Tivoli Provisioning Manager for OS Deployment

Reference Guide

Tivoli Provisioning Manager for OS Deployment

Reference Guide

Contents

Chapter 1. Keyboard navigation on the target

When a target is controlled by the OS deployment server and is in the PXE kernel, you can navigate using the keyboard.

The keys described in Table 1 are assigned navigational functions when the target is controlled by the OS deployment server and is in the PXE kernel. If the controlled target is in the PXE kernel, the word *kernel* appears in the lower right corner of the target screen.

Table 1. Keys and actions

Key or key combination	**Action**
Enter	Selects
Tab	Goes to the next control
Esc	Deletes content of a text field Exits from a menu
Ctrl+S and Ctrl-Esc	Opens the start menu
Alt+F4	Closes the window
Alt+Tab	Toggles between windows
Alt+Right Arrow	Scrolls right
Alt+Left Arrow	Scrolls left
Alt+Page Down	Scrolls down
Alt+Page Up	Scrolls up
Print Screen	Takes a screen snapshot and stores it under `files/global/snapshot.png`
Alt+Print Screen	Takes a window snapshot and stores it under `files/global/snapshot.png`
Ctrl+Alt+Backspace	Reinitializes USB devices

If you have difficulties in getting the focus on a window when only one is open, open and close the start menu with **Ctrl+S**, followed by **Esc**.

Chapter 2. The web interface

The web interface is the most common way of interacting with the provisioning server. It is an Web-based console viewed in a standard browser and does not have to run on the same computer as the provisioning server.

The supported browsers are Internet Explorer and Mozilla Firefox. For the supported Internet Explorer and Mozilla Firefox versions, see Web interface requirements.

Note: Mozilla Firefox 2.0.0.2 does not work on Linux PowerPC® 64-bits.

To reach the web interface, type the name of the computer on which the provisioning server is located (or `localhost` if the browser is on the same computer as the server), with port `8080`.

To access the web interface, you need a valid user name and password. You entered these into the provisioning server during the product installation.

On the web interface login page, there is a link to navigational tips for first time web interface users. You are strongly encouraged to follow the link and read these tips.

The web interface is divided into five components:

- Server status
- Server parameters
- Server history
- OS deployment
- Server log files
- Advanced features:
  - Deployment engines
  - Hardware configuration
  - Image monitor

You can access these components

- by clicking on their respective icons in the main frame of the web interface
- by using the menu provided in the left frame of the web interface

Server status

Server status information is further subdivided into four sections: General information, Installation check, Network connections and web interface extension.

General information

Items in the **General information** section are read-only and contain:

- General server information, such as the OS deployment server version, the number of active targets, and the last warning message
- OS deployment server statistics, including the number of successful boots, and ignored boots, and the current boot module load

- File server statistics, containing information about MCAST, unicast, and PCAST sessions, and TCP connections
- Web interface modules, including statistics on the HTTP connections and operations

Installation check

Items in the **Installation check** section are read-only. It provides a summary of all errors that have happened in the past 30 minutes. They are presented in distinct sections representing their associated log file.

If you are not familiar with Tivoli® Provisioning Manager for OS Deployment, use **Check PXE Boot** at the bottom of the screen. A wizard helps you understand and check all of the necessary steps to ensure that the target works with your OS deployment server.

Network connections

Displays the active connections onto the OS deployment server. They are presented with thee different links: *Unicast* connections, *Multicast* connections, and *HTTP* connections.

The numbers **Traffic IN** and **Traffic OUT** are in Kilobytes (total number of KB seen on this connection).

web interface extension

Displays the status of the web interface extension on the computer that is running the browser.

Server parameters

Server parameters are used in the OS configuration of the Provisioning server.

Server parameters are used to configure the OS deployment server. They are divided in to five sections:

OS configuration
provides read and write access to the main server OS configuration parameters.

HTTP console security
allows you to change the administrator name and password, and to create security roles to secure access to the OS deployment server through the web interface.

Predefined channels
is the location to define TCP tunnels and authentication domains.

Hardware handling
provides information about the known compatibility and incompatibility of hardware devices with Tivoli Provisioning Manager for OS Deployment.

Server replication
allows you to replicate parent and child servers. See "Multiserver infrastructure" in the Installation Guide for more information.

Server history

The Server history component helps you to follow the task of your OS deployment server.

It is divided into four sections:

Server statistics
There are three different pages that show graphically the load of the processor, the network and the number of targets working with the OS deployment server. The graphs are displayed for the past 24 hours, the past week and the past month.

Each of the **Network traffic** graphs has two scales. The left scale shows the total inbound and outbound traffic, and the right scale shows multicast traffic. Numbers are in Kilobytes per minute (KB/min).

Deployment statistics
Deployment statistics show you, through tables, when deployments were performed over the last 24 hours, the last month, or the last two years. You can select to view successful or failed deployments by OS configuration, by deployment scheme, by computer model, or by administrative group. You can export the deployment statistics into a CSV format by clicking **Download as CSV data file**.

Modification history
Keeps track of creations, deletions, and modifications on OS configurations, system profiles, deployment schemes, and software modules. For every update, the page displays the date and time of the update, its author, its item description, and its new version number.

Tasks tasks include most actions performed from the OS deployment server on targets. Examples of tasks are deployment, creation of cloning profiles, detection of the operating system currently on a target.

Displays a description, an execution and an expiration date, current state, and progress rate. You can expand a task by clicking on its + sign to view its target or targets.

To cancel a task, select **Cancel task** from the contextual menu when a target is selected.

See also deploy/tosd_monactivity.dita.

OS deployment

The OS deployment component provides access to the main functions of Tivoli Provisioning Manager for OS Deployment, allowing you to manage targets and to prepare and run deployments.

It is divided in to four sections:

Target Monitor
to manage the targets

Task template
to manage screen layouts on the target and deployment schemes

Profiles
to manage the operating system profiles and OS configurations

Software modules
to manage software to be deployed with an operating system

Server log files

Server log files help in diagnosing problems and recording commands.

Log file description

There are two sets of files for the logs contained in the `TPMfOS Files` folder: `.log` and `.trc`. For example, there is a `vm.log` file and a `vm.trc` file. The .log files contains information understandable by all users. It is in the language in which the product was installed. The .trc files contain more detailed information and are designed for the support team. It is always in English. When sending logs to support, make sure to always include the `.trc` files.

There are several distinct log files:

`files/logs/activities.log`
This log allows you to view when a replication task was started. Its content cannot be viewed from the web interface.

`files/global/hostactivities.log`
This directory contains the list of all target tasks. Its content can be viewed from the **Tasks** page of the **Server history**, where it is merged with the information provided by `activities.log`. To send the logs of a given task to support, ideally select the task and select **Export debug data** in the contextual menu.

Boot log file or `files/logs/boot.log`
This log is useful when diagnosing DHCP/TFTP problems. Its content can be viewed in the**Server log files** page of the web interface.

File log fileor `files/logs/File.log`
This log is useful to diagnose multicast file transfer problems. Its content can be viewed in the **Server log files** page of the web interface.

HTTP log file or `files/logs/HTTP.log`
This log records requests made by the web interface. Its content can be viewed in the **Server log files** page of the web interface.

NBP log file or `files/logs/NBP.log`
This log records remote target commands such as authentication, tracking logs, and so on. Its content can be viewed in the **Server log files** page of the web interface.

TCP log file or `files/logs/TCP.log`
This log is used to diagnose unicast file transfer problems. Its content can be viewed in the **Server log files** page of the web interface.

VM log file or `files/logs/VM.log`
This log records events generated by background tasks running on the server. Its content can be viewed in the **Server log files** page of the web interface.

Sync log file or `files/logs/rsync.log`
This log file contains information specific on replication: checking files, finding them or not, copying them, and so on. Its content can be viewed in the **Server log files** page of the web interface.

Note: The `files` directory corresponds to the main data directory. Under Windows operating systems, it is `TPMfOS Files` by default.

Retrieving pertinent information in the logs

To help you retrieve pertinent information easily, the following features are available in the web interface:

- Logs are color coded. Blue indicates no warning or errors, yellow indicates warnings, red indicates errors.
- Logs are hierarchical. Clicking on the expand sign allows you to see more details.
- Logs are chronological. If you want to go to a specific date and time in a log file, click **Jump to ...**, enter the date and time, and click **Jump**. The log section with the given date and time expands. If no section has the exact given date and time, the section just before the given time and date expands.

Log file cleaning

From the **Server Log Files** page, you can also clean the contents of the log files. You can either delete the content of individual server log files or perform a partial cleaning of all the server log files. You can also delete or perform a partial cleaning of all target log files.

Partial cleaning of a log allows you to keep the most recent content only. You can keep information of the latest

- 30 minutes,
- 60 minutes,
- 6 hours,
- 12 hours,
- 24 hours,
- 1 week,
- 2 weeks,
- or 1 month.

Log data older than the selected time is deleted.

Chapter 3. Database schema

The database schema documentation describes the most important items of the database schema used by Tivoli Provisioning Manager for OS Deployment to store and manage targets and images. Further details can be found in the reference database file, AutoDeployDistrib.ini, in the program installation directory.

Database schema overview

In Tivoli Provisioning Manager for OS Deployment, database schemas are used to store and manage targets and images.

The database schema can be divided in two parts:

- The Bill of Material-related tables (BOM, DiskInventory, DMIInventory, PCIInventory, PCIDescription, Deployment, ErrorLog, Settings, UserProfile) that store all target and user-dependent information
- The OS configuration-related tables (SoftwareProfile, SoftwareItem, GroupingRule, Groups, SystemProfile) that retain information related to the software modules and operating system that can be deployed

Bill of Material-related tables

There is one Bill of Material (BOM) record for each remote-boot target that has been started on the OS deployment server. The DeplCount field is a counter that allows you to track the number of deployments that have been run for a particular target(the summary of each deployment can be found in table deployment). The field Status indicates the current state of this target:

- If the target was last used to create an image, or if the deployment is configured with no database update, the status is empty
- If the target is showing the parameter edition window and waiting for BOM information to be reviewed (in the database or on the target), the status is editbom
- If the target is determining its software configuration (at the beginning of the deployment), the status is 0/
- During the deployment, the status is x/y, where x is the current stage number and y is the last stage number (each stage ends with a restart)
- After a successful deployment, the status is ok
- In case of error, the status is error and the error time and message can be found in the table ErrorLog

Most target-specific parameters are stored directly in the BOM table. User-specific parameters are stored in the UserProfile table (and identified by the UserID key). The Locale field stores the Microsoft locale code, and the TimeZone field stores the Microsoft time zone index (a list of which can be found in the Windows registry). There are also nine freely configurable user categories that can be used to store information regarding the user (such as position, department, and location), and that can be used in the software matching mechanism.

The hardware inventory is stored in the three tables on the left, at the beginning of the deployment, after the BOM edition stage. There is one record in the

DiskInventory table per disk detected, one record per target in the DMIInventory table (see the database for the exhaustive list of fields) and one record per PCI adapter in the PCIInventory table.

The Settings table lists deployment schemes. The actual settings for each deployment scheme are stored in an .ini file on the OS deployment server, whose prefix is made using the DeplSet identifier.

OS configuration-related tables

Every OS configuration is stored as one row in the Groups table. Each OS configuration is bound to exactly one system profile, but for multiboot systems there can be several OS configurations for a single system profile.

All system profile-dependent parameters (such as partitioning scheme) are stored in the SystemProfile table. Primary partitions are stored in the PrimPart field and logical partitions in the LogiPart field, as a space-separated list of partition type:size (size is in MB). The column OSPart is the index of the partition on which the operating system shall be installed. The image fields store the name of various image files on the OS deployment server (OSImage is the prefix used for the OS partition, DiskImage is the prefix used for all other partitions). The Model field is optional and used to perform computer model checking/locking if requested, to ensure that the correct operating system is deployed on each model of computer (check is made as a substring, that is, a system profile flagged for Supra matches any computer model with Supra in its model name, as found in the DMI tables). There are four freely configurable system categories, that can be used to store information regarding the system profile such as preconfiguration, distribution, and that can be used in the software matching mechanism (see the information later in this section for the explanation of table SoftwareProfile).

The relation between the BOM table and the Groups table is not direct, but is done through the GroupingRule table. In the simplest case, a row of this table only includes the BomID of a target record and the Group ID of the selected configuration. If several OS configurations are bound to a target, several rows are used. In a more complex case (automatic binding rules), the table acts as a dynamic relational index and follows a kind of rule-matching behavior. For each record of the table, if all specified values are matching those of a target, the corresponding OS configurations will be bound to it. The field Condition can include an additional Rembo-C condition to check before in addition to the pattern matching mechanism.

The SoftwareItem table (on the bottom right) record information regarding every software modules that can be deployed. Each software is identified by its SoftItemID. There is typically one record per software module, but there can be more than one if a package involves for instance both a file copy and a command line (each of which is described by a record). For each record, the type field identifies the kind of item:

- copy items are used to copy a tree of files to the OS partition. The Source is the file archive on the server, the Dest is the destination path.
- run items are used to start a command-line, for instance to run an unattended setup. The Dest is the full command line to run.
- ini items are used to make changes to an .ini (or .inf) file on the OS partition. The Source is the name of the .ini file on the OS deployment server, the Dest is the destination path and file name of the file to update.

- `diff` items are used to install a system snapshot. The `Source` is the file archive on the server.
- `floppy` items are used to start a non-DOS virtual floppy-disk image (in a ramdisk), for instance to run a vendor-specific firmware upgrade utility. The `Source` is the raw image name on the server.
- `part` items are used to start a DOS virtual floppy-disk image (as a ramdisk or on a partition), for instance to run a BIOS upgrade utility. The `Source` is the partition archive on the server, the `Dest` is the destination partition and mode.

Other software module parameters are the `Pass` number at which the software must be applied, and a boolean `SysOnly` that prevents the software from being made visible to the user in the BOM edition stage when turned on.

The choice of what software is to be installed is controlled by the `SoftwareProfile` table. This table acts as a dynamic relational index and follows a kind of rule-matching behavior. For each record of the table, if all specified values in the first sixteen columns are matched, the software item specified by the identifier in `SoftItemID` will be applied. For instance, software modules can be associated to a specific target by adding a record with the `BomID` of the target and the `SoftItemID` of the software. In the same way, software can be bound to a specific system profile or group. More complex OS configurations can also be used to implement the software binding rules.

Chapter 4. OS deployment server configuration

These topics describe the different parameters available to configure the OS deployment server, and on how to modify the values of these parameters.

Global parameters

Global parameters are the main server side OS configuration parameters that can be modified either on the web interface or by editing the `rembo.conf` file.

The parameters are divided into seven categories that mirror the sections found in the web interface. They are:

- Debug level information
- Base parameters
- Web interface parameters
- Boot module parameters
- Network boot module parameters
- File access module parameters
- Network share module parameters
- HTTP Console Security parameters
- New targets default parameters

Most parameters can be modified in the web interface. Go to **Server** > **Server parameters** > **Configuration**.

Base parameters

- *Debug level information*

  GlobalDebugLevel `number` specifies the level of details that must be placed in the log file.

- *IP address of the backup server*

  Backup `ip-addr` defines the IP address of a backup server that can be used by the target if the primary server fails. The backup server must have the same `Net Password` as this server. For the backup recovery system to work, the backup server must also contain an exact copy of the files stored on primary server.

- *Maximum size of the server log files*

  MaxLogSize `number` limits the size of the log file generated by the OS deployment server. The maximum log size is specified in bytes, and applies to all log files created by the OS deployment server. If you do not specify this parameter is not set, or set to `0`, then log files are not limited in size. If a limit is set, and the server reaches this limit, the current log file is backed up and a new file is created.

- **CountersInterval** `minutes` specifies the interval, in minutes, for the server statistics measures.

- *Network interfaces used by the server*

  Interfaces `ip-addr` specifies the list of network interfaces that the OS deployment server uses when sending and receiving packets to and from targets.

If you leave this option unspecified, the server uses the network interface corresponding to the official host name of the server. The format is a list of IP addresses, separated by spaces.

web interface parameters

- *Disable the HTTP module*

  DisableHTTP disables the web interface. If you set this parameter, run the `-c` command-line option to change server parameters.

  Note:

  – Disabling the HTTP module prevents you from using the Java API.
- *Disable HTTP SSL encryption*

  DisableSSL disables encryption in the Web interface and enables unencrypted HTTP connections.
- *TCP port for HTTP requests, TCP port for encrypted HTTP requests (SSL)*

  HTTPServerPort `port` specifies the TCP port used by the Web interface when listening for unencrypted HTTP requests. You can set this parameter to `80` if you want the Web interface to be accessible by typing the server host name in your Web browser. To make the Web interface accessible by typing `https` followed by the server host name, set this parameter to `443`.
- *Inactivity timeout before a session expires*

  HTTPSessionTimeout `minutes` specifies the timeout (in minutes) before Web interface users are automatically logged out. A typical value is 5 minutes.
- *Disable HTML contextual menus*

  DisableContextualMenu disables the contextual menu. When you use a right-click in the web interface, you will see the regular contextual menu of your browser.
- *Disable HTML bottom-left menu*

  DisableBottomLeftMenu disables the HTML bottom-left menu.
- *Disable welcome checklist page*

  DisableChecklistPage disables the welcome checklist page.
- *Disable HTML drag-and-drop*

  DisableDragAndDrop disables web interface drag-and-drop operations on selected icons. Drag-and-drop works on most common browsers, but it can be made unavailable in case of incompatibility.
- *Disable javascript drop-down lists*

  DisableJSDropDown disables enhanced menus in favor of standard menus. Standard menus do not recognize DHTML layers under Internet Explorer, therefore this parameter can cause problems.
- *Disable HTTP transfer optimization*

  DisableHTTPOptim disables the optimization of HTTP transfer in the web interface. Set this parameter if you do not want the web interface to group JavaScript, stylesheets, and images into large files to reduce HTTP traffic and enhance the cache process. You must disable this optimization if you are creating console pages.

Boot module parameters

- *Disable DHCP/BINL module*

  DisableBOOT disables the PXE component of the OS deployment server.

When this parameter is set to **Yes**, it is not possible to create network boot media.

To prevent targets booting into the OS deployment server without disabling network boot media generation, you have two options:

- Put the server in *closed server* mode by setting **Completely ignore unknown targets (closed OS deployment server)** to **yes** in the **Idle state** task template
- Temporarily **disable DHCP proxy functionality**, which prevents any non-directed PXE request

- *Disable the DHCP proxy functionality*

  DisableDHCPProxy disables the DHCPProxy service on the OS deployment server. The server does not send extended DHCP replies (PXE replies) to DHCP discovery packets sent on the network by remote-boot target. When the DHCP proxy is unavailable the server must be configured to provide a complete PXE answer to PXE targets, including option 60 and option 43. DHCP proxy mode must be made unavailable if you have more than one OS deployment server operating on the same subnet.

- *Disable the multicast BINL functionality*

  BootNoMulticastDiscovery disables PXE multicast discovery support on the OS deployment server. When PXE multicast discovery is unavailable, target computers with multicast discovery configured in DHCP option 43 are not able to find the OS deployment server.

- *UDP port for TFTP requests*

  TFTPPort is the port used on the OS deployment server to receive TFTP request packets from PXE targets. The default value is 69. Note that unless your network equipment is automatically routing TFTP packets to a non-standard part, you must not change this setting because PXE always sends TFTP requests to port 69, without any possibility to change port.

- *Max. TFTP Segment Size*

  MaxTFTPSegSize `bytes` is the maximum size in bytes for TFTP segments. The default value is 512 but a smaller one can be given if the network configuration requires it.

- *Seconds to wait before starting a MTFTP stream*

  MTFTPStartDelay `secs` specifies the time, in seconds, to wait before starting a MTFTP transfer. This period of time is used by target computers to replicate together. The higher the value is, the more replicated the deployment engine is. However, latency is introduced when booting target computers individually. The PXE standard default value is `2`.

- *IP address used by targets for MTFTP*

  MTFTPClients `port [:target-port ]` specifies the multicast IP address and port used by the OS deployment server when sending MTFTP data to target computers.

- *UDP port for MTFTP requests*

  MTFTPPort `port` specifies the UDP port used by the OS deployment server to receive MTFTP request packets from target computers. The default value is `4015`.

- *Seconds to wait before sending DHCP/BINL answers*

  BootReplyDelay `secs` specifies the number of seconds to wait before answering DHCP/BINL request packets sent by target computers. If you have more than one OS deployment server on the same subnet, target computers use the server with the lowest shortest reply delay. The default value is `0` (no delay).

- *Max. number of requests per minute (load balancing)*

BootReplyLimit `number` specifies the maximum number of DHCP/BINL requests to an OS deployment server in one minute. The server discards boot requests coming from target computers when the specified limit is reached. A number of `0` indicates unlimited requests. This parameter can be used to implement load balancing over multiple servers.

- *UDP port for DHCP/BINL requests*

  BootDiscoveryPort `port` specifies the UDP port that the OS deployment server uses when listening for BINL boot requests. The default port used for a standard target computer is `4011`.

- *Multicast IP address used for BINL requests*

  BootDiscoveryAddress `ip-addr` specifies the multicast IP address that the OS deployment server listens to when waiting for BINL multicast requests coming from target computers. This value must match the value set in the DHCP option `43` sent to targets.

- *No-boot schedule*

  NoBootSchedule specifies the frequency at which the OS deployment server stops providing a PXE boot service. It is used in conjunction with `NoBootDuration`. Modify this parameter using the web interface only.

- *No-boot duration in minutes*

  NoBootDuration specifies the duration in minutes during which the OS deployment server stops providing a PXE boot service if a no-boot schedule is specified. It is used in conjunction with the parameter `NoBootSchedule`. Modify this parameter using the web interface only.

Network boot parameters

- *Disable the NBP module*

  DisableNBP disables the NBP component of the OS deployment server. Because NBP is used by target computers when booting, target computers are not able to boot if NBP is unavailable.

- *UDP port for NBP requests*

  NBPServerPort `port` specifies the UDP port used by the OS deployment server to receive NBP requests from target computers. The default value is `4012`. Only change this value if the default port is already used by another application on your computer.

Note: It is also possible to remote boot targets without using PXE. See Booting targets without using PXE.

File access module parameters

- *Disable the FILE module*

  DisableFILE disables the FILE (NetFS, MCAST) component of the provisioning server. If you disable the FILE server module, target computers are not able to boot on this server.

- *Base directory for server files*

  BaseDir `string` specifies the base directory for the provisioning server. All paths included in the OS configuration are relative to this base directory. This is a mandatory parameter, and has no default value (if you are using file-based configuration mode, you must set it before starting the provisioning server for the first time).

- *Relative directory for the shared repository*

SharedDir `path` specifies the path relative to the server file directory for storing the shared repository internal files. By default, this parameter is set to `shared`.

- *First multicast IP address used for MCAST transfers*

  FileMCASTAddress `ip-addr:port` specifies the multicast IP address used when sending packets to target computers. The OS deployment server can generate different multicast IP addresses using this parameter as the base for the first address in a range.

- *Number of multicast addressed to use for MCAST transfers*

  FileMCASTAddrCount `number` specifies the number of multicast addresses that the provisioning server can use when sending multicast data to target computers. This parameter sets the size of the range of multicast addresses that the server can use. This range begins with the address set by the parameter `FileMCASTAddress`.

