Redbooks Paper

Alexander Koch
Erich Amrehn

Linux Software Management with Aduva OnStage

Introduction

This IBM® Redpaper describes Aduva OnStage, an application for software configuration management of Linux and Solaris systems. Aduva OnStage supports multiple Linux® distributions running on various hardware platforms. In this Redpaper, we focus on managing Linux servers running on IBM System z9™ and zSeries.

A Linux system is an assembly of many applications, utilities, and libraries running under the control of the Linux kernel. This software is available as source code from developers all around the world. While the loosely connected collection of projects works well as a process of developing the components of a Linux system, it introduces some fundamental problems.

Although you can get all the sources, compile them, and put your own Linux system together, this would be a time consuming process that requires specific knowledge. Keeping such a system up to date, installing bug fixes, and patching security issues is even worse without the proper tools.

But there is an easier way. Several distributions bring order to this chaos and provide selected software in form of packages and come with a way to install and configure a Linux system. They use package management systems to introduce a level of abstraction that hides the complexity of the underlying process.

As this may be sufficient for individual users or small environments, at least when the number of systems you manage increases, the problems inherited from the development process have just shifted:

- The constant stream of bug fixes and updates, definitely has advantages in terms of security, and makes new features available early. But you can also wind up in a sisyphean task, doing things over and over again.
- Many administrators hesitate to update their systems. To be prepared for problems, you have to keep track of every change you make, and even then it is difficult to revert to a previous state.

- Distributions have their own way of managing a software configuration and have a distinct package base. You are limited thinking on the level of packages instead in terms of the functionality your system should provide.
- Embedded in software packages is information like which other packages need to be installed (a dependency) or which packages must not be installed (a conflict). Sometimes these rules are not tested to their full extent and complex configurations can produce bugs that didn't show up in a developers test environment.
- Traditional ways of Linux software configuration management work on the basis of one system and often fail to provide a single point of administration for your whole infrastructure.

Aduva OnStage offers a level of abstraction to hide complexity and ease software configuration management of Linux systems. Aduva OnStage offers:

- Automated deployment of patches and updates with results recorded in system log files.
- A rollback mechanism to easily undo every change.
- Use of profiles to define software configurations. This allows you to think in terms of system functionality (extending across Linux distributions and hardware architectures).
- Distribution packages are certified and tested by the Aduva Certification Lab to ensure safe and successful deployment.
- Software can be installed or uninstalled on multiple hosts. Systems can be administered from either a graphical user interface or using a command line interface.

In this Redpaper, we discuss:

- Aduva OnStage architecture
- A deployment scenario
- Installation of Aduva OnStage
- Working with Aduva OnStage

Aduva OnStage architecture

Aduva OnStage is made up of several applications:

System Dependency Server (SDS)	The server application, it consists of:
Aduva Server	An apache based server that communicates with the Universal Server at Aduva.
Dependency Manager	Prepares and initiates the jobs to run on hosts.
Knowledge Base	A cache that holds the deployment rules and software packages from the Universal Server.
Agent	The client that runs on managed servers.
Director Console	The graphical user interface.
Command Line Interface (CLI)	The command line interface.
Proxy Server	Additional SDS for load balancing.
Universal Server	Master server for software packages and deployment rules. Located at knowledge.aduva.com.

The Director Console and the Command Line Interface connect to the SDS and allow you to manage your hosts. A host in Aduva OnStage terminology is a managed server with the Agent installed. Tasks like changing the software configuration or performing tests are done

by running jobs. Jobs use deployment rules and packages fetched from the Universal Server via the SDS.

Network connections

Aduva OnStage relies on TCP/IP for network connectivity. This makes it is possible to use it in your whole infrastructure, even with different subnetworks or in a DMZ. For network connectivity in an Aduva OnStage setup:

- The SDS must be able to contact the Universal Server.
- The clients (Agents, Director Console CLI, and Python API) require a bidirectional connection to the SDS.

Connection to the Universal Server

The Universal Server is located at knowledge.aduva.com. It holds the master repository of certified software packages and deployment rules, known as the Knowledge Base. The System Dependency Server contacts it regularly to update its channel information. When a job needs software packages, they are pulled from the Universal Server along with the necessary rules to deploy them.

All connections are initiated by the System Dependency Server and at no time information is pushed to the Universal Server.

In order to access the Universal Server, the SDS must be able to contact knowledge.aduva.com over HTTPS on port 443. A proxy can be used and is configured in the installation process or afterwards (“Configuration of the proxy settings” on page 38).

Connection between the SDS and its clients

During the installation process the default ports for services are assigned, or if not free, chosen from a default range. They can be changed later by editing the configuration files. See “Reconfiguration of server and ports” on page 38.

There are three connections that must be possible between the SDS and its clients. Consult Figure 1 on page 4 and the following list:

- Client to Aduva Server: Agents pull data from the Universal Server via the SDS. The Director Console and the CLI fetch information from the SDS. The SDS is listening on port 8002 as a default. The default range is 8000-8100.
- Client to Dependency Manager: The SDS is listening for Agent login and messages. The Director Console and the CLI initiate jobs through the SDS. The default port is 8100. The default range is 8100-8200.
- Dependency Manager to Agent: Agents listen for reconnect on port 8200, the default range is 8200-8300.

