sin errores. El archivo de registro contiene un mensaje de que no se ha encontrado ningún archivo de configuración.

El parámetro **Product_Version** del parámetro **Product_Code** asociado indica la versión del producto. Es el código de 8 dígitos para representar la versión, release, modificación y nivel con dos dígitos para cada parte del código; por ejemplo, 7.3.21 es 07032100. **Product_Version** es un parámetro opcional. Si no lo establece, Prerequisite Scanner comprueba la última versión disponible.

Puede establecer uno o muchos parámetros **Product_Code** con el parámetro opcional **Product_Version**, cada uno separado por una coma.

Importante: Cuando establezca más de un parámetro **Product_Code** con el parámetro opcional **Product_Version**, incluya los parámetros entre comillas. Si no lo hace así, Scanner fallará.

En este ejemplo se comprueban los requisitos previos de la última versión de Tivoli Monitoring Operating System Agent para Windows y la versión 6.2.1 de Tivoli Monitoring Agent para DB2 .

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

[detail]

Este parámetro opcional indica si se muestran los resultados detallados de la exploración en la interfaz de línea de comandos.

Importante: No incluya este parámetro entre comillas.

Cuando se establece el parámetro **detail**, los resultados detallados incluyen:

- La versión de Prerequisite Scanner
- La versión del sistema operativo en el que se ejecutó Scanner
- El nombre de los productos o componentes para los que se ejecutaron las comprobaciones de los requisitos previos
- Para cada propiedad de requisito previo: el nombre de la propiedad de requisito previo comprobado, el resultado `PASS` o `FAIL`, el valor real y el valor esperado
- Para todos los componentes: el nombre de la propiedad de requisito previo general, el resultado `PASS` o `FAIL`, el valor real y el valor esperado
- El resultado general `PASS` o `FAIL`

Prerequisite Scanner también guarda estos resultados en el archivo *ips_output_dir*`/result.txt`. Guarda los resultados en el archivo de texto independientemente de si se establece el parámetro **`detail`**.

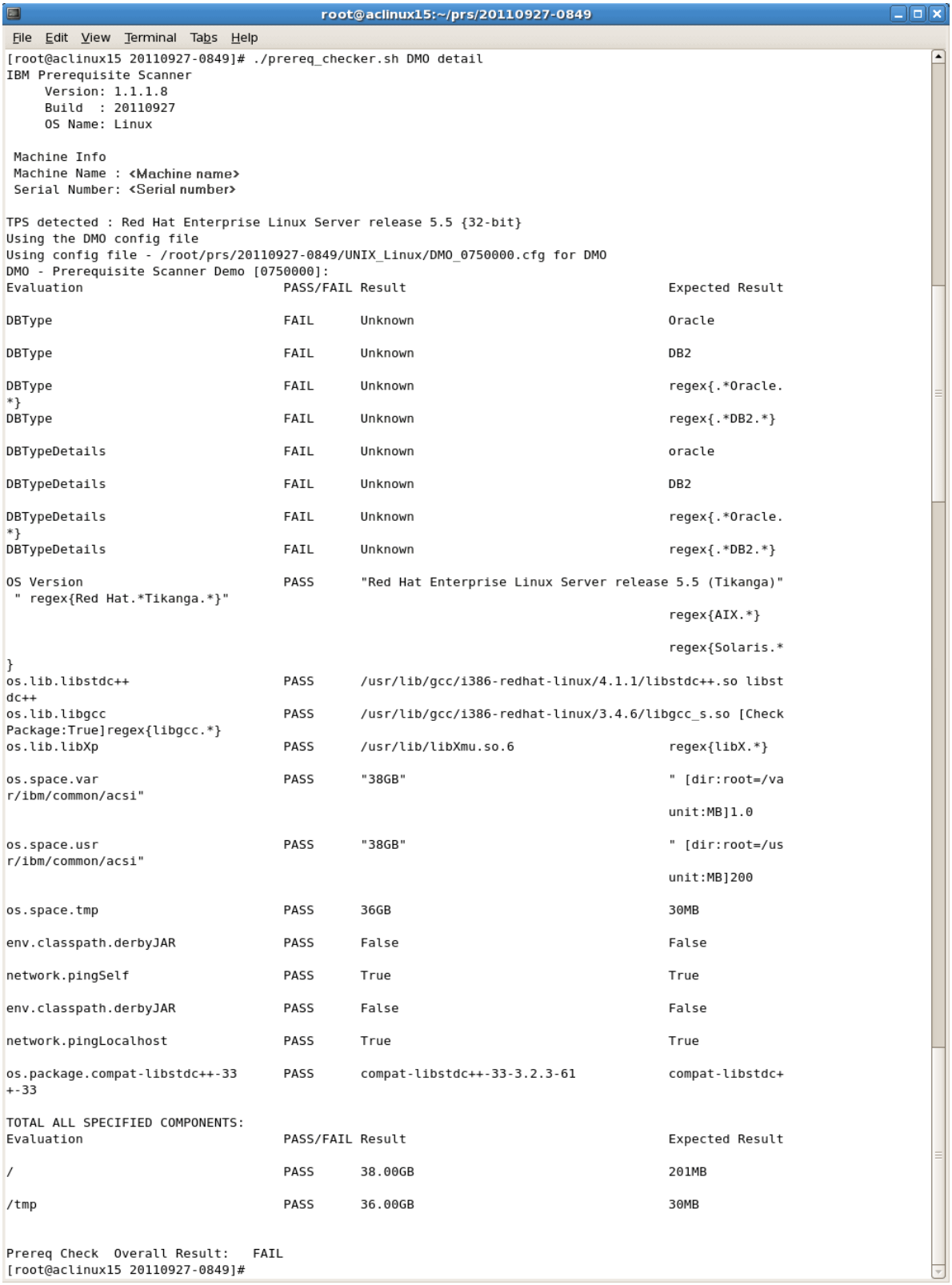


Figura 10. Ejecución de la secuencia de comandos y establecimiento del parámetro de detalle en sistemas UNIX

Si no se establece el parámetro **detail**, Scanner muestra solo el resultado PASS o FAIL en la interfaz de línea de comandos.

Figura 11. Ejecución de la secuencia de comandos sin establecer el parámetro de detalle en sistemas Windows

[outputDir="*ips_output_dir*"]

Este parámetro opcional indica que desea establecer el directorio de salida para los resultados del análisis y los archivos de registro para Prerequisite Scanner .

Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se establece el parámetro opcional **outputDir**, Prerequisite Scanner coloca los archivos de texto de resultados, XML y de registro en el directorio especificado por el valor del parámetro. A lo largo de toda la documentación, se hace referencia a este valor como *ips_output_dir*.

Si no establece este parámetro, la ubicación de salida predeterminada es *ips_root* .

Debe utilizar el parámetro para especificar una ubicación si opta por ejecutar Prerequisite Scanner desde un CD, un DVD o una unidad de red de solo lectura. Debe tener permisos de escritura para grabar en *ips_output_dir*; de lo contrario, Prerequisite Scanner fallará.

Importante: Si el directorio de salida no existe, Prerequisite Scanner lo creará. Debe tener permisos de escritura para crear o grabar en el directorio de salida en el que Prerequisite Scanner guarda los archivos.

[xmlResult]

Este parámetro opcional indica que desea mostrar los resultados en el archivo de resultados XML, además de en el archivo de resultados de texto sin formato.

Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se establece el parámetro opcional **xmlResult**, Prerequisite Scanner incluye los resultados en el archivo *ips_output_dir*/result.xml.

Si no establece este parámetro, los resultados se muestran solo en el archivo de texto sin formato.

[PATH="*product_root*"]

Este parámetro opcional indica los directorios de instalación de los productos.

Importante: En Windows, no establezca la ruta solo a una letra de unidad, es decir, C:. Asegúrese de establecer una ruta válida.

Si no establece el parámetro **path**,Scanner comprueba los directorios de instalación predeterminados para los productos de IBM Tivoli:

- En sistemas **UNIX:** /opt/ibm/itm
- En sistemas **Windows:** C:\IBM\itm

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

El distintivo opcional **-p** indica que los parámetros de procedimiento deben pasarse a un archivo de secuencia de comandos para una comprobación de requisitos previos adicional.**<Product_Code>** es el código de producto. Solo se pasa a la secuencia de comandos cada conjunto de *instance.parameter=value*. Puede pasar varios conjuntos de parámetros, separados por una coma.

La secuencia de comandos a la que se pasan los parámetros está determinada por las siguientes opciones:

- Con un prefijo **Product_Code**, los parámetros se pasan a la secuencia de comandos con el **Product_Code** asociado
- Sin el prefijo **Product_Code**, los parámetros se pasan a los recopiladores comunes.

Ejemplo 1-p KUD.inst1.DB2_INST_OWNER=db2inst1, KUD.inst2.DB2_INST_OWNER=db2inst2 Este distintivo con parámetros pasa db2inst1.DB2_INST_OWNER=db2inst1 y db2inst2.DB2_INST_OWNER=db2inst2 al archivo de secuencia de comandos KUD.**Product_Version**.bat.

Ejemplo 2

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

Este distintivo con parámetros pasa SERVER=IP.PIPE://mymachine:1918 al recopilador común para comprobar los puertos.

Nota: Esta secuencia de comandos acepta los parámetros de **-p** como tacmd createNode .
Puede definir los parámetros SERVER, PROTOCOL, PORT, BACKUP y BSERVER en *ips_root*/lib/common_configuration. Prerequisite Scanner da prioridad a los parámetros pasados desde la interfaz de línea de comandos respecto a los parámetros establecidos en el archivo common_configuration.

[debug]

Este parámetro opcional indica que desea activar la depuración al ejecutar Prerequisite Scanner.

Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se establece el parámetro opcional **debug**, Prerequisite Scanner muestra información detallada de procesamiento, mensajes de error y de aviso y los resultados del análisis en el archivo de registro. Es el archivo *ips_output_dir*/prs.debug en sistemas UNIX y el archivo *ips_output_dir*/precheck.log en sistemas Windows.

Importante: La función de depuración de Scanner está desactivada de forma predeterminada.

[trace]

(Solo sistemas UNIX) Este parámetro opcional indica que desea activar el registro de seguimiento al ejecutar Prerequisite Scanner.

Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se establece el parámetro opcional **trace**, Prerequisite Scanner muestra información de seguimiento en el archivo *ips_output_dir*/prs.trc.

Importante: La función de seguimiento de Scanner está desactivada de forma predeterminada.

Ejecución de Prerequisite Scanner desde la línea de comandos

Puede ejecutar IBM Prerequisite Scanner desde la interfaz de línea de comandos e introducir los parámetros de entrada correspondientes a la secuencia de comandos.

Antes de empezar

Asegúrese de comprobar la documentación sobre la instalación del producto o las notas técnicas para conocer los pasos adicionales que debe realizar antes de ejecutar Prerequisite Scanner. Por ejemplo, puede que necesite ajustar la variable de entorno que indica a Prerequisite Scanner qué componentes o características se están instalando en el equipo cliente y, por lo tanto, qué requisitos previos deben comprobarse.

Procedimiento

1. Abra la interfaz de línea de comandos y abra el directorio *ips_root*.
2. Ejecute el archivo de secuencia de comandos Prerequisite Scanner, **prereq_checker**, como se indica a continuación:

   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,...]
   ```

   En el siguiente ejemplo se ejecuta Prerequisite Scanner para Autonomic Deployment Engine utilizando un archivo de configuración y su código de producto asociado ADE:

   ```
   ./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,...]
   ```

   En el siguiente ejemplo se ejecuta Prerequisite Scanner para Tivoli Provisioning Manager para Windows 2003 y 2008 utilizando los códigos de producto COX y COY.

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

   En el siguiente ejemplo se ejecuta Prerequisite Scanner para Tivoli zEnterprise Monitoring Agent utilizando el código de producto KZE. También establece la

ubicación de los resultados y los archivos de registro en *ips_output_dir* utilizando el parámetro opcional **outputDir**.

Importante: Debe utilizar el parámetro **outputDir** para especificar una ubicación si opta por ejecutar Prerequisite Scanner desde un CD, un DVD o una unidad de red de solo lectura. Debe tener permisos de escritura para grabar en *ips_output_dir*; de lo contrario, Prerequisite Scanner fallará.

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

Scanner incluye los archivos `result.txt file` y `precheck.log` en las siguientes ubicaciones:

- En sistemas Windows: `D:\temp\ips` donde `TEMP` es la variable de entorno de la carpeta temporal.
- En sistemas UNIX: `/tmp/ips`

Importante: Si el directorio de salida no existe, Prerequisite Scanner lo creará. Debe tener permisos de escritura para crear o grabar en el directorio de salida en el que Prerequisite Scanner guarda los archivos.

Ubicaciones de directorios comunes

Existen variables de nombre de ruta para los directorios comunes.

Directorio de instalación de IBM Prerequisite Scanner

ips_root describe la ubicación donde Prerequisite Scanner está instalado. Esta ubicación se puede especificar durante la instalación.

Directorio de salida de Prerequisite Scanner

ips_output_dir describe la ubicación donde se guardan los resultados del análisis y archivos de registro de Prerequisite Scanner. Esta ubicación se puede especificar con el parámetro de entrada **outputDir** cuando ejecute Scanner. Si no establece este parámetro, la ubicación de salida predeterminada es *ips_root* .

Nota: Prerequisite Scanner crea archivos temporales durante su ejecución, pero estos archivos se eliminan antes de que Scanner termine su ejecución. Estos archivos temporales se encuentran ubicados en el subdirectorio *ips_output_dir*/temp. Scanner también suprime el subdirectorio *ips_output_dir*/temp, a menos que el subdirectorio contenga los archivos de depuración y de rastreo que se han generado únicamente en sistemas UNIX.

Capítulo 5. Solución de problemas de Prerequisite Scanner

Puede resolver problemas de IBM Prerequisite Scanner mediante archivos de registro y funciones de registro cuando cree comprobaciones de requisitos previos personalizadas.

Prerequisite Scanner genera códigos de retorno en función de los resultados del análisis y si debe salir debido a errores. Estos códigos de retorno se graban en los archivos de registro. Por ejemplo, si Prerequisite Scanner no puede ejecutar el análisis porque no puede leer el archivo de configuración, genera el código de retorno 2.

Solución de problemas en sistemas Windows

Cuando se ejecuta IBM Prerequisite Scanner, crea un archivo de registro de forma predeterminada. Contiene información detallada de cada paso y función que Scanner realiza en secuencia. El archivo también contiene indicaciones de fecha y hora, incluidas las horas de inicio y finalización de cada función y paso. Puede depurar y revisar el archivo de registro para determinar dónde y cuándo se produjo el error.

Prerequisite Scanner muestra información de procesamiento, mensajes de error y de aviso y los resultados del análisis en el archivo *ips_output_dir*/`precheck.log`. Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se define el parámetro opcional **`debug`**, Prerequisite Scanner muestra mensajes adicionales de depuración en este archivo.

Figura 12 en la página 76 muestra un ejemplo del archivo de registro cuando el parámetro opcional **`debug`** se establece y Figura 13 en la página 77 muestra el archivo de registro cuando no se ha establecido el parámetro.

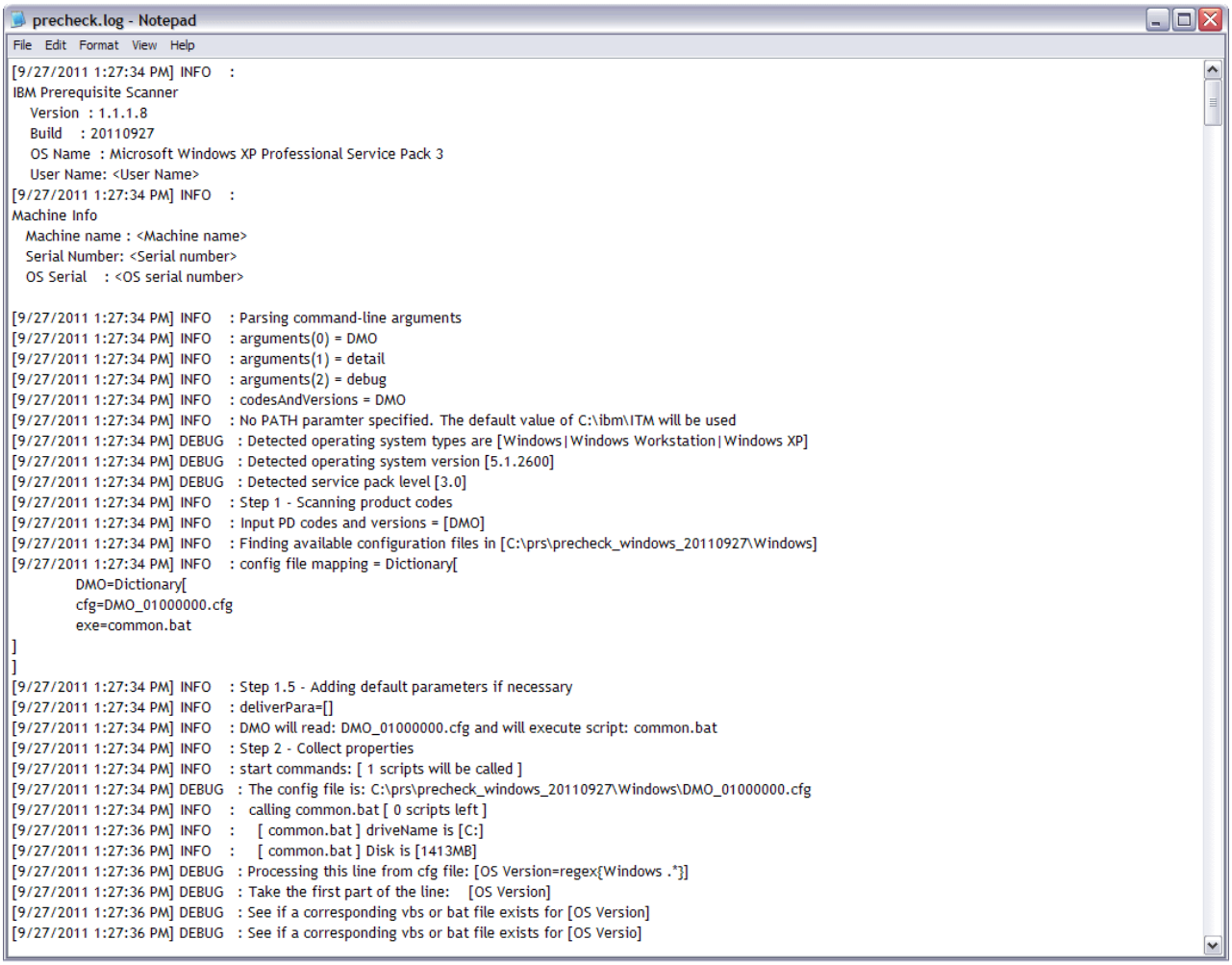


Figura 12. Archivo precheck.log con datos de depuración

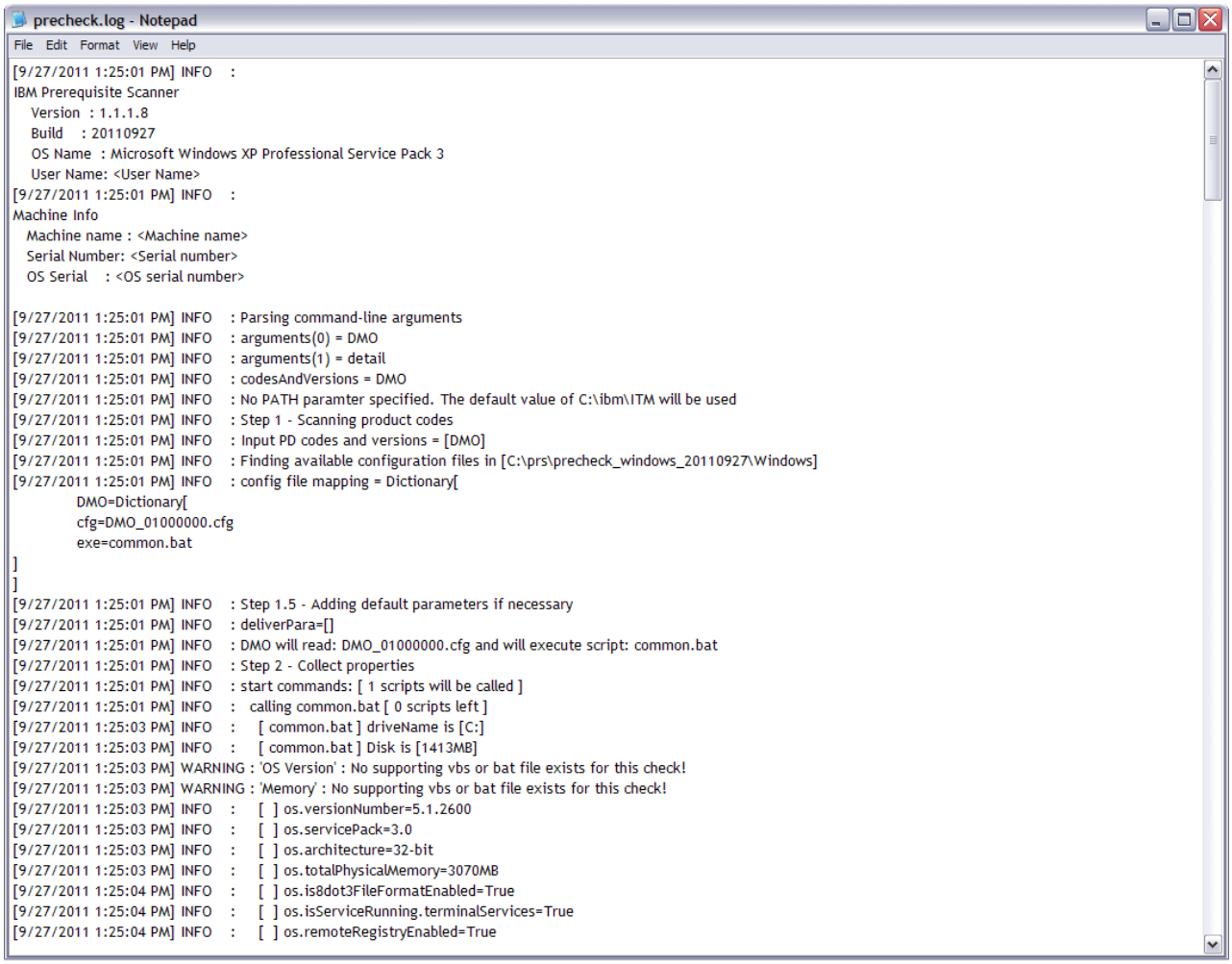


Figura 13. Archivo precheck.log sin datos de depuración

Solución de problemas en sistemas UNIX

La escritura de mensajes en archivo de registro está inhabilitada de forma predeterminada en sistemas UNIX. Para habilitar las funciones de depuración y de seguimiento, utilice los parámetros de entrada **debug** y **trace**. Scanner escribe los datos de depuración y de seguimiento en diferentes archivos de registro y utiliza indicaciones de fecha y hora para señalar las horas de inicio y finalización de los pasos o las funciones. Puede utilizar ambos archivos para correlacionar y solucionar un determinado problema, función, o comprobación de requisitos previos.

Archivo de registro de depuración

Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se establece el parámetro opcional **debug**, Prerequisite Scanner muestra información de procesamiento, mensajes de error y de aviso y los resultados del análisis en el archivo *ips_output_dir*/temp/prs.debug.Contiene información detallada de cada paso y función que Scanner realiza en secuencia. El archivo también contiene indicaciones de fecha y hora, incluidas las horas de inicio y finalización de cada

función y paso. El subdirectorio *ips_output_dir*/temp también contiene los archivos provisionales result1.txt y result2.txt que proporcionan la entrada al archivo final *ips_output_dir*/result.txt. Puede utilizar estos archivos provisionales para determinar los problemas con los resultados de determinadas comprobaciones de requisitos previos.

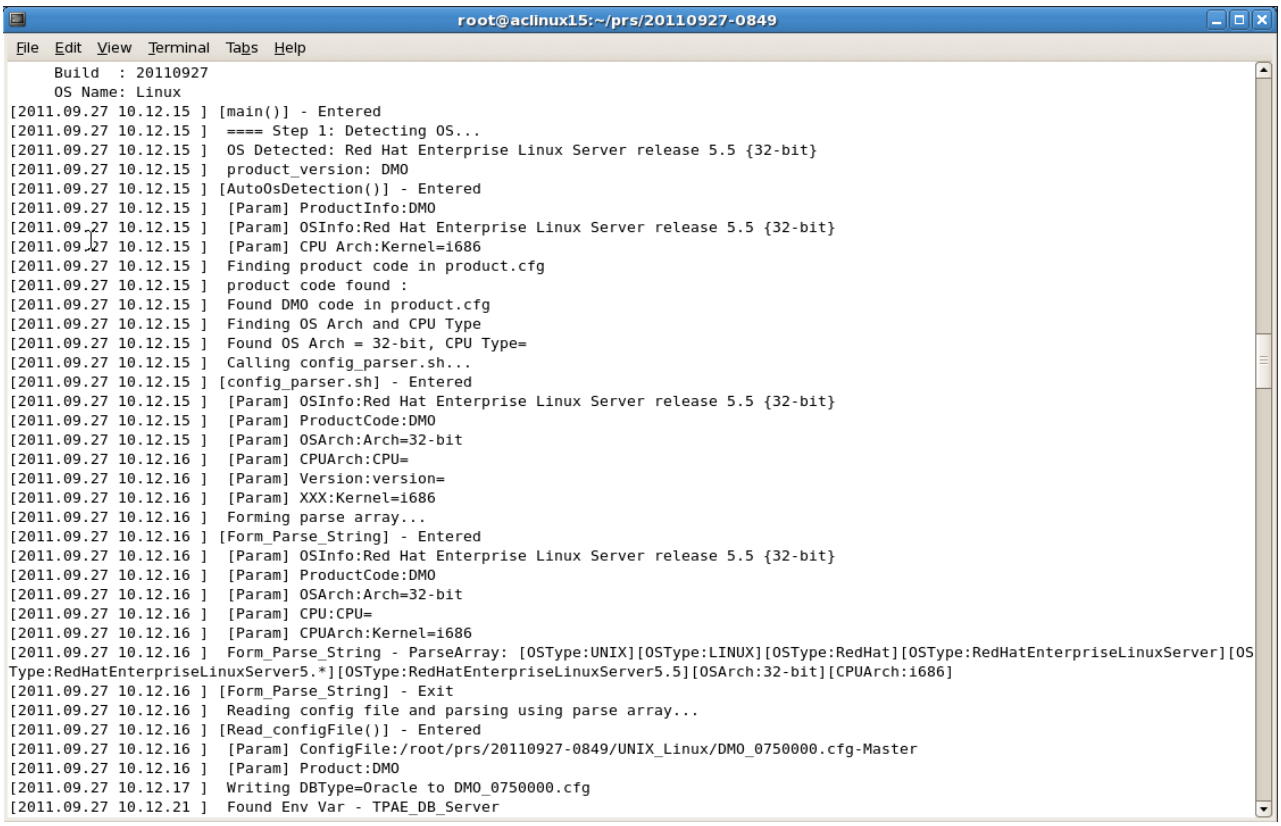


Figura 14. Archivo prs.debug en sistemas UNIX

Archivo de registro de seguimiento

Cuando se ejecuta la secuencia de comandos Prerequisite Scanner y se establece el parámetro opcional **trace**, Prerequisite Scanner muestra información de seguimiento en el archivo *ips_output_dir*/temp/prs.trc. Contiene información detallada de cada función que Scanner realiza en secuencia. El archivo también contiene indicaciones de fecha y hora, incluidas las horas de inicio y finalización de cada función.

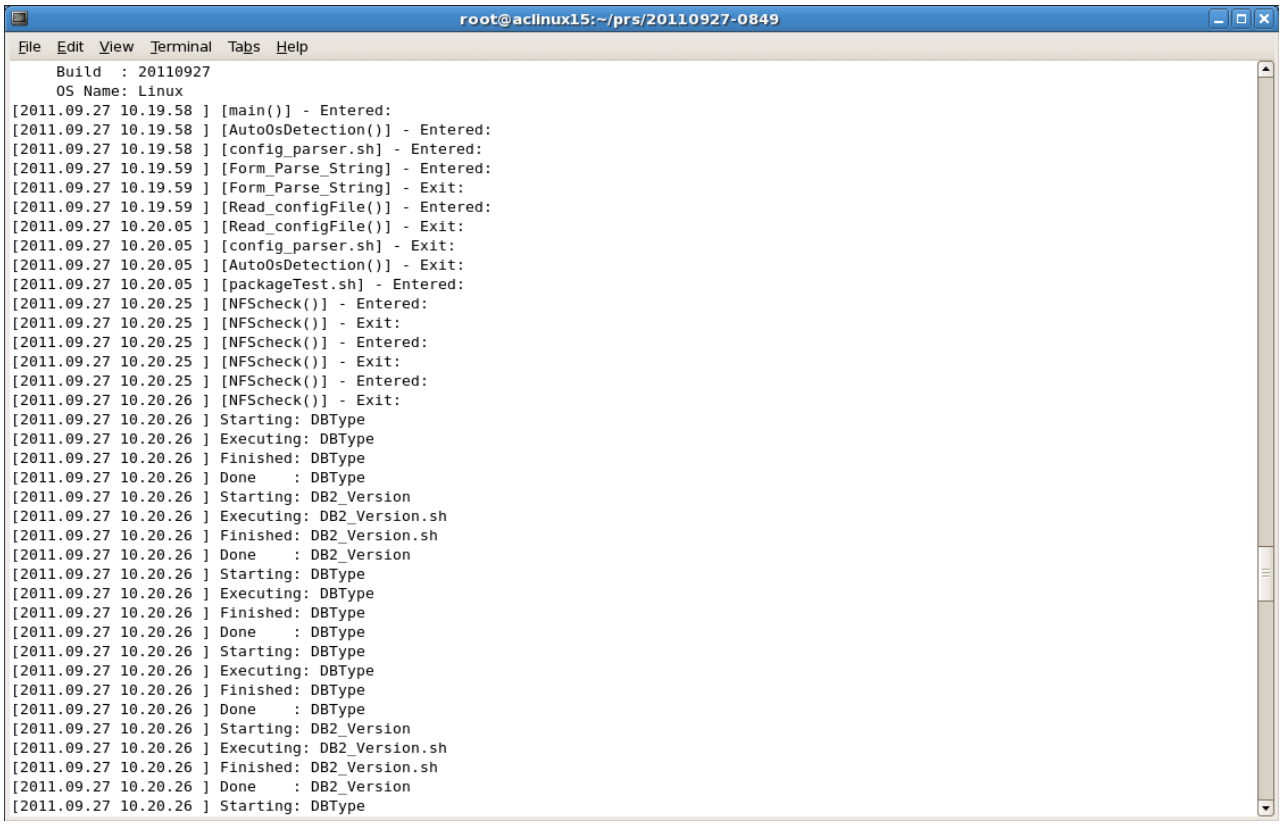


Figura 15. Archivo prs.trc en sistemas UNIX

Problemas relativos a la ejecución

Puede utilizar la lista de comprobación de problemas de ejecución para resolver los errores que puede encontrar al ejecutar Prerequisite Scanner.

Ejecute la secuencia de comandos Prerequisite Scanner con los parámetros de entrada opcionales **debug** y **trace** para ayudar en la depuración de problemas.

Tabla 13. Lista de comprobación de problemas de ejecución

Comprobación	Elemento
☐	Si se establece el parámetro opcional **outputDir** en la línea de comandos y el directorio de salida no existe, Prerequisite Scanner crea el directorio. Debe tener permisos de escritura para crear o grabar en el directorio de salida en el que Prerequisite Scanner guarda los archivos. Si no tiene permisos de escritura, se escribe el mensaje de error siguiente en la interfaz de línea de comandos: `ERROR: Cannot create files in output directory ips_output_dir. Exit.`
☐	Antes de ejecutar Prerequisite Scanner, asegúrese de que el disco en el que desea ejecutar Prerequisite Scanner y guardar los resultados en el directorio de salida no está lleno; de lo contrario, el siguiente mensaje de error se escribe en la interfaz de línea de comandos: `ERROR: Cannot create files in output directory ips_output_dir. Exit.`

Tabla 13. Lista de comprobación de problemas de ejecución (continuación)

Comprobación	Elemento
☐	Si Prerequisite Scanner genera un código de retorno de 2, es posible que se produzca un error de uso de secuencia de comandos o de recopilador. Revise las causas asociadas con este código de error. Si se ha producido un error de uso de la secuencia de comandos, vuelva a ejecutar Prerequisite Scanner utilizando la sintaxis correcta.

Conceptos relacionados:

La escritura de mensajes en archivo de registro está inhabilitada de forma predeterminada en sistemas UNIX. Para habilitar las funciones de depuración y de seguimiento, utilice los parámetros de entrada **debug** y **trace**. Scanner escribe los datos de depuración y de seguimiento en diferentes archivos de registro y utiliza indicaciones de fecha y hora para señalar las horas de inicio y finalización de los pasos o las funciones. Puede utilizar ambos archivos para correlacionar y solucionar un determinado problema, función, o comprobación de requisitos previos.

Prerequisite Scanner genera códigos de retorno en función de los resultados del análisis y si debe salir debido a errores. Estos códigos de retorno se graban en los archivos de registro.

La secuencia de comandos `prereq_checker` ejecuta IBM Prerequisite Scanner y comprueba los requisitos previos en función del conjunto de parámetros que se especifique al ejecutar la secuencia.

Códigos de retorno

Prerequisite Scanner genera códigos de retorno en función de los resultados del análisis y si debe salir debido a errores. Estos códigos de retorno se graban en los archivos de registro.

Prerequisite Scanner genera códigos de retorno en función de un conjunto de resultados definidos, como se indica a continuación:

Código de retorno	Descripción
0	Devuelve 0 cuando Prerequisite Scanner se ejecuta correctamente y todos los resultados de la exploración son PASS.
1	Devuelve 1 cuando Prerequisite Scanner se ejecuta correctamente, pero una o muchas comprobaciones de requisitos previos devuelven FAIL.
2	Devuelve 2 cuando Prerequisite Scanner no se ejecuta correctamente y debe cerrarse debido a un error clasificado del siguiente modo: • Errores de uso de secuencias de comandos • Errores de recopilador • Otros errores

Errores de uso de secuencias de comandos

Prerequisite Scanner puede cerrarse debido a cualquiera de los siguientes errores de uso cuando se ejecuta la secuencia de comandos:

- El parámetro de entrada **Product_Code** no es válido; por ejemplo, no se ha encontrado o no tiene un formato compatible.
- El patrón de los parámetros de entrada **Product_Code** y **Product_Version** no es válido; por ejemplo, se han incluido varios códigos y versiones entre comillas o el patrón no se encuentra incluido entre comillas.
- Los parámetros de entrada **Product_Version** no son válidos; por ejemplo, la versión de producto no tiene exclusivamente caracteres numéricos.
- No se ha introducido ningún parámetro de entrada en la interfaz de línea de comandos.
- La sintaxis era incorrecta cuando se introdujo en la interfaz de línea de comandos; por ejemplo, se ha introducido un argumento de línea de mandatos no admitido.
- No se ha introducido ningún parámetro de entrada **Product_Code** necesario.

Errores de recopilador

Prerequisite Scanner puede cerrarse debido a alguno de los siguientes errores de recopilador:

- El archivo de resultados temporal del recopilador no se ha encontrado en el directorio *ips_output_dir*/temp.
- El archivo de secuencia de comandos del recopilador no se ha ejecutado correctamente.

Otros errores

Prerequisite Scanner puede cerrarse porque el usuario no tiene permiso de escritura al directorio de salida *ips_output_dir*.

Conceptos relacionados:

IBM Prerequisite Scanner genera una salida para la siguiente pantalla y formatos de archivo legibles para los humanos: salida en la interfaz de línea de comandos, archivos de registro de depuración y seguimiento y archivos XML y de texto para los resultados.

Apéndice A. Referencia de códigos de producto

IBM Prerequisite Scanner utiliza un código de varios caracteres, *product_code*, para identificar el producto, la plataforma individual compatible y la versión de sistema operativo. El archivo *ips_root*/codename.cfg contiene los pares de nombre-valor para representar el código de producto del producto, su plataforma compatible y la versión del sistema operativo.

Tabla 14 describe el conjunto actual de los códigos de producto predefinidos.

Restricción: IBM Tivoli Monitoring y Tivoli Composite Application Manager tienen códigos predefinidos de productos que Prerequisite Scanner considera como reservados. Estos códigos no deben utilizarse como códigos de producto de Prerequisite Scanner a menos que se refieran a sus agentes asociados IBM Tivoli Monitoring y Tivoli Composite Application Manager. Para obtener más información sobre los códigos de producto, consulte el apartado ITM 6.X Product Codes Technote.

Restricción: Sólo en UNIX: al introducir el valor para el código de producto en el archivo, evite el uso de `for`. Es una palabra reservada y puede afectar a la ejecución de Prerequisite Scanner.

Tabla 14. Códigos predefinidos de productos

Código predefinido de producto	Plataforma	Versión del producto, plataforma, sistema operativo
ADE	Todas	Autonomic Deployment Engine
BSM	Todas	Tivoli Business Service Manager
CDB	Todas	Tivoli Composite Application Manager (ITCAM) for Applications: DB2
COA	UNIX	Tivoli Provisioning Manager para UNIX
COB	AIX	Tivoli Provisioning Manager para AIX
COC	AIX	Tivoli Provisioning Manager para AIX V5.3.0.0 {64 bits}
COD	AIX	Tivoli Provisioning Manager para AIX 6.1
COE	Linux	Tivoli Provisioning Manager para Linux
COF	Linux	Tivoli Provisioning Manager para Red Hat Linux
COG	Linux	Tivoli Provisioning Manager Versión 7.2 para Red Hat Enterprise Linux 5 x86 64 bits
COH	Linux	Tivoli Provisioning Manager para Red Hat Enterprise Linux 5 System z 64 bits
COI	Linux	Tivoli Provisioning Manager para SUSE 10
COJ	Solaris	Tivoli Provisioning Manager Versión 7.2 para Solaris
COK	HP-UX	Tivoli Provisioning Manager Versión 7.2 para HP-UX
COL	Linux	Tivoli Provisioning Manager Versión 7.2 para SUSE zSeries 10
COM	Linux	Tivoli Provisioning Manager Versión 7.2 para SUSE 11

Tabla 14. Códigos predefinidos de productos (continuación)

Código predefinido de producto	Plataforma	Versión del producto, plataforma, sistema operativo
CON	Linux	Tivoli Provisioning Manager Versión 7.2 para SUSE zSeries 11
COX	Windows	Tivoli Provisioning Manager Versión 7.2 para Windows 2008
COY	Windows	Tivoli Provisioning Manager Versión 7.2 para Windows 2003
COZ	Windows	Tivoli Provisioning Manager Versión 7.2 para Windows
DMO	Todas	Demo de Prerequisite Scanner
GYM	UNIX	IBM Tivoli Netcool Performance Manager
KCJ	Windows	Tivoli Enterprise Portal Client
	UNIX	Tivoli Enterprise Portal Client para UNIX
KCQ	Windows	Tivoli Enterprise Portal Server
	UNIX	Tivoli Enterprise Portal Server para UNIX
KHD	Todas	Warehouse Proxy Agent
KHE	UNIX	Warehouse Proxy Agent para UNIX
KIS	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Internet Service Monitoring
KLZ	UNIX	Tivoli Monitoring Operating System Agent para Linux
KM6	Windows	IBM Tivoli Composite Application Manager Agent for WebSphere MQ File Transfer Edition
KMQ	Todas	Tivoli Composite Application Manager Agent for WebSphere MQ
KMS	Windows	Tivoli Enterprise Monitoring Server
	UNIX	Tivoli Enterprise Monitoring Server para UNIX
KNT	Windows	Tivoli Monitoring Operating System Agent para Windows
	UNIX	Windows OS Monitoring Agent para UNIX
KOR	Windows	Tivoli Monitoring Agent para Oracle
KQI	Todas	Tivoli Composite Application Manager Agent para WebSphere Message Broker
KSY	Windows	Summarization and Pruning Agent
	UNIX	Summarization and Pruning Agent para UNIX
KUD	Windows	Tivoli Monitoring Agent para DB2
	UNIX	Tivoli Monitoring Agent para DB2
KTO	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Reporter
KTU	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Collector
KT3	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Application Management Console

Tabla 14. Códigos predefinidos de productos (continuación)

Código predefinido de producto	Plataforma	Versión del producto, plataforma, sistema operativo
KT4	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Client Response Time
KT5	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Web Response Time
KT6	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Robotic Response Time
KZE	Todas	Tivoli zEnterprise Monitoring Agent
LCM	Windows	Tivoli License Compliance Manager
	UNIX	Tivoli License Compliance Manager para UNIX
NCI	Todas	Tivoli Netcool/Impact
NOC	Todas	Componentes de servidor y componentes de escritorio de Tivoli Netcool/OMNIbus
NOD	Todas	Componente de escritorio de Tivoli Netcool/OMNIbus
NOS	Todas	Componentes de servidor de Tivoli Netcool/OMNIbus
PAE	Todas	Tivoli Process Automation Engine
TAD	Windows	Tivoli Asset Discovery for Distributed
	UNIX	Tivoli Asset Discovery for Distributed para UNIX
TCR	Todas	Tivoli Common Reporting
TPM	Todas	Tivoli Provisioning Manager

Apéndice B. Referencia de archivos de configuración

IBM Prerequisite Scanner ofrece un conjunto predefinido de archivos de configuración que se pueden editar. Estos archivos se encuentran en *ips_root*/UNIX_Linux o *ips_root*/Windows. Los archivos tienen la extensión .cfg.

La Tabla 15 incluye los archivos de configuración predefinidos admitidos actualmente.

Tabla 15. Archivos de configuración predefinidos

Archivo de configuración	Plataforma	Versión del producto, plataforma, sistema operativo
ADE_01040000.cfg	Todas	Autonomic Deployment Engine Versión 1.4
BSM_04210000.cfg	Todas	Tivoli Business Service Manager Versión 4.2.1
BSM_06100000.cfg	Todas	Tivoli Business Service Manager Versión 6.1
CDB_06220000.cfg	Todas	Tivoli Composite Application Manager (ITCAM) for Applications: DB2 Versión 6.2.2
COA_07200000.cfg	UNIX	Tivoli Provisioning Manager Versión 7.2 para UNIX
COB_07200000.cfg	AIX	Tivoli Provisioning Manager Versión 7.2 para AIX
COC_07200000.cfg	AIX	Tivoli Provisioning Manager Versión 7.2 para AIX V5.3.0.0 {64 bits}
COD_07200000.cfg	AIX	Tivoli Provisioning Manager Versión 7.2 para AIX 6.1
COE_07200000.cfg	Linux	Tivoli Provisioning Manager Versión 7.2 para Linux
COF_07200000.cfg	Linux	Tivoli Provisioning Manager Versión 7.2 para Red Hat Linux
COG_07200000.cfg	Linux	Tivoli Provisioning Manager Versión 7.2 para Red Hat Enterprise Linux 5 x86 64 bits
COH_07200000.cfg	Linux	Tivoli Provisioning Manager Versión 7.2 para Red Hat Enterprise Linux 5 System z 64 bits
COI_07200000.cfg	Linux	Tivoli Provisioning Manager Versión 7.2 para SUSE 10
COJ_07200000.cfg	Solaris	Tivoli Provisioning Manager Versión 7.2 para Solaris
COK_07200000.cfg	HP-UX	Tivoli Provisioning Manager Versión 7.2 para HP-UX
COL_07200000.cfg	Linux	Tivoli Provisioning Manager Versión 7.2 para SUSE zSeries 10
COM_07200000.cfg	Linux	Tivoli Provisioning Manager Versión 7.2 para SUSE 11
CON_07200000.cfg	Linux	Tivoli Provisioning Manager Versión 7.2 para SUSE zSeries 11
COX_07200000.cfg	Windows	Tivoli Provisioning Manager Versión 7.2 para Windows 2008
COY_07200000.cfg	Windows	Tivoli Provisioning Manager Versión 7.2 para Windows 2003
COZ_07200000.cfg	Windows	Tivoli Provisioning Manager Versión 7.2 para Windows
DMO_00000000.cfg	Todas	Demo de Prerequisite Scanner
DMO_01000000.cfg	Todas	Demo Prerequisite Scanner Versión 1.0
GYM_01030200.cfg	UNIX	IBM Tivoli Netcool Performance Manager Versión 1.3.2
KCJ_06200000.cfg	Windows	Tivoli Enterprise Portal Client Versión 6.2
KCJ_06210000.cfg	UNIX	Tivoli Enterprise Portal Client Versión 6.2.1
KCJ_06220000.cfg	Todas	Tivoli Enterprise Portal Client Versión 6.2.2
KCQ_06200000.cfg	Windows	Tivoli Enterprise Portal Server Versión 6.2

Tabla 15. Archivos de configuración predefinidos (continuación)

Archivo de configuración	Plataforma	Versión del producto, plataforma, sistema operativo
KCQ_06210000.cfg	UNIX	Tivoli Enterprise Portal Server Versión 6.2.2
KCQ_06220000.cfg	Todas	Tivoli Enterprise Portal Server Versión 6.2.2
KHD_06200000.cfg	Windows	Warehouse Proxy Agent Versión 6.2
KHD_06210000.cfg	Todas	Warehouse Proxy Agent Versión 6.2.1
KHD_06220000.cfg	Todas	Warehouse Proxy Agent Versión 6.2.2
KHE_06220000.cfg	UNIX	Warehouse Proxy Agent Versión 6.2.2
KIS_07200000.cfg	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Internet Service Monitoring Versión 7.2
KIS_07300000.cfg	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Internet Service Monitoring Versión 7.3
KLZ_06210000.cfg	UNIX	Tivoli Monitoring Operating System Agent para Linux Versión 6.2.1
KLZ_06220000.cfg	UNIX	Tivoli Monitoring Operating System Agent para Linux Versión 6.2.2
KM6_070100000.cfg	Windows	Tivoli Composite Application Manager Agent for WebSphere MQ File Transfer Edition Versión 7.1
KMQ_070100000.cfg	Todas	Tivoli Composite Application Manager Agent for WebSphere MQ Versión 7.1
KMS_06200000.cfg	Windows	Tivoli Enterprise Monitoring Server Versión 6.2
KMS_06210000.cfg	Todas	Tivoli Enterprise Monitoring Server Versión 6.2.1
KMS_06220000.cfg	Todas	Tivoli Enterprise Monitoring Server Versión 6.2.2
KNT_06200000.cfg	Windows	Tivoli Monitoring Operating System Agent para Windows Versión 6.2
KNT_06210000.cfg	Windows	Tivoli Monitoring Operating System Agent para Windows Versión 6.2.1
KNT_06220000.cfg	Windows	Tivoli Monitoring Operating System Agent para Windows Versión 6.2.2
KOR_06220000.cfg	Windows	Tivoli Monitoring Agent para Oracle Versión 6.2.2
KQI_07010000.cfg	Todas	Tivoli Composite Application Manager Agent para WebSphere Message Broker Versión 7.1
KSY_06200000.cfg	Windows	Summarization and Pruning Agent Versión 6.2
KSY_06210000.cfg	Todas	Summarization and Pruning Agent Versión 6.2.1
KSY_06220000.cfg	Todas	Summarization and Pruning Agent Versión 6.2.2
KTO_07200000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Reporter Versión 7.2
KTO_07200200.cfg	Windows	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Reporter Versión 7.2.2
KTO_07300000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Reporter Versión 7.3
KTU_07200000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Collector Versión 7.2
KTU_07200200.cfg	Windows	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Collector Versión 7.2.2
KTU_07300000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Collector Versión 7.3
KT3_07300000.cfg	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Application Management Console Versión 7.3

Tabla 15. Archivos de configuración predefinidos (continuación)

Archivo de configuración	Plataforma	Versión del producto, plataforma, sistema operativo
KT4_07300000.cfg	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Client Response Time Versión 7.3
KT5_07300000.cfg	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Web Response Time Versión 7.3
KT6_07300000.cfg	Todas	Tivoli Composite Application Manager (ITCAM) for Transactions: Robotic Response Time Versión 7.3
KUD_06100000.cfg	Windows	Tivoli Monitoring Agent para DB2 Versión 6.1
KUD_06200000.cfg	Todas	Tivoli Monitoring Agent para DB2 Versión 6.2
KUD_06210000.cfg	Todas	Tivoli Monitoring Agent para DB2 Versión 6.2.1
KUD_06220000.cfg	Todas	Tivoli Monitoring Agent para DB2 Versión 6.2.2
KZE_06020300.cfg	Todas	Tivoli zEnterprise Monitoring Agent Versión 6.2.3
LCM_01000000.cfg	Todas	Tivoli License Compliance Manager Versión 1.0
LCM_02300000.cfg	Todas	Tivoli License Compliance Manager Versión 2.3
NCI_06100000.cfg	Todas	Tivoli Netcool/Impact Versión 6.1
NOC_07310000.cfg	Todas	Componentes de servidor y componentes de escritorio de Tivoli Netcool/OMNIbus Versión 7.3.1
NOD_07310000.cfg	Todas	Componente de escritorio de Tivoli Netcool/OMNIbus Versión 7.3.1
NOS_07310000.cfg	Todas	Componentes de servidor de Tivoli Netcool/OMNIbus Versión 7.3.1
PAE_07500000.cfg	Todas	Tivoli Process Automation Engine
TAD_07200000.cfg	Todas	Tivoli Asset Discovery for Distributed Versión 7.2
TAD_07220000.cfg	Todas	Tivoli Asset Discovery for Distributed Versión 7.2.2
TCR_02010100.cfg	Todas	Tivoli Common Reporting
TPM_07210000.cfg	Todas	Tivoli Provisioning Manager Versión 7.2.1

Apéndice C. Referencia de propiedades de requisitos previos

En esta referencia se describen las propiedades de requisitos previos para cada categoría predefinida de requisitos previos de hardware y software.

Tabla 16 describe las categorías predefinidas de requisitos previos de software y hardware.

Tabla 16. Categorías predefinidos de propiedades de requisitos previos

Categoría de datos	Descripción	Identificador de prefijo necesario	Referencia
Común	Las propiedades de datos comunes comprueban requisitos previos comunes, como la velocidad del procesador, la memoria RAM, el disco y el espacio temporal.	Ninguna	"Propiedades de datos comunes" en la página 92
Autonomic Deployment Engine	Las propiedades de datos de Autonomic Deployment Engine comprueban requisitos previos de Autonomic Deployment Engine, como la unidad de instalación.	`de`	"Propiedades de datos Autonomic Deployment Engine" en la página 97
Software instalado	Las propiedades de datos de software instalado comprueban los requisitos previos del software instalado, como los programas registrados en el registro de Windows y si cygwin y gskit están instalados.	Ninguna	"Propiedad de datos de software instalado" en la página 113
Usuario	Las propiedades de datos de usuario comprueban los requisitos previos de los usuarios; por ejemplo, si el usuario conectado tenía derechos de administrador o es el usuario root.	`user`	"Propiedades de datos de usuario" en la página 113
Sistema operativo	Las propiedades de los datos del sistema operativo comprueba los requisitos previos del sistema operativo, como versión, arquitectura, memoria total, memoria disponible y memoria física total.	`os`	"Propiedad de los datos del sistema operativo" en la página 101
Conectividad	Las propiedades de datos de conectividad comprueban los requisitos previos de conectividad; por ejemplo si Telnet se está ejecutando y a qué direcciones IP y puertos puede conectarse Scanner.	Ninguna	"Propiedades de datos de conectividad" en la página 98
Red	Las propiedades de datos de red comprueban los requisitos previos de red que pueden ser comunes a todas las plataformas; por ejemplo si hay puertos disponibles.	`network`	"Propiedades de datos de red" en la página 99
Red de Windows	Las propiedades de datos de red de Windows comprueban los requisitos previos de red; por ejemplo, si se han habilitado NetBIOS y DHCP en la máquina y las propiedades de hacer ping.	`network`	"Propiedades de datos de red Windows" en la página 114

Tabla 16. Categorías predefinidos de propiedades de requisitos previos (continuación)

Categoría de datos	Descripción	Identificador de prefijo necesario	Referencia
Red de UNIX	Las propiedades de datos de red de UNIX comprueban los requisitos previos de red; por ejemplo, si se han habilitado NetBIOS y DHCP en la máquina y las propiedades de hacer ping.	`network`	"Propiedades de datos de red de UNIX" en la página 114
Internet Explorer	Las propiedades de datos de Microsoft Internet Explorer comprueban los requisitos previos de Internet Explorer, como la versión.	`internetExplorer`	"Propiedades de datos de Internet Explorer" en la página 99
Servidor de bases de datos, DB2	Las propiedades de datos DB2 comprueban los requisitos previos de DB2, como la versión.	`DB2`	"Propiedades de datos DB2" en la página 98
Servidor de base de datos, MS SQL	Las propiedades de datos de MS SQL Server comprueban los requisitos previos del servidor de MS SQL, como la versión.	`mssql`	"Propiedades de datos de MS SQL Server" en la página 98
Servidor de base de datos, Oracle	Las propiedades de datos de Oracle comprueban requisitos previos de Oracle, como la versión.	`Oracle`	"Propiedades de datos de Oracle" en la página 100
Variables de entorno	Las variables de entorno comprueban los requisitos previos de las variables de entorno; por ejemplo si se ha establecido la variable de entorno.	`env`	"Propiedades de los datos de la variable de entorno" en la página 115

Propiedades de datos comunes

Las propiedades de datos comunes comprueban requisitos previos comunes, como la velocidad de la CPU, la memoria RAM, el disco y el espacio temporal. En sistemas Windows, utiliza la secuencia de comandos primaria IBM Prerequisite Scanner. Con sistemas UNIX, utiliza la secuencia de comandos primaria Prerequisite Scanner y el recopilador común, *ips_root*`/Unix_Linux/common.sh`.

Tabla 17 describe las propiedades de requisitos previos de datos comunes. Esta categoría de propiedades de requisitos previos no requiere un identificador de prefijo.

Tabla 17. Propiedades de requisitos previos de datos comunes

Propiedad de requisito previo	Plataformas	Descripción	Valores válidos
`CPU Name`	Todas	El nombre de la CPU y se utiliza con fines de visualización sólo en los resultados	No aplicable
`CpuArchitecture`	UNIX	Arquitectura del sistema operativo	Cadena, con varios valores compatibles separados por una coma; por ejemplo: `x86_64,s390x,ppc64,AMD64`

Tabla 17. Propiedades de requisitos previos de datos comunes (continuación)

Propiedad de requisito previo	Plataformas	Descripción	Valores válidos

DBType	Todas	Comprueba los tipos de servidor de bases de datos instalados en el equipo.
		
		Para Oracle solo en sistemas UNIX: el recopilador espera las variables de entorno ORACLE_BASE y ORACLE_HOME para establecer en el archivo $HOME/.profile; por ejemplo:
		`export ORACLE_BASE=/home/oracle/app/oracle/product/11.2.0/`
		`export ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1`
		
		donde $HOME debe ser /home/oracle, el directorio de inicio del usuario de Oracle.
			El valor puede ser cualquiera de los siguientes tipos:
			• Cadena que representa cualquier tipo de servidor de bases de datos; por ejemplo:
			`any`
			• Cadena que representa el tipo de servidor de bases de datos; por ejemplo:
			`Oracle`
			• `regex{`*`str`*`}`, una expresión regular con el parámetro de entrada, *str*, que representa el patrón de búsqueda del tipo de servidor de bases de datos; por ejemplo:
			`regex{.*MSSQL.*|DB2.*}`
			
			Comprueba si el tipo de servidor de base de datos es MS SQL o DB2 en sistemas Windows.
			• Cadena que representa ningún tipo de servidor de bases de datos; por ejemplo:
			`unknown`

DBTypeDetails	Todas	Los tipos de servidor de bases de datos instalados en el equipo.
		
		Para Oracle solo en sistemas UNIX: el recopilador espera las variables de entorno ORACLE_BASE y ORACLE_HOME para establecer en el archivo $HOME/.profile; por ejemplo:
		`export ORACLE_BASE=/home/oracle/app/oracle/product/11.2.0/`
		`export ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1`
		
		donde $HOME debe ser /home/oracle, el directorio de inicio del usuario de Oracle.
		
		La propiedad de requisito previo escribe los detalles sobre el tipo de servidor de bases de datos, es decir, el tipo de servidor de bases de datos, la ubicación instalada y la versión del archivo result.txt. Los detalles de varios tipos de bases de datos de servidor están separados por puntos y comas
			El valor puede ser cualquiera de los siguientes tipos:
			• Cadena que representa cualquier tipo de servidor de bases de datos; por ejemplo:
			`any`
			• Cadena que representa un tipo de servidor de bases de datos; por ejemplo:
			`DB2`
			• `regex{`*`str`*`}`, una expresión regular con el parámetro de entrada, *str*, que representa el patrón de búsqueda del tipo dfe servidor de bases de datos; por ejemplo:
			`regex{.*MSSQL.*|DB2.*}`
			
			Comprueba si el tipo de servidor de base de datos es MS SQL o DB2 en sistemas Windows.

Tabla 17. Propiedades de requisitos previos de datos comunes (continuación)

Propiedad de requisito previo	Plataformas	Descripción	Valores válidos

`Disk`	Windows	La cantidad de espacio en disco libre, con los siguientes atributos de calificación opcionales:
		• Atributo `dir`, para determinar qué ruta al directorio se comprobará
		• Atributo `unit`, para determinar qué unidades de espacio de disco se utilizarán
			El valor puede ser cualquiera de los siguientes tipos:
			• Cadena con el siguiente formato calificador:
			`[dir:`*`dir_path`*`, unit:`*`unit_name`*`] `*`disk_space`*
			Por ejemplo:
			`Disk= [dir:C:\Program Files\IBM\SQLLIB, unit:MB]1431`
			• Formato numérico en MB o GB:
			`disk_space``MB|GB`
			Por ejemplo:
			`Disk=250MB`

`Disk`	UNIX	La cantidad de espacio libre en disco	Formato numérico en GB o MB; por ejemplo: `2 GB`
`intel.cpu`	Todas	La velocidad de la CPU para el procesador Intel	Formato numérico en GHz, y en Windows solo en MHz también; por ejemplo: `2GHz`
`Memory`	Todas	La cantidad total de memoria física que se encuentra actualmente disponible en la máquina. **Consejo:** Compruebe por separado la cantidad de memoria virtual y física disponible utilizando las propiedades de requisitos previos de la categoría de sistema operativo.	Formato numérico en GB o MB; por ejemplo: `300MB`

Tabla 17. Propiedades de requisitos previos de datos comunes (continuación)

Propiedad de requisito previo	Plataformas	Descripción	Valores válidos

`OS Version`	Todas	El nombre completo y la versión del sistema operativo que se ejecuta en la máquina; también, puede usar una expresión regular para pasar una cadena que representa las distintas variantes de un sistema operativo.
		Consejo: Utilice esta propiedad de requisito previo en conjunción con `os.servicePack` y `os.architecture` para comprobar el Service Pack y la arquitectura del sistema actuales.
			El valor puede ser cualquiera de los siguientes tipos:
			• Cadena que representa varias versiones,con cada versión separada por una coma; por ejemplo:
			`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`
			Restricción: En sistemas Windows, el comodín `*` solo se admite con una expresión regular.
			• `regex{str}`, una expresión regular con el parámetro de entrada, *str*, que representa el patrón de búsqueda de la versión; por ejemplo:
			`regex{Windows 200[3-8]}`
			Comprueba si el sistema operativo actual coincide con cualquier versión desde Windows 2003 a Windows 2008.
			`regex{Red Hat*.*}`
			Comprueba si el sistema operativo actual coincide con una variante de Red Hat Linux .
			Nota: Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.

`numCPU`	Todas	El número de núcleos o procesadores independientes del sistema. Si la herramienta explora el sistema y no encuentra núcleos o si encuentra procesadores que no son núcleos, devuelve un resultado que indica que no se ha encontrado.	Número; por ejemplo, `4`
`risc.cpu`	UNIX	La velocidad de la CPU de un procesador RISC	Formato numérico en GHz; por ejemplo: `1.4GHz`
`Temp`	UNIX	El espacio de disco disponible para el sistema de archivos *Temp* especificado	Formato numérico en GB o MB; por ejemplo: `300MB`

Conceptos relacionados:

Prerequisite Scanner trata la comprobación de la propiedad de requisito previo `Memory` de forma distinta en función de si ya se está ejecutando un agente de Tivoli Monitoring o Tivoli Composite Application Manager en el equipo.

Referencia relacionada:

Las propiedades de los datos del sistema operativo comprueba los requisitos previos del sistema operativo, como versión, arquitectura, memoria total, memoria disponible y memoria física total. Solo para sistemas Windows, utiliza los recopiladores VBScript del sistema operativo en el directorio *ips_root*`/lib`, con el identificador de prefijo `os` en sus nombres de archivo. Solo para sistemas UNIX, utiliza los recopiladores del sistema operativo UNIX en el directorio *ips_root*`/UNIX_Linux`, con el identificador de prefijo `os` en sus nombres de archivo.

Comportamiento del sistema de la propiedad de requisito previo de memoria y agentes de Tivoli Monitoring

Prerequisite Scanner trata la comprobación de la propiedad de requisito previo `Memory` de forma distinta en función de si ya se está ejecutando un agente de Tivoli Monitoring o Tivoli Composite Application Manager en el equipo.

Si ya se ha instalado un agente, Prerequisite Scanner utiliza un valor esperado para la propiedad de requisito previo `Memory` en función de la diferencia entre el valor esperado de los archivos de configuración nuevos y existentes si el archivo de configuración existente sigue en el equipo; de lo contrario, trata el valor esperado según el comportamiento predeterminado.

Cuando se ejecuta Prerequisite Scanner para comprobar los requisitos previos de un agente de Tivoli Monitoring que se está actualizando o se ha reinstalado, comprueba primero si el agente ya se está ejecutando en el equipo. Si el agente se está ejecutando, Prerequisite Scanner busca el archivo de configuración asociado con la versión existente del agente en ejecución. Se produce el siguiente comportamiento en función del resultado de esta búsqueda:

- Si no puede encontrar el archivo de configuración, Prerequisite Scanner asume que el entorno de destino no se ha explorado con anterioridad; por lo tanto, Prerequisite Scanner utiliza el valor esperado de la propiedad de requisito previo`Memory` que se ha especificado en el archivo de configuración nuevo, que sigue el comportamiento predeterminado. Prerequisite Scanner escribe esta valor esperado en la salida de resultado.
- Si encuentra el archivo de configuración, Prerequisite Scanner compara el valor esperado de la propiedad de requisito previo `Memory` de la versión existente con el valor esperado del archivo de configuración de la versión nueva. Si hay una diferencia entre los valores, y el nuevo valor es mayor que el valor esperado existente, Prerequisite Scanner establece esta diferencia como el valor esperado. Prerequisite Scanner escribe la diferencia entre los valores esperados en la salida de resultado. Por ejemplo, el archivo de configuración de la versión de agente 1 especifica 1 GB como valor esperado. El archivo de configuración nuevo de la versión de agente 2, especifica 1,5 GB como valor esperado; por lo tanto, Prerequisite Scanner utiliza y escribe 0,5 GB como la diferencia entre los valores esperados.

Propiedades de datos Autonomic Deployment Engine

Las propiedades de datos de Autonomic Deployment Engine comprueban requisitos previos de Autonomic Deployment Engine, como la unidad de instalación. Solo en sistemas Windows, utiliza los recopiladores Autonomic Deployment Engine del directorio *ips_root*/lib/, con el prefijo de en sus nombres de archivo. Solo en sistemas UNIX, utiliza los recopiladores de UNIX Autonomic Deployment Engine del directorio *ips_root*/UNIX_Linux, con el prefijo de en sus nombres de archivo.

Tabla 18 describe las propiedades de requisitos previos. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo de.

Tabla 18. Propiedades de datos Autonomic Deployment Engine

Propiedad de requisito previo	Platafor.	Descripción	Valores válidos
de.installed	Todas	Comprueba si está instalado	Booleano; por ejemplo: true|false
de.installationUnit	Todas	Comprueba si la unidad de instalación especificada se ha instalado con el comando **listIU** - v	El valor puede ser cualquiera de los siguientes tipos: • Cadena para representar una unidad de instalación única; por ejemplo la unidad de instalación de Tivoli Integrated Portal: C37109911C8A11D98E1700061BDE7AEA, B24209911C8A11D98E1700061BDE7AEA • Cadena para representar varias unidades de instalación; por ejemplo: 5FFE79F918DF3BA0D67511FD3F7C358E • regex {str}, expresión regular con el parámetro de entrada, str, para representar el patrón de búsqueda de la unidad de instalación, versión y ruta de instalación; por ejemplo, para comprobar la unidad de instalación, la versión de WebSphere Application Server y la ruta de instalación de Tivoli Integrated Portal, el patrón de búsqueda debe ser el siguiente: • regex{.*C00DA95AFD9B7E0397153CD944B5A255. *6.1.0.2100.*SIU eWAS. *C:\\IBM\\tivoli\\tip.*} **Nota:** También puede utilizar una variable de entorno para la ruta de instalación; por ejemplo, sustituyendo la ruta con la variable de entorno TIPHOME, el patrón de búsqueda es: regex{.*C00DA95AFD9B7E0397153CD944B5A255. *6.1.0.2100.*SIU eWAS. *%TIPHOME%.*} • Varios argumentos regex {str} para representar varias comprobaciones; por ejemplo: regex{.*C37109911C8A11D98E1700061BDE7AEA.*}, regex{.*B24209911C8A11D98E1700061BDE7AEA.*}

Propiedades de datos de conectividad

Las propiedades de datos de conectividad comprueban los requisitos previos de conectividad; por ejemplo si Telnet se está ejecutando y a qué direcciones IP y puertos se puede conectar Scanner. Con sistemas Windows, utiliza el recopilador de conectividad, *ips_root*/lib/connectivity_plug.vbs. Con sistemas UNIX, utiliza el script primario de IBM Prerequisite Scanner y el recopilador de conectividad, *prs_root*/Unix_Linux/connectivity_plug.sh. La salida se pasa únicamente al archivo de registro de depuración.

Propiedades de datos DB2

Las propiedades de datos DB2 comprueban los requisitos previos de DB2, como la versión. Solo en sistemas Windows, utiliza el recopilador DB2, *ips_root*/lib/db2_version_plug.bat. Solo en sistemas UNIX, utiliza los recopiladores DB2 de UNIX en el directorio *ips_root*/UNIX_Linux, con el prefijo db2 en sus nombres de archivo.

Tabla 19 describe la propiedades de requisito previo de DB2. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo DB2.

Tabla 19. Propiedades de datos DB2

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
DB2 Version	Todas	La versión de DB2 que está actualmente instalada en la máquina	Cadena, por ejemplo: v9.5.100.179FP4
db2.home.space	UNIX	El espacio de disco disponible para el directorio de inicio DB2	Formato numérico en GBs; por ejemplo: 8GB

Propiedades de datos de MS SQL Server

Las propiedades de datos de MS SQL Server comprueban los requisitos previos de MS SQL Server, como la versión y la ubicación. Solo en sistemas Windows, utiliza los recopiladores de MS SQL Server del directorio *ips_root*/Windows, con el prefijo mssql en sus nombres de archivo.

Tabla 20 escribe las propiedades de requisitos previos de MS SQL Server. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo mssql.

Tabla 20. Propiedades de datos de MS SQL Server

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
mssql.Client	Windows	Comprueba la versión del cliente de MS SQL que está actualmente instalada en la máquina	El valor de cadena esperado pueden ser varias versiones, separadas por una coma; por ejemplo: 10.50.1600.1 **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.

Tabla 20. Propiedades de datos de MS SQL Server (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`mssql.Server`	Windows	Comprueba la versión del servidor de MS SQL que está actualmente instalada en la máquina	El valor de cadena esperado pueden ser varias versiones, separadas por una coma; por ejemplo: `10.50.1600.1` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`mssql.Server.Location`	Windows	Comprueba el directorio de inicio del servidor de bases de datos de MS SQL	Cadena, por ejemplo: `any`

Propiedades de datos de Internet Explorer

Las propiedades de datos de Microsoft Internet Explorer comprueban los requisitos previos de Internet Explorer, como la versión. Utiliza el recopilador de Internet Explorer, *ips_root*`/lib/internetExplorer_plug.vbs`.

Tabla 21 describe las propiedades de requisitos previos de Internet Explorer. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo `internetExplorer`.

Tabla 21. Propiedades de datos de Internet Explorer

Propiedad de requisito previo	Descripción	Valores válidos
`internetExplorer.version`	La versión de Internet Explorer instalada en la máquina.	Formato numérico; por ejemplo, `7.0+` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.

Propiedades de datos de red

Las propiedades de datos de red comprueban los requisitos previos de red que pueden ser comunes a todas las plataformas; por ejemplo si hay puertos disponibles. Utiliza los recopiladores de red del directorio *ips_root*`/lib`, con el identificador de prefijo `network` en sus nombres de archivo.

Tabla 22 en la página 100 describe las propiedades de requisitos previos de red que son comunes a todas las plataformas. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo `network`.

Tabla 22. Propiedades de datos de red

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`network.availablePorts.`*`app_type`*	Todas	Utilice este convenio de denominación para comprobar si el puerto o rango de puertos está disponible para el tipo de aplicación *app_type*. Compruebe qué puertos no se escuchan; por ejemplo: • `network.availablePorts.DB2` comprueba los puertos del servidor de bases de datos DB2, donde *app_type* es `DB2` • `network.availablePorts.WAS` comprueba los puertos de WebSphere Application Server, donde *app_type* es `WAS`	Enteros positivos; por ejemplo: • `network.availablePorts.DB2 = 50000-50005` • `network.availablePorts.WAS = 8080` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`network.portsInUse.`*`app_type`*	Todas	Utilice este convenio de denominación para comprobar si el puerto o rango de puertos está en uso para el tipo de aplicación *app_type*. Compruebe qué puertos se están escuchando; por ejemplo: • `network.availablePorts.DB2` comprueba los puertos del servidor de bases de datos DB2, donde *app_type* es `DB2` • `network.availablePorts.WAS` comprueba los puertos de WebSphere Application Server, donde *app_type* es `WAS`	Enteros positivos; por ejemplo: • `network.portsInUse.DB2 = 50900-50905` • `network.portsInUse.WAS = 8080` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`network.validate HostsFile`	Windows	Comprueba si todos los sistemas host que están enumerados en el archivo de hosts tienen el formato: *`IP_Address Host_Name Short_Name`* donde: • *`IP_Address`* es la dirección IP del equipo; por ejemplo, `127.0.0.1` • *`Host_Name`* es el nombre de host completo del equipo; por ejemplo, `localhost.localdomain` • *`Short_Name`* es el nombre abreviado del equipo; por ejemplo, `localhost`	Valor booleano; por ejemplo, `True`

Propiedades de datos de Oracle

Las propiedades de datos de Oracle comprueban requisitos previos de Oracle, como la versión. Solo en sistemas Windows, utiliza el recopilador de Oracle. Solo en sistemas UNIX, utiliza los recopiladores de UNIX del directorio *ips_root*`/UNIX_Linux`, con el prefijo `oracle` en sus nombres de archivo.Solo en sistemas Windows, utiliza los recopiladores Oracle de Windows del directorio *ips_root*`/lib`, con el prefijo `oracle` en sus nombres de archivo.

Tabla 23 describe las propiedades de requisitos previos de Oracle. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo `oracle`.

Tabla 23. Propiedades de datos de Oracle

Propiedad de requisito previo	Platafor.	Descripción	Valores válidos
`ORACLE Version`	Windows	Comprueba la versión de Oracle que está actualmente instalada en la máquina	El valor de cadena esperado pueden ser varias versiones, separadas por una coma; por ejemplo: `9.2, 10.1, 10.2`
`oracle.Client`	Todas	Comprueba la versión del cliente de Oracle que está actualmente instalada en la máquina	El valor de cadena esperado pueden ser varias versiones, separadas por una coma; por ejemplo: `9.2.0.8+` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`oracle.Client.Location`	Todas	Comprueba el directorio de inicio del cliente de Oracle.	Cadena, por ejemplo: `/opt/oracle/products/10.1.0/client_1`
`oracle.Server`	Todas	Comprueba la versión del servidor de Oracle que está actualmente instalada en la máquina	El valor de cadena esperado pueden ser varias versiones, separadas por una coma; por ejemplo: `10.2.0.4g,11g R1` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`oracle.Server.Location`	Todas	Comprueba el directorio de inicio del servidor de bases de datos de Oracle	Cadena, por ejemplo: `/opt/oracle/product/10.1.0/Db_1`

Propiedad de los datos del sistema operativo

Las propiedades de los datos del sistema operativo comprueba los requisitos previos del sistema operativo, como versión, arquitectura, memoria total, memoria disponible y memoria física total. Solo para sistemas Windows, utiliza los recopiladores VBScript del sistema operativo en el directorio *ips_root*`/lib`, con el identificador de prefijo `os` en sus nombres de archivo. Solo para sistemas UNIX, utiliza los recopiladores del sistema operativo UNIX en el directorio *ips_root*`/UNIX_Linux`, con el identificador de prefijo `os` en sus nombres de archivo.

Tabla 24 describe las propiedades de requisitos previos de sistema operativo. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo `os`.

Tabla 24. Propiedad de datos de sistema operativo

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.architecture`	Todas	Comprueba la arquitectura del sistema	`32-bit|64-bit`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.automount`	UNIX	Comprueba si las características de montaje automático funcionan	Valor booleano, por ejemplo: `True`
`os.autoUpdateEnabled`	Windows	Comprueba si Windows Update se habilita de forma automática; devuelve `True` si está habilitado	Valor booleano, por ejemplo: `True`
`os.availableMemory`	Windows	Comprueba la cantidad de memoria virtual que está disponible actualmente, pero no utiliza el sistema operativo	Formato numérico en MB; por ejemplo: `900MB`

`os.dir.`*`dir_name`*	UNIX	Comprueba el sistema de archivos *dir_name* en función de los siguientes atributo de cualificación:
		• Atributo `dir`, para determinar qué sistema de archivos se comprobará
		• Atributo `type`, para determinar qué atributo del sistema de archivos se comprobará; por ejemplo, la representación de ocho dígitos de *`octal_digits`* para los permisos de acceso a ese sistema de archivos
		dir_name puede representar por ejemplo:
		• `tmp`
		• `home`
			Cadena con el siguiente formato calificador:
			`[dir:`*`dir_name`*`, type:permission] `*`octal_digits`*`+`
			
			Por ejemplo, para comprobar si el directorio de inicio tiene permisos `drwxr-xr-x`:
			`os.dir.home=[dir:/home, type:permission]755+`

`os.diskquota`		Comprueba la cuota de uso de disco del usuario conectado, devuelve el valor de la cuota en kilobytes o `Unlimited`	El valor puede ser cualquiera de los siguientes tipos: • Número para representar kilobytes; por ejemplo, `414000` • Cadena para representar una cuota de disco ilimitado; por ejemplo, `Unlimited`
`os.expectLink`	UNIX	Comprueba si la extensión Expect para TCL se encuentra disponible en la máquina; devuelve `Available` si tiene el estado disponible **Nota:** La propiedad de requisito previo `os.file.expect` comprueba si la extensión Expect está instalada en la máquina.	`Available|Unavailable`
`os.file.`*`script_name`*	UNIX	Comprueba si la cadena de comandos *script_name* se encuentra disponible en la máquina. *script_name* puede representar por ejemplo: • `bash` • `expect` • `gzip` • `tar`	Valor booleano, por ejemplo: `True`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.Firefox`	UNIX	Comprueba si Mozilla Firefox está instalado en la máquina; devuelve `Available` si está instalado	`Available|Unavailable`
`os.FreePagingSpace`	UNIX	Comprueba el tamaño total de la memoria caché de páginas disponible	Formato numérico en MB o GB; por ejemplo: `4GB+` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`os.ftpusers`	UNIX	Comprueba si el usuario root está incluido en el archivo `ftpusers` que determina los usuarios que no tienen privilegios de P ftp; devuelve `Available` si el usuario no está incluido	`Available|Unavailable`
`os.gnu.tar`	UNIX	Comprueba si el programa de utilidad tar GNU está instalado en la máquina; devuelve `Available` si está instalado	`Available|Unavailable`
`os.hostformat`	UNIX	Comprueba si las entradas de `/etc/host` están en el formato correcto	Valor booleano, por ejemplo: `True`
`os.iodevicestatus`	AIX	Comprueba el estado de la E/S asíncrona (`aio0`), esto es, el proceso kernel para mejorar el rendimiento de la operación de E/S; devuelve `Available` si tiene un estado disponible	`Available|Unavailable`
`os.is8dot3File FormatEnabled`	Windows	Comprueba si los formatos de nombre de archivo se están aplicando de forma automática; devuelve `True` si se aplican	Valor booleano, por ejemplo: `True`
`os.localhost InHostsFile`	Todas	Comprueba si hay una entrada en el archivo de hosts que correlaciona el host local con la dirección IP `127.0.0.1`, por ejemplo: `127.0.0.1 localhost`	Valor booleano, por ejemplo: `True`
`os.isService Running.`*`service_name`*	Windows	Utilice este convenio de denominación para comprobar si *service_name* se está ejecutando en la máquina. *service_name* puede representar por ejemplo: • `remoteRegistry` para el Servicio de Registro remoto • `DNSClient` para el servicio Cliente DNS • `TerminalServices` para Servicios de Escritorio remoto o Servicios de Terminal Server	Valor booleano, por ejemplo: `True`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.kernelMode`	AIX	Comprueba la arquitectura de la CPU compatible con el kernel o el modo sin restricciones	`32-bit|64-bit`
`os.kernelParameters`	Linux	Comprueba si los parámetros de kernel están disponibles para el sistema operativo	`Available|Unavailable`
`os.kernelversion`	Linux	Comprueba el release del kernel para sistemas operativos Linux	Formato numérico; por ejemplo, `2.6`
`os.largeFile`	UNIX	Comprueba la compatibilidad con archivos grandes	Valor booleano, por ejemplo: `True`
`os.ldLibPath`	UNIX	Comprueba si existe la variable de entorno `LD_LIBRARY_PATH` y termina con un punto; es decir, `os.ldLibPath=[endsWith=:]`	`Available|Unavailable`
`os.level`	AIX	Comprueba si el nivel del sistema operativo AIX es mayor que 10 para AIX Versión 5.3 o mayor que 3 para AIX Versión 6.1	Valor booleano, por ejemplo: `True`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.lib.lib_name_version`	UNIX	Comprueba que la versión compatible de la biblioteca *lib_name* esté instalada en la máquina. Cadena o expresión regular para representar *lib_name_version*; por ejemplo, en negrita: • Biblioteca **`libstdc++.so.#`** de 32 bits • Biblioteca **`libstdc++.so.#`** de 64 bits • Biblioteca **`libXft.so.#`** de 32 bits • Biblioteca **`libXtst.so.#`** de 32 bits • Biblioteca **`libaio.so.#`** de 64 bits • Nivel de tiempo de ejecución XLC **`xlC.rte`** de 32 bits • Tiempo de ejecución XLC **`xlC.aix50.rte`** de 32 bits para AIX Versión 5.3 • Tiempo de ejecución XLC **`xlC.aix61.rte`** de 32 bits para AIX Versión 6.1 • Biblioteca AIX IOCP **`bos.iocp.rte`** • **`bos.loc.iso.en_us`**, el conjunto de archivos con código ISO para el sistema operativo base AIX	El valor puede ser cualquiera de los siguientes tipos: Cadena, por ejemplo: • `/usr/lib/libstdc++.so.#` como el valor de la biblioteca `libstdc++.so.#` de 32 bits • `/usr/lib64/libaio.so.#` como el valor de la biblioteca `libaio.so.#` de 64 bits • `xlC.aix50.rte.9.0.0.8+` como el valor del tiempo de ejecución XLC `xlC.aix50.rte` de 32 bits para AIX Versión 5.3 • `bos.loc.iso.en_us` para el conjunto de archivos de código ISO `regex {str}`, expresión regular con el parámetro de entrada, *str*, que representa el patrón de búsqueda del nombre de la biblioteca; por ejemplo: `regex{.*libgcc.*}` Comprueba si existe una versión de la biblioteca en tiempo de ejecución del CCG de bajo nivel, `libgcc`, para ese sistema operativo. **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`os.loginVariable`	UNIX	Comprueba si las rutas predeterminadas para el usuario root se han establecido en las variables `PATH` y `SUPATH`; devuelve `Available` si se han establecido.	`Available|Unavailable`
`os.maximoDirectory`	UNIX	Comprueba si el directorio `/export/home/maximo` está disponible	`Available|Unavailable`
`os.maximoDirOwner`	UNIX	Comprueba el propietario del directorio `/export/home/maximo`	`maximo`
`os.maximumProcesses`	UNIX	Comprueba el número máximo de procesos que se pueden ejecutar para cada usuario	Número; por ejemplo, `2048`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.MozillaVersion`	UNIX	Comprueba si hay una versión específica de Mozilla Firefox en la máquina a diferencia del requisito de propiedad `os.Firefox`	Formato numérico; por ejemplo, `3.0+` **Nota:** The values can use the special characters as outlined in Tabla 1 en la página 2.

`os.mountcheck`	UNIX	Checks whether the file system is mounted based on the following qualification attributes:
		• `driveAtributo`, para determinar qué directorio es el sistema de archivos montado
		• Atributo `nosuid`, para determinar si la opción de montaje se establece si se monta el sistema de archivos
			Cadena con el siguiente formato calificador:
			`[drive:`*`dir_name`*`,`
			`mount_option``:`
			`false|true]`
			`True|False`
			
			Por ejemplo, para comprobar si el directorio `/home` está montado y no se ha establecido la opción `nosuid`:
			`os.mountcheck=[drive:/home,`
			`nosuid:false]True`

`os.package.`*`package_name`*	UNIX	Comprueba que la versión compatible del paquete *package_name* esté instalada en la máquina. Cadena para representar *package_name*; por ejemplo, en negrita: • **`bash`**shell • **`expect`** para el paquete de extensión de TCL • **`libgcc`** para el paquete GCC tiempo de ejecución de bajo nivel • **`openssh`** para el Secure Shell de código abierto • **`openssl`** para el conjunto de herramientas de código abierto para SSL/TLS • **`perl`** para el paquete de extensión de Perl • **`rpm`** para los paquetes RPM o RPM Build • **`telnet`** para el paquete Telnet • **`wget`**para el paquete de recuperación de archivos GNU	Cadena, por ejemplo: • `bash-3.2+` for bash shell • `expect-1.2.0` para Expect • `libgcc-3.4.3-9` para libgcc • `openssh-5.0.0.5301-` para openssh • `openssl-4.2.0-` para OpenSSL • `perl-5.8.2` para Perl • `rpm` • `telnet` • `wget` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`os.pagesize`	UNIX	Comprueba el tamaño de página del sistema	Formato numérico en KBs; por ejemplo: `4KB` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`os.RAMSize`	Todas	Comprueba la memoria RAM del sistema	Formato numérico en el GB; por ejemplo, `8GB`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.SeaMonkeyVersion`	UNIX	Comprueba si hay una versión específica de Mozilla SeaMonkey en la máquina puesto que la vía de acceso se ha establecido en la variable de entorno `PATH`	Formato numérico; por ejemplo, `2.10` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.

`os.SELinux`	Linux	Comprueba el estado de obligatoriedad de la característica de mejora de seguridad de Linux en función de los siguientes atributos de cualificación:
		• Atributo `source`, para determinar el comando que se utilizará para el sistema operativo correspondiente
			El valor puede ser cualquiera de los siguientes tipos:
			• Cadena con el siguiente formato calificador:
			`[source:Command] Disabled|Enabled`
			Por ejemplo, para comprobar si la función está desactivada o tiene un estado permisivo en el sistema operativo Red Hat o SUSE:
			`os.SELinux=[source: Command]Disabled`
			• Cadena sin un calificador, donde el sistema operativo es una variante genérica de Linux:
			`os.SELinux=Disabled`

`os.servicePack`	Todas	Comprueba la versión actual del Service Pack que está instalada	Formato numérico, con *majorVersion*. *minorVersion* o solo la versión *majorVersion* Por ejemplo, para comprobar si el Service Pack 2 o posterior está instalado, `2+` **Nota:** The values can use the special characters as outlined in Tabla 1 en la página 2.