- *Maximum number of PCAST connections*

  FileMaxPCASTSessions indicates the maximum number of PCAST sessions that the provisioning server accepts simultaneously. This parameter is relevant if the server is running in UNICAST mode and deploying a group with the UNICAST setting.

- *Port for FILE requests*

  FileServerPort `port` specifies the UDP port used by the provisioning server to receive transfer protocol control requests. The default value is 4013, but you can change this parameter if port `4013` is already used by an application on your computer.

- *Disable the TCP module*

  DisableTCP disables the TCP component of the provisioning server. Target computers might not be able to boot on this server if you disable the TCP server module.

- *Directory for internal files*

  DataDir `path` specifies the path relative to the provisioning server base directory for storing files accessible to the target computer. By default, this parameter is set to `files`.

  Note: You can specify the `DataDir` as the network driver disk by using a UNC path.

- *Disable Multicast across subnets*

  Set the **DisableWideMCAST** parameter to `Yes` if your network infrastructure does not handle multicast routing efficiently, or if you do not want to send multicast packets across subnets and VLANs. If you disable multicast routing, multicast packets will only be sent locally on each subnet. This might increase the OS deployment server load when serving multiple subnets simultaneously.

- *Limit for shared IDX in memory*

  Using the **CachedIdx 64** parameter, you can enter the maximal number of shared indexes cached in RAM. Their size is approximately 2.5 MB. They are cached in RAM for faster access. The default (and maximal value) for this parameter is 64. Setting a value lower than the number of indexes in your OS deployment server shared repository will limit the memory consumption but might also slow down the access to the shared files.

Network share module

- *Network UNC path*

NetworkShare `path` specifies the path of the shared partition directory of the OS deployment server. The OS deployment server shared partition directory is the following:

```
C:\TPMfOS Files\global\partition
```

which is made accessible using a Windows network sharing protocol. This parameter can be used to increase the deployment speed of some operating systems. You need to have set the partition as read-only for the user entitled to access the network share.

- *User entitled*

  NetworkUser `"string"` is the name of the user entitled to access the network share. If the user is part of a domain, use the syntax `Domain/User`.

- *Password of the user*

  NetworkPasswd `"string"` is the password used to access the network share. The password is stored in an encrypted format in the `rembo.conf` file.

Note: If the `Network UNC path`, the `User entitled`, or the `Password of the user` are not correct, you get an error when attempting deployment using the network share. If the `Network UNC path` is empty, the deployment does not try to use a network share.

New targets default parameters

Targets are assigned a set of default parameters.

The following parameters can be modified in the web interface, or in the `rembo.conf` file.

- *Completely ignore unknown targets/ Let unknown computers contact another PXE server/ Make unknown computers boot on their hard disk/ Make unknown computers boot on their hard disk when there is no pending task*

  DefaultPXEBootMode { `normal | hdboot | ignore | otherPXE | BootIfIdle`} When the OS deployment server becomes aware of an unknown PXE target trying to boot from the network, it can decide whether to add the target automatically to its internal database. To add the new target, include `normal` in your argument list. To exclude unknown targets, add `ignore` in your argument list. You can leave them alone to discover another PXE server, force them to boot on their hard-disk, or force them to boot on their hard disk when no task is pending. Add `otherPXE`, `HDBoot`, or `BootIfIdle` in your argument list to obtain the wanted behavior.

- *Tivoli Provisioning Manager for OS Deployment kernel options*

  DefaultOptions [`autoboot ] [ , KernelFree ] [, nousb ] [, noautousb ] [, novesa ] [, noapm ] [, unicast ] [, noudma ] [, noprotpart ] [, realmodedisk ] [, realmodePXE ][, routepxeirq ]` The argument of this parameter is a list composed of the following elements, separated by commas: `autoboot` reboots automatically on unrecoverable errors, `NoUSB` disables USB devices, , KernelFree defines the target boot in kernel-free mode, `noautousb` enables USB devices only when no PS/2 connection is detected, `NoVESA` disables the graphic interface, `NoAPM` disables the APM, `Unicast` disables multicast, `NoIGMP2` disables IGMP version 2, `NoUDMA` disables Ultra-DMA, `NoProtPart` disables ATA-5 features, `RealModeDisk` disables enhanced disk access, `RealModePXE` disables enhanced PXE access, and`RoutePXEIRQ` reroutes network IRQ separately from the disk controller IRQ .

- *Human interface locking*

DefaultLockOut [none] [, mouse] [, keys] [, screen] [, all] The argument of this parameter is a list composed of the following elements, separated by commas: Mouse disables the mouse, Keys disables the keyboard, Screen disables the screen, and ALL disables the mouse, keyboard, and screen.

Other parameters for new targets can be configured only by editing the rembo.conf file.

- *Preferred screen resolution*

  DefaultResolution "string" specifies the default screen resolution for new targets.

- *Authentication domain*

  DefaultAuthDomain "string" specifies the default authentication domain to use when a target wants to check credentials. This domain can be overridden individually for each target.

- *Server for files services*

  DefaultFileServer ip-adr:port specifies the default file server, given by its IP address (and port number), for new targets.

- *Boot redirection servers*

  DefaultBootRedirectionip-adr [, ip-adr] specifies the IP address of an OS deployment server to which boot requests of the targets must be redirected. Optionally, a second redirection server can be indicated, in which case you must separate the two IP addresses by a comma.

HTTP Console Security parameters

In the web interface, these parameters can be modified on the **Server** > **Server parameters** > **HTTP Console Security** page.

- *Super-user login*

  HTTPAdminName "name" specifies the superuser login of the OS deployment server administrator, in order to access all OS configuration parameters of the server. There is only one superuser login.

- *Super-user password*

  NetPassword "string" specifies the password used by the main OS deployment server administrator This password is used to protect the server files against illegitimate remote access.

- *New security role*

  HTTPRole "name" {Members "stringlist" AllowPages "stringlist" AllowGroups "stringlist"} allows new members to access the web interface by specifying which pages are available to them and which administrative groups these new members can manage.

Authentication domains

An authentication domain is a group of parameters related to the authentication of users by Tivoli Provisioning Manager for OS Deployment.

The term *domain* has nothing to do with Windows NT domains, or NIS+ domains. Parameters contained in an authentication domain define how user and password information entered on a target workstation are checked by the OS deployment server. For example, they can be checked against the local server database of users,

or through a remote authentication device. Additionally, Tivoli Provisioning Manager for OS Deployment allows you to restrict the search for matching users to a single user group for greater flexibility.

Note: In order to setup security roles, you must create an authentication domain named `HTTP`.

The OS deployment server supports several authentication protocols, depending on the platform of the server:

- On Windows, a user identity can be verified with the local user database, or a remote user database (but still on a Windows server)
- On UNIX, the user identity is verified with the standard user database functions, which can be configured to use local files, NIS or NIS+. PAM is used if the server is running on Linux
- On both platforms, the OS deployment server can use the authentication standard *Radius* to perform the authentication with a device supporting the Radius protocol, or with a Radius gateway for Netware (NDS).

  Note:

  1. On Windows, authentication can be done only if the user running Tivoli Provisioning Manager for OS Deployment has the `Act as part of the Operating System` privilege. By default, the user account used for services has this privilege (the `SYSTEM` account).
  2. If you are using NT remote authentication, the OS deployment server must have this privilege on the remote computer. You must start the IBM® Tivoli OS Deployment Server (*remboserver*) and IBM Tivoli Web Interface Extension (*remboagent*) services with domain accounts that have the NT remote authentication privileges.
  3. Authentication with redeployment does not work if the target is offline (the target has no network connection and boots from the hard disk). A message warns the user.

If you are modifying the server configuration in the web interface, you can add a new authentication domains by clicking **New auth. domain** in **Server** > **Server parameters** > **Predefined channels** and entering the appropriate information.

If you are maintaining your server configuration directly in the text file `rembo.conf`, you must add authentication domains in the configuration file. Each domain starts with `AuthLocalDomain`, `AuthNTDomain` (on Windows only), or `AuthRadiusDomain`, and is followed by the name that you want to attribute to this domain. For example:

```
AuthLocalDomain HTTP {}
```

There are three types of authentication domains:

- *Local domains* uses the local user database to authenticate a user. The optional `UserGroup` parameter can be used to restrict the verification to a specific group of users. This type of domain is supported by both Windows NT and UNIX.
- *Remote NT domains* forwards authentication requests to the NT server specified by the mandatory parameter `AuthServerName`. The parameter `UserGroup` can be used to restrict the verification to a specific group of users. This type of domain is supported only in Windows.
- *Radius domains* forwards authentication requests to the Radius-compliant device specified by the parameter `AuthServerAddress`. The value of the parameter

RadiusSecret is used as the password for the Radius communication, and must match the password stored in the configuration of the Radius device for the protocol to work.

TCP tunnels

A TCP tunnel is a way to provide TCP connectivity to target computers.

- In Tivoli Provisioning Manager for OS Deployment, you must have an ODBC tunnel to access the database. This tunnel is present by default.
- A *sendmail* TCP tunnel is mandatory to receive e-mail notification at the end of a deployment.

Creating a new hardware rule

Hardware rules are useful to work around problems related to some specific hardware.

At boot on the Tivoli Provisioning Manager for OS Deployment, a target will be compared to the hardware rules and if there is a match within the list, some specific features (potentially making problems for that hardware) will be unavailable, for instance: "enhanced PXE access", "ATA-5 features", and so on. When Tivoli Provisioning Manager for OS Deployment is shipped there is a built-in default hardware rule list that is a list of the known hardware making problems at the time of the build. However, new hardware coming out is not listed here and if any new hardware causes problems it can be resolved by manually disabling specific features.

At remote-boot the hardware of a target is immediately detected and compared to the hardware rule list, the selected features will then be unavailable if necessary.

To create a **New hardware rule**:

1. Go to **Server** > **Server parameters** > **Hardware handling**.
2. Click **New hardware rule**.
3. Select one or several of the offered options.

Specific PCI device present:

PCI (Peripheral Component interconnect) devices are peripheral components connected on the PCI bus of the system board. These are typically: network cards, sound cards, Disk/USB controllers and so on.You can specify the PCI device that the rule is to be based on and enter information in the following fields:

- Vendor ID (4 hex digits)
- Device ID (4 hex digits)
- Revision (optional, 2 hex digits)

Note: To view Vendor/Device/Revision information,go to **Server** > **OS deployment** > **Target Monitor**. Double-click on a target to view its details. Look in the **Inventory** tab, under **PCI devices**.

Specific Model name:

You can specify a model name for the rule it is based on. For example: eServer™ xSeries®.

Specific Linux distribution
You can specify the Linux distribution name for the rule it is based on. For example: SUSE , Red Hat.

Specific type of chassis
You can specify the Type of chassis for the rule it is based on. For example: Single Chassis , Multi Chassis.

Specific OS Architecture
You can specify if you are working with a 32–bit or a 64 bit architecture.

4. Depending on your selections, the wizard provides steps with easy-to-follow instructions to create the new hardware rule.
5. Select what the rule will do if chosen criteria are met.
   - Change Tivoli Provisioning Manager for OS Deployment kernel flags. The options available are:
     - Use kernel-free flow
     - Reboot on unrecoverable errors
     - Disable USB
     - Disable Auto USB
     - Disable graphic interface
     - Disable APM
     - Disable multicast
     - Disable IGMP version 2
     - Disable Ultra-DMA
     - Disable ATA-5 features
     - Disable enhanced disk access
     - Use BIOS for CD/DVD ROM access
     - Disable enhanced PXE access
     - Try to optimize IRQ
   - Change Linux kernel parameters. Changing the Linux kernel parameters option requires a command line parameter to be entered that will be used when Linux is deployed and rule criteria are met. Enter the command line in the Linux kernel parameters field.
   - Enable remote power management feature, using either Intel Active Management Technology or by performing the following, Go to **Server** > **Server parameters** > **Hardware handling** > **New hardware rule** . Here you can define a rule that will show the hardware where Remote power management is available. In the Host details page if there is a matching hardware rule for remote power management then **Switch on** and **Switch off** buttons will be visible.

     Note: These targets can also be switched on by checking the **Try to wake up targets using management interface** in the deployment wizard. If you use the **Command line** technology to remotely power on and off the target, you can specify the [IF] parameter to customize the command. In the **Remote power management** section of the **Target details** page, assign a target-specific value to the [IF] parameter, for example specify the target IP address. This value replaces the [IF] parameter in the command line to power on or off the specific target machine.
6. Follow the wizard instructions to complete the rule.

At remote-boot the hardware of a target is immediately detected and compared to the hardware rule list, the selected features will then be unavailable if necessary.

Default port numbers

During the installation process, you can choose to accept the default port numbers proposed or you can specify alternate unused port numbers.

The following table lists the default port numbers that are used by Tivoli Provisioning Manager for OS Deployment.

Table 2. Default port numbers used by the server

Name of port	Default port number
DHCP	67 UDP
PXE BINL	4011 UDP
TFTP	69 UDP
MTFTP	4015 UDP
NBP	4012 UDP
FILE	4013 UDP and TCP
MCAST	10000-10500 UDP Address: 239.2.0.1 - 239.2.1.1

Table 3. Default port numbers used by the web interface extension (rembo agent)

Name of port	Default port number
Listening port	4014 UDP
The web interface extension (rbagent) connects to ports 4012 and 4013 on the OS deployment server to retrieve information and files. Port 4012 is normally used when sending commands, while 4013 is used when dealing with file transfers. The source port of the agent when connecting to 4012 and 4013 is random.	

Table 4. Default port numbers used by the client PXE

Name of port	Default port number
DHCP	68 UDP
MTFTP	8500-8510 UDP Address: 232.1.0.1
MCAST	9999 UDP
Remote control (agent)	4014 UDP

Configuring OS deployment server with a text file

A single text file (config.csv) can be used to configure all the OS deployment servers in a complex environment.

OS deployment servers can be configured using a file named `config.csv` (comma separated values). The advantage of this configuration file over other configuration methods is that the same file can be used to supply different configuration parameters to a large number of OS deployment servers installed in a big organization.

You must place the file `config.csv` in directory `<datadir>/global/rad`, for instance `c:\TPMfOS Files\global\rad` on a Windows server.

The same configuration file can be used for all the OS deployment servers in the organization. But it must be deployed on all the OS deployment servers. The reason is that each OS deployment server reads the configuration file from its local file system, and filters out all irrelevant information. The file can also be different for different parts of the organization, or even for each OS deployment server. It is up to the operator to select the most convenient management strategy.

The file is read automatically at server startup, or on demand, using the web interface extension command **rad-configure**. Some of the changes automatically trigger a server restart to be properly applied.

Each line in the file describes one OS deployment server. The fields are separated by a semicolon **;**. Fields can be optionally quoted with double quotes ("). The double quotation mark itself must be repeated to differentiate it from a quotation mark. Fields containing a semicolon or a double quotation mark must be quoted (see Table 5).

Table 5. Code

Field	Interpretation
abcd1234	abcd1234
"abcd1234"	abcd1234
"abcd;1234"	abcd;1234
"abcd""1234"	abcd"1234
abcd""1234	abcd1234
abcd";"1234	abcd;1234

Note: The first line of the `config.csv` file must contain the label of the fields.

This file format can be conveniently imported and exported by Microsoft Excel and OpenOffice.

Field description

Within one line, each field is interpreted according to its position. The interpretations are detailed in the information shown later in this section.

HostName

This can be either the host name (for example: rembomaster) or the fully qualified name (rembomaster.rembo.com) of the target targeted by this line of the file.

There are two very important constraints on this field.

1. All the letters in the name must be lowercase.
2. The name must be immediately followed by the semicolon.

If one of these two constraints is not true, the script which parses the file will not be able to recognize the line and the server will not be configured properly.

As a side effect, all the lines which do not start with a valid host name are considered to be a comment. The prefix ";RBO " is reserved for internal use (for example: ";RBO;DB2_BIN_DIR;")

When an OS deployment server has recognized its HostName, the parameters found in the line overrides the values that were supplied earlier, for instance when running the setup program or by the means of a rembo.conf file.

Interfaces

The list of IP addresses used by this OS deployment server. If the server has several interfaces, PXE remote boot is available only for the interfaces listed in this field. Addresses must be given in the dotted notation (for example: 192.168.168.16), and separated by one space character.

DbName DbUser DbPass

Connection information for the server database used by this OS deployment server. Typical value for DbName is "AutoDeploy". The password can be hidden using the web interface extension command **rad-hidepassword**.

Note: If *MasterDbName* is used, then *DbName* is mandatory.

DbName can be either the name of an existing ODBC source, or a valid string for SQLDriverConnect, enclosed in square brackets.

Example (MS SQL Server)

```
[DRIVER={SQL Server};SERVER=<hostname>;UID=<user>;PWD=<pass>;
DATABASE=<db>[DRIVER={SQL Server};]
```

Example (DB2®)

```
[DRIVER={IBM DB2 ODBC DRIVER};SERVER=<hostname or IP>;DATABASE=<db>;
UID=<user>;PWD=<pass>;]
```

With this syntax, which is not natively recognized by DB2, the product will automatically use or create a DB2 Node that matches <hostname or IP>, and a catalog entry that matches <db>.

Additional fields (DB2 database)

Some optional parameters can be added to the db2 command:

- - REMOTEPORT: this parameter will be set to 50000 if not present
- - REMOTE_INSTANCE: this parameter will be added if present
- - SYSTEM: this parameter will be added if present
- - OSTYPE: this parameter will be added if present

Example:

```
[DRIVER={IBM DB2 ODBC DRIVER};SERVER=srvkludge-1.rembo.private;
DATABASE=GwA;UID=administrator;PWD=poiuz;OSTYPE=AIX; REMOTEPORT=1288;
REMOTE_INSTANCE=devtinv4; SYSTEM=rtxidb2;]
```

will generate the command **db2 catalog tcpip node SRVKLDGR remote srvkludge-1.rembo.private server 1288 REMOTE_INSTANCE devtinv4 SYSTEM rtxidb2 OSTYPE AIX**

MasterIP

There are three choices for this field:

- "" (empty string) indicates that the server is a stand-alone server and that no replication is necessary.
- "SELF" indicates that the server is a parent at the top of the hierarchy. Some replication tasks are performed, but only in the "downward" direction.
- IP address of the *parent* OS deployment server server which is used for replication.

MasterDbName MasterDbUser MasterDbPass

Connection information for a second optional OS deployment server database. These fields must be left empty if the field MasterIP is "" or "SELF". Otherwise, they must lead to the same database as the parent's DbName. The password can be hidden using the web interface extension command **rad-hidepassword**.

Note: When MasterDBName is included, DbName becomes mandatory.

BinDir

This can only be used for radtcm.pak.

The directory where the wapmrpt.exe is located. Using forward slashes as separators, for example: `c:/temp`

Description

This can only be used radtcm.pak.

The Tivoli endpoint identifier of this server.

Reporting

This can only be used for radtcm.pak.

A string of letters controlling which reporting to wapmrpt must be performed by this OS deployment server. The letter **u** stands for **updates** and activates the reporting of branch update events, expected after the web interface extension command **sync-validate** .The letter **d** stands for **deployments** and activates the reporting of deployment events, expected after web interface extension command **rad-deployhost**. The order of letters in the string is irrelevant. If the string is empty, no reporting is done. The letter **k** stands for **keep files** and can be used for debugging.

AutoSync

A string of letters controlling server replication. The order of letters in the string is irrelevant. If the string is empty, no replication is done. The letters have the following interpretation:

f Replicates files: when this letter is present, the files on this OS deployment server are copied from the parent periodically. File replication is the mechanism to retrieve the "missing" files for the objects stored in the local database of the server. The *SyncSchedule* flag performs the same operation at a given time. Therefore, the *f* flag must not be set if another mechanism is used for file replication, for example: sync.pak or *SyncSchedule*.

c Replicates OS configurations: when this letter is present, the OS

configurations in this server database are copied from the parent's database periodically. This should be set if sync.pak is used to synchronize the files, but the server has its own database (for example: on spokes).

s Replicates software modules: same as **c**, but for software modules.

d Replicates deployment schemes: same as **c**, but for deployment schemes.

h Replicates targets: when this letter is present, the targets in this server's database will be copied to/from the parents's database periodically. This should be set at the branch level, to provide target management from the spokes.

l Light target replication: when this letter is present, do not copy parent's targets which are known to belong to a different branch of theOS deployment servers hierarchy. Copy only new targets and targets which have booted on this server, or one of its children. With this letter, **h** is automatically added.

b Replicates blacklist: when this letter is present, the hardware compatibility blacklist is copied from the parent periodically. The new values are unconditionally propagated to other servers using the same DbName.

PollInterval

The interval in minutes between two server replications. The default value is 1.

DebugLevel

The required global debug level for this server.

DB2BinDir

This can only be used for DB2

The directory where the command **db2cmd** is located. Using forward slashes as separators, for example: `c:/db2/bin` The default location is `c:/Program Files/IBM/SQLLIB/BIN`. It can be overriden globally for all servers with a special comment line **;RBO;DB2_BIN_DIR;<path>**, and also individually for each server with this column.

SyncSchedule

You can use it only if the flag *f* in *AutoSync* and the sync.pak package are not used. Replication occurs according to the timestamp of the different servers. For example if in config.csv, the replication time is set to 9.00 PM for a level 2 server, the replication occurs at 9.00 PM of level 2 server and not 9 PM of the parent server.

The schedule for file replication. Tells the server when it should copy the files from its parent. This should not be set if another mechanism is used for file replication, for example: sync.pak.

Example: synchronize once on 2006-02-22, at 3PM:

```
type=once;date=2006-2-22;time=15:00
```

Example: synchronize every day, at 3PM:

```
type=daily;period=1;time=15:00
```

Example: synchronize every week, every day except Thursday, at 3PM ("days" is a bit mask, where Monday=1, Tuesday=2, Wednesday=4, Thursday=8, Friday=16, Saturday=32, Sunday=64):

```
type=weekly;period=1;days=119;time=15:00
```

Example: synchronize on the first day of each month, at 3PM:

```
type=monthly;day=1;time=15:00
```

If flag **f** is used in `AutoSync`, scheduling a file replication here is not necessary.

SyncBandlim

This can only be used when sync.pak is not used

When SyncSchedule is used, SyncBandlim can be used to limit the network bandwidth used for file replication. The syntax is a number followed by a white space and one of "Mbits/s" or "Kbits/s". For instance "15 Mbits/s" or "192 Kbits/s". It is also possible to give just one integer, in this case the units are kilobytes per second.

APISecret

This can be used for the Java API, since version 5.1.0.2

The password that must be supplied when connecting to this server through the Java API. Must be hidden using the web interface extension command **rad-hidepassword**.

APITrusted

This can only be used for the Java API, since version 5.1.0.3

The list of trusted targets that can use the Java API when there is no APISecret. If APISecret is defined, this field is not used and any Java target that knows the secret can create a new session. If APISecret is empty or undefined, the server checks that the target IP address is in this list before creating a new session. Addresses must be given in the dotted notation (for example: 192.168.168.16), and separated by one space character

TPMReporting

This can only be used for TPM 7.1 and Director 6.1

A string of letters controlling events reported to TPM. The order of letters in the string is irrelevant. If the string is empty, no event is reported. The letters have the following interpretation:

b bare-metal targets : when this letter is present, targets without Description are announced to TPM. An attempt is made every time the OS deployment server is restarted. Fields TPMUser, TPMPass, TPMPort and TPMBinDir are required when this letter is present.