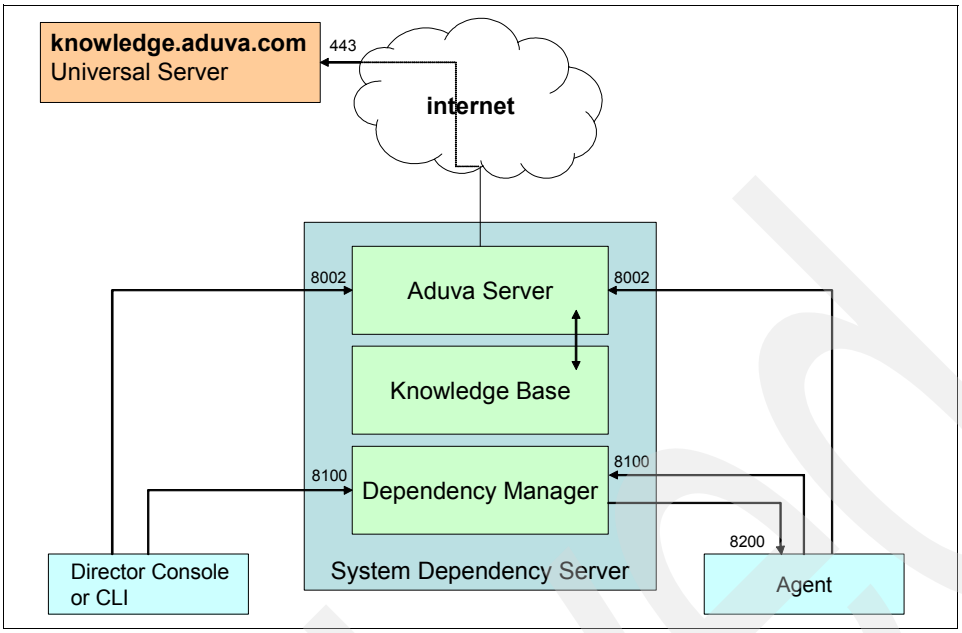

Figure 1 Network connections

The connection to the Universal Server and the connection between a client and the Aduva Server are secured with HTTPS, the other connections are secured with an RSA public key algorithm.

A deployment scenario

We use a basic scenario in some examples to show the features of Aduva OnStage, but they can be easily applied to your own needs. Look at Figure 2 to get an overview of the sample scenario.

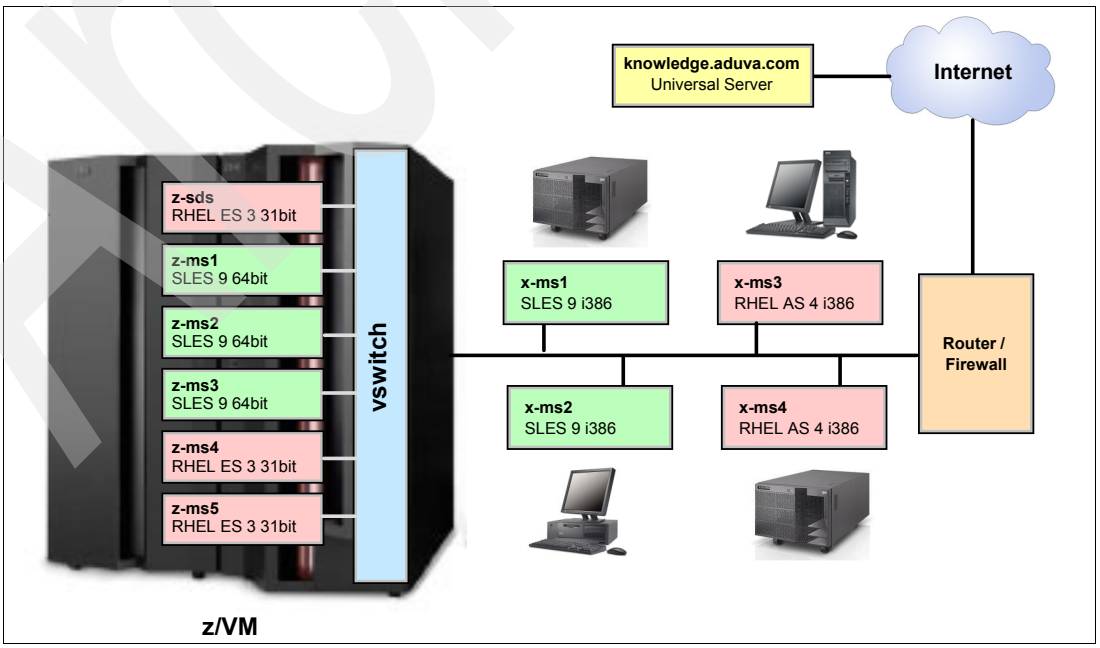

Figure 2 Sample scenario

While the System Dependency Server is installed on z-sds, we run the Agent on every host.

Installation of Aduva OnStage

Prerequisites

Before you install Aduva OnStage, a couple of requirements must be fulfilled.

First, refer to “Supported distributions” on page 5 to see which distributions can be managed with Aduva OnStage. The sections “Hardware prerequisites” on page 6 and “Software prerequisites” on page 6 tell you about existing system requirements. Finally, use the information from “Network connections” on page 3 and configure your network to allow the necessary connections between the Aduva OnStage components.

Supported distributions

Aduva OnStage manages whole Linux systems. Every software package of a distribution is tested and certified by Aduva to ensure a safe and successful deployment. A current overview of supported distributions can be found on the Aduva Web site:

http://www.aduva.com/index.php?page=distributions

Note: The Aduva OnStage term for a supported distribution is channel. A channel is a specific version of a distribution on a specific architecture. We will use the terms channel and distribution interchangeably in this Redpaper.