`os.shell.default`	UNIX	Checks whether the default shell script is installed	String to represent the shell script, for example, `bash`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.space.dir_name` Prerequisite Scanner tiene tres variantes de la propiedad `os.space.dir_name`: • `os.space.dir_name` que comprueba si existe suficiente espacio en disco disponible para el sistema de archivos especificado independientemente de si el usuario conectado es siempre root o no root. Utilice esta variante de requisito previo cuando desee comprobar la ruta especificada del sistema de archivos, pero no importe si el usuario conectado es siempre root o no root. **Nota:** No puede utilizar esta variante dos veces para el mismo sistema de archivos, pero diferentes tipos de usuario, en un único archivo de configuración; utilice en su lugar una combinación de las otras dos variantes. • `os.space.dir_name_nonroot` que comprueba si existe suficiente espacio en disco disponible para el sistema de archivos especificado del usuario no root. Utilice esta variante de requisito previo cuando esté conectado como usuario no root y desee comprobar explícitamente la ruta especificada del sistema de archivos. **Nota:** El usuario no root debe ser el mismo usuario que instala el producto en el sistema de destino. • `os.space.dir_name_root` que comprueba si existe suficiente espacio en disco disponible para el sistema de archivos especificado del usuario root. Utilice esta variante de requisito previo cuando esté conectado como usuario root y desee comprobar explícitamente la ruta especificada del sistema de archivos. Puede especificar las variantes `os.space.dir_name_nonroot` y `os.space.dir_name_root` en el mismo archivo de configuración. Prerequisite Scanner outputs `NOT_REQ_CHECK_ID` en la celda de resultados reales para la variante no aplicable. Por ejemplo, si el usuario conectado es root, Prerequisite Scanner outputs `NOT_REQ_CHECK_ID` para la variante `os.space.dir_name_nonroot`.			

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos

os.space.*dir_name*	UNIX	Comprueba el espacio en disco disponible para el sistema de archivos <*dir_name*> especificado, en función de uno o varios de los siguientes atributos de cualificación:
		• Atributo `dir`, para determinar qué ruta al sistema de archivos se comprobará
		• Atributo `unit`, para determinar qué unidades de espacio de disco se utilizarán
		
		El valor del atributo `dir` depende del usuario conectado; de este modo, el valor es un par de valor-nombre para representar el tipo de usuario, esto es, root o no root, y la ruta asociada.
		
		dir_name puede representar por ejemplo:
		• `home`
		• `opt`
		• `tmp`
		• `usr`
		• `var`
		
		Nota: No puede utilizar esta variante dos veces para el mismo sistema de archivos, pero diferentes tipos de usuario, en un único archivo de configuración. Utilice una combinación de las variantes `os.space.`*`dir_name`*`_nonroot` y `os.space.`*`dir_name`*`_root`.
			Cadena con el siguiente formato calificador para el sistema de archivos de un usuario root:
			`[dir:`**`root`**`=<`*`dir_path`*`>, unit:<`*`unit_name`*`>] <`*`disk_space`*`>`
			
			Por ejemplo:
			`os.space.usr= [dir:root=/usr/ibm/common/ acsi, unit:GB]200`
			
			Cadena con el siguiente formato calificador para el sistema de archivos de un usuario no root:
			`[dir:`**`non_root`**`=<`*`dir_path`*`>, unit:<`*`unit_name`*`>] <disk_space>`
			
			Por ejemplo:
			`os.space.home= [dir:non_root=USERHOME/ .acsi_HOST, unit:MB]200`
			
			Cadena con el siguiente formato calificador, que usa solo un calificador:
			`[dir:<`*`dir_path`*`>] <`*`disk_space`*`> MB`
			
			Por ejemplo:
			`os.space.home= [dir:/home/sat]250MB`
			
			Formato numérico en MB o GB; por ejemplo:
			`os.space.opt=11GB`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos

os.space.*dir_name*_nonroot	UNIX	Comprueba el espacio en disco disponible para el sistema de archivos *dir_name* del usuario no root en función de uno o varios de los siguientes atributos de cualificación:
		• Atributo `dir`, para determinar qué ruta al sistema de archivos se comprobará
		• Atributo `unit`, para determinar qué unidades de espacio de disco se utilizarán
		dir_name puede representar por ejemplo:
		• `home`
		• `opt`
		• `tmp`
		• `usr`
		• `var`
			Cadena con el siguiente formato calificador para el sistema de archivos de un usuario no root:
			`[dir:`**`non_root`**`=`*`dir_path`*`, unit:`*`unit_name`*`] disk_space`
			Por ejemplo:
			`os.space.home_nonroot= [dir:non_root= USERHOME/.acsi_HOST, unit:MB]200`
			Cadena con el cualificación de cualificación `dir` solo para el sistema de archivos de un usuario no root:
			`[dir:`**`non_root`**`=`*`dir_path`*`] `*`disk_space`*`GB|MB`
			Por ejemplo:
			`os.space.opt_nonroot= [dir:non_root=/opt/IBM/ITM] 1024MB`

os.space.*dir_name*_root	UNIX	Comprueba el espacio en disco disponible para el sistema de archivos *dir_name* del usuario root en función de uno o varios de los siguientes atributos de cualificación:
		• Atributo `dir`, para determinar qué ruta al sistema de archivos se comprobará
		• Atributo `unit`, para determinar qué unidades de espacio de disco se utilizarán
		dir_name puede representar por ejemplo:
		• `home`
		• `opt`
		• `tmp`
		• `usr`
		• `var`
			Cadena con el siguiente formato calificador para el sistema de archivos de un usuario root:
			`[dir:`**`root`**`=`*`dir_path`*`, unit:`*`unit_name`*`] `*`disk_space`*
			Por ejemplo:
			`os.space.usr_root= [dir:root= /usr/ibm/common/acsi, unit:GB]200`
			Cadena con el cualificación de cualificación `dir` solo para el sistema de archivos de un usuario root:
			`[dir:`**`root`**`=`*`dir_path`*`] `*`disk_space`*`GB|MB`
			Por ejemplo:
			`os.space.opt_root= [dir:root=/opt/IBM/ITM] 1024MB`

`os.sshdConfig`	UNIX	Comprueba si el inicio de sesión root se ha configurado para las sesiones del daemon SSH	`Available|Unavailable`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.swapSize`	UNIX	Comprueba si el espacio de intercambio debe ser mayor que el tamaño de la RAM o el total del espacio de intercambio	El valor puede ser cualquiera de los siguientes tipos: • Valor booleano, por ejemplo: `True` • Formato numérico en MB o GB; por ejemplo: `2GB`

`os.tmpdir`	UNIX	Comprueba si los permisos asignados al sistema de archivos `/tmp`, incluidos los permisos especificadlos establecidos por distintivos de derecho de acceso; por ejemplo, los bits `sticky`, `setuid` o `setgid` en los dígitos octales	Número para representar los dígitos octales `octal_digits` de los permisos de acceso
			Por ejemplo, para comprobar si el directorio temp tiene permisos `drwxrwxrwt` con el permiso de bit de permanencia habilitado:
			`1777`
			Otro ejemplo, para comprobar si el directorio temp tiene permisos `drwxrwxrwx`, excluido el de bit de permanencia:
			`777`

`os.totalMemory`	Windows	La cantidad total de memoria virtual a la que el sistema operativo puede acceder	Formato numérico en MB o GB; por ejemplo: `4GB`
`os.totalPhysicalMemory`	Windows	La cantidad total de memoria física a la que el sistema operativo puede tener acceso, pero no indica la cantidad de memoria física auténtica en el equipo de destino	Formato numérico en MB o GB; por ejemplo: `2030MB`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos

`os.ulimit`	UNIX	Comprueba si se puede ejecutar un número ilimitado de procesos en función de los siguientes atributos de calificación:
		• Atributo `type`, para determinar el límite adicional que se comprobará; por ejemplo `filedescriptorlimit` comprueba el límite del número de descriptores de archivo que los procesos pueden abrir
		
		También, comprueba si se han establecido los siguientes límites para los dominios especificados en el archivo `/etc/security/limits.conf`:
		`root - stack unlimited`
		`ctginst1 - stack unlimited`
		`root - nofile 8192`
		`tioadmin - nofile 32767`
			El valor puede ser cualquiera de los siguientes tipos:
			• Cadena con el siguiente formato calificador:
			`[type:`*`limit_name`*`]`
			`limit_value``,`
			`limited|unlimited`
			Por ejemplo, para comprobar si el límite del descriptor de fichero es mayor que 8192, con un número ilimitado de procesos:
			`os.ulimit=[type:`
			`filedescriptorlimit]`
			`8192+,`
			`unlimited`
			
			Los tipos válidos de límites para comprobar, donde *limit_name* representa el tipo de límite, son los siguientes:
			– `ALL`, comprueba todos los límites
			– `corefilesizelimit`
			– `datasegmentlimit`
			– `filedescriptorlimit`
			– `filesizelimit`
			– `hardlimit`
			– `processlimit`
			– `maxmemorysizelimit`
			– `maxprocesseslimit`
			– `stacksizelimit`
			– `threadlimit`
			• `Available|Unavailable` para especificar si los dominios pertinentes tienen conjuntos de límites en el archivo `/etc/security/limits.conf`.

`os.umask`	UNIX	Comprueba los permisos de la máscara de creación de modalidad de archivo	Número para representar los dígitos octales *`octal_digits`* de los permisos de acceso. Por ejemplo, para establecer que solo el propietario pueda escribir los nuevos archivos, configure el dígito octal como `0022`
`os.userLimits`	UNIX	Comprueba si el tamaño de pila máximo es ilimitado; devuelve `Available` si es ilimitado	`Available|Unavailable`

Tabla 24. Propiedad de datos de sistema operativo (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`os.versionNumber`	Windows	Comprueba la versión actual del sistema operativo que está instalado en la máquina	Formato numérico; por ejemplo, `5.0+` **Nota:** The values can use the special characters as outlined in Tabla 1 en la página 2.
`os.windowManager`	UNIX	Checks whether GNOME or KDE is available as a graphical desktop	`Available|Unavailable`

Propiedad de datos de software instalado

Las propiedades de datos de software instalado comprueban los requisitos previos del software instalado, como los programas registrados en el registro de Windows y si cygwin y gskit están instalados. Solo en sistemas Windows, utiliza los recopiladores de software instalado en el directorio *ips_root*`/lib`, con el identificador de prefijo `installedSoftware, cygwin, or gskit` en sus nombres de archivo.

Tabla 25 describe las propiedades de requisitos previos de datos comunes. Esta categoría de propiedades de requisitos previos no requiere un identificador de prefijo.

Tabla 25. Propiedad de datos de software instalado

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`installedSoftware`	Windows	Analiza el registro del sistema operativo para los programas instalados con ubicaciones	Cadena, con múltiples aplicaciones separadas por una coma.
`cygwinVersion`	Windows	Comprueba la versión de cygwin que está instalada en el equipo; devuelve 0.0 si no hay ninguna versión instalada	Entero positivo; por ejemplo, `1.5` **Nota:** Los valores pueden utilizar los caracteres especiales como se indica en Tabla 1 en la página 2.
`gskit7Version`	Windows	Comprueba si gskit Versión 7 está instalado en la máquina; devuelve 0.0 si la versión 7 no está instalada	Entero positivo; por ejemplo, `7.0`
`gskit8Version`	Windows	Comprueba si gskit Versión 8 está instalado en la máquina; devuelve 0.0 si la versión 8 no está instalada	Entero positivo; por ejemplo, `8.0`

Propiedades de datos de usuario

Las propiedades de datos de usuario comprueban los requisitos previos de los usuarios; por ejemplo, si el usuario conectado tenía derechos de administrador o es el usuario root. Solo en sistemas Windows, utiliza el recopilador de usuario del directorio *ips_root*`/lib`, con el identificador de prefijo `user` en sus nombres de archivo. Solo en sistemas UNIX, utiliza el recopilador de usuario en *ips_root*`/lib/packageTest.sh`.

Tabla 26 describe las propiedades de requisitos previos de usuario. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo `user`.

Tabla 26. Propiedades de datos de usuario

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`user.userID`	Windows	El ID del usuario actualmente conectado	Cadena; por ejemplo, `smithj`
`user.isAdmin`	Todas	Comprueba si el usuario conectado es un miembro del grupo Administradores	Valor booleano; por ejemplo, `True`

Propiedades de datos de red Windows

Las propiedades de datos de red de Windows comprueban los requisitos previos de red; por ejemplo, si se han habilitado NetBIOS y DHCP en la máquina y las propiedades de hacer ping. Utiliza los recopiladores de red de Windows del directorio *ips_root*`/lib`, con el identificador de prefijo `network` en sus nombres de archivo. .

Tabla 27 describe las propiedades de requisitos previos de red que son comunes a todas las plataformas Windows. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo `network`.

Tabla 27. Propiedades de datos de red Windows

Propiedad de requisito previo	Descripción	Valores válidos
`network.DHCPEnabled`	Comprueba si al menos un adaptador con una dirección IP válida ha obtenido la dirección IP mediante DHCP; devuelve `True` si hay al menos una.	Valor booleano; por ejemplo, `False`
`network.netBIOSEnabled`	Comprueba si al menos un adaptador con una dirección IP válida tiene habilitado NetBIOS como protocolo; devuelve `True` si hay al menos uno.	Valor booleano; por ejemplo, `True`
`network.pingLocalhost`	Comprueba si el host local responde al protocolo de ping; devuelve `True` en caso afirmativo.	Valor booleano; por ejemplo, `True`
`network.pingSelf`	Comprueba si el nombre del equipo local se ha resuelto mediante el uso de DHCP y si puede hacer ping; devuelve `True` si es así.	Valor booleano; por ejemplo, `True`
`network.ValidateHostsFile`	Comprueba si las entradas de `C:\WINDOWS\system32\drivers\etc\hosts` tienen el formato correcto; devuelve `True` si el formato es válido.	Valor booleano; por ejemplo, `True`

Propiedades de datos de red de UNIX

Las propiedades de datos de red de UNIX comprueban los requisitos previos de red; por ejemplo, si se han habilitado NetBIOS y DHCP en la máquina y las propiedades de hacer ping. Utiliza los recopiladores de red en el directorio *ips_root*`/UNIX_Linux`.

Tabla 28 en la página 115 describe las propiedades de requisitos previos de red que son comunes a todas las plataformas UNIX. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo `network`.

Tabla 28. Propiedades de datos de red UNIX

Propiedad de requisito previo	Descripción	Valores válidos
`network.DHCPEnabled`	Comprueba si al menos un adaptador con una dirección IP válida ha obtenido la dirección IP mediante DHCP	Valor booleano; por ejemplo, `False`
`network.dns`	Comprueba si la entrada DNS para la máquina host es correcta	Valor booleano; por ejemplo, `True`
`network.fqdn`	Comprueba si se ha configurado el nombre de dominio completo de la máquina host	Valor booleano; por ejemplo, `True`
`network.pingLocalhost`	Comprueba si el host local responde al protocolo de ping	Valor booleano; por ejemplo, `True`
`network.pingSelf`	Comprueba si el nombre del equipo local se ha resuelto mediante el uso de DHCP y si puede hacer ping	Valor booleano; por ejemplo, `True`

Propiedades de los datos de la variable de entorno

Las propiedades de los datos de la variable de entorno comprueban los requisitos previos de variable de entorno que pueden ser comunes a todas las plataformas; por ejemplo si se ha establecido una variable de entorno o el valor de una variable de entorno. Sólo para sistemas Windows, utiliza los recopiladores de variable de entorno en el directorio *raíz_ips*`/lib`, con el identificador de prefijo `env` en los nombres de archivo. Sólo para sistemas UNIX, utiliza los recopiladores de variable de entorno UNIX en el directorio *raíz_ips*`/UNIX_Linux` con el identificador de prefijo `env` en los nombres de archivo.

Tabla 29 describe las propiedades de requisito previo de la variable de entorno comunes para todas las plataformas. Esta categoría de propiedades de requisitos previos requiere el identificador de prefijo `env`.

Tabla 29. Propiedades de los datos de la variable de entorno

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`env.var.set.` *`env_var_name`*	UNIX	Utilice este convenio de denominación para comprobar si se ha establecido la variable de entorno *nombre_var_ent* especificada en el equipo, por ejemplo: • `env.var.set.HOME` comprueba si se ha establecido la variable de entorno para el directorio inicial, donde *nombre_var_ent* es el nombre de la variable de entorno `HOME` • `env.var.set.JAVA_HOME` comprueba si se ha establecido la variable de entorno del directorio inicial para Java, donde *nombre_var_ent* es el nombre de la variable de entorno `JAVA_HOME`	Valor booleano; por ejemplo, `True`

Tabla 29. Propiedades de los datos de la variable de entorno (continuación)

Propiedad de requisito previo	Plataforma	Descripción	Valores válidos
`env.var.set.`*`env_var_name`*`[type:`*`env_var_type`*`]`	Windows	Utilice este convenio de denominación para comprobar si se ha establecido la variable de entorno *nombre_var_ent* para el tipo de variable de entorno *tipo_var_ent*, por ejemplo: • `env.var.set.HOME` comprueba si se ha establecido la variable de entorno para el directorio inicial, donde *nombre_var_ent* es el nombre de la variable de entorno `HOME` • `env.var.set.JAVA_HOME[type:User]` comprueba si la variable de entorno del directorio inicial de Java se ha establecido para el usuario conectado, donde *nombre_var_ent* es el nombre de la variable de entorno `JAVA_HOME` y *tipo_var_ent* es el tipo de variable de entorno `User` El tipo de variable de entorno *tipo_var_ent* es opcional y representa los tipos de variables de entorno admitidos por el sistema operativo Windows del siguiente modo: • `Process` • `System` • `User` • `Volatile` El tipo predeterminado si no se especifica es `Process`.	Valor booleano; por ejemplo, `True`
`env.classpath.derbyJAR`	Todas	Comprueba si la vía de acceso al archivo Derby JAR está en la variable de entorno de la vía de acceso de clase	Valor booleano; por ejemplo, `True`
`env.CIT.homeExists`	Windows	Comprueba si existen las variables de entorno `HOMEDRIVE` y `HOMEPATH`	Valor booleano; por ejemplo, `True`

Apéndice D. Recopiladores predefinidos para sistemas UNIX

Existen recopiladores individuales para las comprobaciones de propiedades de requisitos previos en sistemas UNIX que están en el directorio *ips_root*/lib. Puede revisar estos recopiladores y sus parámetros de entrada antes de crear recopiladores personalizados.

Tabla 30 describe los recopiladores predefinidos de sistemas UNIX.

Tabla 30. Recopiladores de UNIX

Recopilador	De propiedad de requisito previo	Datos de entrada
DB2_Version	DB2 Version	Ninguno
DBType	DBType	Ninguno
DBTypeDetails	DBTypeDetails	Ninguno
env.classpath.derbyJAR	env.classpath.derbyJAR	Ninguno
env.var.set	env.var.set*env_var_name* *nombre_var_ent* es el nombre de la variable de entorno que se debe comprobar	*nombre_var_$ent*
network.DHCPEnabled	network.DHCPEnabled	Ninguno
network.dns	network.dns	Ninguno
network.fqdn	network.fqdn	Ninguno
network.pingSelf	network.pingSelf	Ninguno
network.port	network.availablePorts.* network.portsInUse.*	$ports
oracle.Client	oracle.Client	Ninguno
oracle.Client.Location	oracle.Client.Location	Ninguno
oracle.Server	oracle.Server	Ninguno
oracle.Server.Location	oracle.Server.Location	Ninguno
os.architecture	os.architecture	32 bit|64 bit
os.automount	os.automount	Ninguno
os.cmd	os.lookup	nslookup
os.cmd	os.tar|os.gnu.tar	tar|gtar
os.dir	os.dir.*dir_name* *dir_name* puede representar por ejemplo: • tmp • home	Cadena con el siguiente formato: [dir:*dir_name*, type:permission] *octal_digits*+ Por ejemplo, para comprobar si el directorio *dir_name*, es decir, el directorio de inicio tiene permisos drwxr-xr-x: [dir:/home, type:permission]755+
os.diskquota	os.diskquota	Ninguno

Tabla 30. Recopiladores de UNIX (continuación)

Recopilador	De propiedad de requisito previo	Datos de entrada
os.expectLink	os.expectLink	Ninguno
os.filepath	os.file.*script_name* *script_name* puede representar por ejemplo: • bash • expect • gzip • tar	Ruta al archivo de secuencia de comandos, donde la ruta puede ser: • /bin/bash • /usr/bin/expect • /usr/bin/gzip • /usr/bin/tar
os.Firefox	os.Firefox	Ninguno
os.FreePagingSpace	os.FreePagingSpace	Ninguno
os.ftpusers	os.ftpusers	Ninguno
os.hostformat	os.hostformat	Ninguno
os.iodevicestatus	os.iodevicestatus	Ninguno
os.kernelMode	os.kernelMode	Ninguno
os.kernelParameters	os.kernelParameters	Ninguno
os.kernelversion	os.kernelversion	Ninguno
os.largeFile	os.largeFile	Ninguno
os.ldLibPath	os.ldLibPath	Producto
os.level	os.level	Ninguno

Tabla 30. Recopiladores de UNIX (continuación)

Recopilador	De propiedad de requisito previo	Datos de entrada
os.lib	os.lib.*lib_name_version* Cadena o expresión regular para representar *lib_name_version*; por ejemplo, en negrita: • Biblioteca **libstdc++.so.#** de 32 bits • Biblioteca **libstdc++.so.#** de 64 bits • Biblioteca **libXft.so.#** de 32 bits • Biblioteca **libXtst.so.#** de 32 bits • Biblioteca **libaio.so.#** de 64 bits • Nivel de tiempo de ejecución XLC **xlC.rte** de 32 bits • Tiempo de ejecución XLC **xlC.aix50.rte** de 32 bits para AIX Versión 5.3 • Tiempo de ejecución XLC **xlC.aix61.rte** de 32 bits para AIX Versión 6.1 • Biblioteca AIX IOCP **bos.iocp.rte** • **bos.loc.iso.en_us**, el conjunto de archivos con código ISO para el sistema operativo base AIX • **regex{*str*}**, expresión regular con el parámetro de entrada, *str*, que representa el patrón de búsqueda del nombre de la biblioteca; por ejemplo, .*libgcc.*	*path_to_library* o *lib_name_version* Por ejemplo, para comprobar el valor real de la propiedad de requisito previo os.lib.libstdc++.so, los parámetros de entrada son /usr/lib/libstdc++.so.5 y libstdc++.so: os.lib /usr/lib/libstdc++.so.5 libstdc++.so Por ejemplo, para comprobar si una versión de la biblioteca en tiempo de ejecución del CCG de bajo nivel, libgcc, existe en la máquina, el parámetro de entrada es: regex{.*libgcc.*}
os.loginVariable	os.loginVariable	Ninguno
os.maximoDirectory	os.maximoDirectory	Ninguno
os.maximoDirOwner	os.maximoDirOwner	Ninguno
os.maximumProcesses	os.maximumProcesses	Ninguno
os.MozillaVersion	os.MozillaVersion	Ninguno
os.mountcheck	os.mountcheck	Cadena con el siguiente formato calificador: [drive:*dir_name*, *mount_option*: false|true] True|False Por ejemplo, para comprobar si el directorio /home está montado y no se ha establecido la opción nosuid: os.mountcheck=[drive:/home, nosuid:false]True

Tabla 30. Recopiladores de UNIX (continuación)

Recopilador	De propiedad de requisito previo	Datos de entrada
`os.package`	`os.package.`*`package_name`* Cadena para representar *package_name*; por ejemplo, en negrita: • **`bash`**shell • **`expect`** para el paquete de extensión de TCL • **`libgcc`** para el paquete GCC tiempo de ejecución de bajo nivel • **`openssh`** para el Secure Shell de código abierto • **`openssl`** para el conjunto de herramientas de código abierto para SSL/TLS • **`perl`** para el paquete de extensión de Perl • **`rpm`** para los paquetes RPM o RPM Build • **`telnet`** para el paquete Telnet • **`wget`**para el paquete de recuperación de archivos GNU	*package_name*, por ejemplo, donde *package_name* es `rpm`: `os.package rpm`
`os.pagesize`	os.pagesize	Ninguno
`os.RAMSize`	os.RAMSize	Ninguno

`os.SELinux`	`os.SELinux`	• Cadena con el siguiente formato calificador:
		`[source:Command] Disabled|Enabled`
		Por ejemplo, para comprobar si la función está desactivada o tiene un estado permisivo en el sistema operativo Red Hat o SUSE:
		`os.SELinux=[source: Command]Disabled`
		• Si no hay ningún calificador, no se pasa ningún valor al recopilador.

`os.servicePack`	`os.servicePack`	Valor de Service Pack
`os.shell.default`	`os.shell.default`	El valor esperado de la propiedad de requisito previo; por ejemplo, `bash`

Tabla 30. Recopiladores de UNIX (continuación)

Recopilador	De propiedad de requisito previo	Datos de entrada
os.space	os.space.*dir_name* *dir_name* puede representar por ejemplo: • usr • home • tmp • var	Cadena con el siguiente formato calificador para el sistema de archivos de un usuario root: [dir:**root**=*dir_path*, unit:*unit_name*] *disk_space* Por ejemplo: os.space.usr= [dir:root=/usr/ibm/common/acsi, unit:GB]200 Cadena con el siguiente formato calificador para el sistema de archivos de un usuario no root: [dir:**non_root**=*dir_path*, unit:*unit_name*] disk_space Por ejemplo: os.space.home= [dir:non_root=USERHOME/ .acsi_HOST, unit:MB]200 Cadena con el siguiente formato calificador, que usa solo un calificador: [dir:*dir_path*] *disk_space* MB Por ejemplo: os.space.home= [dir:/home/sat]250MB
os.sshdConfig	os.sshdConfig	Ninguno
os.swapSize	os.swapSize	Ninguno
os.tmpdir	os.tmpdir	Ninguno

Tabla 30. Recopiladores de UNIX (continuación)

Recopilador	De propiedad de requisito previo	Datos de entrada
os.ulimit	os.ulimit	Cadena con el siguiente formato calificador: [type:*limit_name*] *limit_value*, limited|unlimited Por ejemplo, para comprobar si el límite del descriptor de fichero es mayor que 8192, con un número ilimitado de procesos: os.ulimit= [type:filedescriptorlimit] 8192+,unlimited Los tipos válidos de límites para comprobar, donde *limit_name* representa el tipo de límite, son los siguientes: • ALL, comprueba todos los límites • corefilesizelimit • datasegmentlimit • filedescriptorlimit • filesizelimit • hardlimit • processlimit • maxmemorysizelimit • maxprocesseslimit • stacksizelimit • threadlimit
os.umask	os.umask	Ninguno
os.userLimits	os.userLimits	Ninguno
os.windowManager	os.windowManager	Ninguno

Apéndice E. Funciones comunes para los sistemas Windows

Prerequisite Scanner tiene un conjunto de funciones comunes en el archivo `/lib/common_function.vbs` para ejecutar comprobaciones en sistemas Windows.

Tabla 31. Funciones de common_function.vbs

Función	Descripción
"allFiles()" en la página 124	Lee los nombres de los archivos de un directorio especificado en una matriz.
"arrayToString()" en la página 125	Crea una representación en forma de cadena de la matriz.
"bigthan()" en la página 125	Calcula la diferencia entre el valor esperado y real de la propiedad de requisito previo si esa propiedad de requisito previo es un tamaño en MB o GB.
"changeMG()" en la página 126	Convierte el parámetro de entrada a MB o GB para las propiedades de requisito previo de espacio en disco o de memoria.
"checkItemToString()" en la página 126	Crea una representación en forma de cadena del objeto `CheckItem`.
"dictionaryToString()" en la página 127	Crea una representación en forma de cadena del objeto del diccionario de secuencias de comandos.
"exeCommand()" en la página 127	Ejecuta el comando especificado y devuelve el resultado de la ejecución del comando.
"filterCommand()" en la página 128	Ejecuta el comando especificado y devuelve las líneas del resultado del comando que coincide con el patrón especificado.
"filterFile()" en la página 128	Lee y filtra el contenido de un archivo a un objeto de diccionario de secuencias de comandos.
"findNewest()" en la página 129	Localiza el archivo de configuración más reciente.
"findSuitableFile()" en la página 129	Localiza el archivo de configuración pertinente de un producto y versión.
"fmt()" en la página 130	Modifica una cadena añadiéndole un número determinado de caracteres de otra cadena y rellenando la otra cadena con caracteres de espacios si la otra cadena es muy corta o truncándola si es muy larga.
"formatForDisplay()" en la página 131	Da formato al parámetro de entrada y lo hace legible.
"formatSizeForDisplay()" en la página 131	Toma el parámetro de entrada y agrega o recorta la parte fraccionaria del parámetro de entrada a dos decimales, por ejemplo, 123 MB por 123,00 MB MB o 12,123 MBs por 12,12 MBs
"getDecimalSeparator()" en la página 132	Determina el separador decimal que se utiliza con el entorno local actual.

Tabla 31. Funciones de common_function.vbs (continuación)

Función	Descripción
"getFirstMatch()" en la página 132	Obtiene la primera coincidencia de la cadena de búsqueda en la matriz.
"isMatch()" en la página 132	Comprueba si el patrón de búsqueda se encuentra en la cadena.
"notInLatter()" en la página 133	Filtra la primera matriz para determinar si el contenido se encuentra en la segunda matriz. En función del valor del parámetro de entrada `in_or_not`, la función devuelve el contenido de la primera matriz, incluyendo o excluyendo lo que coincide con la segunda matriz.
"passOrFail()" en la página 133	Compara los valores esperados y los reales de la propiedad de requisito previo y determina si la propiedad de requisito previo supera la comprobación. Los parámetros de entrada pueden ser números genéricos, un tamaño en MB o GB, la velocidad de la CPU en MHz o GHz, un booleano o cadenas.
"ppread()" en la página 134	Lee el contenido de un archivo en un objeto de diccionario de secuencias de comandos y divide aún más cada línea del archivo por el parámetro de entrada separador especificado si ese separador existe en la línea.
"readFile()" en la página 135	Lee cada línea de un archivo en una entrada de índice de una matriz.
"unitMGTOG()" en la página 135	Suma los contenidos de una matriz para obtener el número total de megabytes.
"varToString()" en la página 135	Crea una representación en forma de cadena de una variable. La variable para comprobar puede ser una cadena, un número, un objeto de diccionario de secuencias de comandos, una matriz o un objeto `CheckItem`.

allFiles()

Lee los nombres de los archivos de un directorio especificado en una matriz.

Finalidad

Esta función obtiene la lista de archivos del parámetro de entrada de directorio y los añade a la matriz. Devuelve la matriz.

Sintaxis