TPMUser

This can only be used for TPM 7.1 and Director 6.1

User name for the**soapcli** command line.

TPMPass

This can only be used for TPM 7.1 and Director 6.1

Password for the **soapcli** command line, hidden in the same way as APISecret.

TPMPort

This can only be used for TPM 7.1 and Director 6.1

HTTP port for the**soapcli** command line. Defaults to 4080.

TPMBinDir

This can only be used for TPM 7.1 and Director 6.1

The directory where soapcli.cmd or soapcli.sh is located. Using forward slashes as separators, `c:/program files/ibm/director/tpm/tools/soapclient`

Mandatory fields

The only mandatory field in `config.csv` is HostName.

However, a typical configuration for a multidatabase architecture also includes

- DbName
- MasterIP
- MasterDbName
- AutoSync

Moreover, the fields DbUser DbPass MasterDbUser MasterDbPass are often needed.

For a use with the JavaAPI, a typical configuration contains only HostName and APISecret.

Example

Typical settings for a large organization

HUB:

Single server at the top of the production servers hierarchy. Contains the latest version of all the deployment objects but not of the targets for scalability reasons.

- C) DbName: name of an ODBC source that points to the database "AutoDeployHUB"
- F) MasterIP: "SELF"
- G) MasterDbName: "", no master
- L) Reporting: "", no event reporting
- M) AutoSync: "", this server owns the up-to-date database

SPOKE:

Main management point for a large unit in the organization. Synchronizes its deployment objects from the HUB. Stores target data for all the servers in the list but doesn't propagate it upwards for scalability reasons.

- C) DbName: name of an ODBC source that points to the database "AutoDeploySPOKE"
- F) MasterIP: HUB's IP address
- G) MasterDbName: name of an ODBC source that points to the database "AutoDeployHUB"

- L) Reporting: "ud", this is the right place to report to TCM Tivoli Provisioning Manager for OS Deployment plugin
- M) AutoSync: "csd" to synchronize deployment objects (with sync.pak), "fcsd" to synchronize files as well (without sync.pak)

PROXY:

OS deployment server for a medium unit in the organization. Has its own database to perform deployments even when not connected to the SPOKE.

- C) DbName: name of an ODBC source that points to the database "AutoDeployPROXY"
- F) MasterIP: SPOKE's IP address
- G) MasterDbName: name of an ODBC source that points to the database "AutoDeploySPOKE"
- L) Reporting: "", no event reporting
- M) AutoSync: "csdh" to synchronize deployment objects and all targets (with sync.pak), "fcsdh" to synchronize files as well (without sync.pak), "csdl" or "fcsdl" to synchronize only a subset of targets

BRANCH:

OS deployment server for a small unit in the organization. Requires good connectivity with the PROXY.

- C) DbName: name of ODBC source that points to the database "AutoDeployPROXY"
- F) MasterIP: PROXY's IP address
- G) MasterDbName: "", no DB replication is needed because we use the same database as the PROXY
- L) Reporting: "", no event reporting
- M) AutoSync: "" if file replication is done using sync.pak, **f** if Tivoli Provisioning Manager for OS Deployment must synchronize files periodically

Parameter reference

Parameter reference contains syntax information on OS deployment server parameters.

Global parameters

The following list provides detailed explanations of the parameters used in the `rembo.conf` file and the web interface.

Backup

Name
Backup — IP address of backup server

Synopsis
Backup `ip-addr`

Location in the server configuration

- *web interface* > **Server Parameters > Server configuration: Base parameters**

- *rembo.conf:* Beginning of the file

Description

This parameter defines the IP address of a backup server for this OS deployment server. The backup server IP address is sent to all target computers connected to this server. If this server fails, the targets detect that the primary server has failed, and automatically switch to the backup server.

Here are some considerations regarding the OS deployment server running at the IP address specified by this parameter (*the backup*):

- The backup must be configured to use the same UDP ports as this server for the NBP, FILE, MCAST and PCAST services (that is for all specific services);
- Its file system must be strictly identical to the file system of this server, so that boot agents can switch from one server to the other in the middle of a file transfer;
- Ideally, the backup must be configured to answer DHCP discovers and PXE discovers for the same targets as this server, but with a higher delay (see (BootReplyDelay). This ensures that the remote-target boots on the backup server on the next boot.

BaseDir

Name

BaseDir — Directory for all OS deployment server files

Synopsis

BaseDir " directory"

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: File access module**
- *rembo.conf:* Beginning of the file

Description

By default under Windows, BaseDir points to c:\Program Files\Common Files\IBM Tivoli.

You must set this parameter to the directory where you have installed the OS deployment server executable program. If you are using the Windows version the BaseDir parameter is automatically configured during the setup process.

BaseDir contains a subdirectory named packages which holds the .pak server extensions.

Note: Always use forward slashes (/) in all path names you enter in the text-based configuration file.

Examples

```
BaseDir "c:/Program Files/Common Files/IBM Tivoli"
BaseDir "/usr/local/tpmfos"
```

BootDiscoveryAddress

Name

BootDiscoveryAddress — Multicast address used for PXE discovery requests

Synopsis

BootDiscoveryAddress `ip-addr`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Boot module**
- *rembo.conf:* Beginning of the file

Description

If PXE multicast boot discovery is enabled on the OS deployment server (see `(BootNoMulticastDiscovery)`), this option must be set to the multicast IP address used by the remote-boot targets when performing PXE discovery.

If this option is not set, Tivoli Provisioning Manager for OS Deployment uses the IP address `232.2.0.1`.If the OS deployment server is in the same subnet as the DHCP server, PXE discovery is not required because the remote-boot targets know that the PXE server is on the same computer as the DHCP server (option 60 set to `PXEClient`). If the OS deployment server and the DHCP server run on different computers, but are on the same IP subnet, PXE discovery is still not required because the OS deployment server intercepts DHCP requests and provides PXE information at the DHCP stage. You only require PXE discovery if the remote-boot targets are not on the same subnet as the OS deployment server, or if your existing PXE infrastructure is already setup for PXE discovery.

Example

```
BootDiscoveryAddress 232.5.6.7
```

BootDiscoveryPort

Name

BootDiscoveryPort — IP port used when listening to PXE discovery requests

Synopsis

BootDiscoveryPort `port`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Boot module**
- *rembo.conf:* Beginning of the file

Description

This parameter must be set to the port used by remote-targets when sending PXE boot requests (multicast boot discovery, or BINL discovery). The default value works with PXE bootroms which have not been modified, therefore there is no reason to change this value unless you know what you are doing.

If this option is not set, Tivoli Provisioning Manager for OS Deployment uses the IP port `4011`.

Example

```
BootDiscoveryPort 5011
```

BootNoMulticastDiscovery

Name

BootNoMulticastDiscovery — Disables PXE multicast discovery support

Synopsis

BootNoMulticastDiscovery `yes/no`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Boot module**
- *rembo.conf:* Beginning of the file

Description

If the parameter `BootNoMulticastDiscovery` (without argument) is present in the configuration file (rembo.conf config), then PXE multicast discovery is disabled on the server. The OS deployment server does not answer PXE multicast packets received from remote-boot targets.

PXE multicast discovery is used by remote-boot targets if your existing PXE infrastructure is configured to use PXE discovery. Multicast discovery is not needed if the remote-boot targets and the OS deployment server are on the same subnet, because the targets use other mechanisms to discover the OS deployment server (DHCP proxy, or DHCP option 60).

If you know that you do not need PXE multicast discovery, you can disable it by setting this parameter to true. This is suggested if you are in a large company or institution where other PXE targets can be configured to use multicast discovery on other subnets, but you do not want your server to reply to these targets.

Example

```
BootNoMulticastDiscovery
```

BootReplyDelay

Name

BootReplyDelay — Sets the delay before answering discovery requests

Synopsis

BootReplyDelay `secs`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Boot module**
- *rembo.conf:* Beginning of the file

Description

This parameter sets the number of seconds to wait before answering a PXE discovery request (PXE discovery enabled), or a DHCP request (ProxyDHCP enabled). The default value is 0 (no delay).

The only reason to delay a PXE reply is when two or more OS deployment servers are configured to run on the same subnet, with the same list of targets. The parent server can be configured with a delay of 0 seconds (answer immediately), and other servers with a delay of 2 seconds. If the parent server fails, backup servers handle remote-boot target requests.

Example

```
BootReplyDelay 2
```

DataDir

Name

DataDir — Directory for internal files

Synopsis

DataDir " path"

Location in the server configuration

- *web interface* > **Not available**
- *rembo.conf:* Beginning of the file

Description

The server uses the directory specified by DataDir to store its internal files, in particular all the files stored on its file system.

You can use this parameter to specify a new directory for the internal files of the server, for example if you want to store the server files on a new storage device with more space, but you want to keep the executable files and configuration files on the original location.

Note: You can specify the DataDir as the network driver disk by using a UNC path.

All the files stored in the directory specified by DataDir must not be modified, or the OS deployment server might not work correctly. If the specified directory does not exists, the server tries to create it.

Examples

```
DataDir "/mnt/morespace/rembo"
DataDir "C:/TPMfOS"
```

DefaultAuthDomain

Name

DefaultAuthDomain — Default authentication domain

Synopsis

DefaultAuthDomain " domain"

Location in the server configuration

- *web interface* > **:Not available**
- *rembo.conf:* Beginning of the file

Description

This is the name of the default authentication domain to use when a target sends an authentication request to the server to identify a user. This name must match the name of an existing authentication domain as defined in *Authentication domains*.

Example

```
DefaultAuthDomain "HTTP"
```

DefaultFileServer

Name

DefaultFileServer — Defines a new server for FILE services

Synopsis

DefaultFileServer ip-addr

Location in the server configuration

- *web interface* > **Not available**
- *rembo.conf:* Beginning of the file

Description

This parameter can be used to specify the default IP address of the target serving the NETfs and MCAST protocols. If you want to use the same OS deployment server for all services, you do not have to specify this parameter.

The target defined by this parameter must be configured to use the same network password (`NetPassword`) as the server containing this parameter.

This parameter is applied to all the targets contained in the group.

Example

```
DefaultFileServer 172.16.8.9
```

DefaultGroup

Name

DefaultGroup — Default group for unknown targets

Synopsis

DefaultGroup " `name`"

Location in the server configuration

- *web interface* > **OS deployment >Target Monitor: Default group**
- *rembo.conf:* Beginning of the file

Description

This parameter defines the default administrative group for unknown targets.

The "`name` " argument must be an existing administrative group.

Example

```
DefaultGroup "MyNewHosts"
```

DefaultLockOut

Name

DefaultLockOut — Locks peripherals on unknown targets

Synopsis

DefaultLockOut [`none`] [**,** `mouse`] [**,** `keys`] [**,** `screen`] [**,** `all`]

Location in the server configuration

- *web interface* >**OS deployment > Target Monitor: Default group**
- *rembo.conf:* Beginning of the file

Description

Use this parameter to lock peripherals on the remote-boot targetsduring the time Tivoli Provisioning Manager for OS Deployment is active on the unknown target. You can for example lock the mouse or/and the keyboard so that the user cannot interrupt an automatic image restoration process.

Peripherals are automatically unlocked when Tivoli Provisioning Manager for OS Deployment gives the control to the operating system (with a `HDBoot`, `LXBoot`, `RDBoot` or `DeviceBoot`).

Example

```
DefaultLockOut keys, mouse
```

DefaultOptions

Name

DefaultOptions — Startup options for the remote target computer

Synopsis

DefaultOptions [admin] [, autoboot] [, KernelFree] [, nousb] [, noautousb] [, novesa] [, noapm] [, unicast] [, noigmp2] [, noudma] [, noprotpart] [, realmodedisk] [, realmodepxe] [, routepxeirq]

Location in the server configuration

- *web interface* >**OS deployment > Target Monitor: Default group**
- *rembo.conf:* Beginning of the file

Description

Twelve different options can be set for targets. These options are used when unknown targets run under Tivoli Provisioning Manager for OS Deployment:

- `Admin:` forces the target to run in administrative mode. Administrative mode adds options in the start menu on the target: `Interact`, to run the interactive evaluator, and `Purge cache` to purge the local disk cache. Additionally, functions defined in the `admin.rbx` plugins are available (File Manager, TextEdit,...).
- `Autoboot:` tells the target to automatically reboot on unrecoverable errors. Unrecoverable errors are low-level errors which cannot be intercepted with the exception mechanisms. By default unrecoverable errors are displayed on the screen and the computer is halted. You can define the `autoboot` option if you want to reboot on unrecoverable errors.
- `KernelFree:` defines how the OS deployment serverhandles the boot of unknown targets. If the `KernelFree` value is set in the list of DefaultOptions, targets boot in kernel-free mode. When you set `KernelFree`, the following options are not applied: `NoUSB`, `NoAutoUSB`, `NoVESA`, `NoAPM`, `NoUDMA`, `NoProtPart`, `RealModeDisk`, `RealModePXE`, `RoutePXEIRQ`.
- `NoUSB:` disables USB devices on the remote-boot target.
- `NoAutoUSB:` enables USB devices only when non PS/2 connection is detected.
- `NoVESA:` disables all the graphical interface on the remote-boot target.
- `NoAPM:` disables advanced power management on the remote-boot target.
- `Unicast:` uses unicast instead of multicast when downloading files from the OS deployment server.
- `NoIGMP2:` disables IGMP version 2 to force using IGMP version 1 on the remote-boot target.
- `NoUDMA:` disables UltraDMA support for all IDE hard-disks on the remote-boot target. This can solve problems on poorly designed hardware or buggy chipsets, but at a high cost in terms of performance.
- `NoProtPart:` disables ATA-5 features.
- `RealModeDisk:` disables enhanced disk access.
- `RealModePXE:` disables enhanced PXE access.

- `RoutePXEIRQ`: reroutes network IRQ separately from the disk controller IRQ.

Example

```
DefaultOptions admin
```

DefaultPXEBootMode

Name

DefaultPXEBootMode — Selects default PXE boot behavior

Synopsis

DefaultPXEBootMode { `normal | hdboot | ignore` }

Location in the server configuration

- *web interface* >**OS deployment > Target Monitor: Default group**
- *rembo.conf:* Beginning of the file

Description

This option defines how the OS deployment server answers PXE boot requests coming from unknown targets. When mode is `normal`, the OS deployment server processes the boot request. When mode is `hdboot`, the OS deployment server tells the unknown target to immediately boot on the hard disk. When mode is `ignore`, the OS deployment server does not answer the boot request, thus giving the opportunity for another PXE server to answer.

Example

```
DefaultPXEBootMode normal
```

DefaultResolution

Name

DefaultResolution — Default screen resolution for new targets

Synopsis

DefaultResolution " `path` "

Location in the server configuration

- *web interface* > **Not available**
- *rembo.conf:* Beginning of the file

Description

This parameter defines the default screen resolution for target on a new target.

Example

```
DefaultResolution "800x600"
```

DisableBOOT

Name

DisableBOOT — Disables PXE services

Synopsis

DisableBOOT

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Boot module**
- *rembo.conf:* Beginning of the file

Description

Use this option to disable the PXE component of the OS deployment server. Targets are not able to boot on PXE anymore if this option is included in the configuration file. Use this option if you want to use the OS deployment server for remote file access only (for example when setting up the server).

Not included in the default configuration.

Example

```
DisableBOOT
```

DisableContextualMenu

Name

DisableContextualMenu — Disables the specific contextual menus in the web interface

Synopsis

DisableContextualMenu

Location in the server configuration

- *web interface* > **Server Parameters >server configuration: web interface**
- *rembo.conf:* Beginning of the file

Description

Add this option to the configuration file if you want to disable the specific contextual menus in the web interface. This can be useful if you want access to the regular contextual menu of your Web browser, in order, for example, to view the source page.

Not included in the default configuration.

Example

```
DisableContextualMenu
```

DisableCookies

This parameter is deprecated from version 7.1.1.3 onwards. The web interface does not use cookies anymore.

DisableDHCPProxy

Name

DisableDHCPProxy — Disables answers to DHCP discovers (DHCPProxy)

Synopsis

DisableDHCPProxy `yes/no`

Note: In rembo.conf configuration, the keyword `DisableDHCPProxy` is used without argument. If the keyword is present, it is assumed to be set to the value `yes` automatically.

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Boot module**
- *rembo.conf:* Beginning of the file

Description

If this parameter is set to `yes`, the OS deployment server does not send PXE replies to DHCPDISCOVER packets sent by remote-boot targets.

You can set this parameter to `yes` if you do not need this feature because either the DHCP server and the OS deployment server are on the same target, or you are using PXE multicast discovery.

The default value is `false` (that is DHCP Proxy is enabled by default).

Example

```
DisableDHCPProxy
```

DisableDragAndDrop

Name

DisableDragAndDrop — Disables the drag-and.drop feature of the web interface

Synopsis

DisableDragAndDrop

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: web interface**
- *rembo.conf:* Beginning of the file

Description

Add this option to the configuration file if you want to disable the drag-and-drop feature of the web interface. This can be useful in case of severe incompatibility with unsupported browsers. Note that the feature in known to work on most common browsers.

Not included in the default configuration.

Example

```
DisableDragAndDrop
```

DisableFILE

Name

DisableFILE — Disables FILE components of the OS deployment server

Synopsis

DisableFILE

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: File access**
- *rembo.conf:* Beginning of the file

Description

Add this option to the configuration file if you want to disable the FILE component of the OS deployment server. This can be useful if you are accessing the server from TCP targets.

Note: Do not set this option if targets are booting on this OS deployment server. If you set `DisableFile`, you must also set `DisableTCP`

Not included in the default configuration.

Example

```
DisableFILE
```

DisableHTTP

Name
DisableHTTP — Disables the console

Synopsis
DisableHTTP

Location in the server configuration
- *web interface* > **Server parameters > Server configuration: web interface**
- *rembo.conf:* Beginning of the file

Description
Add this option to the configuration file if you want to disable the HTTP component of the OS deployment server, that is the web interface. Disabling HTTP is not recommended as it forces you to run the server in command-line only. If you set `DisableHTTP`, you must run the `-c` command line option to change server parameters.

Note: A full server restart is necessary when this parameter is modified.

Not included in the default configuration.

Example

```
DisableHTTP
```

DisableHTTPOptim

Name
DisableHTTPOptim — Disables the web interface

Synopsis
DisableHTTPOptim

Location in the server configuration
- *web interface* > **Server parameters > Server configuration: deployment servers**
- *rembo.conf:* Beginning of the file

Description
Add this option to the configuration file if you want to prevent the web interface to group JavaScript, stylesheet, and image files into large files to reduce HTTP traffic.

Note: If you are creating console pages, you must set this parameter to disable optimization.

Not included in the default configuration.

Example

```
DisableHTTPOptim
```

DisableJSDropDown

Name
DisableJSDropDown — Disables JavaScript menus in the web interface

Synopsis
DisableJSDropDown

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: web interface**
- *rembo.conf:* Beginning of the file

Description

Add this option to the configuration file if you want to disable enhanced menus in the web interface, and use standard drop-down selectors instead. Note that standard drop-down selectors do not respect DHMTL layers under Internet Explorer, resulting in strange effects.

Not included in the default configuration.

Example

```
DisableJSDropDown
```

DisableNBP

Name

DisableNBP — Disables NBP services

Synopsis

DisableNBP

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Network boot module**
- *rembo.conf:* Beginning of the file

Description

Including this option in the configuration file causes the NBP component of the OS deployment server to stop answering requests coming on the NBP port. Do not set this option if your server is used by targets.

Not included in the default configuration.

Example

```
DisableNBP
```

DisableSSL

DisableSSL — Disables SSL encryption

Synopsis

DisableSSL

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: web interface**
- *rembo.conf:* Beginning of the file

Description

Add this option to the configuration file if you want to disable the SSL encryption and use unencrypted HTTP connections. This can be useful if you want a faster download time, but it is not as safe.

Note: A full server restart is necessary when this parameter is modified.

Not included in the default configuration.

Example

```
DisableSSL
```

DisableTCP

Name

DisableTCP — Disables TCP services

Synopsis

DisableTCP

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: File access**
- *rembo.conf:* Beginning of the file

Description

This option disables the TCP component of the OS deployment server. You can include this option with caution if you are not using server replication

Note: Do not set this option if targets are booting on this server. If you set `DisableTCP`, you must also set `DisableFile`.

Not included in the default configuration.

Example

```
DisableTCP
```

FileMCASTAddress

Name

FileMCASTAddress — Multicast address used for the MCAST protocol

Synopsis

FileMCASTAddress `ip-addr:port`

FileMCASTAddrCount `number`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Not available**
- *rembo.conf:* Beginning of the file

Description

`FileMCASTAddress` defines the destination multicast address and destination port used by the OS deployment server when sending multicast UDP datagram to targets requesting a file with the MCAST protocol. The default value is `239.2.0.1:10000`.

`FileMCASTAddrCount` defines the upper limit of the multicast address range used by the server (that is the number of different multicast addresses used). When combined with `FileMCASTAddress`, this parameter can control the lower and upper limits of the range of multicast addresses used by the server. The default value is `256` (that is use no more than 256 multicast addresses).

The MCAST protocol is used by targets to download large files from the server. This protocol is a proprietary protocol made using multicast UDP.

Example

To force the OS deployment server to use 1000 multicast addresses starting at `232.1.1.1`, use:

```
FileMCASTAddress 232.1.1.1
FileMCASTAddrCount 1000
```

FileMCASTEncrypt

Name

FileMCASTEncrypt — Encrypts MCAST datagrams

Synopsis

FileMCASTEncrypt

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Not available**
- *rembo.conf:* Beginning of the file

Description

If this parameter is present in a rembo.conf configuration file, or set to `yes` in the web interface, MCAST datagrams exchanged between the OS deployment server and the remote-boot targets are encrypted.

The MCAST protocol is used by targets to download large files from the server. This protocol is a proprietary protocol made using multicast UDP.

MCAST datagrams are not encrypted by default.

Example

```
FileMCASTEncrypt
```

FileServerPort

Name

FileServerPort — Port used on the server for NETfs and MCAST requests

Synopsis

FileServerPort `port`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: File access module**
- *rembo.conf:* Beginning of the file

Description

This value is used by the OS deployment server as the port for all file-related requests sent by targets. File-related requests include NETfs requests and MCAST requests.

The default value is `4013`. The value of this parameter is sent to targets during the NBP phase of the boot process. If you change this parameter, targets automatically use the new value on next boot.

Example

```
FileServerPort 5013
```

HTTPAdminName

Name

HTTPAdminName — User name of the main administrator of the OS deployment server

Synopsis

HTTPAdminName " `username` "

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: HTTP Console Security: Base Parameters**
- *rembo.conf:* Beginning of file

Description

This is the superuser name intended to be used only by the main OS deployment server administrator, in order to get access to the all configuration parameters of the server. There is only one superuser login.

Example

```
HTTPAdminName "Admin"
```

HTTPRole

Name

HTTPRole — Security roles with access to the web interface console

Synopsis

HTTPRole `"RoleName"{ Members "membername" [,...] Allowpages "pagename" [,...] Allowgroups "groupname" [,...] Policies "policyname" [,...] }`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: HTTP Console Security: Security Role list**
- *rembo.conf:* After global parameters

Description

An HTTP role allows specific members to access predefined pages on the web interface, as well as preselected administrative groups. One can also define security policies. Parameters are a single string or a list of strings separated by commas. The star `*` means *all*, without restriction.