Consult the following list to see which distributions that Aduva OnStage can manage at the time of writing:

- Intel® i386™ architecture
  - Fedora Core 2
  - Novel Linux Desktop 9
  - Red Hat 7.2, 7.3, 8.0, 9.0
  - RH Enterprise Linux AS 2.1, 3.0, 4.0
  - RHEL ES 2.1, 3.0, 4.0
  - RHEL WS 2.1, 3.0, 4.0
  - SLES 8, 9
  - SuSE Professional 9.0, 9.1, 9.2
- Intel IA64 architecture
  - RHEL WS 3.0
- AMD 64 bit architecture
  - RHEL AS 3.0, 4.0
  - RHEL ES 4.0
  - RHEL WS 3.0, 4.0
  - SLES 8, 9
  - SuSE Professional 9.0, 9.1
- IBM 64 bit mainframe environment
  - RHEL AS 4.0 (announced)
  - SLES 8, 9
- IBM 31 bit mainframe environment
  - RHEL AS 3.0
  - SLES 7, 8, 9

- PowerPC® 64 bit architecture
  - RHEL AS 3.0
  - SLES 9
- Sun™ SPARC environment
  - Solaris™ 8, 10

Hardware prerequisites

Besides choosing the right distribution for an architecture there are other hardware requirements you must consider.

Architecture requirements

For IBM 31 bit and 64 bit mainframe, any zSeries or S/390 with support for Linux will do. For Intel i386 we recommend at least Intel Pentium® IV or equivalent. PowerPC 64 support is intended for IBM @server iSeries™, JS Blade, OpenPower™ and pSeries® servers.

While the Agent runs on every supported platform, other applications do not:

- The System Dependency Server and the Proxy Server run on Intel i386 or zSeries and S/390.
- The Director Console runs only on Intel i386 plattforms (Linux or Microsoft® Windows®).
- The Command Line Interface runs on Intel i386, S/390 or Solaris.

Memory and Storage requirements

Consult Table 1 for the recommended memory and minimum storage requirements.

Table 1 Required memory and storage

Component	Memory		Storage
	Intel	Mainframe	
System Dependency Server	512 MB	256 MB	500 MB + 512 MB in /usr/local during installation
Director Console	512 MB	256 MB	256 MB + 20 MB for every active channel
Agent	256 MB	128 MB	256 MB

Swap space should correspond to the double amount of RAM.

Software prerequisites

Usually a basic installation of a supported distribution, with a configured TCP/IP stack is sufficient to install Aduva OnStage applications. However, you should check the following requirements:

- During the installation of the System Dependency Server the **`zip`** utility is used to create a zip file for the Windows version of the Director Console. If you don't want to use it, you can ignore this requirement and the warning that occurs when the installer fails to invoke the command.
- The Aduva Server runs as the user nouser of the group nogroup. Make sure they exist before you start the installation of the SDS.
- The Director Console (Linux version) needs an X Windows environment to run.

Installation of the System Dependency Server

The installation of the System Dependency Server (SDS) is done with an installer called **`ezInstall`**. This program guides you through the configuration of Aduva OnStage, installs the SDS and creates installation files for the Agent and the other applications.

Please refer to the *Aduva OnStage Admin Guide* for an in depth description of the installation process.

Refer to “Prerequisites” on page 5 before you start with the installation. If you run into problems, see “Problems with the installation” on page 12 for help.

Starting the installation

Transfer the Aduva OnStage tarball and the license file to the machine the System Dependency Server is to be installed and place them in an arbitrary directory. The tarball is named this way:

```
OnStage-<version>-<release>.tgz
```

As root user go to the directory you created and unpack the tarball. Issue the following command:

```
tar xzvf OnStage-*.tgz
```

This places the contents of the tarball in a directory called `OnStage-<version>-<release>`. We will refer to this directory as the installation directory. Change into that directory and run the installer:

```
./ezInstall [-channels] [-s]
```

Refer to “Installation options” on page 7 to see what the parameters in square brackets mean.

Installation options

You can give parameters to the installer, to change some default behavior:

- You can install the SDS with the requirement to supply a password in order to start or stop it. This is called *secure mode*, and is enabled when you start the installer with the option “-s”.
- Channels are activated as soon as the first Agent with a new channel connects. It is possible to already activate channels in the installation process. The advantage of this is that you can use them to predefine hosts; see “Predefine hosts” on page 17. Start the installer with the option “-channels” to enable this.

Handling the installer

The handling of the installer is easy. Use the arrow keys or Tab to move around. Select options with the Spacebar. Confirm a screen with Enter, which then brings up the next screen.

On some screens you need to enter text. If you need to delete it, use Backspace and not Delete.

Be careful to press Enter only once. Sometimes it takes a moment until you see the next screen. If you press Enter at that time, the new screen is confirmed before you can make any changes!

The installation process

The installer takes you through the following 11 on page 11 steps:

1. License agreement

   The first screen, shown in Figure 3, displays the license agreement for Aduva OnStage and various third party products. Read the full license agreement and use the arrow keys to move around. When you are finished press Enter to accept it. Choose **Decline** to abort the installation. Use Tab to change between the two.

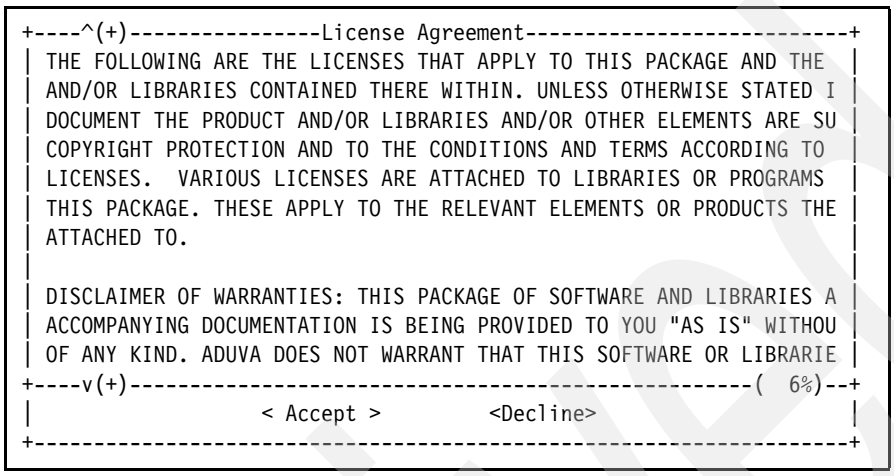

Figure 3 License agreement

2. Select the install option.

   After you have accepted the license, choose **Install OnStage Suite** and press Enter. See Figure 4.