```
allFiles(filepath)
```

Parámetros de entrada

`String` *`filepath`*
La ruta al directorio que contiene los archivos.

Valores de retorno

Array *fileNames*
Devuelve la matriz que contiene los nombres de los archivos en el directorio especificado.

arrayToString()

Crea una representación en forma de cadena de la matriz.

Finalidad

Esta función toma la matriz que se pasa como parámetro de entrada y devuelve una representación en forma de cadena del contenido de la matriz.

Sintaxis

```
arrayToString(arr)
```

Parámetros de entrada

Array *arr*
Contiene la matriz.

Valores de retorno

String *result*
Devuelve una representación en forma de cadena de la representación en forma de cadena, con cada elemento separado por una coma.

bigthan()

Calcula la diferencia entre el valor esperado y real de la propiedad de requisito previo si esa propiedad de requisito previo es un tamaño en MB o GB.

Finalidad

Esta función llama primero a la función "changeMG()" en la página 126 para modificar los valores esperados y reales de la propiedad de requisito previo a MB en caso necesario. A continuación, comprueba si alguno de los valores devueltos de las funciones es nulo, y en ese caso, el valor devuelto de la función es 0 MB y la función se cierra. Comprueba los MB o GB en cualquiera de los dos valores y los convierte a MB en caso necesario. Calcula la diferencia entre los valores finales con formato y devuelve el resultado.

Sintaxis

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

Parámetros de entrada

String *expect*
El valor esperado de la propiedad de requisito previo.

String *real*
El valor real de la propiedad de requisito previo.

Valores de retorno

String *bigthan*
Devuelve la diferencia en MB o 0 MB si no hay ninguna diferencia.

changeMG()

Da formato al parámetro de entrada para eliminar caracteres de agrupación de dígitos de él y devuelve el parámetro con formato, salvo si el parámetro de entrada contiene MB o GB. En este caso, convierte el parámetro de entrada a GB o MB respectivamente.

Finalidad

Esta función llama a la función "getDecimalSeparator()" en la página 132 para determinar el separador de decimales del entorno local actual y después elimina todos los caracteres de agrupación de dígitos adicionales para ese entorno local del parámetro de entrada de número. A continuación, llama a la función "getFirstMatch()" en la página 132 para determinar si el valor está en MB o GB y convierte el valor a MB o GB respectivamente.

Sintaxis