Role members can be either individuals or groups defined by the authentication authority. An individual, who does not belong to any role, trying to log on to the web interface is denied access to the console. An individual, belonging to several roles, cumulates the page access rights and administrative group access rights of all the roles s/he belongs to. Security policies are also cumulative, resulting in a more restrictive policy when an individual belongs to several roles.

The security policies include:

- *CONF_RO* to deny changes to the server configuration
- *HOST_RO* to deny addition and removal of targets

To use `HTTPRole`, you must first have set a local authentication domain named `HTTP` with `( )AuthLocalDomain`.

Example

```
HTTPRole "RestrictedAccess" {
    Members "rembo"
    Allowpages "*"
    AllowGroups "local1", "local2"
    Policies "HOST_RO"
}
```

HTTPServerPort

Name
HTTPServerPort — TCP port for HTTP requests

Synopsis
HTTPServerPort `port`

Location in the server configuration
- *web interface* > **Server parameters > Server configuration: web interface**
- *rembo.conf:* Beginning of the file

Description
This is the TCP port used by the web interface when listening for unencrypted HTTP requests. You can set this parameter to `8080` if you want the web interface to be accessible by typing the server host name in your Web browser. The OS deployment server always listens on this port even if connections are encrypted, in order to redirect unencrypted connections to the encrypted TCP port.

Note: A full server restart is necessary when this parameter is modified.

Example
```
HTTPServerPort 8080
```

HTTPSServerPort

Name
HTTPSServerPort — TCP port for encrypted HTTP requests (SSL)

Synopsis
HTTPSServerPort `port`

Location in the server configuration
- *web interface* > **Server parameters > Server configuration: web interface**
- *rembo.conf:* Beginning of the file

Description
This is the TCP port used by the web interface when listening for encrypted HTTP requests. You can set this parameter to `443` if you want the web interface to be accessible by typing the server host name in your Web browser, prefixed with the `https` URL syntax.

Note: A full server restart is necessary when this parameter is modified.

Example
```
HTTPSServerPort 443
```

HTTPSessionTimeout

Name
HTTPSessionTimeout — Time in minutes before a web interface session expires.

Synopsis

HTTPSessionTimeout `minutes`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: web interface**
- *rembo.conf:* Beginning of the file

Description

This is the inactivity timeout (in minutes) before web interface users are automatically logged out. A typical safe value is 5 minutes, but you might want to make it longer if you receive the message `Session timed out` too often and are sure that you will never forget to log out when you leave your computer.

Example

```
HTTPSessionTimeout 30
```

Interfaces

Name

Interfaces — List of network interfaces used by the OS deployment server

Synopsis

Interfaces `ip-addr1 [ ip-addr2 ] [ ip-addr3... ]`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Base parameters**
- *rembo.conf:* Beginning of the file

Description

You will find this option very useful if you are running on a multihomed computer (that is a target with more than one network card, or with a network card and a dial-up adapter).

This option lets you specify the list of network interfaces used by the OS deployment server when receiving and sending packets to targets. If you leave this option unset, the server can use one or more interfaces (usually all of them).

You must specify a list of IP addresses to use. Each IP address must correspond to the IP address to one of the network interfaces of the OS deployment server. The list must contain at least one address.

Note: You are strongly encouraged to set this option if your computer is multihomed. Otherwise the server might not receive the network packets not originating from its official interface.

Examples

```
Interfaces 192.168.1.1
Interfaces 192.168.1.1 192.168.10.1 10.1.1.1
```

MaxLogSize

Name

MaxLogSize — Maximum size of log files created by the OS deployment server

Synopsis

MaxLogSize `size_in_bytes`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Base Parameters**
- *rembo.conf:* Beginning of the file

Description

This parameter can be used to limit the size of the log file generated by the OS deployment server. The maximal log size must be specified in bytes, and apply to all the log files created by the server (file, nbp, http, tcp, VM, and boot). If you do not specify this parameter, or set the limit to `0`, then log files are not limited in size.

Example

To limit the size of logs to 10MB:

```
MaxLogSize 10000000
```

To disable log size limit:

```
MaxLogSize 0
```

MaxPCASTSessions

Name

MaxPCASTSessions — Maximum number of simultaneous PCAST sessions

Synopsis

MaxPCASTSessions `number`

Location in the server configuration

- *web interface* > **:Not available**
- *rembo.conf:* Beginning of the file

Description

This is the maximum number of PCAST sessions that the OS deployment server accepts simultaneously. This parameter is mostly relevant if the server is running in UNICAST mode and deploying a group with UNICAST setting.

The default value is `8`. Because each session uses around 16MB of server memory, higher values must be avoided on servers with no more than 256Mb.

Example

```
MaxPCASTSessions 8
```

MaxTFTPSegSize

Name

MaxTFTPSegSize — Maximum size of a TFTP segment

Synopsis

MaxTFTPSegSize `number`

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Boot module**
- *rembo.conf:* Beginning of the file

Description

This is the maximum size of TFTP segment in bytes. This parameter can be used to reduce the size of packets sent by the OS deployment

server if required by the underlying network. This may be useful for instance when network traffic is encrypted by routers.

The default value is `512`.

Example

```
MaxTFTPSegSize 256
```

MTFTPClients

Name

MTFTPClients — The destination address and port used by the MTFTP server

Synopsis

MTFTPClients `ip-addr` [: `port`]

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Boot module**
- *rembo.conf:* Beginning of the file

Description

This is the destination IP address and port used by the server when sending multicast MTFTP datagrams to boot agents. The IP address must be a valid multicast address and must match the address sent to the target during the DHCP phase. If the deployment engine has received its PXE parameters from the OS deployment server during the DHCP phase, this address is automatically included so that the deployment engine always uses the same address/port as the server.

The default value is `232.1.0.1:8500`.

Example

```
MTFTPClients 232.8.7.6
```

MTFTPPort

Name

MTFTPPort — The port used by the MTFTP server

Synopsis

MTFTPPort `port`

Location in the server configuration

- **web interface > server configuration: Boot module**
- *rembo.conf:* Beginning of the file

Description

This is the port used by the server when listening to MTFTP requests. If the remote-boot targets are using the OS deployment server to get their PXE parameters, this value is sent in the DHCP/PXE reply so that the remote-boot target always uses the correct port.

The default value is `4015`.

Example

```
MTFTPPort 5015
```

MTFTPStartDelay

Name
MTFTPStartDelay — The initial delay on target computers before sending first MTFTP request

Synopsis
MTFTPStartDelay `secs`

Location in the server configuration
- *web interface* > **Server parameters > Server configuration: Boot Module**
- *rembo.conf:* Beginning of the file

Description
This is the delay used by remote-boot targets before sending the first MTFTP request to the OS deployment server. This delay is used by the target to listen to an existing MTFTP transfer. If the MTFTP file they are about to request is already being sent by the server on the multicast address, then the target does not request the file and listens to the packets. The longer this delay is, the more probability you have that many targets reuse the same MTFTP channel instead of requesting the file. But if you want to have a fast boot process, you must set this value to 1 (the minimum value).

The default value is 2 (seconds).

Example
```
MTFTPStartDelay 1
```

NBPServerPort

Name
NBPServerPort — Port used on the OS deployment server for NBP requests

Synopsis
NBPServerPort `port`

Location in the server configuration
- *web interface* > **Server parameters > Server configuration: Network Boot Module**
- *rembo.conf:* Beginning of the file

Description
This value is used by the OS deployment server as the port for all NBP requests sent by remote-boot targets. NBP is a protocol used by target computers to get their startup parameters (startup page, groupname,...) and to send authentication requests to the server.

The default value is 4012. The value of this parameter is sent to targets as part of the last MTFTP packet when the server sends the initial part to the PXE bootrom. If you change this parameter, targets automatically use the new value on next boot.

Example
```
NBPServerPort 5012
```

NetPassword

Name
NetPassword — Network password for remote file access

Synopsis

NetPassword " password "

Location in the server configuration

- *web interface* > **Server Parameters > HTTP Console Security: Base parameters**
- *rembo.conf:* Beginning of the file

Description

This is the password used by the OS deployment server to accept or deny network requests coming from a target, the web interface or NetClnt.

You must change this option when you install a new OS deployment server and select a password to protect your files against unpermitted access. If you are using the Windows version of the server, you are asked to enter the password during the setup.

If you are using the web interface, or NetClnt to access the files stored in your OS deployment server, the password you enter in NetClnt (or the web interface) must match the word set in `NetPassword`.

Note: This password is an important piece of your security. If you require optimal security, you are strongly encouraged to protect the configuration of the server. By default, this file is created by the installer and the password is stored encrypted in it.

Example

```
NetPassword "mypassword"
```

NetworkShare

Name

NetworkShare — UNC path of the shared partition directory of the OS deployment server.

Synopsis

NetworkShare "path "

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Network share module**
- *server.ini* Beginning of the file

Description

This is the path of the shared partition directory of the OS deployment server. This parameter can be used to increase the deployment speed of some operating systems. The target computer access the network share directly to retrieve the necessary installation files. You need to have set the partition as read-only for the user entitled to access the network share.

Example

```
NetworkShare "provisioningServer/partition"
```

NetworkUser

Name

NetworkUser — name of a user entitled to access the network share.

Synopsis

NetworkUser "name "

Location in the server configuration

- *web interface* > **Server parameters > Server configuration: Network share module**
- *rembo.conf:* Beginning of the file

Description

The name of the user entitled to access the network share, which is given in `NetworkShare`. If the user is part of a domains, you must use the syntax `Domain/User`. This user name is used by the targetcomputer to access the network share on the OS deployment server.

Example

```
NetworkUser "ClientComputer"
```

NetworkPasswd

Name

NetworkPasswd — Network password used by the target computer to access the network share

Synopsis

NetworkPasswd `"password "`

Location in the server configuration

- *web interface* > **> Server parameters > Server configuration: Network share module**
- *rembo.conf:* Beginning of the file

Description

This is the password used by the target computer to access the network share located on the server, using the `NetworkUser` user name. The password is stored in the `rembo.conf` file.

Example

```
NetworkPasswd "clientPassword"
```

TCPServerPort

Name

TCPServerPort — Defines the TCP port used

Synopsis

TCPServerPort `port`

Location in the server configuration

- *web interface* > **Server Parameters > HTTP Console Security: File access Module**
- *rembo.conf:* Beginning of the file

Description

This option defines the TCP port used by the TCP component of the OS deployment server when listening to requests coming from other servers.

The default value is `4013`

Example

```
TCPServerPort 2048
```

Authentication domains

The following information details the authentication domain references.

AuthLocalDomain

AuthLocalDomain

Name

AuthLocalDomain — Local authentication domain

Synopsis

AuthLocalDomain

AuthLocalDomain { UserGroup "group" }

Location in the OS configuration

- *web interface:* `Predefined channels/Authetication domain`
- *rembo.conf:* After global parameters

Description

A local domain uses the server local user database.

On UNIX servers, if PAM is correctly configured, a local domain uses PAM to authenticate users against any PAM-supported server (including LDAP, Smb, and so on).

Attention: Do not forget to configure PAM if you wan to use it.

Example

```
AuthLocalDomain {
    UserGroup "rembousers"
}
```

AuthNTDomain

AuthLocalDomain

Name

AuthLocalDomain — Remote authentication domain

Synopsis

AuthNTDomain { AuthServerName "hostname" UserGroup "group" }

Location in the OS configuration

- *web interface:* `Predefined channels/Authetication domain`
- *rembo.conf:* After global parameters

Description

A Remote NT domain (supported on Windows platform only) gets users from a given Windows server, for example a domain PDC.

Example

```
AuthNTDomain {
    AuthServerName "domain-pdc"
    UserGroup "rembousers"
}
```

AuthRadiusDomain

AuthLocalDomain

Name

AuthRadiusDomain — Radius authentication domain

Synopsis

AuthRadiusDomain { AuthServerAddress ip-addr RadiusSecret "secret" }

Location in the OS configuration

- *web interface:* `Predefined channels/Authetication domain`
- *rembo.conf:* After global parameters

Description

A Remote Radius domain gets the user list from a Radius compatible server.

Example

```
AuthRadiusDomain {
    AuthServerAddress 192.168.1.15
    RadiusSecret "testing123"
}
```

TCP tunnels

The following information details the TCP Tunnel references.

Remotetarget

Name Remotetarget — Host name or IP address of remote TCP target

Synopsis

RemoteHost "target"

Location in the OS configuration

- *web interface:* `Predefined channels/TCP tunnels`
- *rembo.conf:* After global parameters

Description

`RemoteHost` is a string representing the TCP remote targetto contact when a target opens this TCP tunnel. You can either specify a host name or an IP address for this parameter. Note that you need to use double quotation marks(") around the host name or IP address if you are using `rembo.conf`.

Examples

```
TCPTunnel finger {
    RemoteHost "finger.company.com"
    RemotePort 79
}

TCPTunnel ODBC {
    RemoteHost "127.0.0.1"
    RemotePort 2020
}
```

RemotePort

Name RemotePort — Numeric TCP port of remote connection

Synopsis

RemotePort port

Location in the OS configuration

- *web interface:* `Predefined channels/TCP tunnels`
- *rembo.conf:* After global parameters

Description

`RemotePort` is a number representing the TCP port to connect to when a target opens this TCP tunnel. You can get the complete list of valid port numbers in the file `/etc/services` on any UNIX computer, or in the file `/system32/drivers/etc/services` on a Windows computer.

Examples

```
TCPTunnel finger {
    RemoteHost "finger.company.com"
    RemotePort 79
}

TCPTunnel ODBC {
    RemoteHost "127.0.0.1"
    RemotePort 2020
}
```

Chapter 5. OS deployment object parameters

This section contains information on the parameters of various OS deployment objects.

Target parameters and details

To view information on target parameters, go to **Server** > **OS deployment** > **Target Monitor**. Double-click on a target to view its details.

General target info

This section contains information used to identify the target, such as the serial number, the UUID and the MAC address (also called the NIC hardware address). It also contains the host name and a textual description for the computer and its hardware model name (only valid after a hardware inventory, during deployment). You can enter the name of your organization and an administrative password. You can also configure the time zone used by the operating system, and the locale. You can also decide to enable the IPv6 protocol.

Note: Linux Windows Setting **Enable IPv6 support** to false for Windows or Linux operating system disables the IPv6 kernel module even if, for some administrative tools, it might still appear enabled.

Common networking info and Advanced network settings

This section contains information used to configure the networking part of the operating system. **Common networking info** contains information that is compatible with earlier versions. **Advanced network settings** allows you to specify several interfaces for your target. A link in this section allows you to switch back and forth between the two modes.

Note:

- When switching back to basic IP settings mode, all manually created interfaces are lost.
- If your target has more than one network adapters, use **Advanced network settings**. Using **Common networking info** can result in incorrect values at deployment time.
- **Common networking info**
  - In the **TCP/IP** settings section, you can set the `TCP/IP mode` of the target being deployed. The default value for TCP/IP settings is **Undefined, use configuration parameters**. TCP/IP parameters can be set to use DHCP if DHCP is supported on the end-user's network. If the target has to be configured with a fixed IP address, **TCP/IP settings** has to be set to either **Static IP, import from DHCP server** or **Static, manual**. In the last case, you need to switch to **Advanced IP settings mode** to specify the value of the static IP address.

    Note: Option **Undefined, use configuration parameters** is incompatible with option **IP, Netmask and Gateway defined in target** in the OS configuration.

Note: It is not possible to import DHCP settings on PowerPC and SPARC targets. Therefore, you must not use **Static, import from DHCP server** on these types of targets.

– Default gateway, DNS servers 1 to 3, and DNS domain are some of the advanced TCP/IP parameters which can be entered manually if the TCP/IP mode above is set to manual.

Note: The value of **DNS domain** is not used in Windows Vista/2008/7 deployments.

– The **Domain Suffix Search Order** defines the order in which domains are searched for target information.

– The last two fields are the WINS servers used for resolving a name.

Note:

– Any value provided in the OS configuration overrides information entered at the target level.

– Values provided in the fields following **TCP/IP settings** are overridden by DHCP if DHCP provides any value. However, values entered and visible in these fields are used if DHCP does not provide these values, independently of the value of **TCP/IP settings**.

- **Advanced network settings**

  In this mode, you get an additional link on each interface, enabling you to remove the current interface. Moreover, on the most recently created interface, you have a link allowing you to set yet another new interface.

  After clicking on **Edit**, you get three additional fields on top of the ones described for the **Common networking info** section.

  – NIC Device, which is retrieved by theOS deployment server.

  – The MAC address, provided when the interface was created.

  – A connection name, that you can freely select.

  Moreover, it is now possible to change the IP address and the subnet mask, and the other fields, if **TCP/IP settings** is set to **Static, manual.**.

  Note:

  – Any value provided in the OS configuration overrides information entered at the target level.

  – Values provided in the fields following **TCP/IP settings** are overridden by DHCP if DHCP provides any value. However, values entered and visible in these fields are used if DHCP does not provide these values, independently of the value of **TCP/IP settings**.

Windows-specific info

Contains Windows-specific information used in the deployment:

- **Windows product key** is used to set the product key of your operating system. If you have a special license with Microsoft, and you have one single product key for all your computers, it is more convenient to set the product key as a fixed value in the OS configuration you are deploying on these targets, rather than setting the product manually for each target.
- If the computer is part of a workgroup, the field network type must be set to **Stay in a workgroup**. If the computer is part of a Windows NT domain, the network type must be set to either **Join a Windows domain**

during deployment, or **Join a Windows domain on-site**, that is the first time the target boots after the deployment has been completed successfully.

- In **Workgroup name**, you can enter the workgroup or domain name to join.
- The domain administrator name and password corresponds to the user name and password to use to join the computer to the Windows NT domain. This is not used if the computer is configured to run in a workgroup.
- You can also configure video parameters such as horizontal and vertical resolution, the color depth, and the vertical refresh rate in Hertz. Typical values are 1024, 768, 32 bpp, and 60 Hz. These fields are not mandatory.
- It is also possible to enter an administrator name for your target.

Note: If a fixed property field is left empty, the value currently on the target is not altered by deployment. This is of particular importance for the administrator name, as an empty administrator name field does not mean that the administrator name will be null on the deployed target. The value of the administrator name must be checked in case of user login trouble.

UNIX-specific info

Contains UNIX-specific information used in the deployment:

- You must give a name resolution method. Select the type of service to use to convert target names to IP addresses (and vice-versa).

  Note: Some name resolution services might not be supported on all Linux distributions.
- If you are using NIS/NIS+, enter the name and IP address of the NIS server, in the form `name(ipaddress)`. If you are using LDAP, enter the IP address of the LDAP profile server. For DNS, leave this field empty (DNS servers are specified in the standard IP information panel).
- If you are using LDAP, provide the name of the LDAP profile to use.
- Kerberos is a standard way of ensuring network security in a UNIX environment. To use Kerberos, you must first configure a Kerberos server, providing security tokens for its realm. Then you can enter Kerberos parameters to automatically join newly deployed computers to your Kerberos realm. The Kerberos administration server is the central repository for managing principals (targets, services, and so on) in the Kerberos environment.
- You must provide one to three key distribution centers that provides authentication tokens when a secured operation has to be performed.
- If you want to replicate the clock of the deployed targets to a reference server using the NTP protocol, enter the server IP address or target name.
- For Solaris, you might configure the default terminal emulation to use (see `/usr/share/lib/terminfo` for possible values). The default is `sun-cmd`.

User details

You can configure the registered owner of your operating system (valid for Windows operating systems) in this section. Enter the full name and the organization for the registered owner of the operating system.

You can also enter a user login name and a user domain name if you have selected the following options for the OS configuration you are deploying on this target:

- `Add domain user to local admin group` (uses the user login name and domain name to build the domain user name)
- `Create a local account for the user` (uses the user login name to determine the local account name to create)

It is possible to enter the values of the site-specific database fields as well. These values are used to store site-specific data, such as the localization of a PC or the name of an application server. These values are then be used during deployment when working with `.ini` updates packages.

Boot settings

The boot settings section provides information about the last known PXE OS deployment servers, on the PXE boot mode, on boot engine options, and on human interface locking.

- **PXE boot mode** Consists of three sub-options:
  - `Use alternate PXE server`
  - `Boot on hard-disk`
  - `Boot on hard-disk if idle`

  Set `Use alternate PXE server` if you do not want the remote-boot target to boot on the OS deployment server but want to force it to search for another PXE server. Set `Boot on hard-disk` to force the target to boot on its hard-disk. Set `Boot on hard-disk if idle` to force the target to boot on its hard-disk only if the target is idle and there is no pending task. For redeployment, `Boot on hard-disk if idle` must not be set.
- **Deployment Engine options** Consists of the following sub-options:
  - `Reboot on unrecoverable errors`
  - `Disable USB`
  - `Disable Auto USB`
  - `Disable graphic interface`
  - `Disable APM`
  - `Disable multicast`
  - `Disable IGMP version 2`
  - `Disable Ultra-DMA`
  - `Disable ATA-5 feature`
  - `Disable enhanced disk access`
  - `Disable enhanced PXE access`
  - `Try to optimize IRQ`
- **Human interface locking options** Consists of three sub-options:
  - `Disable mouse`
  - `Disable keyboard`
  - `Disable screen`

  You can use this parameter to lock a specific peripheral during the time the deployment engine is active on the remote-boot target.
- **Boot engine options** Consists of ten self-explanatory sub-options:
  - `Use kernel-free flow` If you set this option, you cannot set any additional boot option.

- Reboot on unrecoverable errors
- Disable USB
- Disable Auto USB
- Disable graphic interface
- Disable APM
- Disable multicast
- Disable IGMP version 2
- Disable Ultra-DMA
- Disable ATA-5 feature
- Disable enhanced disk access
- Disable enhanced PXE access
- Try to optimize IRQ

- **User interface locking options** Consists of three sub-options:
  - Disable mouse
  - Disable keyboard
  - Disable screen

  You can use this parameter to lock a specific peripheral during the time the deployment engine is active on the remote-boot target.
- **Boot redirection server** and **Alternative redirection server** lets you specify one or two servers to which PXE requests from the target are automatically redirected, without needing to modify DHCP options.

Hardware details

If the target has been deployed at least once, and if the deployment scheme used for the deployment was configured to collect hardware information, then the **Target details** page shows information about hardware installed in the computer.

CPU, Memory and Disks provides information about the processor, the amount of main memory (RAM) and the size of the hard-disks found.

Note: This information is also available on Windows hypervisors, but not on Linux ones.

PCI devices provides a list of all the PCI devices found on the computer.

You can remotely turn on and off your targets using the **Switch on** and **Switch off** buttons at the bottom of the **Target details** page. Every time a wizard launches a task, it asks you for switching on the target using the **Management Interface Parameter** identifier.

OS configuration parameters

The OS configuration parameters are divided in the web interface along several tabs.

General

This tab is further divided in to **OS deployment** and **General settings**.

The **OS deployment** section contains operating system information including an operating system version, its architecture, its language, the root of the system installation and boot parameters.

In the **OS deployment** section, you can also specify (for a multi-partition image) partitions that must be excluded from one-time deployment and redeployment. In the text fields, enter the number of the partitions you want to exclude, keeping in mind that partitions with numbers 1 to 4 are primary partitions, while those with number 5 or above are logical partitions. If you want to exclude more than one partition, separate the partition numbers with commas. For example, `2,5` excludes the second primary partition and the first logical partition.