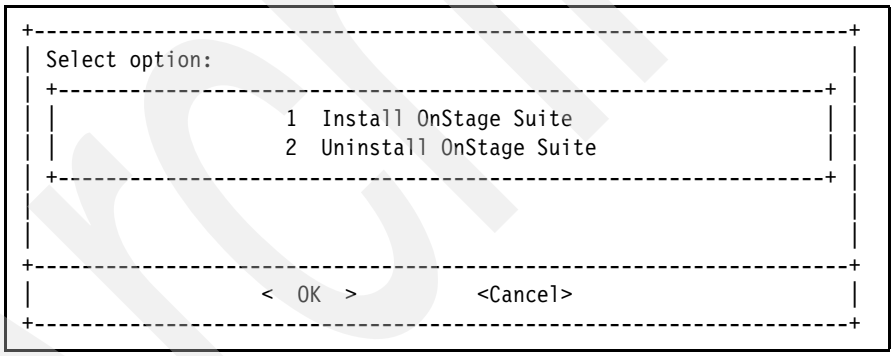

Figure 4 Select install option

3. Provide the path to the license file.

   On the next screen, shown in Figure 5 on page 9, provide the path to the license file and confirm with Enter.

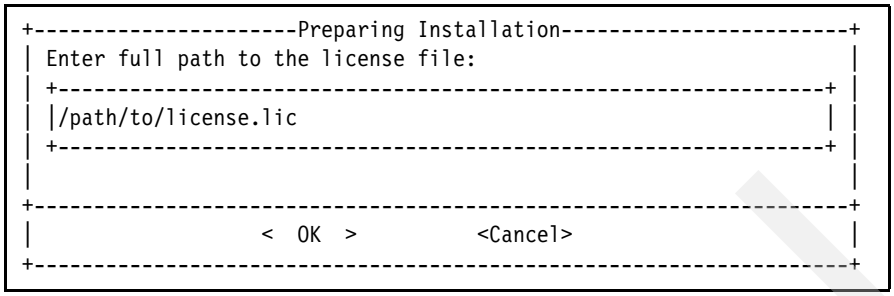

Figure 5 Provide path to license file

4. Retrieving channels configuration.

   Now the installer tries to connect to the Universal Server to retrieve information about the channels. See Figure 6. This screen closes by itself. Don't press Enter!

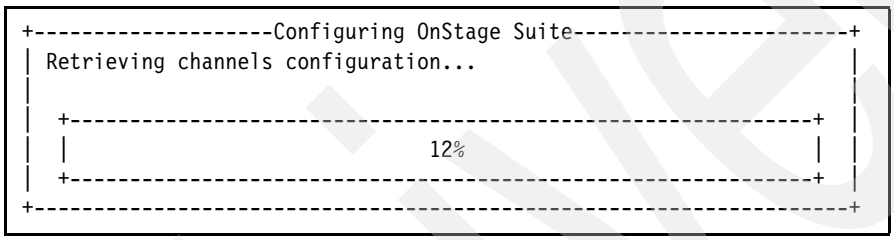

Figure 6 Retrieving channels configuration

   If the connection fails, you can enter information about a proxy server on the screens that follow. Provide the server name, the proxy server port and an optional user name and password, if needed.

5. Registering with Sun service.

   To use Aduva OnStage to manage hosts running Solaris, you must enter a user name and password for the www.sun.com service in the screen shown in Figure 7. Otherwise, simply leave blank. Make your choice and press Enter to proceed.

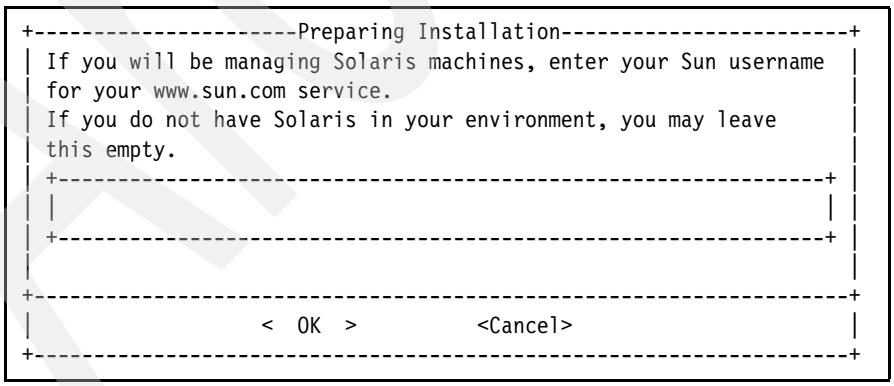

Figure 7 Registering with sun service (optional)

6. The next screen in Figure 8 on page 10 shows that the installation is prepared. Again this screen closes by itself. Don't press Enter!

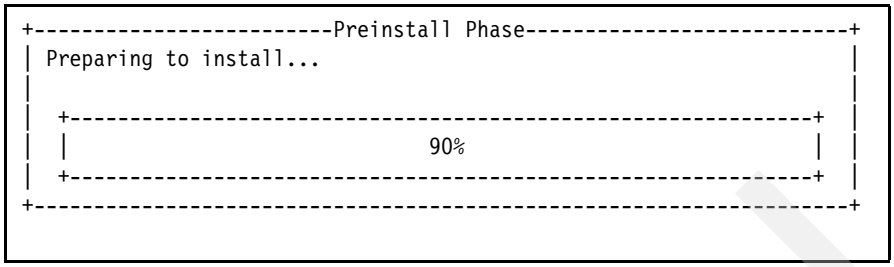

Figure 8 Preinstall phase

7. Predefined channels.

If you have started the installer with the -channels parameter, you can now define channels that are activated during installation. This is not necessary, because they will be activated as soon as they are needed. But you may use it if you want to predefine hosts; see “Predefine hosts” on page 17.