```
changeMG(tochange)
```

Parámetros de entrada

String *tochange*
Contiene el formato de valor y lo convierte si es necesario.

Valores de retorno

String *changeMG*
Devuelve el número con formato sin los caracteres de agrupación de dígitos o el número en MB o GB.

checkItemToString()

Crea una representación en forma de cadena del objeto `CheckItem`.

Finalidad

Esta función toma el objeto `CheckItem` que se pasa como parámetro de entrada y devuelve una representación en forma de cadena que comprende los valores de las diferentes propiedades para esa instancia del objeto `CheckItem`.

Sintaxis

```
checkItemToString(var)
```

Parámetros de entrada

CheckItem *var*
Contiene la instancia del objeto `CheckItem`.

Valores de retorno

String *result*
Devuelve una representación en forma de cadena de las propiedades del objeto `CheckItem` como se indica a continuación:

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

dictionaryToString()

Crea una representación en forma de cadena del objeto del diccionario de secuencias de comandos.

Finalidad

Esta función toma el objeto de diccionario que se pasa como parámetro de entrada y devuelve una representación en forma de cadena del contenido de ese objeto de diccionario.

Sintaxis

```
dictionaryToString(dic)
```

Parámetros de entrada

Dictionary *dic*
Contiene el objeto del diccionario.

Valores de retorno

String *result*
Devuelve una representación en forma de cadena del objeto de diccionario, con cada clave y elemento separados por el símbolo igual.

exeCommand()

Ejecuta el comando especificado y devuelve el resultado de la ejecución del comando.

Finalidad

Esta función ejecuta el parámetro de entrada del comando. Si hay algún error, llama a la subrutina `logWarning` para mostrar los errores; en caso contrario, devuelve el resultado de la ejecución de ese comando.

Sintaxis

```
exeCommand(cmd)
```

Parámetros de entrada

String *cmd*
El nombre del comando para ejecutar.

Valores de retorno

String *result*
Devuelve una cadena que contiene el resultado de la ejecución de ese comando.

filterCommand()

Ejecuta el comando especificado y devuelve las líneas del resultado del comando que coincide con el patrón especificado.

Finalidad

Esta función ejecuta el parámetro de entrada del comando. Analiza el resultado de la ejecución de ese comando y comprueba si cualquier línea del resultado coincide con el parámetro de entrada del patrón de línea. Si existe una coincidencia, llama a la función "getFirstMatch()" en la página 132 para determinar si existe también una coincidencia entre el parámetro de entrada de la línea de información y el resultado del comando. Si existe, utiliza la función `Join` para devolver el contenido del objeto de diccionario de la función `getFirstMatch()`.

Sintaxis

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

Parámetros de entrada

String *cmd*
El nombre del comando para ejecutar.

String *line_patt*
El patrón de la línea que desea buscar en el resultado de la ejecución de ese comando.

Number *after_line*
El número de líneas después del cual se detiene la búsqueda del patrón de información.

String *info_patt*
El patrón de información para buscar en cada línea del resultado del comando.

Valores de retorno

String *filterCommand*
Devuelve el contenido del objeto de diccionario como una única cadena.

filterFile()

Lee y filtra el contenido de un archivo a un objeto de diccionario de secuencias de comandos.

Finalidad

Esta función lee cada línea del archivo y pasa cada línea con el patrón de búsqueda a la función "getFirstMatch()" en la página 132. Si devuelve una coincidencia y la línea no existe realmente en el objeto del diccionario, la línea se escribe en el objeto del diccionario. La función recorre en bucle el archivo hasta que alcanza el final y devuelve el objeto de diccionario.

Sintaxis

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

Parámetros de entrada

String *fileName*
El archivo que se va a filtrar.

String *patt*
El patrón que se buscará en cada línea del archivo.

Valores de retorno

Dictionary *dic.keys*
Devuelve el objeto de diccionario *dic* con las líneas filtradas del archivo.

findNewest()

Localiza el archivo de configuración más reciente de una matriz.

Finalidad

Esta función recorre en bucle la matriz y determina qué archivo de la matriz es el archivo de configuración más reciente. Devuelve el nombre del archivo.

Sintaxis

```
findNewest(arr)
```

Parámetros de entrada

Array *arr*
Contiene el conjunto de archivos de configuración para comprobar.

Valores de retorno

String *result*
Devuelve el nombre del archivo de configuración más reciente.

findSuitableFile()

Localiza el archivo de configuración pertinente de un producto y versión.

Finalidad

Esta función llama a la función “getFirstMatch()” en la página 132 para obtener el conjunto de archivos que tiene el parámetro de entrada de extensión como el archivo de extensión de la lista de archivos devueltos por la función “allFiles()” en la página 124. A continuación, vuelve a llamar a la función “getFirstMatch()” en la página 132 para devolver el conjunto de archivos que contiene el parámetro de entrada de código de producto en el nombre de archivo. Llama a la misma función para obtener el conjunto de archivos que contiene el parámetro de entrada de versión en el nombre de archivo. Si la función encuentra una o varias coincidencias para la versión, llama a la función “findNewest()” para obtener la versión más reciente de ese archivo y devuelve el nombre de ese archivo; en caso contrario, devuelve el archivo `common.bat` o utiliza las subrutinas `logScreen` y `logWarning` antes de devolver la versión más reciente del archivo de configuración del código de producto.

Sintaxis

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

Parámetros de entrada

String *pd*
El código de producto asociado con el archivo para localizar, como se especifica en el archivo de código de producto, *ips_root*/codename.cfg.

String *version*
La versión del producto asociada con el producto para localizar. <*version*> es el código de 8 dígitos para representar la versión, release, modificación y nivel con dos dígitos para cada parte del código; por ejemplo, 7.3.21 es 07032100.

String *suf*
La ampliación del tipo de archivo para localizar; por ejemplo, cfg o bat.

String *filepath*
La ruta al directorio que contiene el archivo para localizar.

Valores de retorno

String *findSuitableFile*
Devuelve uno de los siguientes nombres de archivo en función de los resultados de las funciones llamadas:

- *pd_version*.cfg, la versión más reciente del código de producto y versión asociados.
- common.bat si el valor del parámetro de entrada de extensión de archivo es bat.
- *pd*.cfg, la versión más reciente del archivo de configuración genérico del producto, si no se ha encontrado ningún archivo que contenga el parámetro de entrada de versión.

fmt()

Modifica una cadena añadiéndole un número determinado de caracteres de otra cadena y rellenando la otra cadena con caracteres de espacios si la otra cadena es muy corta o truncándola si es muy larga.

Finalidad

Esta función busca la expresión %#s dentro del parámetro de entrada *s* de la cadena de tipo. La expresión %#s determina el número # de caracteres especificados del parámetro de entrada *args* que se añaden a la primera cadena en la posición de esa expresión. Si el # especificado es mayor que la longitud del parámetro de entrada *args*, la diferencia se rellena con caracteres de espacio. Si el # especificado es menor que la longitud del parámetro de entrada *args*, la cadena se trunca según esa diferencia. Si el # es 0, se añade la longitud total del parámetro de entrada *args* a la primera cadena en la posición adecuada de la cadena.

Sintaxis

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

Parámetros de entrada

String *s*
Contiene la cadena para modificar según el número # de caracteres en la expresión %#s dentro de la cadena.

Array *args*
Contiene el conjunto de caracteres que modifican el parámetro de entrada *s*.

Valores de retorno

String *result*
Devuelve la cadena modificada.

Ejemplo

```
fmt("Hello %5s!",array("Neo")) returns "Hello Neo  !" padded with extra space characters
fmt("Hello %5s!",array("Mr. Anderson")) returns "Hello Mr. A!" truncated to add only "Mr. A"
fmt("Hello %0s!",array("Mr. Anderson")) returns "Hello Mr. Anderson!"
```

formatForDisplay()

Da formato al parámetro de entrada y lo hace legible.

Finalidad

Esta función llama a la función "formatSizeForDisplay()" para dar formato al parámetro de entrada.

Sintaxis

```
formatForDisplay(val)
```

Parámetros de entrada

Variable *val*
La variable para dar formato.

Valores de retorno

String *vartoString*
Devuelve el resultado de la función "formatSizeForDisplay()" a la que se ha llamado.

formatSizeForDisplay()

Toma el parámetro de entrada y agrega o recorta la parte fraccionaria del parámetro de entrada a dos decimales, por ejemplo, 123 MB por 123,00 MB MB o 12,123 MBs por 12,12 MBs

Finalidad

Esta función cuenta el número de caracteres del parámetro de entrada, comprueba si es un número o una cadena y divide el parámetro de entrada en partes enteras y fraccionarias. Dependiendo de la parte fraccionaria, anexa o recorta a dos posiciones decimales. Devuelve el resultado.

Sintaxis

```
formatSizeForDisplay(size)
```

Parámetros de entrada

Integer *size*
El valor para redondear con dos decimales.

Valores de retorno

Integer *val*
Devuelve el valor redondeado a dos decimales.

getDecimalSeparator()

Determina el separador decimal que se utiliza con el entorno local actual.

Finalidad

Esta función crea un número fraccionario y luego utiliza la función `Mid ()` para determinar el separador decimal que se utiliza en ese número fraccionario.

Sintaxis

```
getDecimalSeparator()
```

Parámetros de entrada

None

Valores de retorno

Character *sep*
Devuelve el separador de decimal , o . para la configuración regional.

getFirstMatch()

Obtiene la primera coincidencia de la cadena de búsqueda en la matriz.

Finalidad

Esta función utiliza una expresión regular para buscar el patrón, que se pasa como parámetro de entrada, en la matriz que también se pasa como parámetro de entrada. Cuando encuentra la primera coincidencia del patrón de la matriz, añade el valor de la matriz al objeto de diccionario de secuencias de comandos.

Sintaxis

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

Parámetros de entrada

String *patt*
Contiene el patrón para buscar.

Array *arr*
Contiene la matriz en la que buscar el patrón de búsqueda.

Valores de retorno

Dictionary *keys*
Devuelve las claves del objeto de diccionario de secuencia de comandos.

isMatch()

Comprueba si el patrón de búsqueda se encuentra en la cadena.

Finalidad

Esta función llama a la función “getFirstMatch()”, pasando el patrón y la cadena (contenida en una matriz) como parámetros de entrada de esta función. Invoca a la función `ubound`para comprobar si el valor devuelto de la función `getFirstMatch()`

es mayor que o igual a 0. Si es así, existe una coincidencia; en caso contrario, no la hay.

Sintaxis

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

Parámetros de entrada

String *patt*
Contiene el patrón para buscar.

String *str*
Contiene la cadena en la que buscar el patrón de búsqueda.

Valores de retorno

Boolean *True|False*
Devuelve `True` si existe una coincidencia; de lo contrario, devuelve `False`.

notInLatter()

Filtra la primera matriz para determinar si el contenido se encuentra en la segunda matriz. En función del valor del parámetro de entrada `in_or_not`, la función devuelve el contenido de la primera matriz, incluyendo o excluyendo lo que coincide con la segunda matriz.

Finalidad

Sintaxis

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

Parámetros de entrada

Array *arr1*
copiar desde algún sitio

Array *arr2*
a otro sitio

String *in_or_out*
Contiene `"in"` o `"not"` dependiendo de si la función debe devolver el contenido de la primera matriz filtrada para devolver solo los contenidos que coincidían con la segunda matriz (`"in"`) o los contenidos que no coincidían con la segunda matriz (`"not"`).

Valores de retorno

Dictionary *keys*
Devuelve las claves del objeto de diccionario de secuencias de comandos que contiene la primera matriz filtrada para tener solo los contenidos que coincidían con la segunda matriz `(In_or_not = "en")` o los contenidos que no coincidían con la segunda matriz o `(In_or_not = "no")` .

passOrFail()

Compara los valores esperados y los reales de la propiedad de requisito previo y determina si la propiedad de requisito previo supera la comprobación. Los parámetros de entrada pueden ser números genéricos, un tamaño en MB o GB, la velocidad de la CPU en MHz o GHz, un booleano o cadenas.

Finalidad

Esta función llama en primer lugar a la función “changeMG()” en la página 126 para dar formato y si es necesario convertir los valores reales y esperados. Comprueba si alguno de los valores es 0, y si es así, devuelve "FAIL" y se cierra. Si los valores no son 0, la función comprueba si los valores son del tipo booleano, numérico, tamaño en MB o GB, velocidad de la CPU en MHz (sólo Windows) o GHz, o cadenas. A continuación compara los valores y devuelve el resultado.

Sintaxis

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

Parámetros de entrada

String *expect*
El valor esperado de la propiedad de requisito previo.

String *real*
El valor real de la propiedad de requisito previo.

Valores de retorno

String *passOrFail*
Devuelve `"PASS"` o `"FAIL"` dependiendo de si el valor esperado es igual o mayor que el valor real.

ppread()

Lee el contenido de un archivo en un objeto de diccionario de secuencias de comandos y divide aún más cada línea del archivo por el parámetro de entrada separador especificado si ese separador existe en la línea.

Finalidad

Esta función lee cada línea del archivo, elimina los espacios iniciales o finales, y comprueba si contiene el separador. Si contiene el separador, divide la línea por el separador, añadiendo cada fragmento como un elemento del objeto de diccionario; de lo contrario, añade una línea recortada a un elemento de objeto de diccionario. Devuelve una matriz que contiene el objeto de diccionario como el primer índice.

Sintaxis

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

Parámetros de entrada

String *fileName*
El nombre del archivo para leer en el objeto de diccionario.

Character *sep*
Carácter que representa el separador por el que se dividirá una línea en el archivo.

Valores de retorno

Array *array(dic)*
Devuelve una matriz con el objeto de diccionario (`DIC`) como su primer índice.

Ejemplo

Ejemplo que debe proporcionarse.

readFile()

Lee cada línea de un archivo en una entrada de índice de una matriz.

Finalidad

Esta función abre el archivo y lee cada línea del archivo en una entrada de índice de la matriz. Devuelve la matriz.

Sintaxis

```
readFile(fileName)
```

Parámetros de entrada

String *fileName*
El nombre del archivo para leer en la matriz.

Valores de retorno

Array *fileContents*
Devuelve la matriz con el contenido del archivo.

unitMGTOG()

Suma los contenidos de una matriz para obtener el número total de megabytes.

Finalidad

Esta función convierte el valor de cada índice de la matriz a MB y los suma.

Sintaxis