Note:

1. This option is valid only for the first physical disk.
2. This option is valid only for partitions defined in the system profile. The partition scheme defined in the system profile is always written on the disk of the deployed target. Therefore, if your target has two partitions but your system profile only one, the deployment always creates the one partition of the system profile and deletes the second partition of your target.
3. Partitions can be preserved only when the target has the same partition scheme as the one defined in the system profile. The way partition sizes are defined in a system profile makes it unlikely for a target not previously deployed with this system profile to have the same partition scheme.
4. For unattended setup profiles, all partitions must have the option **Must be deployed** set to **yes** for a successful deployment.

Disks

You can manage partition schemes: you can change partition size, you can add new partitions, you can change mount points.

Targets

In the **Fixed target properties** section, you can specify fixed parameters common to all targets using this OS configuration, to avoid specifying them for each target individually.

Note:

Values in **Fixed target properties** can contain special keywords that are replaced by dynamic information. For example, [IP] is replaced by the full IP address of the target being deployed, while [MAC] is replaced by the hardware address, also known as Media Access Control (MAC) address. To set names based on the MAC address, you can enter the following value in the **target hostname to set** field: `pc[MAC]`. The computer with the MAC address 00:01:02:03:04:05 is named pc000102030405.

The following keywords are supported:

- `[IP]`: full IP address (received by DHCP)
- `[MAC]`: hardware address
- `[SN]`: serial number as found in DMI (SMBIOS)
- `[BOMID]`: unique target identifier in the OS deployment server database
- `[AT]`: DMI asset tag
- `[GRP]`: deepest administrative group name to which the target belongs
- `[DHCPNAME]`: target name as known to the DHCP server

Every keyword supports a *range* extension if you want to include only part of the dynamic information. The range starts at value `0`. [IP3] corresponds to the last byte of the IP address (in IP addresses, bytes are separated by dots. `pc-[IP3]` becomes `pc-12` if IP address is 192.168.0.12). [IP1-3] corresponds to bytes 1 to 3. [MAC3-5] is replaced by the last three bytes of the MAC address (MAC addresses are typically represented in hexadecimal, with colons to separate the bytes). For `AT`, `GRP`, and `DHCPNAME`, the range corresponds to a substring.

If you need more flexibility, and you want to use a program to change the values of the target properties, you must access the ODBC/JDBC database directly. The BOM table contains almost all of the items that are used by a target when filling the Sysprep answer file. If your program knows how to access an ODBC/JDBC database, then you can change everything without using the web interface.

The **Target hostname to set** is limited to 15 effective characters. However, as you can use keyword substitution in this field, you can enter more than 15 characters in the field, provided these characters include character substitution encoding. An attempt is then performed to estimate the final length of the hostname. A warning message is displayed if the result of the worse-case estimation is longer than 15.

You can set the TCP/IP mode of the target being deployed. The value you can select are the following:

IP, Netmask and Gateway defined in target
: The deployment uses the values of the IP, netmask, and gateway defined in the target page.

  Note: This is not compatible with option **Undefined, use configuration parameters** in the **Target details** page.

Import all parameters from DHCP server
: All the parameters are taken from the DHCP and set as fixed values.

Force all parameters as dynamic (DHCP)
: All the parameters are set dynamically by the target requests to the DHCP server.

If a DHCP server is available on the end-user's network, you can set TCP/IP settings to use DHCP.

Note:

- Any value provided in the OS configuration overrides information entered at the target level.
- Values provided in the fields following **TCP/IP settings** are overridden by DHCP if DHCP provides any value. However, values entered and visible in these fields are used if DHCP does not provide these values, independently of the value of **TCP/IP settings**.

Note: The value of **DNS domain** is not used in Windows Vista/2008/7 deployments.

Users

In the **Fixed user properties** section, you can enter the organization name once and for all. Other required fields are a name, an organization, the time zone and the language.

Bindings

The **OS configuration binding rules** summarizes all the bindings linked to the current OS configuration

Operating system

The name of this section varies depending on the operating system of the OS configuration.

Windows **Windows**

Windows-specific info

Vista 2008 Windows 7 If you have a Volume License, set **Volume licensing** to **Yes** to avoid a product key entered on the target or on the OS configuration to be taken into account.

Note: Tivoli Provisioning Manager for OS Deployment supports the Key Management Service (KMS) key only. If you have a Multiple Activation Key, select **Volume licensing, no product key required** on the product key screen of the wizard.

Windows You can enter a Windows product key if you have an Open License or a similar site license that uses the same product key for all targets. You can copy the key in its xxxxx-xxxxx-xxxxx-xxxxx-xxxxx format and paste it directly in the product key fields by first pressing and holding down Ctrl and then pressing V.

Note: This is not a regular Ctrl+V, therefore make sure to first press Ctrl and then V.

You can provide an Administrator name and password to be set on the deployed target. The **Admin password** is limited to 15 characters.

System customization

Two options are mutually exclusive: **Add domain user to the local admin group** and **Create a local user account**.

Add domain user to the local admin group
If set, the user is added as an administrator.

Create a local user account
If set, the user is added in a local account.

If either option is set, add the administrator name as an administrator of the domain.

The option **Force user to put a new administrator password** works only in conjunction with the option **Create a local user account**. It forces the user to change the password of the local user.

UNIX **UNIX**

Net boot device indicates the network card to be used when deploying PowerPC with multiple interfaces. You must set it to the network interface that was used when registering the target in theOS deployment server. If this value is incorrect for a given target, the installation switches to interactive mode.

Network settings

For more information about the network settings, see “Target parameters and details” on page 55.

Note: If the **Join a Windows domain on-site** option is selected, the domain name must be expressed by its NETBIOS name when we deploy an unattended setup profile, or by the fully qualified domain name (FQDN) when we deploy a cloning profile.

Chapter 6. Java API

The Java API is a set of classes that map to readable and editable data structures in the server (OS deployment server).

Java classes are provided to interface directly with all objects and settings. Previously, Rembo-C scripts were used for these tasks. You can still make Rembo-C calls through the Java API but you can now write applications that use the technology without writing a single script in Rembo-C code.

Capabilities of the Java API

At a high level, the Java API offers the following capabilities:

- Booting targets with Wake-on-LAN technology within tasks
- Scheduling tasks
- Creating tasks. Tasks available with the Java API include:
  - Deployment of targets
  - Operating system restoration on targets
  - Image capture of targets (cloning)
  - Booting a preboot image on a targets: DOS, LinPE and WinPE are supported
  - Running customized Rembo-C scripts on a PXE booted computer, a web interface extension, or the OS deployment server itself
  - Object replication between OS deployment servers
  - Configuration of server parameters
  - Creation of software modules
  - Creation of unattended system profiles from the installation media
  - Creation of bootable CDs, which can be used to start the deployment engine on a target without using PXE
  - Exporting objects to RAD files (RAD export)
  - Importing RAD files (RAD import)
  - Blanking of hard disks on targets
  - Conversion of existing Rembo Toolkit images into system profiles
- Creation of *open* tasks. Every target (and every newly added target) runs an open task once and only once
- Monitoring tasks
- Cancelling tasks
- Managing server configuration
- Managing targets, such as adding targets, removing targets or changing target properties
- Managing system profiles
- Managing OS configurations
- Managing software modules
- Managing deployment template

Getting started with Java API

The `javaapi.zip` file contains the `rbapi.jar` file required for the Java API, a set of Java class examples that you can use, and Javadoc for the entire API.

When you installed the product a file called `javaapi.zip` was copied. This file contains the `rbapi.jar` file necessary to use the Java API. The `javaapi.zip` file is located in the top level folder in the directory. By default on OS deployment servers with a Windows operating system, this is `c:\Program Files\IBM\TPMfOSd`.

The communication protocol between the Java API and the OS deployment server is HTTP and XML. The Java API generates XML code which is passed to the OS deployment server over HTTP. This communication handles all the function calls from the Java API to the OS deployment server.

Configuring the OS deployment server to use the Java API

To work with the Java API, some configuration steps are required for the product.

1. By default, there is no password set up to access the OS deployment server through the Java API. When no password is set up, the OS deployment server accepts connections from trusted computers only. Therefore, you must either set up your own password to connect to the OS deployment server through the Java API, or indicate a list of trusted computers.
   - To set up your own password to connect to the server through the Java API, complete the following steps:
     a. Open a command prompt in the same directory that contains the `rbagent.exe` executable file. On Windows operating systems, the path is generally `C:\Program Files\Common Files\IBM Tivoli\rbagent.exe`.
     b. Run the `rbagent` command to encrypt your chosen password:

        ```
        rbagent.exe -d -s <ip_of_tpmfosd_server>:<tpmfosd_web_password>
        rad-hidepassword <your_new_password>
        ```

        Your new encrypted password is generated and can be found in the `Result` string. For example, an encrypted password can be `A846025095B4AA763231579210233951`.
     c. Create a file called `config.csv` that looks like this:

        ```
        HostName;APISecret
        <your_hostname>;<your_APISecret>
        ```

        where <*your_hostname*> is your Tivoli Provisioning Manager for OS Deployment host name and where <*your_APISecret*> is your encrypted password.

        This is an example of a `.csv` file:

        ```
        HostName;APISecret
        testhost;A846025095B4AA763231579210233951
        ```

        Note:

        1) <*your_hostname*> must be either a short host name or a fully qualified host name, it cannot be an IP address
        2) <*your_hostname*> must be written in lowercase only
        3) <*your_hostname*> must be immediately followed by the semicolon (`;`) without any intervening space

4) Inside config.csv, the APISecret can either be in plain or encrypted format. However, do not use the hidden form on the Java side, only <*your_new_password*> of step 2.b.

d. Copy the config.csv file to thefiles/global/rad directory. The full path of the directory is generally C:\TPMfOS Files\global\rad.

- To create a list of trusted computers, complete the following steps:

a. Create a file called config.csv that looks like this:

```
HostName;APITrusted
<your_hostname>;<list_of_trusted_ip>
```

where <*your_hostname*> is your product host name (<*your_hostname*> must be either a short host name or a fully qualified host name, it cannot be an IP address) and where <*list_of_trusted_ip*> is the list of IP addresses of the trusted computers. Trusted IP addresses must be given in the dotted notation (for example, 192.168.168.16) and separated by one space character.

This is an example of a .csv file:

```
HostName;APITrusted
testhost;198.162.32.04 198.162.35.23 198.162.12.45
```

b. Copy the config.csv file to thefiles/global/rad directory. The full path of the directory is generally C:\TPMfOS Files\global\rad.

2. Stop and start your OS deployment server.

The product is now configured to work with the Java API.

Note: There is a known problem when using HTTPS and the IBM JRE 1.5. Other versions of the IBM JRE work, as well as JRE 1.5 from other brands. If you are using IBM JRE 1.5, follow these steps to ensure the Java API works with HTTPS enabled:

1. Check if there is a ibmjsseprovider2.jar under %JAVA_HOME%/jre/lib/ext.
2. If it exists, edit the file java.security under %JAVA_HOME%/jre/lib/security.
3. In the section headed by *security.provider*, replace security.provider.x=com.ibm.jsse.IBMJSSEProvider with security.provider.x=com.ibm.jsse2.IBMJSSEProvider2, wherex is he numeric value indicating the order of the providers being called.
4. Save the java.security file and try again to use the Java API with HTTPS enabled.

Examples

The javaapi.zip compressed file contains many ready to use Java code examples for all kinds of tasks in the examples sub-folder.

Compiling and running examples

The example code in javaapi.zip can be compiled with any Java 1.5 or higher Java Developer's Kit (JDK).

To compile the Java code (on a Windows operating system, with apidir the directory where rbapi.jar is extracted), type the following commands :

```
c:\> cd apidir
c:\apidir> javac -classpath .;rbapi.jar examples\*.java
```

Where apidir is the directory where rbapi.jar is extracted

To run an example class, in this instance `TestRADImport`, with `10.10.10.10` as server IP address, on the default port (`8080`), with an API secret of `mypassword`, type:

```
c:\> cd apidir
c:\apidir> java -cp rbapi.jar;. examples.TestRADImport 10.10.10.10 8080 mypassword
```

Understanding the sequence of procedure calls

Understanding the sequence of procedure calls between the Java client application, its objects, the OS deployment server, the SQL database, and the web interface extension (rbagent) is necessary to use the Java API adequately.

`TestRADImport.java`, which imports a RAD file to the OS deployment server through the Java API, is a good example to detail the process. To get the most of this example, follow the source code of the example, provided in `javaapi.zip`, while you view the explanatory diagrams and read their legend.

RAD file import is a process in four sequential top-level steps. For each top-level step, there is a diagram containing items such as the Java client application, Java objects, and the OS deployment server, among others. These items are linked by lettered arrows representing procedure calls. Follow the arrows in alphabetic order. You can discover the meaning of each arrow in the list which follows each diagram.

1. Asking the OS deployment server about known targets

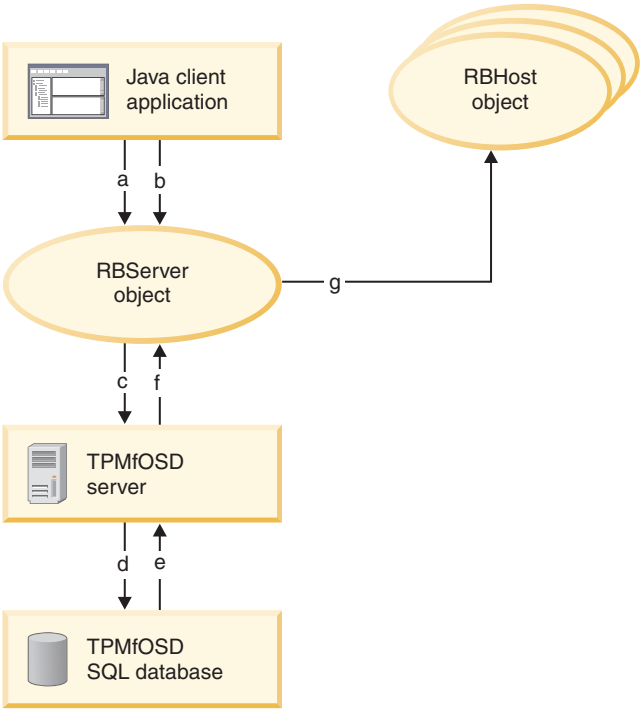

Figure 1. Step 1

a. The Java client application creates a new `RBServer` object, the OS deployment server is referenced by IP address and port
b. The Java client application calls the `RBServer.getHosts()` method
c. `RBServer` sends a remote procedure call to the OS deployment server (XML over HTTP)

d. The OS deployment server sends an SQL query to the database to get the list of targets-->
e. The database returns an array of targets is returned -->
f. The OS deployment server encodes the result in XML
g. The `RBServer` object creates one `RBHost` object for each targetID returned by the query

2. Checking the status of the web interface extension on the chosen target

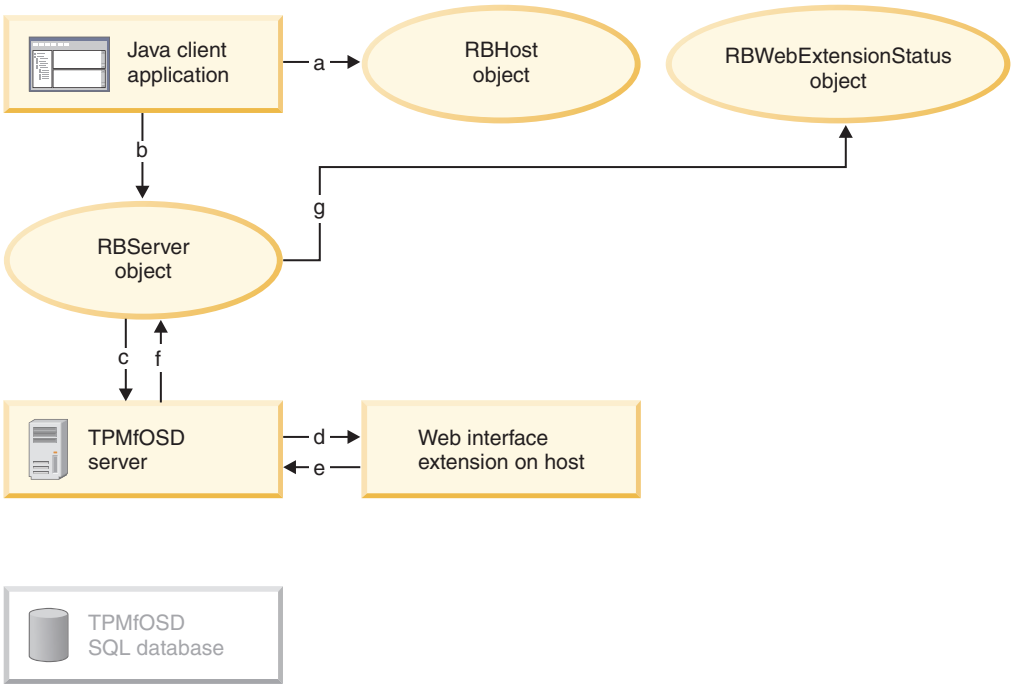

Figure 2. Step 2

a. The Java client application calls `RBHost.getIPAddress()` to lookup attributes of the target
b. The Java client application calls `RBServer.isWebExtensionReady(IP)` to check the status of the web interface extension (rbagent) on the target selected by the user
c. `RBServer` sends a remote procedure call to the OS deployment server
d. The OS deployment server sends a *probe* packet to the web interface extension
e. The web interface extension returns its status to the OS deployment server
f. The OS deployment server encodes the result in XML
g. The Java client application gets the status of the web interface extension in a new `RBWebExtensionStatus` object

3. Performing the RAD import task

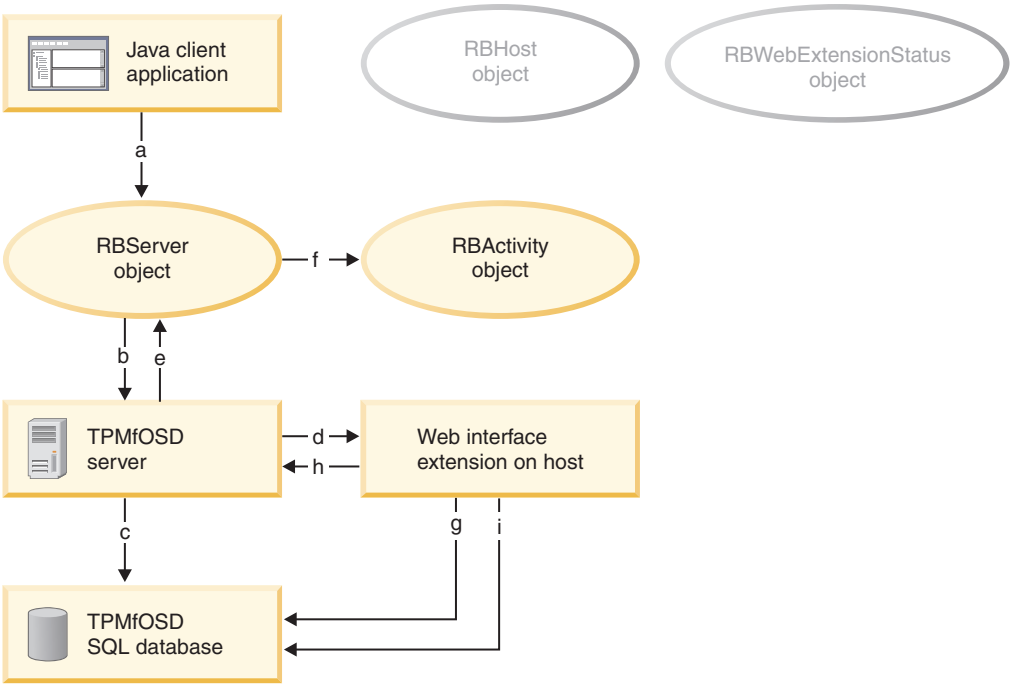

Figure 3. Step3

a. The Java client application calls `RBServer.registerRADImportActivity` with the name of the RAD file and the target `RBHost`
b. The `RBServer` object sends a remote procedure call to the OS deployment server
c. The SQL database records a new task with its type (RAD import) and parameters (file name, target, schedule, ...)
d. The OS deployment server sends a signal to notify the target that there is something to do
e. The OS deployment server sends, in XML, a reference to the new task to `RBServer`
f. The target receives a new `RBActivity` object
g. The web interface extension finds the new task to perform in the SQL database
h. The web interface extension uploads the RAD file content on the OS deployment server
i. The web interface extension reports the progress and the completion of the task to the SQL database

4. Monitoring task progress

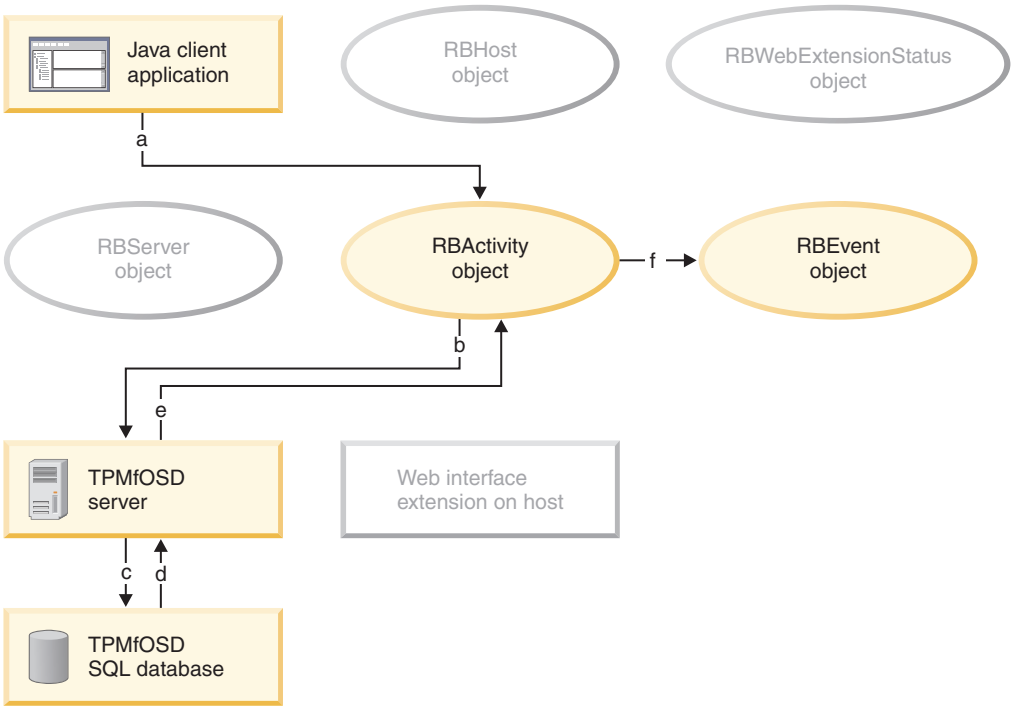

Figure 4. Step 4

a. The Java client application calls `RBActivity.getEventsFromServer` to check the task progress and status
b. `RBActivity` sends a remote procedure call to the OS deployment server
c. The OS deployment server uses an SQL query to get the list of events for this task
d. The SQL database send the task status to the OS deployment server
e. The OS deployment server encodes this task in XML
f. The Java client application receives a new `RBEvent` object that describes the current status of the task

Example classes

Here is the list of example classes present in `javaapi.zip` and a short description of what each one does.