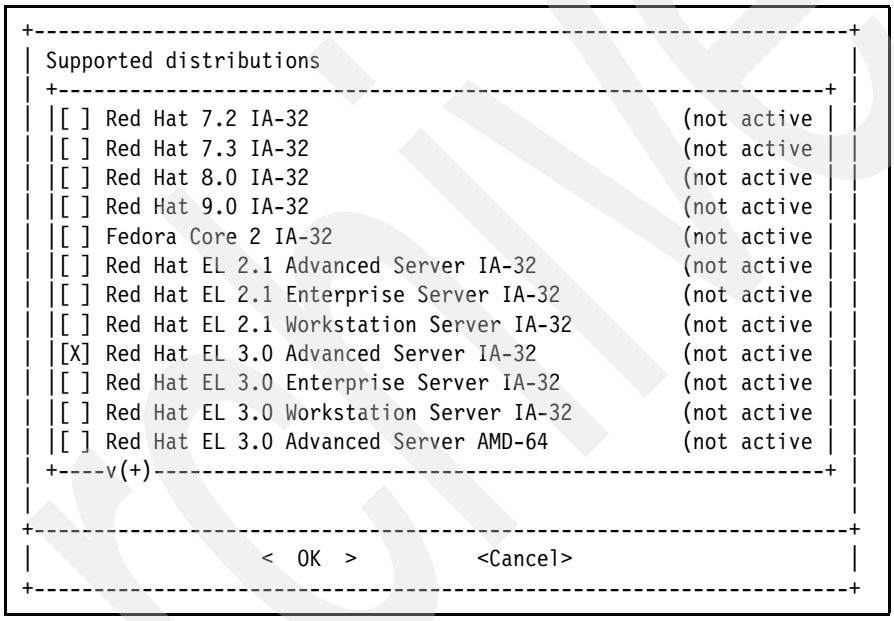

Figure 9 Select predefined channels

8. Confirm host name.

   It is important to use the correct host name or IP address. This information will be used by the clients to contact the SDS. Change the value if necessary and confirm it with Enter.

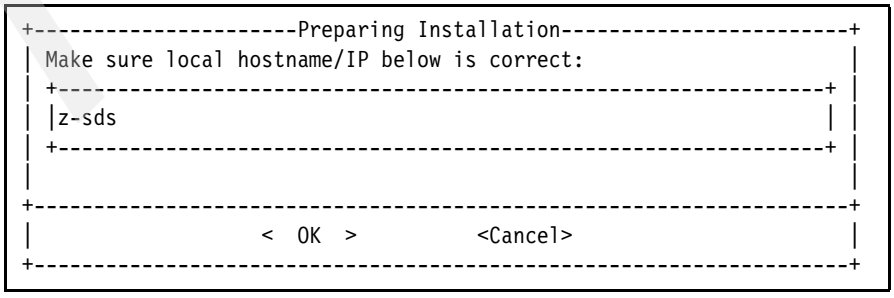

Figure 10 Confirm host name or IP address

9. Configure MS Windows Director Console

   The MS Windows version of the Director Console must be pre configured with the path, you want to install it in. Change if necessary and press Enter.

```
+----------------------Preparing Installation-----------------------+
| Enter the path where the Windows console will be installed:       |
| +---------------------------------------------------------------+ |
| |C:\director_console                                            | |
| +---------------------------------------------------------------+ |
|                                                                   |
+-------------------------------------------------------------------+
|                   <  OK  >          <Cancel>                      |
+-------------------------------------------------------------------+
```

Figure 11 Provide the path where the Windows console will be installed

10. Create a certificate.

    A SSL certificate is used to secure the communication between Agents and the SDS. Choose **Yes** to create your own certificate, and supply the correct information in the following screens. Choose **No** to create a default certificate.

```
+----------------------Preparing Installation-----------------------+
| EZinstaller generates the SSL certificate for the Universal       |
| Server. If you want to have your organization's details in the    |
| certificate, type yes below.                                      |
|                                                                   |
| Do you want to enter your details into the certificate?           |
|                                                                   |
+-------------------------------------------------------------------+
|                     < Yes >           < No  >                     |
+-------------------------------------------------------------------+
```

Figure 12 Create a certificate

11. Finishing the installation.

    In the last step, the installer:

    – Assigns the ports that will be used for communication.
    – Installs the System Dependency Server.
    – Installs the Dependency Manager and the Knowledge Base.
    – Prepares the installation files for the Agent, the Director Console, the Command Line Interface and the Proxy Server.
    – Creates the directory /usr/local/aduva/install/.
    – Opens a screen that informs you about the result of the installation.

After **`ezInstall`** has finished, the installation files for the Agent and other components are found in the installation directory:

- `agent-<MONTH><DAY>.tar.gz`, installation files for the Agent.
- `cli-<MONTH><DAY>.tar.gz`, installation files for the Command Line Interface.
- `console-<MONTH><DAY>.tar.gz`, installation files for the Director Console.
- `console-<MONTH><DAY>.zip`, installation files for the MS Windows Director Console.
- `proxy-<MONTH><DAY>.tar.gz`, installation files for the Proxy Server.

Additionally they are stored at /usr/local/aduva/install/.

Install the Agent on all hosts you want to manage with Aduva OnStage. Install the Director Console and optionally the Command Line Interface on hosts, from which you want to work with Aduva OnStage.

The Proxy Server is intended for load balancing, for more information, refer to the *Aduva OnStage Admin Guide*.

Problems with the installation

If you run into problems with the installation, look at the log files in the subdirectory logs in the installation directory.