```
unitMGTOG(arr)
```

Parámetros de entrada

Array *arr*
Contiene la matriz.

Valores de retorno

String *unitMGTOG*
Devuelve el número total de los contenidos de la matriz en MB y anexa `"MB"` al total.

varToString()

Crea una representación en forma de cadena de una variable. La variable para comprobar puede ser una cadena, un número, un objeto de diccionario de secuencias de comandos, una matriz o un objeto `CheckItem`.

Finalidad

Esta función comprueba los datos o el tipo de objeto de la variable y llama a la función pertinente para crear una representación en forma de cadena de esos datos o del tipo de objeto.

Tabla 32. Función llamada para cada tipo de variable.

Tipo de variable	Función llamada
Matriz	“arrayToString()” en la página 125
Objeto `CheckItem`	“checkItemToString()” en la página 126
Objeto de diccionario de secuencias de comandos	“dictionaryToString()” en la página 127

Sintaxis

```
varToString(var)
```

Parámetros de entrada

`Variable` *`var`*
Las variables admitidas son: cadena, número, objeto de diccionario de secuencias de comandos, matriz u objeto CheckItem

Valores de retorno

`String` *`vartoString`*
Devuelve una representación en forma de cadena de la variable incluidos los valores devueltos por las funciones llamadas en los casos necesarios.

Apéndice F. Subrutinas de programa de utilidad de registro para sistemas Windows

IBM Prerequisite Scanner posee un conjunto de subrutinas de archivos comunes en el archivo `preq.vbs` para mostrar mensajes en la pantalla o escribir en el archivo de registro.

Tabla 33 describe las utilidades de registro.

Tabla 33. Subrutinas de programa de utilidad de registro

Subrutina	Descripción	Parámetros de entrada
`deleteLogFile`	Borra el archivo de registro, si existe.	Ninguno
`log(level, msg)`	Escribe el mensaje en el archivo de registro mediante el uso de la función "fmt()" en la página 130. El registro también incluye la fecha y hora actuales.	• `level`, cadena que establece el tipo de mensaje; por ejemplo, informativo o aviso • `msg`, cadena que representa el mensaje para registrar
`logDebug(msg)`	Llama a la función `log()`, pasando `"DEBUG"` como parámetro de entrada de nivel.	`msg`, cadena que representa el mensaje para registrar
`logError(msg)`	Llama a la función `log()`, pasando `"ERROR"` como parámetro de entrada de nivel.	`msg`, cadena que representa el mensaje para registrar
`logInfo(msg)`	Llama a la función `log()`, pasando `"INFO"` como parámetro de entrada de nivel.	`msg`, cadena que representa el mensaje para registrar
`logScreen(msg)`	Escribe el mensaje en la pantalla.	`msg`, cadena que representa el mensaje para escribir en la pantalla
`logScreenWith Replacement (msg, replaceStr)`	Escribe el mensaje en la pantalla, pasando por un código de mensaje y una cadena como parámetros de entrada.	• `msg`, mensaje de código que representa la cadena de mensaje para escribir en la pantalla • `replaceStr`, cadena que sustituye la *%variable* en el valor de código de mensaje
`logScreenWith MultiReplacements (msg, replaceStrArray)`	Escribe el mensaje en la pantalla, pasando por un código de mensaje y una matriz de cadena como parámetros de entrada.	• `msg`, mensaje de código que representa la cadena de mensaje para escribir en la pantalla • `replaceStrArray`, matriz de cadena con cada índice de la matriz que sustituye a una *%variable* en el valor de código de mensaje
`logWarning(msg)`	Llama a la función `log()`, pasando `"WARNING"` como parámetro de entrada de nivel.	`msg`, cadena que representa el mensaje para registrar

Apéndice G. Subrutinas de programa de utilidad de archivado para sistemas Windows

Prerequisite Scanner posee un conjunto de subrutinas de archivos comunes en el archivo `/lib/common_function.vbs` para gestionar archivos. También tiene un conjunto de funciones para gestionar archivos.

Tabla 34 describe los programas de utilidad de archivado.

Tabla 34. Subrutinas de programa de utilidad de archivado

Subrutina	Descripción	Parámetros de entrada
`appendToFile(text, fileName)`	Añade el texto al final del archivo especificado.	• `text`, cadena que contiene el texto para agregar al archivo • `filename`, cadena que representa el nombre del archivo para modificar
`writeToFile(text, fileName)`	Escribe el texto en el archivo especificado, sobrescribiendo el contenido existente si es necesario.	• `text`, cadena que contiene el texto para grabar en el archivo • `filename`, cadena que representa el nombre del archivo para modificar

Tabla 35 describe las funciones de archivado que gestionan archivos.

Tabla 35. Funciones de programa de utilidad de archivado

Función	Descripción
“allFiles()” en la página 124	Lee los nombres de los archivos de un directorio especificado en una matriz.
“filterFile()” en la página 128	Lee y filtra el contenido de un archivo a un objeto de diccionario de secuencias de comandos.
“findNewest()” en la página 129	Localiza el archivo de configuración más reciente.
“findSuitableFile()” en la página 129	Localiza el archivo de configuración pertinente de un producto y versión.
“ppread()” en la página 134	Lee el contenido de un archivo en un objeto de diccionario de secuencias de comandos y divide aún más cada línea del archivo por el parámetro de entrada separador especificado si ese separador existe en la línea.
“readFile()” en la página 135	Lee cada línea de un archivo en una entrada de índice de una matriz.

Apéndice H. Otras funciones comunes y subrutinas para sistemas Windows

Prerequisite Scanner tiene un conjunto de otras funciones comunes y subrutinas que se usan en varios archivos.

describe las otras funciones comunes y subrutinas.

Tabla 36. Otras funciones comunes y subrutinas para sistemas Windows

Función o subrutina	Descripción
"ffirstMatch()"	Obtiene la primera coincidencia de la cadena de búsqueda en la matriz.
"getValue()" en la página 142	Obtiene el espacio de disco disponible para un directorio especificado.
"removeSpecialCharacters()" en la página 143	Quita la marca registrada u otros caracteres especiales para facilitar las comparaciones.
"versionCompare()" en la página 143	Analiza los parámetros de entrada que representan los valores reales y esperados de una propiedad de requisito previo y los compara para determinar si la propiedad de requisito previo supera la comprobación de requisitos previos.La función espera cadenas de versión separadas por puntos como parámetros de entrada; pro ejemplo, 1.0.0.4, 2.3, 3.40.26.7800 o 2.3.*.

ffirstMatch()

Obtiene la primera coincidencia de la cadena de búsqueda en la matriz.

Finalidad

Esta función utiliza una expresión regular para buscar el patrón, que se pasa como parámetro de entrada, en la matriz que también se pasa como parámetro de entrada. Cuando encuentra la primera coincidencia del patrón de la matriz, añade el valor de la matriz al objeto de diccionario de secuencias de comandos.

Funciones principales

Tabla 37. Funciones principales que llaman a ffirstMatch

Función principal, secuencia de comandos	Descripción
`ud620db2level(expect, real) in DB2_Version_compare.vbs`	Compara los valores reales y esperados de la propiedad de requisito previo de versión de DB2.
`oslevelcompare(expect, real) in OS_Version_compare.vbs`	Compara los valores reales y esperados de la propiedad de requisito previo de versión de SO.

Sintaxis

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

Parámetros de entrada

String *patt*
Contiene el patrón para buscar.

Array *arr*
Contiene la matriz en la que buscar el patrón de búsqueda.

Valores de retorno

Dictionary *keys*
Devuelve las claves del objeto de diccionario de secuencia de comandos.

getValue()

Obtiene el espacio de disco disponible para un directorio especificado.

Finalidad

Esta subrutina utiliza la instancia del objeto de sistema de archivos para llamar a la función `getDriveName ()` del parámetro de entrada path y, a continuación, utiliza la propiedad `FreeSpace` para obtener el espacio en disco disponible y lo convierte a MBs. El parámetro de entrada de la propiedad de requisito previo y su valor se graban en el archivo de texto temporal asociado con el archivo de secuencia de comandos.

Secuencias de comandos

Tabla 38. Secuencias de comandos que utilizan getValue()

Secuencia de comandos	Descripción
`DEZ_01040000.vbs`	Secuencia de comandos para obtener propiedades de requisitos previos y ponerlas a disposición solo del archivo de configuración DEZ 01040000
`LCM_TAD_common.vbs`	Secuencia de comandos para obtener propiedades de requisitos previos y ponerlas a disposición solo de los archivos de configuración LCM 02300000 y TAD 07200000
`TAD722_impl.vbs`	Secuencia de comandos para obtener propiedades de requisitos previos y ponerlas a disposición solo del archivo de configuración TAD 07220000

Sintaxis

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

Parámetros de entrada

File system object *fso*
Instancia del objeto de sistema de archivos.

String *sKey*
Contiene una cadena con el nombre de la propiedad de requisito previo y el símbolo igual.

String *drvPath*
Contiene la ruta para la que se obtendrá el espacio en disco disponible.

Valores de retorno

None

removeSpecialCharacters()

Quita la marca registrada u otros caracteres especiales para facilitar las comparaciones.La función está en el archivo `/lib/common.vbs`.

Finalidad

Esta función llama a la función `Replace ()` para sustituir la marca, copyright y símbolos de marca registrada en `""` .

Sintaxis

```
removeSpecialCharacters(s)
```

Parámetros de entrada

String *s*
Contiene la cadena de que la que se deben quitar los caracteres

Valores de retorno

String *s*
Devuelve la cadena sin los caracteres especiales.

versionCompare()

Analiza los parámetros de entrada que representan los valores reales y esperados de una propiedad de requisito previo y los compara para determinar si la propiedad de requisito previo supera la comprobación de requisitos previos.La función espera cadenas de versión separadas por puntos como parámetros de entrada; pro ejemplo, 1.0.0.4, 2.3, 3.40.26.7800 o 2.3.*.

Finalidad

Esta función se encarga primero de los casos especiales, donde uno o los dos parámetros de entrada están vacíos y devuelve códigos de devolución para representar estos casos. Divide cada versión en varias partes mediante el separador de punto. Si la última parte de la versión es el carácter comodín *, la función considera todas las partes de la versión que faltan para suplir el carácter comodín; por ejemplo, 2.* coincide con 2.1 o 2.3.*. A continuación, recorre la lista de componentes de cada versión y los compara. Devuelve códigos de retorno dependiendo de si el valor esperado es menor que, igual que o mayor que el valor real.

Funciones principales

Tabla 39. Funciones principales que llaman a versionCompare

Función principal, secuencia de comandos	Descripción
`cygwinVersion_compare.vbs`	Compara los valores reales y esperados de la propiedad de requisito previo de versión de cygwin.

Tabla 39. Funciones principales que llaman a versionCompare (continuación)

Función principal, secuencia de comandos	Descripción
gskit7Version_compare.vbs	Compara los valores reales y esperados de la propiedad de requisito previo de gskit Versión 7.
gkit8Version_compare.vbs	Compara los valores reales y esperados de la propiedad de requisito previo de gskit Versión 8.
internetExplorer.version_compare.vbs	Compara los valores reales y esperados de la propiedad de requisito previo de la versión de Internet Explorer.
os.servicePack_compare.vbs	Compara los valores reales y esperados de la propiedad de requisito previo del Service Pack de SO.
os.versionNumber_compare.vbs	Compara los valores reales y esperados de la propiedad de requisito previo de versión de SO.

Sintaxis

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

Parámetros de entrada

String *ver1*
Contiene la versión prevista de una propiedad de requisito previo.

String *ver2*
Contiene la versión real de una propiedad de requisito previo.

Valores de retorno

Integer *0*
Devuelve el código de retorno 0 si los dos parámetros de entrada son iguales. La función principal devuelve "PASS".

Caso especial: devuelve el código de retorno 0 y se cierra si los dos parámetros de entrada están vacíos.

Integer *-1*
Devuelve el código de retorno -1 si el primer parámetro de entrada es menor que el segundo. La función principal devuelve "FAIL".

Caso especial: devuelve el código de retorno -1 y se cierra si el primer parámetro de entrada está vacío.

Integer *1*
Devuelve el código de retorno 1 si el primer parámetro de entrada es mayor que el segundo. La función principal devuelve "PASS".

Caso especial: devuelve el código de retorno -1 y se cierra si el segundo parámetro de entrada está vacío.

Apéndice I. Funciones comunes de los sistemas UNIX

Prerequisite Scanner tiene un conjunto de funciones comunes en el archivo `/lib/common_function.sh` para ejecutar comprobaciones en sistemas basados en UNIX.

Tabla 40. Funciones de common_function.sh

Función	Descripción
"AddMG()" en la página 146	Comprueba si los parámetros de entrada están en MB o GB y agrega los parámetros.
"changeMG()"	Convierte el parámetro de entrada a MB o GB para las propiedades de requisito previo de espacio en disco o de memoria.
"compare()" en la página 147	Analiza los parámetros de entrada que representan los valores reales y esperados para una propiedad de requisito previo y los compara para determinar si el primer valor (real) es mayor que el segundo valor (esperado).
"cutdown()" en la página 147	Analiza los parámetros de entrada que representan los valores reales y esperados para una propiedad de requisito previo y los compara para determinar si el primer valor (real) es mayor que el segundo valor (esperado).A continuación, se imprime la diferencia entre los dos valores si el primer valor no es menor que el segundo valor.
"findOSInfo()" en la página 149	Localiza la versión de sistema operativo, el nivel y versión de release del SO y los datos de implementación de hardware del sistema.
"mes4path()" en la página 148	Localiza el espacio libre en disco para cada sistema de archivos montado.
"mes4Path1()" en la página 148	Localiza el espacio libre en disco para cada sistema de archivos montado en un sistema Solaris únicamente.
"NFScheck()" en la página 150	Comprueba el estado NFS de los montajes en un sistema basado en UNIX.
"telnetNFS()" en la página 150	Comprueba si puede realizar envíos mediante telnet a la dirección IP de un sistema de archivos montado en el puerto predeterminado 2049.

changeMG()

Convierte el parámetro de entrada a MB o GB para las propiedades de requisito previo de espacio en disco o de memoria.

Finalidad

Esta función comprueba primero que la función reciba un parámetro de entrada. Si recibe un parámetro de entrada, determina si el valor está en MB o GB y convierte el valor a MB o GB respectivamente.

Sintaxis

```
changeMG val
```

Parámetros de entrada

String *$val*
Contiene el valor de espacio de disco o memoria en MB o GB.

Valores de retorno

Integer *1*
Devuelve 1 si la función no recibe un parámetro de entrada.

String *printf "%.0fM%s",mm[1]*1024,mm[2];*
Devuelve el valor en MB.

String *printf "%.2fG%s",mm[1],mm[2];*
Devuelve el valor en GB.

AddMG()

Comprueba si los parámetros de entrada están en MB o GB y agrega los parámetros.

Finalidad

Esta función comprueba primero que la función reciba parámetros de entrada. Si recibe un parámetro de entrada, determina si el valor está en MB o GB y luego suma los valores.

Sintaxis

```
AddMG val1 val2
```

Parámetros de entrada

String *$val1*
Contiene el valor de espacio en el disco o la memoria en MB o GB que se añadirá al otro parámetro de entrada.

String *$val2*
Contiene el valor de espacio en el disco o la memoria en MB o GB que se añadirá al otro parámetro de entrada.

Valores de retorno

Integer *1*
Devuelve 1 si la función no recibe dos parámetros de entrada.

String *val*
Devuelve los valores añadidos en MB o GB.

compare()

Analiza los parámetros de entrada que representan los valores reales y esperados para una propiedad de requisito previo y los compara para determinar si el primer valor (real) es mayor que el segundo valor (esperado).

Finalidad

Esta función comprueba primero que la función reciba dos parámetros de entrada. Si los recibe y no son falsos, determina si los valores son en MB o GB y luego compara los dos valores para comprobar si el primer valor es menor que el segundo. Si es así, devuelve un valor false; de lo contrario, devuelve un valor pass.

Sintaxis

```
compare real expected
```

Parámetros de entrada

String *$real*
Contiene el valor real de una propiedad de requisito previo.

String *$expected*
Contiene el valor previsto de una propiedad de requisito previo.

Valores de retorno

Integer *1*
Devuelve 1 si la función no recibe dos parámetros de entrada.

String *"FAIL|PASS"*
Devuelve la cadena "FAIL" si el valor real es menor que el valor esperado; de lo contrario, devuelve la cadena "PASS".

cutdown()

Analiza los parámetros de entrada que representan los valores reales y esperados para una propiedad de requisito previo y los compara para determinar si el primer valor (real) es mayor que el segundo valor (esperado).A continuación, imprime la diferencia entre los dos valores si el primer valor no es menor que el segundo.

Finalidad

Esta función comprueba primero que la función reciba dos parámetros de entrada. Si los recibe, determina si los valores están en MB o GB y los convierte a MB si están en GB. A continuación, compara los dos valores para comprobar si el primer valor es menor que el segundo. Si lo es, devuelve un valor "0MB"; de lo contrario, devuelve la diferencia entre los dos valores en MB.

Sintaxis

```
cutdown real expected
```

Parámetros de entrada

String *$real*
Contiene el valor real de una propiedad de requisito previo.

String *$expected*
Contiene el valor previsto de una propiedad de requisito previo.

Valores de retorno

Integer *1*
Devuelve 1 si la función no recibe dos parámetros de entrada.

String *"FAIL|PASS"*
Devuelve la cadena "FAIL" si el valor real es menor que el valor esperado cuando ningún valor está en MB o GB; de lo contrario, devuelve la cadena "PASS".

String *"OMB|Real-ExpectedMB"*
Devuelve la cadena "OMB" si el valor real es menor que el valor esperado; de lo contrario, devuelve una cadena que representa la diferencia entre los dos valores convertidos en MB.

mes4path()

Localiza el espacio libre en disco para cada sistema de archivos montado.

Finalidad

Esta función toma una ruta como entrada, llama al comando **uname** para determinar el sistema operativo; a continuación, llama a la función NFScheck para determinar si el sistema está funcionando y los montajes. Después, llama al comando df para determinar el espacio de disco libre para cada montaje de un sistema. Devuelve el valor del espacio libre en disco.

Sintaxis

```
mes4Path path
```

Parámetros de entrada

String *$path*
Ruta de acceso al sistema para comprobar si hay espacio libre en disco.

Valores de retorno

Integer *1*
Devuelve el código de retorno 1 si la función no recibe un parámetro de entrada.

Integer *2*
Devuelve el código de retorno 2, si el parámetro de entrada no es una ruta.

String *$NF*
Devuelve el espacio libre en disco para cada montaje.

String *"$path Server NotAvailable Responding for $path"*
Devuelve un mensaje para indicar que el servidor de la ruta no está disponible.

mes4Path1()

Localiza el espacio libre en disco para cada sistema de archivos montado en un sistema Solaris únicamente.

Finalidad

Esta función toma una ruta como entrada, llama al comando **uname** para determinar si el sistema operativo es Solaris. Después, llama al comando df para

determinar el espacio de disco libre para cada montaje del sistema. Devuelve el valor del espacio libre en disco.

Sintaxis

```
mes4Path1 path
```

Parámetros de entrada

String *$path*
Ruta de acceso al sistema para comprobar si hay espacio libre en disco.

Valores de retorno

Integer *1*
Devuelve el código de retorno 1 si la función no recibe un parámetro de entrada.

Integer *2*
Devuelve el código de retorno 2, si el parámetro de entrada no es una ruta.

String *$NF*
Devuelve el espacio libre en disco para cada montaje.

findOSInfo()

Localiza la versión de sistema operativo, el nivel y versión de release del SO y los datos de implementación de hardware del sistema.

Finalidad

Esta función ejecuta el comando `uname` y analiza su salida para la la versión del sistema operativo, nivel de release de sistema operativo y versión, así como datos de implementación de hardware del sistema.

Sintaxis

```
findOSInfo
```

Parámetros de entrada

Ninguno

Valores de retorno

String *$oo*
Salida de **uname** sin la información básica del sistema.

String *$kk*
Versión del sistema operativo

String *$hh*
Implementación de hardware representada como I para el hardware i386 o Z para el hardware s390.

String *$rr*
Nivel de release del sistema operativo

String *$vv*
Versión de nivel de release del sistema operativo

telnetNFS()

Comprueba si puede realizar envíos mediante telnet a la dirección IP de un sistema de archivos montado en el puerto predeterminado 2049.

Finalidad

Esta función toma una IP como entrada y llama al comando **telnet** para probar si la conexión remota se ha realizado correctamente en el puerto telnet predeterminado 2049. Intenta la conexión remota 10 veces. Si el comando **telnet** falla, la función devuelve el valor "FALSE"; en caso contrario, devuelve el valor "PASS".

Sintaxis

```
telnetNFS ipaddr
```

Parámetros de entrada

String *$ipaddr*
La dirección IP para comprobar si se pueden realizar envíos mediante telnet.

Valores de retorno

String *"FALSE|TRUE"*
Devuelve el resultado de la comprobación de telnet. Devuelve "TRUE" si la comprobación es correcta; de lo contrario, devuelve "FALSE".

NFScheck()

Comprueba el estado NFS de los montajes en un sistema basado en UNIX.

Finalidad

Esta función toma una ruta como entrada y llama al comando mount para obtener la lista de sistemas de ficheros montados. Llama al comando **uname** para determinar el sistema operativo. A continuación, llama al comando **ping** para hacer ping a cada sistema montado y si puede hacer ping, llama a la función **telnetNFS** para comprobar si se puede realizar una conexión remota. Si las acciones ping o telnet fallan, la función devuelve el valor "FALSE"; en caso contrario, devuelve el valor "PASS".

Sintaxis

```
NFScheck path
```

Parámetros de entrada

String *$path*
Toma una ruta válida a un directorio como entrada.

Valores de retorno

Boolean value *TRUE* **or** *FALSE*
Devuelve TRUE si la comprobación NFS es correcta, es decir, si hace ping de forma satisfactoria a la dirección IP asociada o puede utilizar telnet para establecer una conexión con la dirección IP de cada sistema de archivos; en caso contrario, devuelve FALSE.

Ejemplo

Este ejemplo de uso es de ola función **mes4Path()**:

```
# 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
     ...
```

Apéndice J. Otras funciones de los sistemas UNIX

Prerequisite Scanner tiene un conjunto de funciones comunes en varios archivos.

Tabla 41 describe el conjunto de funciones en varios archivos.

Tabla 41. Funciones comunes de varios archivos

Función	Descripción
"formatSizeDisplay()" en la página 154	Toma el parámetro de entrada y agrega o recorta la parte fraccionaria del parámetro de entrada a dos decimales, por ejemplo, 123 MB por 123,00 MB MB o 12,123 MBs por 12,12 MBs
"versionCompare()" en la página 154	Analiza los parámetros de entrada que representan los valores reales y esperados de una propiedad de requisito previo y compara cada parte de la versión para determinar si la propiedad de requisito previo supera la comprobación de requisitos previos.

Tabla 42 describe el conjunto de funciones del archivo `UNIX-Linux/TAD722_impl.sh` para ejecutar comprobaciones para Tivoli License Compliance Manager y Tivoli Asset Discovery for Distributed.

Tabla 42. Funciones comunes de TAD722_impl.sh

Función	Descripción
"checkSunOS()" en la página 156	Verifica si la versión del sistema operativo Solaris es para plataformas SPARC o X86.
"checkHpux()" en la página 156	Verifica si la versión del sistema operativo HP-UX es para plataformas IA64 o PARISC.
"checkLinux()" en la página 156	Comprueba si la versión del sistema operativo Linux es para plataformas System p, System z o x86.
"getSystemId()" en la página 158	Llama a diferentes funciones del sistema operativo para comprobar las plataformas del sistema operativo pertinente.
"getValue()" en la página 157	Obtiene el valor de una clave en un archivo especificado si la clave existe.
"setValue()" en la página 157	Establece el valor de una clave en un archivo especificado si la propiedad de requisito previo existe.
"copyValue()" en la página 157	Obtiene y establece la propiedad de requisito previo (clave) basada en el producto y el sistema operativo.
"parseDirParameter()" en la página 159	Analiza el parámetro de la lista de parámetros para el distintivo -p del escáner y coloca su valor en la lista.
"getClosestExistingParentDir()" en la página 158	Obtiene el directorio principal más cercano o a sí mismo.

Tabla 42. Funciones comunes de TAD722_impl.sh (continuación)

Función	Descripción
"printDirSize()" en la página 159	Comprueba el estado NFS del sistema de archivos montado y luego obtiene el espacio de disco del sistema de archivos o su directorio padre.

formatSizeDisplay()

Toma el parámetro de entrada y agrega o recorta la parte fraccionaria del parámetro de entrada con dos decimales, por ejemplo, MB 123 a 123.00 MB o 12.123 MBs a 12.12 MB

Finalidad

Esta función cuenta el número de caracteres del parámetro de entrada, comprueba si es un número o una cadena y divide el parámetro de entrada en partes enteras y fraccionarias. Dependiendo de la parte fraccionaria, anexa o recorta a dos decimales. Devuelve el resultado.

Secuencias de comandos principales

Los siguientes scripts contienen la función:

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

Sintaxis

```
formatSizeDisplay val
```

Parámetros de entrada

Integer *$val*
El valor para redondear con dos decimales.

Valores de retorno

Integer *val*
Devuelve el valor redondeado a dos decimales.

versionCompare()

Analiza los parámetros de entrada que representan los valores reales y esperados para una propiedad de requisito previo y se compara cada parte de la versión para determinar si el primer valor (real) es mayor que el segundo valor (esperado).

Finalidad

Esta función comprueba primero que la función recibe dos versiones como parámetros de entrada. Se utiliza awk para analizar y dividir cada versión en sus partes cuando "." es el delimitador para dividir el valor en partes. A continuación, realiza un bucle para comparar cada parte de la primera versión contra la misma parte de la segunda versión y ver si son iguales.

Funciones principales

Tabla 43. Funciones principales que llaman a versionCompare

Función principal, script	Descripción
db2.home_compare.sh	Compara los valores reales y previstos para el espacio en disco para la propiedad de requisito previo HOME de DB2.
oracle.Client_compare.sh	Compara los valores reales y esperados de la propiedad de requisito previo de cliente de Oracle.
os.locale_compare.sh	Compara los valores reales y esperados de la propiedad de requisito previo de entorno local de SO.
os.MozillaVersion_compare.sh	Compara los valores reales y esperados de la propiedad de requisito previo de Mozilla Firefox.
os.package.perl_compare.sh	Compara los valores reales y esperados de la propiedad de requisito previo de paquete de Perl.Se llama a sí misma.
os.RAMSize_compare.sh	Compara los valores reales y esperados de la propiedad de requisito previo de RAM.
os.space_compare.sh	Compara los valores reales y esperados de la propiedad de requisito previo de espacio de disco disponible.
OS_Version_compare.sh	Compara los valores reales y esperados de la propiedad de requisito previo de versión de SO.

Sintaxis

```
versionCompare real expected
```

Parámetros de entrada

String *$real*
Contiene el valor real de una propiedad de requisito previo.

String *$expected*
Contiene el valor previsto de una propiedad de requisito previo.

Valores de retorno

Integer *0*
Devuelve el código de retorno 0 si el valor real y esperado son iguales. La función principal devuelve "PASS".

Caso especial: Devuelve el código de retorno 0 y se cierra si la función recibe parámetros de entrada vacíos.

Integer *-1*
Devuelve el código de retorno -1 si el valor real es menor que el valor previsto. La función principal devuelve `"FAIL"`.

Devuelve el código de retorno -1 y se cierra si la función recibe el segundo parámetro de entrada vacío.

Integer *1*
Devuelve el código de retorno 1 si el valor real es mayor que el valor previsto. La función principal devuelve "PASS".

Devuelve el código de retorno 1 y se cierra si la función recibe el primer parámetro de entrada vacío.

checkHpux()

Verifica si la versión del sistema operativo HP-UX es para plataformas IA64 o PARISC.

Finalidad

Esta función utiliza el distintivo -m del comando **uname** para determinar si el sistema operativo HP-UX es para plataformas IA64 o PARISC.

Sintaxis

```
checkHpux
```

Valores de retorno

String *HPUXIA64*|*HPUXPARISC*
Devuelve "HPUXIA64" si el distintivo -m es "ia64"; en caso contrario, devuelve "HPUXPARISC".

checkLinux()

Comprueba si la versión del sistema operativo Linux es para plataformas System p, System z o x86.

Finalidad

Esta función utiliza el distintivo -m del comando **uname** para determinar si el sistema operativo Linux es para plataformas System p, System z o x86.

Sintaxis

```
checkLinux
```

Parámetros de entrada

Valores de retorno

String *LINUXPSERIES*|*LINUXZSERIES*|*LINUXX86*
Devuelve "LINUXPSERIES" si el distintivo -m es "ppc64" o "ppc". Devuelve "LINUXZSERIES" si el valor es "S390x" o "S390"; de lo contrario, devuelve "LINUXX86".

checkSunOS()

Verifica si la versión del sistema operativo Solaris es para plataformas SPARC o X86.

Finalidad

Esta función utiliza el distintivo -p del comando **uname** para determinar si el sistema operativo Solaris es para plataformas SPARC o X86.

Sintaxis

```
checkSunOS
```

Parámetros de entrada

Valores de retorno

String *SOLARISSPARC|SOLARISX86*
Devuelve "SOLARISSPARC" si el distintivo -p es "sparc"; en caso contrario, devuelve "SOLARISX86".

getValue()

Obtiene el valor de una clave en un archivo especificado si la clave existe.

Finalidad

Sintaxis

```
getValue key file
```

Parámetros de entrada

String *$key*
Contiene la clave para establecer.

String *$file*
Contiene el nombre del archivo que contiene la clave.

setValue()

Establece el valor de una clave en un archivo especificado si la propiedad de requisito previo existe.

Sintaxis

```
setValue key value file
```

Parámetros de entrada

String *$key*
Contiene la propiedad de requisito previo para establecer.

String *$value*
Contiene el valor de la propiedad de requisito previo.

String *$file*
Contiene el nombre del archivo que contiene la propiedad de requisito previo.

copyValue()

Obtiene y establece la propiedad de requisito previo (clave) basada en el producto y el sistema operativo.

Finalidad

Esta función llama a la función **getValue ()** para obtener el valor de la propiedad de requisito especificada para el producto y sistema operativo. A continuación, llama a la función **setValue ()** para establecer el valor de la propiedad de requisito previo en el archivo Prerequisite Scanner.

Sintaxis

```
copyValue key file
```

Parámetros de entrada

String *$key*
Contiene la clave para obtener y establecer.

String *$file*
Contiene el nombre del archivo que contiene la clave.

Valores de retorno

getSystemId()

Llama a diferentes funciones del sistema operativo para comprobar las plataformas del sistema operativo pertinente.

Finalidad

Esta función llama a diferentes funciones del sistema operativo para determinar las plataformas del sistema operativo correspondiente.

Sintaxis

```
getSystemId
```

Parámetros de entrada

Valores de retorno

String *AIX|Linux*
Devuelve "AIX" o "Linux" si el producto es Tivoli License Compliance Manager y el SO es AIX o Linux o "AIX" si el producto es Tivoli Asset Discovery for Distributed u el SO es AIX.

getClosestExistingParentDir()

Obtiene el directorio principal más cercano o a sí mismo.

Finalidad

Sintaxis

```
getClosestExistingParentDir dirpath
```

Parámetros de entrada

String *$dirpath*
Contiene la ruta para obtener su directorio principal o a sí mismo.

Valores de retorno

String *dirpath*
Devuelve el directorio principal o a sí mismo

parseDirParameter()

Analiza el parámetro de la lista de parámetros para el distintivo -p del escáner y coloca su valor en la lista.

Finalidad

Sintaxis

Parámetros de entrada

String

Valores de retorno

printDirSize()

Comprueba el estado NFS del sistema de archivos montado y luego obtiene el espacio de disco del sistema de archivos o su directorio padre.

Finalidad

Esta función llama en primer lugar a la función **NFScheck** para determinar el estado NFS del directorio. Si el estado es verdadero, llama a la función **getClosestExistingParentDir** para devolver el directorio o su directorio padre y, a continuación, utiliza el comando **df** para obtener el espacio libre de disco disponible. Por último, llama a la función **formatSizeDisplay** para redondear el valor a puntos decimales.

Sintaxis

```
printDirSize dirpath
```

Parámetros de entrada

String *$dirpath*
Contiene la ruta del directorio para el que se obtendrá el espacio libre en disco.

Valores de retorno

Integer *dsize*
Devuelve la cantidad de espacio libre en disco con dos decimales.

String *"NFS_NOT_AVAILABLE"*
Devuelve que el sistema de archivos montado no está disponible.

Apéndice K. Funciones de utilidad de registro para sistemas UNIX

Prerequisite Scanner tiene un conjunto de funciones comunes en el archivo `/lib/common_function.sh` para grabar datos de depuración y rastreo en archivos de registro.

Tabla 44 describe las utilidades de registro.

Tabla 44. Funciones de utilidad de registro en sistemas UNIX

Función	Descripción	Parámetros de entrada
`wrlTrace` *log_str1 log_str2*	Escribe las cadenas *log_str1* y *log_str2* en el archivo de rastreo, con la indicación de fecha y hora	*log_str1* y *log_str2*, cadenas de rastreo que representan la acción y el recopilador que se está ejecutando y en los que debe registrarse el archivo de rastreo. Por ejemplo: **`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*	Pasa la función *fn_name* a `wrlTrace()`	*fn_name*, cadena de rastreo que representa la función a la que se acaba de llamar. Por ejemplo: `` `wrlTraceFuncStart "$1"` ``
`wrlTraceFuncExit` *fn_name*	Pasa la función *fn_name* a `wrlTrace()`	*fn_name*, cadena de rastreo que representa la función que acaba de completarse. Por ejemplo: `` `wrlTraceFuncExit "$1"` ``
`wrlDebug` *log_str1 log_str2*	Pasa las cadenas *log_str1* y *log_str2* a `wrlDebugGeneric()`	*log_str1* y *log_str2*, cadenas de depuración que representan la acción y el recopilador que se está ejecutando y en los que debe registrarse el archivo de depuración. Por ejemplo: `wrlTrace Starting os.lib` `wrlTrace Executing os.lib` **`wrlDebug Starting os.lib`** **`wrlDebug Expected libXp `** `ss=`./os.lib libXp libXp` `wrlTrace Finished os.lib` `echo "os.lib.libXp=$ss"` **`wrlDebug Finished os.lib`** **`wrlDebug OutPutValueIs $ss`** `wrlTrace Done os.lib`
`wrlDebugFuncStart` *fn_name*	Pasa la función *fn_name* a `wrlDebug()`	*fn_name*, cadena de depuración que representa la función a la que se acaba de llamar. Por ejemplo: `` `wrlDebugFuncStart "$1"` ``

Tabla 44. Funciones de utilidad de registro en sistemas UNIX (continuación)

Función	Descripción	Parámetros de entrada
`wrlDebugFuncExit` *fn_name*	Pasa la función *fn_name* a `wrlDebug()`	*fn_name*, cadena de depuración que representa la función que acaba de completarse. Por ejemplo: `` `wrlDebugFuncExit "$1"` ``
`wrlDebugFuncReturn` *result_value*	Escribe el resultado *result_value* devuelto para la función en el archivo de registro.	*result_value*, cadena de depuración que representa el valor devuelto por la función. Por ejemplo: `` `wrlDebugFuncReturn "$versionCompare"` ``
`wrlDebugFuncParam` *param1 param2*	Pasa los parámetros *param1* y *param2* a `wrlDebugFunc()`	*param1* y *param2*, cadenas de depuración que representan títulos de sección analizados, calificadores analizados o argumentos de entrada analizados para las funciones llamadas. Por ejemplo: `` `wrlDebugFuncParam "OSArch" "$3"` ``
`wrlDebugGeneric formatspec` *log_str1 log_str2*	Escribe cadenas *log_str1* y *log_str2* en el archivo de depuración, formateado por el argumento de cadena *formatspec*	• *log_str1* y *log_str2*, cadenas que representan datos específicos para registrar en una línea del archivo de depuración • *formatspec*, argumento de cadena para colocar después de la indicación de fecha y hora, pero antes de la alineación a la izquierda de las cadenas de registro y el carácter de nueva línea Por ejemplo: `` `wrlDebugGeneric "" "$1" "$2"` ``
`wrlDebugFunc` *str*	Pasa un carácter de tabulación y el parámetro de entrada *str* a `wrlDebugGeneric ()`	*str*, cadena que representa datos para registrar, es decir, el estado de una comprobación o una acción que se está realizando. Por ejemplo: `` `wrlDebugFunc "Reading config file and parsing using parse array..." ` ``
`wrlLogFuncStart` *str*	Pasa el parámetro de entrada *str* a `wrlTraceFuncStart()` y `wrlDebugFuncStart()`	*str*, cadena que representa datos para registrar, es decir, el nombre de la función a la que se está llamando. Por ejemplo: `` `wrlLogFuncStart "main()"` ``
`wrlLogFuncExit` *str*	Pasa el parámetro de entrada *str* a `wrlTraceFuncExit()` y `wrlDebugFuncExit()`	*str*, cadena que representa datos para registrar, es decir, el nombre de la función de la que se está saliendo. Por ejemplo: `` `wrlLogFuncExit "main()"` ``

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en los Estados Unidos. Es posible que IBM no ofrezca los productos, servicios o características tratados en este documento en otros países. Póngase en contacto con el representante de IBM local para obtener información sobre los servicios y productos actualmente disponibles en su área. Cualquier referencia a un producto, programa o servicio de IBM no implica que solo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar puede utilizarse cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de productos, programas o servicios no IBM.

IBM puede tener patentes o solicitudes de patente en tramitación que abarquen temas descritos en el presente documento. La entrega de este documento no le otorga ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a:

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

Para las consultas sobre licencias referentes a información de doble byte (DBCS), póngase en contacto con el Departamento de propiedad intelectual de IBM en su país o envíe las consultas, por escrito, a:

Licencias de propiedad intelectual
Ley de propiedad intelectual y legal
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país en el que tales disposiciones sean incoherentes con la legislación local

INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍAS DE NINGÚN TIPO NI EXPLÍCITAS NI IMPLÍCITAS, INCLUYENDO PERO NO LIMITÁNDOSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO.

Algunos países no permiten la renuncia a garantías expresas o implícitas en ciertas transacciones, por lo que el párrafo anterior puede no aplicarse en su caso.

Esta información podría incluir imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; estos cambios se incorporarán en las nuevas ediciones de la publicación. IBM puede realizar en cualquier momento mejoras o cambios en los productos o programas descritos en esta publicación sin previo aviso.

Cualquier referencia que se haga en esta información a sitios web que no sean de IBM se proporciona, únicamente, a efectos de comodidad, y de ninguna manera sirve de endoso de dichos sitios web. La información de esos sitios Web no forma parte de la información del presente producto de IBM y la utilización de esos sitios Web se realiza bajo la responsabilidad del usuario.

IBM puede utilizar o distribuir cualquier información que se le proporcione en la forma que considere adecuada, sin incurrir por ello en ninguna obligación para con el remitente.

Los titulares de licencias de este programa que deseen información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) la utilización mutua de la información intercambiada, deben ponerse en contacto con:

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

Dicha información puede estar disponible, sujeta a los términos y condiciones correspondientes, incluyendo, en algunos casos, el pago de una tarifa.

IBM proporciona el programa bajo licencia descrito en este documento y todo el material bajo licencia disponible para el mismo bajo los términos del Contrato de cliente IBM, el Acuerdo Internacional de Programas bajo Licencia IBM o de cualquier acuerdo equivalente entre las dos partes.

Los datos de rendimiento contenidos en él se determinaron en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar de forma significativa. Algunas mediciones pueden haberse realizado en sistemas a nivel de desarrollo y no se garantiza que sean las mismas en los sistemas comercializados. Además, algunas medidas se pueden haber estimado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables para su entorno específico.

La información relativa a productos no IBM se ha obtenido de los proveedores de estos productos, su publicidad u otras fuentes públicamente disponibles. IBM no ha comprobado estos productos y no puede confirmar la precisión del rendimiento, la compatibilidad ni cualquier otra reclamación relacionada con los productos que no son de IBM. Las consultas acerca de las posibilidades de los productos no IBM deben dirigirse a los proveedores de los mismos.

Todas las declaraciones relativas a la dirección o intenciones de IBM en el futuro están sujetas a cambios o a ser retiradas sin previo aviso, y sólo representan metas y objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos de la manera más completa posible, los ejemplos incluyen los nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es mera coincidencia.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente que ilustran técnicas de programación en varias plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo en cualquier formato sin necesidad de efectuar ningún pago a IBM, con el fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación que se ajusten a la interfaz de programación de aplicaciones para la plataforma operativa para la cual se han escrito los programas de aplicación. Estos ejemplos no se han probado de forma exhaustiva bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni implicar la fiabilidad, servicio o funcionamiento de estos programas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier forma sin pago a IBM con el fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación que se ajusten a las interfaces de programación de aplicaciones de IBM.

Si está visualizando esta información en copia software, es posible que las fotografías y las ilustraciones en color no aparezcan.

Marcas registradas

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp. en muchas jurisdicciones de todo el mundo. Puede que otros productos o nombres de servicio sean marcas registradas de IBM u otras compañías. Una lista actualizada de marcas registradas de IBM está disponible en el sitio Web “Copyright and trademark information” et www.ibm.com/legal/copntrade.shtml.

Adobe, Acrobat, PostScript y todas las marcas registradas basadas en Adobe son marcas registradas o comerciales de Adobe Systems Incorporated en Estados Unidos y/o en otros países.

Cell Broadband Engine y Cell/B.E. son marcas registradas de Sony Computer Entertainment, Inc., en Estados Unidos o en otros países, y se utilizan bajo su licencia.

Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Intel Centrino, el logotipo de Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium y Pentium son marcas registradas o marcas comerciales registradas de Intel Corporation o de sus filiales en Estados Unidos Y/o en otros países.

IT Infrastructure Library es una marca registrada de Central Computer and Telecommunications Agency que ahora forma parte de la Office of Government Commerce.

ITIL es una marca registrada, una marca registrada comunitaria de la OGC británica (Office of Government Commerce), y está registrada en la Oficina de Patentes y Marcas de Estados Unidos.

Linux es una marca registrada de Linus Torvalds en Estados Unidos, en otros países o en ambos.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en Estados Unidos y/o en otros países.

UNIX es una marca registrada de The Open Group en Estados Unidos y/o en otros países.

Java y todas las marcas registradas y los logotipos basados en Java son marcas registradas o marcas registradas comerciales de Sun Microsystems, Inc. en Estados Unidos y en otros países.

Otros nombres de empresas, productos y servicios son marcas registradas o marcas de servicio de terceros.

Información de soporte y feedback

Si tiene algún problema con el software de IBM, deseará resolverlo rápidamente. IBM proporciona diferentes formas para que pueda obtener la asistencia que necesita; por ejemplo, Internet o IBM Support Assistant. También puede proporcionar información o presentar solicitudes de productos para mejoras.

Internet

Los siguientes sitios contienen información sobre resolución de problemas:

- Vaya a la página IBM Prerequisite Scanner en IBM Support Portal.
- Vaya a los temas relativos a Prerequisite Scanner en Service Management Connect. No dude en hacer sus propias aportaciones a estos temas si lo desea.

Utilice los siguientes sitios para proporcionar comentarios, presentar solicitudes o hablar de Prerequisite Scanner :

- Vaya a los temas relativos a Prerequisite Scanner en Prerequisite Scanner at Service Management Connect No dude en hacer sus propias aportaciones a estos temas si lo desea.
- Utilice elIntegrated Service Management Message Board en Service Management Connect.
- Envíe o revise solicitudes de mejoras de producto para Prerequisite Scanner en Tivoli RFE Community.

IBM Support Assistant

IBM Support Assistant (ISA) es un entorno de trabajo local gratuito de provisión de servicio de software que ayuda al usuario a resolver preguntas y problemas relacionados con productos de software de IBM. ISA proporciona un rápido acceso a información relacionada con soporte y herramientas de prestación de servicio para la determinación de problemas. Para instalar el software ISA, vaya a http://www.ibm.com/software/support/isa

Índice

A

C

D

E

F

I

M

N

O

P

R

S

T

U

V

W

X

Impreso en España