TestAbortingActivities.java
Shows ways of cancelling a task that has been registered. This test shows both cancelling the whole task, and cancelling the task on only one specific task target.

TestAbortingActivitiesForHost.java
Shows an example of cancelling all tasks for a specific target.

TestActRepl.java
Shows how to run a task on a child, even if it has been created by connecting to the parent server. First a task template is created on the parent server and then it is replicated to the child workstation by performing an object copy task. This ensures that when running a task that uses the template, the child server has this information. Afterward a task is scheduled that uses the task template in question. If the target boots off the

child server, it will not succeed unless the object copy task of the task template to the parent server was successful.

TestAddRemoveHost.java
Shows a very basic example of adding or removing targets.

TestAgentScript.java
Runs a Rembo-C script on an target running the web interface extension.

TestBlackDevAndMod.java
Displays information that you can retrieve about black-listed PCI devices and models.

TestBootCDCreation.java
Creates a bootable CD from a target running the web interface extension. You can use the bootable CD to start the deployment engine on a target without using PXE.

TestCloneFromImageFile.java
This is an example showing capturing a cloned system profile from a reference image. The reference file can be a Windows Vista WIM image, or a Solaris Flash file. This is basically an example of using the Java API for what you can do using the new profile wizard if you select the option "Cloning from a reference file". Rembo toolkit files can also be cloned into system profiles, however for these it's best to look at the example TestToolkitProfileCreation

TestCloneHost.java
This example shows how to capture a cloned system profile from a reference workstation - Java API equivalent to running the new profile wizard and choosing "Cloning from a reference workstation". This is the preferred way to capture a system profile from a target, and should be used in place of the example class TestRBImageCaptureTemplate

TestImageManagement.java
Copies objects (task templates, system profiles, or software modules) from one OS deployment server to another. To run this test you must have multipleOS deployment servers working together in your environment.

TestLinuxPreboot.java
Boots a LinPE image on a target PXE target.

TestMultiServer.java
Provides information about how to setup a multi-server environment. Also shows a specific multi-boot server scenario of creating a task template, moves it to another server, and changes the scope so that the other server becomes the owner of the object.

TestOpenActivity.java
Runs an open task: this task runs once and only once on all targets that PXE boot off the OS deployment server.

TestPassword.java
This test class tries connecting to the OS deployment server using the Java API without supplying a password - typically this is not allowed, as an APISecret (java api password) needs to be setup in config.csv - however, if you use the method in config.csv of supplying "APITrusted" and list the targets that you trust, and then do not supply an APISecret - you can test that those specific targets can connect to the OS deployment server without a password with this example.

TestRADExport.java
Exports object(s) to a RAD file using the Java API.

TestRADImport.java
Imports a RAD file using the Java API.

TestRBCustomTemplate.java
Runs a custom Rembo-C script on a PXE target.

TestRBCustomTemplateUsingVariant.java
Similar to `TestRBCustomTemplate`, but also demonstrates the use of a variant to override parameters from the template during task registration without modifying the template itself.

TestRBDeviceBlankingTemplate.java
Performs military grade device blanking of a hard drive on a PXE target.

Note: Run this test with caution as it blanks disks.

TestRBEvents.java
Monitors events: you can run this test while other tests are running to monitor their progress.

TestRBHosts
Illustrates some properties of `RBHost` that can be gathered with the Java API.

TestRBImageCaptureTemplate.java
Refer to the TestCloneHost.java example for details.

TestRBLogFileLines.java
This class shows some examples of how to retrieve the contents of log files from the OS deployment server through the Java API.

TestCloneHost.java
Creates a system profile cloned from a reference computer (a PXE target).

TestRBOSDeployment.java
Deploys an OS configuration to a PXE target with some customization.

TestRBOSRestoreTemplate.java
Restores a system profile to a PXE target.

TestRBPrebootExecuteTemplate.java
Boots a DOS image on a PXE target.

TestRBServerConfig.java
Displays information about the OS configuration of the OS deployment server that you can retrieve and modify using the Java API.

TestRBServerReset.java
Restarts the OS deployment server through the Java API.

TestRBServerStatus.java
Gets the status of the OS deployment server through the Java API.

TestSoftwareComponentCreation.java
Creates software modules using the Java API. This example requires a target running the web interface extension.

TestStress.java
Runs many tasks and does inefficient polling to put a heavy load on the OS deployment server.

TestStressSmart.java
Shows a better method than `TestStress.java` for event polling.

TestSubnets.java
This test shows some of the subnet information available using the Java API, and how to probe the subnet again for multicast capability, change the multicast status and so on.

TestToolkitProfileCreation.java
Creates a system profile from an already existing Rembo Toolkit image.

TestUnattendedProfileCreation.java
Creates an unattended system profile from CD/DVD media.

TestWinPE.java
Boots a WinPE image on a PXE target.

TestWinPE2.java
Boots a WinPE2 image on a PXE target

TestWinPE2Update.java
This example shows how to update drivers in an existing WinPE2 image.

Deployment server configuration and maintenance

This documentation describes the initial classes that you will use, settings, tasks, and troubleshooting information.

Server connection and status

The first class that is used to establish a relationship with a OS deployment server is `RBServer`.

It can be instantiated by providing the IP address and port of the OS deployment server and a valid authentication token. When a method returns an object, see the information later in this section for details on the class.

```
RBServer(InetAddress ip, int port, String token)
```

There is another constructor for `RBServer` that takes an additional parameter, `java.util.logging.Level verbosity`. This parameter can be used to set a different level of details in the traces. See for a"Controlling API traces" on page 85 longer discussion on traces.

The following methods should be used on instances of `RBServer` in order to get access to the various aspects of the server status:

- `bool isReady(int timeout)`: Probes the availability of the OS deployment server. The parameter `timeout` is specified in milliseconds.
- `RBServerStatus getStatus()`: Returns an object describing the current server status.
- `RBServerActHistory getActHistory(int StatType, Date start, long duration, long resolution)`: Returns an object describing the specified server task history.
- `bool isWebExtensionReady(InetAddress ip)`: Asks the OS deployment server to check the availability of the web interface extension. The variable `ip` is the address of the computer on which the web interface extension is required. The caller must expect negative answers the first couple of times they call this method because the answer of the agent never comes back immediately. The appropriate way to use this method is in a loop and with a few seconds delay between calls.

An instance of the class `RBServerStatus` represents a view on the server general status at the time it was instantiated, as displayed on the page **General Information** of the web interface. Individual methods such as `getVersion`, `getHostCount`, and `getSubnetCount` provide access the individual values describing the state.

Deployment server settings

The following methods should be used on instances of `RBServer` in order to read and update the server OS configuration:

- `RBServerConfig getConfig()`: Returns an object describing the current server OS configuration
- `void updateConfig(RBServerConfig config)`: Update the server OS configuration according to the object provided. This can trigger a server restart depending on the changes applied.

  Note: When this method returns the attributes of object, `config` can be different because the OS deployment server actually merges differences with its own records and sends back the resulting object. This remark applies to all `update` methods.
- `RBBootServerRecs getBootServers(String filter, String order)`: Returns an object describing one or several server records in the server replication table.
- `RBServerBlackDev[] getBlackListedDevices()`: Returns the full list of black-listed PCI devices .
- `RBServerBlackMod[] getBlackListedModels()`: Returns the full list of black-listed target models.
- `void restart()`: Force a restart of all threads of the OS deployment service.

An instance of the class `RBServerConfig` represents a view on the current server OS configuration, exactly as it is available in the built-in Rembo-C variable `RemboConf` on the OS deployment server. Individual methods are provided to access the individual values of each setting.

The arrays of `RBServerBlackDev` and `RBServerBlackmod` instances represent the list of PCI devices and target models requiring special handling by the deployment engine, as displayed in “Hardware handling” on the Web interface. Individual methods are provided to access the individual values of each record.

Targets

At any time, the OS deployment server can know a large number of targets and the API to access them must remain efficient. For this reason, the API is based on a query and result set model.

Getting the attributes of a target or of a set of targets is, for scalability reasons, a two stage process:

1. Query the database about known targets matching a given search pattern with `getHosts`; a potentially large number of small records (only IDs) is created.
2. Query the actual attributes, a large record, of a small number of targets with `elementAt` or `subList`.

This is the approach used in the Web interface to respond in a timely manner to user actions. It is based on the fact that the console can display only a small

amount of data at a time, so it is not necessary to fetch data that cannot be displayed. Additional records are fetched in a lazy way when they can really be shown to the console operator.

The following method should be used on instances of RBServer in order to query for known targets:

`RBHostRecs getHosts(String filter, String order)` builds a set of records matching the specified SQL filter condition, and sorted by the specified SQL ordering expression. Because SQL syntax is not exactly the same for all DBMS supported by the OS deployment server, only very simple operators should be used, for example, `<,>,=,like`, and `AND`. Field names which correspond to the attributes of `RBHost` must be enclosed in square brackets; this helps the server replace them with the actual field names expected by the database. For example,`getHosts("[IP] like '9. ", "[HostName] asc")`

A current description of all tables and columns in the SQL database can be downloaded directly from the server using a Web browser and the URL:

http://<server_name_or_IP>:8080/virtual/AutoDeployDistrib.ini.

The following methods can then be used on instances of `RBHostRecs`:

- `int size()`
- `RBHost elementAt(int index)`
- `RBHost[] subList(int fromIndex, int toIndex)` with the same parameter semantic as `java.util.Vector.subList` (element at `toIndex` not included in the list)

When several targets should be displayed simultaneously, use the `subList` method rather than making several individual `elementAt` calls.

A record of type `RBHost` includes all attributes editable on the target details page in the Web interface of the OS deployment server. Methods are provided to access them individually, for example, `getHostName` and `setHostName`. After making changes to a `RBHost` instance, call its `update()` method to propagate the change to the OS deployment server. Removing or deleting a target can be done with the following method:

- `void RBHost.remove()`: Removes the specified target record from the server database.

Manually adding a target is different, because it must handle the case where the target already exists in the database, which should not be considered as an error because the computer can have been rightfully discovered. This is handled by the target registration API, where each of the four parameters can be empty if unknown but at least one should be specified:

```
RBHost RBServer.registerHost(String IP, String MAC,
 String UUID, String Serial)
```

This returns the corresponding target record or creates a new one if none exists yet. The IP address must be given in decimal dotted format, for example, *192.168.2.34*. The MAC address must be given in hexadecimal, with or without colons, for example, 00:16:41:17:AF:DE or 00164117AFDE.

`RBHost` methods `getPxeBootMode` and `setPxeBootMode` use an additional class `PxeBootMode` that can take the following several values:

- IGNORE: the PXE server ignores all requests coming from this target.
- HDBOOT: the PXE server sends an answer to cancel the network boot and perform a harddisk boot.
- HDBOOT_IF_IDLE: the target is loaded and performs a harddisk boot if there is no activity for this target.
- NORMAL: the target is loaded and waits for tasks to perform

Deployment objects

The following objects are required to run an OS deployment:

- System profiles (disk images or installation CD images).
- The differentOS configurations that can be defined for each system profile.
- The automatic binding rules that can be defined for each OS configurations.
- Optional software modules (including drivers and DOS floppy images).
- The automatic binding rules that can be defined for each software modules.

Although these objects typically exist in small numbers only on a OS deployment server, a mechanism that allows the caller to select how many objects are loaded together is implemented. It is the same principle as for RBHosts.

The following method should be used on instances of RBServer to work on system profiles:

- RBSystemProfileRecs getSystemProfiles(String filter, String order): Returns an object that knows all system profiles matching the filter expression (same semantic as getHosts) and can fetch subsets from the server using the three methods:
  - int size()
  - RBSystemProfile elementAt(int index)
  - RBSystemProfile[] subList(int fromIndex, int toIndex) with the same parameter semantic as java.util.Vector.subList

There are three ways to create system profiles:

1. Capture an image from a reference computer.
2. Capture the contents of an installation CD or DVD.
3. Capture an image from a reference image file (a windows WIM image or a Solaris Flash Archive file).

The partition scheme used during a deployment is part of the system profile. It is not practical to change this value inside RBSystemProfile for each deployment, and not always possible to specify partition sizes as percentage of the target of the hard disk size. It is possible to select the partition scheme dynamically, and other target parameters as well, every time a deployment task is performed. See the documentation on RBDiskLayout.

RBConfiguration class

Configurations are used to automate the assignment of parameters to targets during the deployment of a system profile.

For a longer description of the properties, in the web interface, go to **Server** > **OS deployment** > **System Profiles**.

The Fixed_ prefix in the property names is a reminder that at deployment time, the value found in the RBConfiguration overrides values found in the RBHost. The

values in RBHost are applied only if the corresponding Fixed_ value in RBConfiguration is empty. After the deployment, the values in RBHost reflect the actual values that have been used. Fixed_ values themselves can be over-ridden with values provided in the RBOSDeploymentTemplate in the argument params Map of RBServer.registerActivity(), or in the params Map of RBActivity.addTarget().

RBConfigurationRecs getConfigurations(String filter, String order) returns an object that knows all configurations matching the filter expressions, (same semantic as getHosts) and can fetch subsets from the server using the three methods.

RBSystemProfile also has a getConfigurations method to retrieve the configurations that belong to a specific system profile.

RBSoftwarePackage class

RBSoftwarePackageRecs getSoftwarePackages(String filter, String order) returns an object that knows all software modules matching the filter expression (same semantic as getHosts) and can fetch subsets from the server using the three methods:

- int size()
- RBSoftwarePackage elementAt(int index)
- RBSoftwarePackage[] subList(int fromIndex, int toIndex) with the same parameter semantic as java.util.Vector.subList.

RBSoftwarePackage provides methods to query individual attributes of the instances, including getDescription and setDescription to virtually rename the object, corresponding to the various fields displayed on the Web interface.

In regards to pass numbers, older versions used to put software modules in *passes* with reboots between the passes. The current version uses a more refined concept of *stages*. The reboot between stages is optional, therefore it is possible to install two software modules in a specific order, without having to reboot between them. When a software module is bound by an automatic binding rule to the target of a deployment task, the system uses SoftSeqID to decide when the package must be installed. The default uses the stage found in the RBSoftwarePackage.

For OS Deployments, an ordered list of software modules can be added to the RBDeploymentParams. After the OS is deployed, the software modules are installed based on the order of the software modules in the softwareSequence and passed to setSoftwareToDeploySequence. It is possible to specify reboots in the softwareSequence between the installation of two software modules.

RBBootServer class

By default, each OS deployment server automatically installs and uses its own private database. On Windows, this is done by creating and using Open DataBase Connectivity (ODBC) source AutoDeploy, linked to the file c:\program files\ibm\tpmfosd\autodeploy.mdb. You can force the server to use another existing ODBC source with MSI properties ODBC_DSN, ODBC_USERNAME, and ODBC_PASSWORD.

After installation, it is also possible to select a different source using the fields DbName, DbUser, and DbPass of config.csv. For more information, go to the Web address: http://www-1.ibm.com/support/docview.wss?uid=swg21247013. The

advantage of using config.csv is that the same configuration file can be used for multiple servers, and that it can be updated or reloaded at any time.

MultipleOS deployment servers can use the same database. In this case, each server automatically adds a record in the Servers table. These records are available in the Java API as `RBBootServer` objects.

A sample config.csv, for three servers sharing a common MS SQL database can be found in the `javaapi.zip` file, inside `examples/TestMultiServer.java`.

`RBBootServerRecs getBootServers(String filter, String order)`: Returns an object that knows all OS deployment servers registered in the database that match the filter expression (the same semantic as `getHosts`).

Deployment server tasks

A task is an operation that the user wants to perform on one or several targets.

Three classes are used for task management:

1. `RBActivityTemplate`: Defines what must be done, for example, deploy a given Windows XP cloning image in multicast, with no bandwidth limitation and power-off when done.
2. `RBActivity`: Defines the temporal aspect, for example, start not before than today 9 p.m. and not after tomorrow 2 a.m..
3. `RBActivityTarget`: Records one intended of the task and keeps track of progress and completion. All the instances are stored in the SQL database in order to persist when a target needs to restart, or when the daemon is restarted.

Task templates

`RBActivityTemplate` is an abstract class with several subclasses designed to represent different kind of tasks.

The role of a task template is to store a set of predefined parameters to use when instantiating future tasks. The constructors and methods reflect the characteristics of the task. The current list of subclasses includes

- `RBPrebootExecuteTemplate` is used for hardware configuration.
- `RBprofileCreationTemplate` creates system profiles from CD-ROM/DVD (unattended setup), from reference computers (cloning), or from image files.
- `RBOSRestoreTemplate` restores cloning profiles without customization.
- `RBCustomTemplate` is used for arbitrary Rembo-C code execution.
- `RBOSDeploymentTemplate` performs customized deployments.
- `RBDeviceBlankingTemplate` performs military-grade device blanking.

Note: For some categories of tasks, typically web interface extension tasks such as creating a software module or creating RAD files, there is no explicit task template. This is because instances of such individual tasks intrinsically need different parameters.

Depending on the template, the targets are either OS deployment servers, web interface extensions, for example, `rbagent.exe`, or PXE clients.

The instances of RBActivityTemplate have a unique internal ID and a longer textual description. They know their type and are managed using the RBServer method :

- RBActivityTemplateRecs getTemplates(String filter,String order)

Methods of RBServer are used to create new templates on the OS deployment server and get corresponding Java objects.

RBActivityTemplate has two methods, getEndAction and setEndAction to manage end actions. The end action indicates what the target must do when the action is complete:

- STAY: the target waits for more tasks.
- HDBOOT: the target performs a boot on harddisk.
- REBOOT: the target performs a warm reboot.
- POWEROFF: the target is switched off.

RBCustomTemplate

RBCustomTemplate registerCustomTemplate(String description, String script, Map defaultParams) creates a new task template on the OS deployment server to run a custom script. The method will return an object that describes the template. With defaultParams, the caller can pass arbitrary pairs, key and value, to the script. Keys and values must be textual strings.

RBPrebootExecuteTemplate

RBPrebootExecuteTemplate registerPrebootExecuteTemplate(String description, RBSoftwarePackage diskImage, String fsType, boolean unloadUndi, int size, RBFile[] filesToAdd, String[] filesToPatch, Map customParams) creates a new task template on the provisioning server for the execution of a Preboot image, for example, a DOS floppy image.

Before a preboot task boots the ramdisk it currently sets the progress to 100 and the status to COMPLETED.

RBOSDeploymentTemplate

RBOSDeploymentTemplate registerOSDeploymentTemplate(String description, RBConfiguration config) creates a new task template on the OS deployment server for the deployment of the given OS configuration

To supply custom values at deployment time in a more flexible way than what the RBConfiguration allows, class called RBDeploymentParams was defined. This class can be instantiated by the target, using setter methods for relevant fields in RBHost.

The customization done at this level is applied to all targets of the task. The properties defined in the RBDeploymentParams overrides the same properties defined in the OS configuration and system profile. In addition, the partition scheme can be set for all of the task targets from the RBDeploymentParams. The partition scheme is first retrieved from the OS configuration with the getDiskLayout method which returns an RBDiskLayout object. This RBDiskLayout can be modified then set in the RBDeploymentParams with the method setDiskLayout. An ordered list of software modules can be specified in the RBDeploymentParams that is installed after the OS deployment.

The installation of software modules is always related to deployment tasks. For a given deployment, the list of software modules to install is built dynamically. The list contains packages selected by an automatic binding rule and packages that are manually selected.

Automatic binding rules are used to install a driver when a PCI device is present on the target, or to add an application every time a given operating system image is deployed.

For manual selection, all instances of `RBOSDeploymentParams` can have an optional attribute `SoftwareToDeploy` that contains an array of `RBSoftwarePackage` objects.

RBImageCaptureTemplate

```
RBImageCaptureTemplate registerImageCaptureTemplate
(String description, boolean withCMOS,
boolean withProtectedPartitions, boolean forceRABCapture,
boolean allowNoSysPrep)
```

This registers a new OS image capture template on the OS deployment server. The task creates a new system profile, with all OS configurations that can be detected on the target.

When registering a task that uses this template, the call to `RBServer.registerActivity()` must have the following specified in the `params` parameter:

1. An entry with key `profiledesc` and value which is a `String` representing the textual description of the profile.
2. An entry with key `profilecomment` and value which is a `String` representing a comment for the profile.

These must be specified because the same template can be used to take several images; therefore the strings used to describe and identify the new profile are given at the task level.

RBOSRestoreTemplate

```
RBOSRestoreTemplate registerOSRestoreTemplate
(String description, boolean restoreCMOS,
boolean restoreProtectedPartitions, boolean runSysPrep)
```

This creates a new task template for the deployment of a cloning profile without customization.

RBSoftwarePackageParams

Parameters used for software module creation tasks are always different, therefore using a template does not make sense. Consequently, software module creation tasks use an implicit template and use the class `RBSoftwarePackageParams` to specify the task parameters directly at the time when the task is scheduled, similarly to what can be done using `RBDeploymentParams`.

Because the type of parameters relevant for the various types of software modules differ, static methods are provided to instantiate and preset most parameters. In any case, various methods enable you to read and update optional parameters, for example, to define static software ordering.

When a set of parameters is obtained, the `RBServer` method, `probeSoftwarePackageCreation`, can be called to simulate the creation of the software module with the specified set of parameters and return the effective

values that would be completed by the software module creation process. For example, when a command line is automatically generated or when a default description is used. Subsequently, or alternatively, the `RBServer` method, `registerSoftwarePackageCreationActivity`, must be called to actually perform the software module creation. Progress monitoring can be done as with any other type of task.

All the software module creation tasks must use as the target a computer running the Web interface extension, `rbagent`. When the target is a OS deployment server, the source path must be visible under the `\import` directory (but symlinks are permitted, including under Windows where the -specific symlink files can be used). In Windows, file paths must be an absolute path preceded with the disk letter, for example, `C:\temp\...` or UNC paths, `\\file-server\share\...`. In order to have UNC path work correctly, the `rbagent` process must run as a domain user having privileges to access the network share.

The software module creation tasks started using the following parameters captures a diskette into a new software module, for the purpose of running the floppy in a RAM disk:

```
RBSoftwarePackageParams newFloppyCreationParams
(String pkgDescr, String comment, String sourcePath)
```

The software module creation tasks started using this parameter captures a WinPE CD into a new software module, for the purpose of running WinPE in a RAM disk:

```
RBSoftwarePackageParams newWinPECreationParams
(String pkgDescr, String comment, String sourcePath)
```

The software module creation tasks started using this parameter captures a Linux kernel and initial RAM disk into a new software module, for the purpose of running Linux in a RAM disk:

```
RBSoftwarePackageParams newLinuxPECreationParams
(String pkgDescr, String comment, String sourcePath,
String kernelName,
String ramdiskName)
```

The software module creation tasks started using the following parameters capture a Windows MSI installation file into a new software module, for the purpose of installing an application:

```
RBSoftwarePackageParams newMSICreationParams
(String pkgDescr, String comment, String sourcePath, String destPath,
String cmdLine)
```

The software module creation tasks started using these parameters captures a Linux RPM installation file into a new software module, for the purpose of installing an application:

```
RBSoftwarePackageParams newRPMCreationParams
(String pkgDescr, String comment, String sourcePath,
String destPath, String cmdLine)
```

The software module creation tasks started using the following parameters captures a Solaris PKG installation file into a new software module, for the purpose of installing an application:

```
RBSoftwarePackageParams newSolarisCreationParams
(String pkgDescr, String comment, String sourcePath,
String destPath, String cmdLine)
```

The software module creation tasks started using these parameters captures a Windows driver installation file into a new software module, for the purpose of installing a driver:

```
RBSoftwarePackageParams newWinDriverCreationParams
(String pkgDescr, String comment, String sourcePath,
String destPath, boolean createRules)
```

The software module creation tasks started using these parameters captures an arbitrary set of files, scripts, ini files, registry exports (in NT4 format), into a new software module to customize the deployments.

```
RBSoftwarePackageParams newFileSetCreationParams
(String pkgDescr, String comment, String sourcePath,
String destPath, String cmdLine,
boolean substKeywords, boolean removeFiles)
```

The software module creation tasks started using the following parameters capture a command line for execution during a deployment.

```
RBSoftwarePackageParams newCmdLineCreationParams
(String pkgDescr, String comment, String cmdLine)
```

Task variants

Each task template has a corresponding task variant class.