Before you restart the installation, first remove the failed installation:

1. Execute **`ezInstall`**, accept the license and select “Uninstall Aduva OnStage”.
2. To be safe, delete the whole installation directory and start again with unpacking the tarball. The installation process seems to cache some information, and we have ended up with failed installations because of this.

Installed files

Aduva OnStage is installed at /usr/local/aduva, which has the following layout:

aduva.lic	The license file.
aduva.lic.sig	The signed license file.
director_engine/	Subdirectory for the Dependency Manager.
director_server/	Subdirectory for the Aduva Server and the Knowledge Base.
install/	Subdirectory for installation files, the installation log and tools for backup.
support_tool.sh	A script to generate support information.

Installation of the Agents

When the System Dependency Server is installed proceed with the installation of the Agent on all hosts you want to manage with Aduva OnStage.

Install an Agent

After the installation of the SDS you find a file called `agent-<MONTH><DAY>.tar.gz` in the installation directory. This tarball contains the public key of the SDS and the files to install the Agent.

Transfer the file to the host on which you want to install the Agent. Unpack the tarball with the following command:

```
tar xzvf agent-*.tar.gz
```

Start the installation:

```
director_agent/Install
```

The installer is similar to **`ezInstall`**. Choose **Install OnStage Agent** and press Enter.

The installation process adds the appropriate init script and starts the Agent. The Agent will try to connect to the SDS and register itself. The Agent is installed at /opt/local/aduva/director_agent and in the subdirectory logs the log files can be found.

The installation is fast and should complete in less than a minute.

Install the other Agents

Repeat the installation procedure for all hosts which you want to manage with Aduva OnStage.

Note: The Agent installer can be started with the -i option for an automatic installation. With the appropriate remote access and a short script you can save a lot of time and work.

Install Solaris Agent

For information about installing an Agent on Solaris, refer to the *Aduva OnStage Admin Guide*.

Installation of the Director Console and the Command Line Interface

After the installation of the SDS you find these files in the installation directory:

```
cli-<MONTH><DAY>.tar.gz
console-<MONTH><DAY>.tar.gz
console-<MONTH><DAY>.zip
```

They contain the public key of the SDS and the files to install the Director Console or the Command Line Interface (CLI) respectively.

The Director Console and the CLI are installed in the same way as the Agent. Transfer the file to the host on which you want to install it and unpack the tarball. Then run **`Install`** in the unpacked directory and choose to install the component.

The Windows Director Console

If necessary, you can install a MS Windows version of the Director Console. Transfer `console-<MONTH><DAY>.zip` to the Windows machine and unzip the file on the drive you specified in the installation process, Step 9 on page 11, for example, drive c: for the default.

Working with Aduva OnStage

This section provides necessary information to work with Aduva OnStage and get comfortable with basic usage and terminology.

The topics discussed in this section are:

- "The first session" on page 14 introduces the graphical user interface Director Console and shows how to map your infrastructure into host groups and users.
- In "Running jobs" on page 19 you learn to perform system checks and updates, and how to create and schedule jobs or view job status information.
- "Working with the inventory" on page 25 shows you how you can add and remove software packages and use profiles to enforce defined software configurations.
- "Local components" on page 31 gives you the knowledge to add your own software packages and manage them with Aduva OnStage.
- Besides the Director Console you can use the Command Line Interface. See "The command line interface" on page 36.
- "Restarting applications and reconfiguration" on page 37 provides information about the start scripts and the configuration files.

The first session

In the first session you change the default password for the admin user. After a short tour through the graphical user interface you verify your hosts and create additional users and groups.

Logging in

Open a console in an X environment on the host you installed the Director Console and enter:

```
director_console &
```

The Director Console starts up and a window appears. It shows the Aduva logo and the text “Acknowledging Aduva technology”.

After a short while the login dialog is displayed. Log in with the user name *admin* and the default password, which is `123`. When you log in with a user for the first time a new window opens, in which you must change the password.

After the login, the hosts and various settings are initialized. When this is finished the main window of the graphical user interface comes up.

Overview of the main window

The main window, shown in Figure 13 on page 15, consists of the menu bar, two rows of tool bars and two panels separated horizontally. Each panel is divided into three lists.

The upper panel is called the inventory panel and the three lists are from left to right:

Host list Shows all registered hosts grouped in channels and self defined groups.

Component list Shows installed and available components for a selected host or channel.

Action list Shows defined actions which can be used to run a job or create a profile.

The lower panel is called the job panel and the three lists are from left right:

Job list Shows running, scheduled and old jobs.

Task list Shows the tasks a selected job consists of.

Host in task list Shows the hosts the selected task runs on.

All lists are explained as they become important to the tasks you perform.

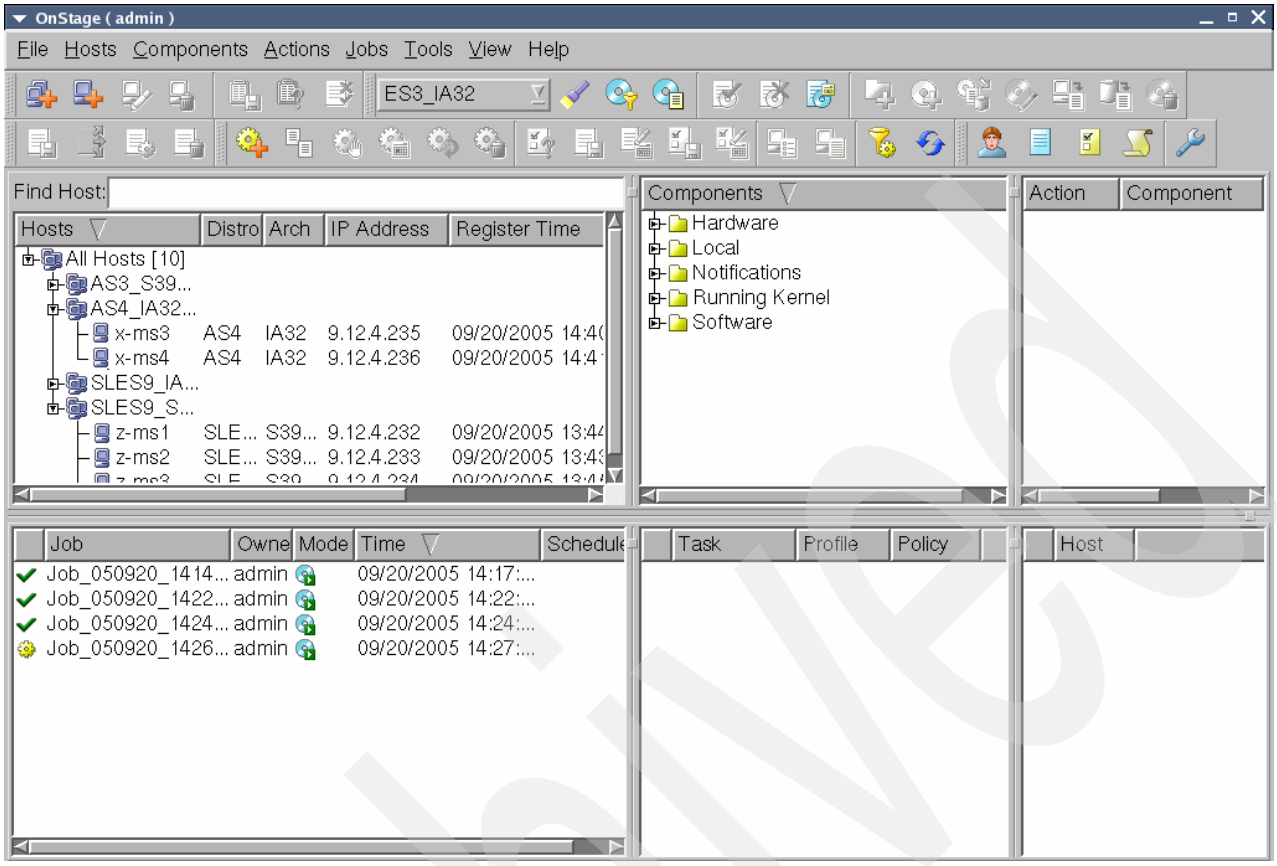

Figure 13 Director Console main window

Executing commands

In Aduva OnStage there are usually three ways of executing a command:

1. Use the menu bar.
2. Use an icon in the tool bar.
3. Use the context menu, activated by right-clicking on an object or into a list.

Both the menu bar and the tool bar are context sensitive. A command only becomes active when it is possible to use it with the object currently selected in one of the lists. Otherwise it is deactivated and appears gray. If nothing is selected only the commands are active, that can always be used.

The context menu shows the commands associated with the object or list you right-clicked. For example, the job list contains the commands for running and managing jobs. But if you right-click a specific job, additional commands are possible.

Although there are different ways to execute a command, most of the time we use the context menu in the examples. It is not difficult to issue the same command using the corresponding tool bar button or the menu bar.

Most menu items in the menu bar and in the context menu display the icon from the tool bar. Both menus use the same names for commands. Moving the mouse over a button in the tool bar opens a quick tip, a short description about the command the button executes.

Common buttons

There are common buttons you find in many dialog boxes. The following list shows how they look and for what they stand:

- The new button .
- The edit button .
- The copy button .
- The delete button .
- The select button .
- The multi distribution button .

Host and group information

Each system that is managed with Aduva OnStage is called a (managed) host, whether it is a single physical system or a server or guest running in a virtual machine or logical partition. You can put hosts in groups to treat them as one unit and run jobs on them easily.

Hosts and groups are displayed in the host list, see Figure 14. The default group All Hosts is a special sort of group. It consists of several sub groups, called distribution groups. Each registered Agent is put in a distribution group, depending on its distribution and architecture.

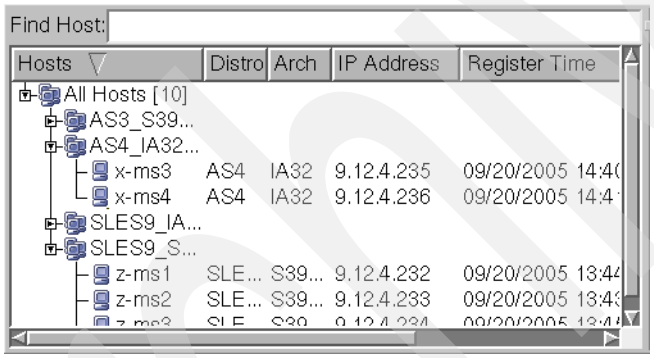

Figure 14 Host list

The column Hosts contains the group or host name. Use the arrow on the left of a group name to fold or unfold the sub entries. After a group name, a number in square brackets indicates the number of hosts in the group. The icon to the left of a name indicates its type and gives additional status information:

- Group icons
  - All Hosts group and distribution group .
  - User defined group .
- Host icons
  - Managed host .
  - Host or distribution not covered by current license .
  - Disconnected host (offline) .
  - Unrecognized host, errors with Agent .

Additional information can be found in the other columns:

Distro The Linux distribution the Agent runs on.
Arch The architecture of the host.

I.P. Address The IP address of the host.
Register Time The time the host registered itself with the SDS.

To verify that all hosts are registered correctly, look whether a host has an entry under the corresponding distribution group and that the host icon indicates no errors.

If you cannot find a host or run into problems, assure that the Agent is installed and the network is set up correct so that he can contact the SDS (see “Aduva OnStage architecture” on page 2).