Variants are used to set or override parameters in the template at run time without modifying the template itself. Each task template contains the method `getVariant();` which returns an `RBActivityVariant` that corresponds to the template in question. Attributes of the variant are the same as the attributes of the template itself (in certain cases, variants also contain additional attributes that apply only at task registration time). This variant can then be modified and passed in when registering the task (with the `registerActivity()` method) instead of the template itself.

For example, if you have an `RBOSRestoreTemplate` that has an end action of `STAY` (the default), but for one particular operating system restore task, you want to change the end action to `HDBOOT` (to boot off the hard drive after restoring the operating system), you can follow these steps:

1. Get an `RBOSRestoreVariant` based on your template by calling the `getVariant()` method on the template
2. Modify the end action of `RBOSRestoreVariant` by calling `setEndAction(RBActivityTemplate.HDBOOT)`
3. Call the `RBServer.registerActivity()` method, but pass in the modified `RBOSRestoreVariant` instead of the `RBOSRestoreTemplate`

Task scheduling

The class `RBActivity` is used to schedule the start of a task. Instances are created using a method of `RBServer`.

Here are a couple of examples:

```
RBActivity registerActivity(String description, RBActivityTemplate template,
boolean suspendOnError, boolean doWakeOnLan, Map params, RBHost target)
```

It creates a new task of the given type on the OS deployment server and returns an object that describes it. This simple class method schedules the task to start immediately and have no expiration date.

```
RBActivity registerActivity(String description, RBActivityTemplate
template, boolean suspendOnError, boolean doWakeOnLan, Map params,
 RBHost[] targetHosts, Date execDate, Date doneDate, Date expireDate)
```

This method creates a new task of the given type and returns an object that describes it. It registers several targets, called `targetHosts`, and gives better control on the scheduling of the task. The parameters of this method, `execDate` and `doneDate` are used to define a window for the beginning of the task. Targets can finish performing the task after the end of the window, but it is too late to start.

The parameter `execDate` can also be used to select between multiple possible tasks. If an idle target has several choices, it takes the task with the smallest `execDate` first.

The expiration date, `expireDate`, is used to cleanup database records after a reasonable amount of time, therefore it is not necessary to manually remove them. `SuspendOnError` is used to prevent the start of other tasks by targets that have not successfully completed this one.

There are two methods in `RBServer` to stop the execution of tasks:

1. `boolean[] abort(RBActivity[] rbActivities)`: Asks all of the targets of given tasks to skip or abandon the task.
2. `boolean[] abort(RBActivityTarget[] rbActivityTargets)`: Asks all of the given targets to skip or abandon the task.

Task targets

Instances of `RBActivityTarget` represent the actual target performing the task. There are several ways to provide access to task targets.

- `RBServer.registerActivity`
- `RBActivity.addTarget`
- `RBActivity.getActivityTargets` returns an `RBActivityTargetRecs` object that gives access to `RBActivityTarget` instances.

When an `RBActivityTarget` generates a result, the method `getActivityTargetResults()` can be used to retrieve the resulting object. The class of the result depends on the type of task. For more information, see the class name and description in the related task templates.

Events

Monitoring events on OS deployment servers is based on the same HTTP communication channel as other API calls. A set of lightweight messages can be used to retrieve lists of events from the server.

The class `RBEvent` provides information about an event initiated from an `RBActivityTarget`. An event is generated when the status of a target changes and is kept in the SQL database until another event overrides it.

There are two methods to retrieve events from a given OS deployment server:

1. `RBEvent[] RBServer.getEvents(Date since)` performs a single call to the OS deployment server and returns all the events that have occurred on this server, regardless of the task. Only events more recent than *since* are returned. This parameter is given in the local time of the Java client, the API does the necessary conversions to the time of the server database.

2. `RBEvent[] RBActivity.getEventsFromServer(Date since)` performs a single call to the OS deployment server and returns all the events that have occurred on all the targets of this task. Only events more recent than *since* are returned.

The class `RBEvent` has the following methods to query event attributes:

- `int getProgress()`
- `String getStatus()`
- `String getLastProgressMsg()`
- `Date getUpdateTime()`

These methods fetch information locally.

The class `RBEvent` has methods to get fresh instances of related objects:

- `RBActity getRBActivity()`
- `RBActivityTarget getRBActivityTarget()`
- `RBHost getRBHost()`

These three methods perform additional server calls to retrieve current values of the objects.

It also has the following methods to retrieve related instances within the Java Virtual Machine (JVM):

- `boolean belongsTo(RBActivity)`
- `boolean belongsTo(RBActivityTarget)`
- `boolean belongsTo(RBHost)`

The purpose of these methods is to pass existing Java objects and to compare internal IDs in order to test for identity.

Controlling API traces

The logger reads `$JAVA_HOME/lib/log.properties` for the log settings. The user can use the system property, `java.util.logging.config.file` to specify the specific location of the log.properties file.

An example setting is `-Djava.util.logging.config.file=c:/Program Files/IBM/common/logging/log.properties`.

This is a sample of a log.properties that writes to both the console and to a file:

```
# Properties file which configures the operation of the JDK
# logging facility.

# The system will look for this config file, first using
# a System property specified at startup:
#
# >java -Djava.util.logging.config.file=myLoggingConfigFilePath
#
# If this property is not specified, then the config file is
# retrieved from its default location at:
#
# JDK_HOME/jre/lib/logging.properties

# Global logging properties.
# ------------------------------------------
# The set of handlers to be loaded upon startup.
# Comma-separated list of class names.
handlers=java.util.logging.FileHandler, java.util.logging.ConsoleHandler
```

```
# Default global logging level.
# Loggers and Handlers because override this level
.level=ALL

# Loggers
# ------------------------------------------
# Loggers are usually attached to packages.
# Here, the level for each package is specified.
# The global level is used by default, so levels
# specified here simply act as an override.

com.ibm.rbapi.level=ALL

# Handlers
# ------------------------------------------

# --- ConsoleHandler ---
# Override of global logging level
java.util.logging.ConsoleHandler.level=FINEST
java.util.logging.ConsoleHandler.formatter=java.util.logging.SimpleFormatter

# --- FileHandler ---
# Override of global logging level
java.util.logging.FileHandler.level=FINEST
java.util.logging.FileHandler.formatter=java.util.logging.SimpleFormatter

# Naming style for the output file:
# (The output file is placed in the directory
# defined by the "user.home" System property.)
java.util.logging.FileHandler.pattern=%h/java%u.log

# Limiting size of output file in bytes:
java.util.logging.FileHandler.limit=50000

# Number of output files to cycle through, by appending an
# integer to the base file name:
java.util.logging.FileHandler.count=1
```

The log level for the rbapi can be set in the log.properties file with the following:

```
com.ibm.rbapi.level=(SEVERE | WARNING | CONFIG | INFO | FINE | FINEST | ALL)
```

The first constructor for RBServer uses the settings defined in log.properties:

```
public RBServer(InetAddress ip, int port, String token)
```

This other constructor overrides the log level defined in log.properties:

```
public RBServer(InetAddress ip, int port, String token, Level verbosity)
```

There are three ways to modify these settings:

1. Restart the application.
2. Instantiate a new `RBServer`.
3. Modify log.properties and then make a call to `LogManager.readConfiguration();`.

Chapter 7. Command-line interface

You can run OS deployment server command-line commands, as well as commands to interface with the OS deployment server.

NetClnt commands are deprecated.

NetClnt

NetClnt commands are deprecated.

NetClnt is a command-line utility, available for Windows, Linux and Solaris. You can use NetClnt to access the files stored on a provisioning server remotely, and to perform some maintenance tasks.

Here is a list of features included in NetClnt:

- Communications between NetClnt and the server are protected with a password and encrypted.
- You can use NetClnt in interactive mode (with an interface similar to a FTP client), or in batch mode.
- You can delete, rename, move, download, and upload files. You can also delete, rename, and move directories.
- You can extract shared files from the server repository, and upload them to the server repository. Archives including both regular files and related shared files can be created and reimported to the server.

Using NetClnt interactively

NetClnt commands are deprecated.

1. Run the file `netclnt` in your installation directory.

   You see the interactive prompt `NETFS>`.
2. Open a connection with a OS deployment server:
   a. Run the `open` command, followed with the IP address (or host name) of the server.
   b. Enter the same password as the one set in the `NetPassword` parameter of the remote OS deployment server.
3. Test the connection by issuing `adir` command. This command downloads and displays the content of the current directory on the remote OS deployment server.
   - If this command works (that is, you see some result), then your connection is working.
   - If you do not see any result, double-check the IP address and password you have entered with the `open` command.
4. You can now use any NetClnt command. For example:
   - Send a new file to the OS deployment server (`put`)
   - Change the current directory (`cd`)
   - Get a server file locally (`get`)

   When you want to send a file to the OS deployment server, the file must exist on your local computer, in the current directory (you can change the current

local directory with `lcd`). Downloaded files are stored in the current local directory. See "NetClnt command reference" on page 89 for more information about commands. At any time, you can use the `help` command to get the list of valid commands, and a command name without argument to get a short summary of its syntax.

Using NetClnt in batch mode

You can use the batch mode of NetClnt to create scripts operating on remote files without any user intervention (for example, file upgrade).NetClnt commands are deprecated.

To use NetClnt in batch mode:

1. Create a text file containing the sequence of commands you want to run with `NetClnt`.
2. Run `netclnt` *`filename`*, where *`filename`* is the name of the text file containing the sequence of commands. The first command must be the `open` command, with a valid IP address and a valid password (password can be passed as the third parameter to `open`).

   You can pass parameters to your `netclnt` script, and use them in the script with the # prefix. For example, you replace every occurrence of `#1` in your script with the first parameter following the script name in the command line used to run `netclnt`. To use this new feature, call `netclnt` with the following syntax:

   ```
   netclnt -f scriptname parameters
   ```

   where *`scriptname`* is the file name of the `netclnt` script to run, and parameters are the parameters passed to the `netclnt` script. You can use the new command `chkparams` to check that the number of parameters passed to the script correspond to what the script is expecting.

Example

Here is an example of batch file for use with NetClnt:

```
open 192.168.1.10 rembopwd
lock
mkdir /myfiles
cd /myfiles
put file1
put file2
put file3
exit
```

Here is an example with parameters:

```
netclnt -f script 192.168.1.10 rembopwd
```

```
chkparams 2 <server-ip> <rembo-password>
open #1 #2
dir
exit
```

Using NetClnt to manage the shared repository

NetClnt commands are deprecated.

Creating a duplicate repository

The commands `getshared` and `putshared` can be used to download and upload all shared files related to a given archive, to and from a local

shared repository. This can be used to duplicate a part of the shared repository locally, for backup purposes or to transfer it to another provisioning server.

Note: The local shared repository uses the deployment engine format of index files, which is slightly different from the server format (the server format is ready to handle a large number of simultaneous targets, while the local format is optimized for local access only). Therefore, you should not copy a shared repository built using `getshared` directly onto a server, but use `putshared` instead.

Exporting shared files

A more convenient way to manipulate disk archives along with the related shared files is to use `.rad` file exports. A `.rad` file can embed several archive headers and other files, and all related shared files. It is a convenient way to backup and transmit archives in a self-contained format. See the description of the commands `radget`, `radput` and `radlist` in the "NetClnt command reference."

Synchronizing shared repositories

When using several OS deployment servers, if archive files are copied manually or mechanically from one server to the other, you might need to replicate the shared repositories to ensure that the destination server has all necessary server files. The `sync`, `msync` and `rsync` commands can do that on a specified subset of server files. These commands ask the targeted server to directly contact the parent server and download any missing shared file. See the documentation for these commands in the reference section "NetClnt command reference." Combined with the command `xcopy`, `rsync` is the best way to replicate the content of a parent provisioning server on a child provisioning server.

NetClnt command reference

NetClnt commands are deprecated.

Here is the complete reference for NetClnt commands:

open, connect

`open` *`host`* [*`password`*]
`connect` *`host`* [*`password`*]

Sets up the connection information required for remote access to the OS deployment server. This is the first command to issue when starting NetClnt. If you do not specify the password in the command line, you are asked to enter the password.

close

`close`

Closes the current connection. This command must be issued if you want to issue a new `connect` command to the same server, but with a different password.

pwd

`pwd`

Displays the current remote directory.

dir, ls

```
dir [path]
ls [path]
```

Displays the content of a directory on the remote server. If you use `dir` without parameters, the content of the current directory on the server is displayed (you can use `cd` to change the current remote directory). If you use a parameter, the content of the directory specified by the parameter is displayed (the path is relative to the current remote directory).

cd, chdir

```
cd path
chdir path
```

Changes the current remote directory. When NetClnt starts, the current remote directory is the root directory of the server file system. The path is relative to the current remote directory.

md, mkdir

```
md path
mkdir path
```

Creates a new directory on the server. The path is relative to the current remote directory.

rd, rmdir

```
rd path
rmdir path
```

Deletes a directory on the server. The directory must be empty. The path is relative to the current remote directory.

deltree

```
deltree path
```

Deletes a remote directory and all its sub-directories. The target directory does not have to be empty. Use this command with caution.

ren, move

```
ren old-name new-name
move old-name new-name
```

Renames or moves a file on the server. Both the old file name and the new file name must be specified. If the new file name is in a different directory from the old file name, the file is moved. Paths are relative to the current remote directory.

del, rm

```
del path
rm path
```

Removes (deletes) a file on the server. The path is relative to the current remote directory. If the `path` is a globbing pattern (that is, it expands file names using a pattern matching notation), all matching files are deleted. Acceptable globbing wildcards are the asterisk and the question mark.

lcd

```
lcd path
```

Changes the current directory on the local target. This command has no effect on the server: it only modifies the current directory on the target which is running NetClnt.

!*localcommand*

```
!localcommand
```

Executes a command on the local target . For example, `!dir` displays the content of the current directory on the local target . `!edit myfile` starts the MS-DOS editor on the file `myfile`.

get, mget, rget

```
get remote-path [local-path]
mget pattern
rget pattern
```

Downloads the remote file `remote-path` from the current remote directory into the file `local-path` in the current local directory. If the second parameter is omitted, the file is stored under the name `remote-path`. The `mget` variant downloads all remote files that match the globbing `pattern` to the local directory. Acceptable globbing wildcards are the asterisk and the question mark. The `rget` variant recursively searches all directories matching the globbing `pattern` (all files within the directory entered are copied).

put, mput, rput

```
get local-path [remote-path]
mput pattern
rput pattern
```

Uploads the local file `local-path` from the local remote directory into the file `remote-path` in the current remote directory. If the second parameter is omitted, the file is stored under the name `local-path` on the server. The `mput` variant uploads all local files that match the globbing `pattern` to the remote directory. The `rput` variant recursively searches all directories matching the globbing `pattern` (all files within the directory entered will be copied).

xcopy

```
xcopy [-d remote-server remote-source-dir local-dest-dir]
```

Copies files recursively, from a remote server to a local server. All the files stored in the directory `remote-source-dir` on the remote server `remote-server` are copied in the directory `local-dest-dir` on the local server. Tests are made on the source and the destination to skip files that have already been copied. This function is therefore used to replicate two servers daily, because it only copies files that have been modified, or added since the last xcopy operation.

By default, `xcopy` does not delete any file on the local server. But if `-d` is specified as the first parameter, `xcopy` then deletes every local file that has not been found on the remote server, creating an exact copy of the remote server directory in the local directory. The server password must be the same on both servers for this command to succeed, because `xcopy` uses the password provided to the `connect` when connecting to the remote server.

sync master, msync master, rsync master

```
sync master [remote-path]
msync master pattern
rsync master pattern
```

Orders the connected server to download all missing shared files for the archive specified by `remote-path` from another OS deployment server specified in `parent`. The server is specified by its IP address or by its DNS name. The `msync` variant triggers the transfer of shared files for all archives matching the globbing `pattern` in the remote directory. Acceptable globbing wildcards are the asterisk and the question mark. The **rsync** variant recursively enters into all directories matching the globbing `pattern` (all files within the directory entered are processed). A typical use for replicating all shared files from a given server is `rsync master *`. Note that the `rsync` command involves no file transfer to and from NetClnt, but only between the two servers.

getshared

`getshared` *`localarchive`* [*`localrepository`*]

Downloads all available shared files related to a given local archive to a local repository. By default, the repository is created in the target current working directory, but you can override the path by specifying a `localrepository` parameter. The `localarchive` can have been previously downloaded from the server, or copied by another mean.

putshared

`putshared` *`remotearchive`* [*`localrepository`*]

Uploads all locally available shared files related to a given server archive to the server shared repository. By default, files are read from a repository in the target working directory, but you can override the path by specifying a `localrepository` parameter. The `remotearchive` can have been previously uploaded to the server, or copied by another means. Both `getshared` and `putshared` optimize network traffic, in the sense that they only transfer files needed by the specified archive and already in the destination shared repository.

radget

`radget` *`radfile`* [*`file*`*]
`radput` *`radfile`* [*`file*`*]
`radlist` *`radfile`*
`radcheck` *`radfile`*

The `radget` command downloads all files specified on the command line (several filenames and globbing patterns are accepted), and all related shared files. The result is stored in a monolithic `.rad` export file, specified in `radfile`. You might need a significant amount of disk space on the deployment engine to perform this operation. The destination file is always overwritten (files are not added to an existing radfile). You can use `radlist` to list the content of a `radfile` export, `radcheck` to verify the internal consistency of a `radfile` export, and `radput` to reupload the files to the same or another server. If you have not specified a file pattern to the `radput` command, then all files in `radfile` are uploaded (including all shared files which are not yet known to the destination server). If you provide a list of files or patterns, only the matching files are uploaded. The `radput` works very well even if the source `radfile` is on a network share, because it reads only the shared files that are not yet present on the destination shared repository.

wake

`wake` *`hardware-address`*

Sends a wake packet to the specified hardware address. You can use this command to wake (remote power-on) targets. The target must be located on the same network as the computer running `netclnt`. Additionally, the target must be equipped with a Wake-On-LAN capable network card. The hardware address must be entered without any delimiters. For example, hardware address `00:10:4B:12:34:56` must be entered as `00104B123456`.

sleep

```
sleep sleeptime
```

Waits `sleeptime` seconds. This command is useful in batch scripts. No command is executed during the waiting time.

lock

```
lock
```

Acquires a unique lock on the computer, which is automatically released when NetClnt exits. This mechanism can be used to ensure that there is only one instance of NetClnt running at a time on a target , to avoid conflicts on the server, when automated batch files are run.

getlock, releaselock

```
getlock lockname
releaselock lockname
```

Acquires or releases a server lock. A server lock is identified by a name (`lockname`) and can be locked by only one computer at a time. These two commands can be used to make sure that no computer is accessing a particular resource during an update. See the `GetLock` function.

start, stop, reload

```
start servername
stop servername
reload servername
```

Controls the Windows service on a remote computer (implemented in Windows `netclnt` executable file only). These commands start, stop or reload the OS deployment server on a remote computer, by using standard Windows remote service control. The user that is logged in when running `netclnt` must have administrator equivalence on the remote server.

timestamp

```
timestamp message
```

Displays a timestamp and a message to the output. This command can be used in batch scripts to output the current time on the screen, or in a log file.

chkparams

```
chkparams reqparamcount [usage]
```

Checks the number of parameters and displays a usage string if parameters are missing. Use this command only in scripts called with the `netclnt -f scriptname params` syntax. `Chkparam` verifies that there is at least `reqparamcount` parameters in the command line, and displays a usage string containing the message in the `usage` parameter if the number of actual parameters is less than the requested parameters count.

sqlinit

```
sqlinit dbgw-ip-addr odbcsource [username password]
sqlclose
sqlexec sql-query
sqlcopy source-dbgw-ip source-odbcsource table-name
        [source-username source-password]
```

This group of functions connects `netclnt` with a database gateway (dbgw) to perform SQL queries remotely. The remote target must be running both dbgw and a OS deployment server. Use `SqlInit` to initiate the connection with the remote database gateway. You must provide the name of the ODBC source to use on the remote target , and the ODBC user name and password if needed. Then, use `SqlExec` to run SQL statements on the remote target , or `SqlCopy` to copy the content of a table from a given database gateway, to the remote target (or local host if `sqlinit` was called to connect to `localhost`). When calling `SqlCopy`, the parameters `source-dbgw-ip`, `source-dobcsource`, `source-username` and `source-password` identify the ODBC source to use as the source of the copy operation, and `table-name` is the name of the database table to copy. Calling `SqlCopy` on a database table is equivalent to the running of a `select` statement on the source, and `delete + insert` statements on the destination.

RbAgent

RbAgent is running under the operating system and is often referred to as the web interface extension in this documentation.

The web interface extension, running either on the OS deployment server or on a deployed computer, can access local resources under the operating system that are unavailable through the web interface. Its built-in operations are oriented to make use of these local resources.

RbAgent command reference

rbagent [-o |-s *srvip:password*] [-p *srvport*] [-f *iface*] [-v *verbosity*] [-d] [-q] [-l *logfile*] [-t *tracefile*] [-arguments]

- `-d`: prints debug info to the standard output, do not run as a daemon (do not detach).
- `-f`: specifies the interface. `iface` is the IP address of the preferred interface/subnet to use.
- `-l`: specifies the alternate log file, instead of rbagent.log/log2.
- `-o`: runs in offline mode (no connection to the OS deployment server)
- `-p`: specifies the port. *srvport* is the NBP port of the server.
- `-q`: quiet (does not display the banner).
- `-s`: specifies the OS deployment server address. *srvip:password* is a server IP address and password, that can be plain text or MD5.
- `-t`: specifies the alternate trace file, instead of rbagent.trc/trc2.
- `-v`: sets the logging verbosity level (from 1 to 6, default is 2)
- `-arguments`: are optional built-in supported operations described in "Built-in RbAgent operations" on page 95.

Built-in RbAgent operations

The following built-in agent operations are used as arguments of `rbagent`. Each operation identifier is followed by a short description and a usage example.

- `hostinfo`

  Displays general information about the computer.

  ```
  C:\rbo\bin\remboc\win32>rbagent hostinfo
  RbAgent 4.0 ($Revision: #12 $) - (c) Copyright 2004-2005 by IBM, Switzerland
  Connect 192.168.1.36 -> 192.168.1.36
  Starting Web interface extension
  model       : 2373FWG ThinkPad T42
  platform    : IAx86
  serial      : 99K5VHM
  uuid        : 261D6A81466E11CBABB7D0EB0A4468DB
  hwaddr      : 000D60D0602F
  ipaddr      : 192.168.1.36
  netmask     : 255.255.252.0
  gateway     : 192.168.1.254
  dhcpserver  : 192.168.1.5
  remboserver : 192.168.1.36
  Stopping Web interface extension

  C:\rbo\bin\remboc\win32>
  ```

- `rad-registerhost <IP|MAC|SN|UUID>= ... [HostName=...] [...]`

  Registers a new target identified by either its IP address, MAC address, serial number or UUID. You can assign values to the target database records of the new target , such as a target name. The database records include the BOM table and the User profile. View the information about keyword substitution in the user's guide for more information.

  ```
  rbagent rad-registerhost IP=192.189.34.32 HostName=Host32@location
  ```

- `rad-unregisterhost <IP|MAC|SN|HostName|Description>= ...`

  Removes from the OS deployment server a target identified by either its IP address, MAC address, serial number, target name, or description.

  ```
  rbagent rad-unregisterhost IP=192.189.34.32
  ```

- `rad-deployhost <IP|MAC|SN|HostName|Description>= ... [Scheme=...|SchemeName=...] [Config=...|ConfigName=...] [bind] [reboot|pxeboot] [wakeup]`

  Deploys the target specified by either its IP address, MAC address, serial number, target name, or description. You can specify a *scheme* and *OS configuration*. Use option *bind* to bind the OS configuration to the target . Option *reboot* attempts to restart the target if it is currently undergoing deployment, on locked screen, or is running the web interface extension. Option *pxeboot* forces the computer to start on the network . Option *wakeup* tries to turn on target currently powered off using Wake-on-LAN technology.

  ```
  rbagent rad-deployhost IP=192.189.34.32 Scheme=Default pxeboot
  ```

- `rad-radput radpath.rad [force|ignore] [spdl=<kbytes/s>] [softstages]`

  Uploads a `.RAD` archive to the OS deployment server. The *force* option adds new objects, even if there are pre-existing objects (profiles, OS configuration, schemes) with the same description on the server. *ignore* does not add new objects when

there are pre-existing objects with the same description. *spdl* is the upload speed limit. If the option *softstages* is present, the software stages included on the RAD file override those of the server.

```
rbagent rad-radput c:\temp\backup.rad force
```

- `radcheck` *radpath.rad*

  Verifies the consistency of a `.RAD` archive. `radcheck` not only checks the file format but tries to read all files in the `.rad` file to verify that they are properly encoded.

  ```
  rbagent radcheck c:\temp\backup.rad
  ```

- `rad-chksoft` *sourcepath ["<attr>=value" ...]*

  simulates the creation of a new software module. *attr* can take the values *descr, content, pkgname, dest, cmdline, pass, flags, dosubst, norules, nopcirules, OSType, OSVersion, OSArch, Model*

  ```
  rbagent rad-chksoft "c:\drivers\ahci"
  ```

- `rad-mksoft` *sourcepath ["<attr>=value" ...]*

  Creates a new software module. *attr* can take the values *descr, content, pkgname, dest, cmdline, pass, flags, dosubst, norules, nopcirules, OSType, OSVersion, OSArch, Model* Before using this command, try *rad-chksoft* to view default values and optionally modify them.

  ```
  rbagent rad-mksoft "c:\drivers\ahci" "descr=DriversAHCI" "OSType=Windows 2003%"
  "OSVersion=%Service Pack 1%" "OSArch=x86-64" "Model=1951 Thinkpad T60"
  ```

- `rad-mkbootcd` *iso-image-path server-ip server-pwd [fixedip] [fixednetmask] [fixedgateway][allowipoverload][allowsrvipoverload]*

  Creates a bootable CD to start without network boot. *iso-image-path* is the path of the iso file. *server-ip* is the IP address of the OS deployment server on which the target must start. *server-pwd* is the OS deployment server password. Options *fixedip, fixednetmask* and *fixedgateway* are used for target properties. Options *allowipoverload* lets the user to modify the IP address of the target , and *allowsrvipoverload* lets the target contact a OS deployment server at another IP address than the one specified by *server-IP*.

  ```
  rbagent rad-mkbootcd myimage.iso 192.168.2.34 mypassword
  ```

  Note: This command requires "PXE activation" which is not available on all hardware. The CD created with this command might not work on specific hardware. See also Booting targets without using PXE.

- `fallback-mbr`

  Installs a special MBR on the hard disk to force PXE booting at the next boot. Use this once; the original MBR is restored afterward with its typical boot order.

  ```
  rbagent fallback-mbr
  ```

- `cmdlines` *file*

  Runs the list of commands specified in `file`, waiting for the end of each command before starting the next one. Error codes returned by the commands are indicated.

  ```
  rbagent cmdlines c:\cmdfile.txt
  ```

- `joindomain` *domain adminuser adminpwd [joinou]*

  joins a Windows domain`joindomain /w workgroup [adminuser adminpwd]` joins a Windows workgroup `joindomain /s domain trustpwd` changes the trust account.

```
rbagent joindomain mydomain myadmin mypassword
rbagent joindomain /w myworkgroup
rbagent joindomain /s mydomain mynewtrustpassword
```

- `checkdevices`

  Lists the devices (under Windows OS) which do not function correctly, with their full name.

  ```
  rbagent checkdevices
  ```

- `rad-osrestore <IP|MAC|SN|HostName|Description>= ... [Config=...|ConfigName=...] [bind][reboot|pxeboot][wakeup][switchon]`

  Performs the same as a manual operating system restoration from the web interface.

  ```
  rbagent rad-osrestore IP=192.168.1.19 configName="Windows XP(setup)" bind
  ```

- `rad-configlist [details]`

  Returns the list of available OS configurations on the server.

  Note: Using the `details` option will display information such as:

  - Operating system configuration name
  - Operating system configuration ID
  - Operating system configuration version
  - Operating system configuration timestamp
  - System profile name
  - System profile ID
  - System profile operating system type
  - System profile version
  - System profile timestamp

  ```
  rbagent rad-configlist [details]
  ```

- `rad-resetscope`

  Used when the server is renamed. All objects are marked in the database as belonging to the new name.

  ```
  rbagent rad-resetscope
  ```

- `rad-radinfo`

  Used to describe the logical content of a .RAD archive.

  ```
  rbagent rad-radinfo
  ```

- `rad-usbget`

  Used to put deployment data on a USB drive.

  ```
  rbagent rad-usbget
  ```

- `rad-mkdrivers`

  Used to create multiple driver modules.

  ```
  rbagent rad-mkdrivers
  ```

- `rad-mkwinsetup`

  Used to create a Windows setup image.

  ```
  rbagent rad-mkwinsetup
  ```

- `rad-mkvistaclone`

  Used to create a Windows WIM image.

```
rbagent rad-mkvistaclone
```

- `rad-mklinuxsetup`

  Used to create a Linux setup image.

  ```
  rbagent rad-mklinuxsetup
  ```

- `rad-mksolarisboot`

  Used to create a Solaris boot image.

  ```
  rbagent rad-mksolarisboot
  ```

- `rad-uploadlogs`

  Used to send local log files to the OS deployment server.

  ```
  rbagent rad-uploadlogs
  ```

- `rad-replicate`

  Used to replicate all database objects from another OS deployment server.

  ```
  rbagent rad-replicate
  ```

- `rad-runtask`

  Used to execute any pending task.

  ```
  rbagent rad-runtask
  ```

- `rad-exportlogs` *`destination=... [mode= 1|2|3|4 [compress]] [activityid=...]`*

  Used to export debugging information. `destination` is the folder where the logs are copied. If `mode=1`, all the logs are placed in `destination`. If `mode=2`, all the logs are placed in `destination`, with only one file for all the tasks. If `mode=3`, all the logs are placed in `destination`, with one file for each task. If `mode=4`, all the logs are placed in `destination` in raw format. `compress` can be used only if `mode=4` and compresses the resulting log file. `activityid` provides the logs only for the specified activity.

  ```
  rbagent rad-exportlogs destination=c:\temp mode=1
  ```

RbAgent and access to remote files on Windows

If you want RbAgent to have access to remote files, for instance to create a software module, you must pay attention to whether RbAgent is running as a service or within a logon session:

- If RbAgent is running within a logon session (user application), the network shares can be accessed through mapped drive letters.
- If RbAgent is running as a service, mapped drive letters are not accessible because drive mappings are created for a specific logon session. However, RbAgent is able to access files on the network share using a UNC path (for example `\\192.168.168.17\c$\install\`).

RbAgent must naturally also have read privileges on the files. These privileges depend on the workgroup/domain account used to run RbAgent.

rad-mountimage command reference

Using this command, you can mount an image on a target to perform offline patching.

`rad-mountimage` [**`readwrite`**] (**`uuid`**|**`descr`**|**`imageID`**) **`cmdline`** [**`mount`**]

Parameter description

readwrite
Needed to update the image, without it the image is read-only.

uuid=<UUID>
`uuid=62fa5eac-3df4-448d-a576-916dd5b432f2`

Provides the universally unique identifier of the image.

descr=<descString>
`descr="SLES 10 32-bit"`

Provides the description of the image you are mounting, according to the **Description** field in the web interface

imageID=<number>
`imageID=003`

Provides the image number.

cmdline=<cmdname>
`cmdline="xterm"`

Provides the command to be performed on the image.

mount=<path>
`mount="/tmp"`

Specify the root path to mount the partitions of the image

This parameter is optional

If a value is provided for the `mount` parameter

- the path must start with a slash, for example `/example`
- if the folder does not exist, it is created
- each partition is mounted as `part1`, `part2`, to `partN` under the root path, where `N` is the number of partitions

If the `mount` parameter is not used

- the default `/tmp/dev` path is used
- each partition in is mounted as `loop1`, `loop2`, to `loopN` under `/tmp/dev`, where `N` is the number of partitions

In all cases, the swap partition is not mounted. At the end of the off-line patching process, the root folder provided by the **`mount`** parameter is kept, but each partition subfolder is deleted.

Example

```
rbagent -v 4 -s 192.168.2.19:abcd rad-mountimage readwrite
descr="SLED-10.2_smallest" cmdline="xterm" mount="/example"
```

rembo command reference

Use this command to perform maintenance operations on the server shared repository.

Syntax

```
rembo [-d][-v loglevel][options][maintenance_options]
```

Parameters

-d
Specifies to run as a console application instead of as a service.

-v Sets the verbosity level. The default value is: 3.

Verbosity levels:

- 0: no output
- 1: log errors
- 2: log errors and warning
- 3: log errors, warnings, and informational messages
- 4: log errors, warnings, informational messages, and notice messages
- 5: log errors, warnings, informational messages, notice messages, and debug messages
- 6: log everything, including network packets

[options]

-c Specifies the configuration file name. The default file name is: `rembo.conf`.

-cert Specifies the file name of a certificate to import.

-convert
Converts the server to the new native file system.

-delcerts
Deletes certificates that were either created automatically or by using the **-cert** option.

-exit Stops the server after processing **-cert** / **-c**

-force Forces conversion even if a conflict is found.

-readonly
Runs the server in read-only mode (child).

-rehash
Reattribute hashed inode IDs to every file.

-sdb Specifies the server database file name. The default file name is: `server.db`.

[maintenance_options]
The following options determine the maintenance operation to be performed on the shared file repository.

-chkshared
Initiates a verification process on the server shared repository. Errors are reported, but no corrective actions are applied. This operation can take from several minutes to several hours, depending on the size and fragmentation of the shared repository. During this time, a connection to the server cannot be established.

-fixshared
Initiates a repair process on the server shared repository. Errors are corrected by recomputing missing indexes if possible, or deleting

corrupted records. The repair process is automatically triggered when the server is killed with unflushed changes to the shared repository. This operation can take from several minutes to several hours, depending on the size and fragmentation of the shared repository. During this time, a connection to the server cannot be established.

-statshared
Initiates a complete analysis on the server shared repository, and displays a summary of shared files usage. This operation can take from several minutes to several hours, depending on the size and fragmentation of the shared repository. During this time, a connection to the server cannot be established.

-sweepshared
Initiates a mark and sweep operation on the server shared repository. Shared files not used in any archives on the server (as reported by the -statshared option) will be cleaned out. This can be used to recover disk space on the server after deleting a significant number of disk images. This operation can take from several minutes to several hours, depending on the size and fragmentation of the shared repository. During this time, a connection to the server cannot be established.

-packshared
Initiates a mark and sweep operation on the server repository, then shrinks all shared repository files to their minimum size to recover any preallocated storage. Note that the use of this function may lead to fragmentation of the shared file repository, and its use is therefore not recommended unless disk space must be recovered for other applications. This operation can take from several minutes to several hours, depending on the size and fragmentation of the shared repository. During this time, a connection to the server cannot be established.

Example

To mark and sweep a set of files on the shared repository to decrease the size of the repository, run the command:

```
rembo -d -v 4 -packshared
```

dbgw command reference

Use this command to install and uninstall the dbgw service. You can also disable remote connections using this command. This command is available only for Windows operating systems. For Linux operating systems a `dbgw.jar` file is provided.

Syntax

```
dbgw [-d] [-v verbosity] [-f logfile] [-t idle-sec] [-l]

dbgw -i

dbgw -u
```

Parameters

-d Specifies to run as a console application instead of as a service.

-v verbosity
Sets the verbosity level. The default value is: 3.

Verbosity levels:

- 0: no output
- 1: log errors
- 2: log errors and warning
- 3: log errors, warnings, and informational messages
- 4: log errors, warnings, informational messages, and notice messages
- 5: log errors, warnings, informational messages, notice messages, and debug messages
- 6: log everything, including network packets.

-f logfile
Writes the output of the gateway to a file. By default, it is sent to standard output.

-t idle-sec
Specifies the timeout before the connection is automatically closed. The default value is 120.

-l Disables remote connections.

-i Installs the dbgw service.

-u Uninstalls the dbgw service.

Example

To run the gateway as a command from the command line and log all the SQL queries, submit the following command:

```
dbgw -d -v 4
```

Chapter 8. Glossary

A

administrative group
A group of related computers. An administrator can create administrative groups to organize target systems into meaningful categories, and to facilitate deployment of software to multiple targets.

B

bare metal computer
A computer on which there is nothing reliable but the hardware. It can be coming straight from factory without any data on its hard disk (out of the box) or it can contain a possibly damaged operating system.

Basic Input/Output System (BIOS)
The code that controls basic hardware operations, such as interactions with diskette drives, hard disk drives, and the keyboard.

BIOS See Basic Input/Output System.

blacklist
In Tivoli Provisioning Manager for OS Deployment, a list of PCI devices or of computer models which are known to raise issues, accompanied by hardware settings which must be used to work around the issues.

C

child An OS deployment server that is a subordinate of another OS deployment server in a replication tree structure. Only the top-level parent OS deployment server is not a child. See also parent.

clone To prepare a reference computer and create a system profile ready for deployment.

D

database server
The computer on which the database application and database are installed.

Deployment
A process which installs an operating system, and possibly other applications and files, on a target computer. During a deployment, data previously stored on the hard drives of the target is deleted.

Deployment scheme
A specific type of task template. A deployment scheme contains parameters for customizing a deployment on a target, and the target display screen layout. See also task template.

DHCP See Dynamic Host Configuration Protocol.

Dynamic Host Configuration Protocol (DHCP)
A communications protocol that is used to centrally manage configuration information. For example, DHCP automatically assigns IP addresses to computers in a network.

F

free-text condition
In Tivoli Provisioning Manager for OS Deployment, a condition written in Rembo-C; syntax, using variables and Java-like logical operators, and which evaluates to true or false.

H

hardware configuration
A set of parameters used to configure hardware before an operating system installation. It includes RAID settings, BIOS update information, BIOS settings, and custom hardware configuration parameters.

M

MCAST
A proprietary transfer protocol of Tivoli Provisioning Manager for OS Deployment computers using multicast. Contrast with unicast and PCAST.

MTFTP
See Multicast Trivial File Transfer Protocol.

multicast
Bandwidth-conserving technology that reduces traffic by simultaneously delivering a single stream of information to many computers.

Multicast Trivial File Transfer Protocol (MTFTP)
Multicast TFTP.

N

network boot
The process of starting up a computer directly over the network rather than on a disk.

O

OS configuration
The operating system parameters of a system profile .

OS deployment server
The computer on which the Tivoli Provisioning Manager for OS Deployment application and files are installed.

P

parent An OS deployment server in a replication tree structure that has at least one dependent OS deployment server. See also child.

PCAST
A proprietary transfer protocol of Tivoli Provisioning Manager for OS Deployment that delivers non-identical sets of files to several target computers using multicast. Contrast with MCAST and unicast.

PCI See Peripheral Component Interconnect.

Peripheral Component Interconnect
A local bus that provides a high-speed data path between the processor and attached devices.

Preboot Execution Environment (PXE)
PXE is an industry standard target/server interface that allows networked

computers that are not yet loaded with an operating system to be configured and booted remotely. PXE is based on Dynamic Host Configuration Protocol (DHCP). Using the PXE protocol, targets can request configuration parameter values and startable images from the server. The PXE process consists of the system initiating the protocol by broadcasting a DHCPREQUEST containing an extension that identifies the request as coming from a target that uses PXE. The server sends the target a list of OS deployment servers that contain the operating systems available. The target then selects and discovers an OS deployment server and receives the name of the executable file on the chosen OS deployment server. The target downloads the file using Trivial File Transfer Protocol (TFTP) and runs it, which loads the operating system.

PXE See Preboot Execution Environment.

R

RAD file
A file containing deployment objects such as task templates, system profiles, and software modules used to archive data or to transfer data between two OS deployment servers. A RAD file has a `.rad` extension.

RAID See Redundant Array of Independent Disks.

redeployment
The process of synchronizing a hard-disk content to its reference image stored on a hidden and protected redeployment partition.

redeployment preload
The process of creating a reference image of a computer at the end of a deployment, and saving this reference image into a protected redeployment partition (invisible to the user and to the operating system itself).

Redundant Array of Independent Disks (RAID)
RAID is a way of storing the same data in different places (thus, redundantly) on multiple hard disks. By placing data on multiple disks, I/O operations can overlap in a balanced way, improving performance. Multiple disks increase the mean time between failure (MTBF) and storing data redundantly increases fault-tolerance.

Rembo-C;
A programming language, descendant of the C language combined with traces of JavaScript and Java.

replicated server
An OS deployment server which shares data with one or several other OS deployment servers. The servers are hierarchically structured with a parent and child servers. A child child can act as parent to replicated servers further down in the hierarchy.

replication
The process of copying files from a parent server to a child server. A selection can be performed on the kind of information that must be replicated. Files that have been modified are copied over.

S

shared repository
In Tivoli Provisioning Manager for OS Deployment, a repository of server

objects where each file is stored only once, even if it belongs to several objects. The shared repository reduces the storage space necessary to hold all server objects.

software module
A group of files, and potentially command lines, packaged together under one name. A software module can be installed on a target during a deployment.

software snapshot
A differential image of software installed on top of a running operating system. Software snapshot creation is deprecated. Any previously created software snapshots can be deployed for compatibility with earlier versions.

system profile
The partition layout and list of files for deployment of an operating system, either by unattended setup or by cloning. A system profile can have several configurations.

system snapshot
For Windows only. The partition layout and list of files for deployment of an operating system, created by cloning without using Sysrep. A system snapshot cannot be parametrized and can only be restored, not deployed.

T

target A computer that is known to an OS deployment server.

target list
A comma-separated-value list of targets used for adding large numbers of targets to the OS deployment server without having to start the targets up individually on the network.

task A set of actions designed to achieve a particular result. A task is performed on a set of targets on a specific schedule.

task template
A group of elements which can be customized on a target computer. These elements are mostly screen layouts which condition the appearance of the target computer screen during the different phases of its control by Tivoli Provisioning Manager for OS Deployment. See also Deployment scheme.

TCP tunnel
A way to provide TCP connectivity to target computers.

TFTP See Trivial File Transfer Protocol.

Trivial File Transfer Protocol (TFTP)
In Internet communications, a set of conventions that transfers files between targets using minimal protocol.

U

unattended setup
Operating system installation on a target, using original installation files and parameters contained in a script defined on the OS deployment server. Contrast with clone.

unicast
Transmission of data to a single destination. In Tivoli Provisioning Manager for OS Deployment, a transfer protocol that delivers a stream of files to a single target. Based on TCP, this protocol is faster when there are

only a few target computers on the receiving end of the transfer. This protocol can also be used in networks where multicast traffic is not properly handled. Contrast with MCAST and PCAST.

universal image
A cloned system profile that has been prepared with all drivers for disk types and hardware abstraction layer variants encountered in the pool of targets to be deployed.

W

Wake on LAN
A technology that enables a user to remotely turn on systems for off-hours maintenance. A result of the Intel-IBM Advanced Manageability Alliance and part of the Wired for Management Baseline Specification, users of this technology can remotely turn on a server and control it across the network, thus saving time on automated software installations, upgrades, disk backups, and virus scans.

Web interface
A user interface for one or more administrative tasks.

Web interface extension
An agent that allows the web interface to have access to the content of the target on which it is running. For example, to browse disks and read and write files.

Z

zone An IP range or domain that is used to logically group computers into regions. You can define one or more zones for each region.

Chapter 9. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

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

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Notice for Windows Automated Installation Kit (AIK)

Windows Automated Installation Kit (AIK) for Windows 7 in English is distributed by Microsoft and is available on the Microsoft website from the following link at the time of publication: http://www.microsoft.com/downloads/details.aspx?familyid=696DD665-9F76-4177-A811-39C26D3B3B34&displaylang=en.

The Windows AIK is licensed to you by the code's owner and not by IBM it is your responsibility to determine whether the license terms offered by the code's owner are acceptable to you.

YOUR USE OF THE WAIK AND ANY URL'S OR MATERIALS ON THIRD PARTY WEBSITES ("THIRD PARTY MATERIALS") IS "AS IS", WITHOUT WARRANTY FROM IBM OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. TO THE EXTENT PERMITTED BY LAW, IBM DISCLAIMS ALL LIABILITY FOR ANY CLAIMS ARISING OUT OF USE OF THE THIRD PARTY MATERIALS.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml

Adobe is either a registered trademark or trademark of Adobe Systems Incorporated in the United States, other countries, or both.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows , and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States, other countries, or both.

Other company, product and service names may be trademarks or service marks of others.

Copyrights

IBM web site pages may contain other proprietary notices and copyright information which should be observed.

Portions of third-party software included in this IBM product is used with permission and is covered under the following copyright attribution statements:

- Copyright (c) 1998-2005, The OpenSSL Project. All rights reserved.
- Copyright (c) 1995-2005 Jean-loup Gailly and Mark Adler, the ZLIB data compression library.
- Copyright 1994-2006, The FreeBSD Project. All rights reserved.

The MD5 Message-Digest Algorithm was developed by Ron Rivest. The public domain C language implementation used in this program was written by Colin Plumb in 1993. Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, without any conditions or restrictions. This software is provided "as is" without express or implied warranty.

Portions include cryptographic software written by Eric Young (<(eay@cryptosoft.com)>). This product may include software written by Tim Hudson (<(tjh@cryptosoft.com)>).

Printed in USA