Try to restart the Agent on the host by entering the following command as root:

```
/etc/init.d/director_agent restart
```

And take a look at the log files of the Agent, stored under /opt/local/aduva/director_agent/logs

Create groups

The following steps add a group called “WebSphere” as an example of how you can create your own groups:

1. Choose the add group dialog from the menu: **`Hosts`** → **`Add Group`**. See Figure 15.
2. Enter the name `WebSphere` in Group Name.
3. Leave Parent Group blank. You could choose one by clicking on the select button  beside the field.
4. Select the hosts z-m1, z-m2, z-m3 and use the arrow keys to add them to the group. All hosts in the right list labeled Selected Hosts will become part of the group.
5. Click the **OK** button to create the group.

You can later change the members of a group: Right-click a group and choose **Edit**. It is also possible to change group membership in the host edit dialog: Right-click a host and choose **Edit**.

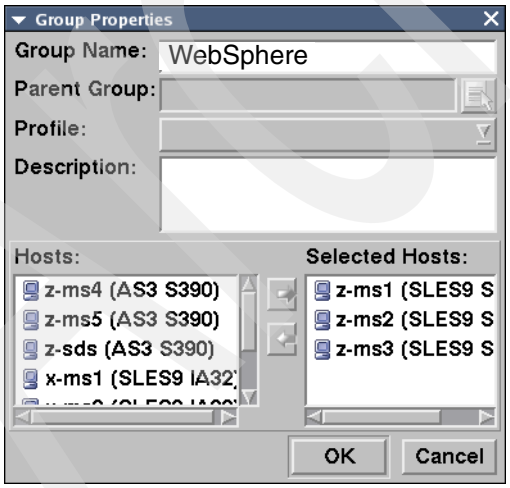

Figure 15 Add group WebSphere

Predefine hosts

It is possible to predefine hosts prior to their integration. Start the Director Console with the following command:

```
director_console manual_host_create true &
```

When you are logged in, choose **Hosts** → **Add Host** from the menu to bring up the add host dialog. Make sure to select the right distribution and enter the actual host name in Unique String.

Create new users

Aduva OnStage has one predefined user account named admin. This account is a super user account with full permissions and the only user, who can manage other user accounts. The admin account cannot be deleted.

You can create additional accounts with full or restricted permissions. A user with full permissions can run jobs on all hosts and manage local components but cannot manage user accounts.

A restricted user cannot manage local components, but may use them in jobs. You can restrict a user to work only with permitted groups or to run only simulations - or both (run only simulations, but only on permitted groups).

A user can only be logged in once. If you login with an account that is already logged in, the other account will be disconnected.

Now we show you how to create a user called WebSphere with limited access to the WebSphere group. Figure 16 on page 19 shows the relevant part of the dialog.

Follow these steps as an example of how you can add your own users:

1. Choose the users dialog box from the menu: **Tools** → **Users**.
2. Click the new button  .
3. Enter `WebSphere` as the user name.
4. Enter twice the initial password for the user.
5. Uncheck the check box **Grant Full Permissions**.
6. Click the select button  on the left of the field Permitted Groups.
7. Select the group WebSphere and add it with the right arrow button . Click **OK** to close this dialog box and go back to the user creation.
8. Under Notifications you can enter an e-mail address and conditions when to notify this user. This service relies on a configured mail system on the system the SDS is installed on.
9. Click the **OK** button to create the user.

Repeat these steps to add all the users you need.

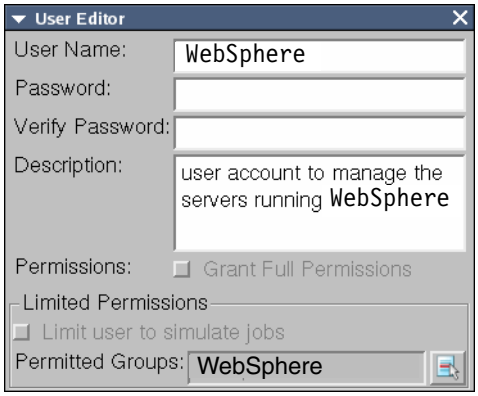

Figure 16 Add WebSphere user (only part of the dialog)

Logout

Your first session has come to an end. You have verified that all agents are up, have created additional users and groups and have taken your first glimpse at the powerful Director Console. Now you may want to look at what Aduva OnStage really has to offer and learn what jobs are and how to run them. Otherwise, the logout is in the menu bar: **File** → **Logout**.

Running jobs

Aduva OnStage uses jobs to run tests or install software by working with components. A component is a logical unit like a software package or a file. Until “Working with the inventory” on page 25, it is sufficient to think of a component as a software package.

A job consists of tasks that are processed one after another. A task itself is a combination of three things:

- A profile
- A set of hosts
- A policy

In short, a profile is a definition of which components must be installed and which not. When a task is executed the profile is applied for every host and a set of actions is created by the Dependency Manager. These actions, such as install, upgrade, or uninstall a software package, are checked against the policy. The policy defines which actions are allowed and which not, or if the user should be asked for a confirmation.

Every job has a mode it runs in. It is either deploy mode, which means that the actions really take place, or simulate mode, which checks what would be done. It is easy to rerun a simulated job in deploy mode. Select Rerun in a jobs context menu or Copy to edit based on the task definitions. We recommend that you simulate a job before you change a productive system.

Predefined profiles

Aduva OnStage comes with a set of predefined profiles:

Check Bugs Fix Every installed software component is checked against the Knowledge Base. If a bug fix exists, it is applied.

Check Security Every installed software component is checked against the Knowledge Base. If a security fix exists, it is applied.

Check System For every installed software component missing dependencies are installed. Conflicts between packages are resolved by downgrading or uninstalling the component with lower priority. Also security fixes are applied.

Check Withdrawn Patches
A Solaris only profile that checks for patches withdrawn by the vendor and does a downgrade or upgrade to remove the patch.

Local Software Review
Checks whether a local software component can be replaced by a certified software component.

Perform Reboot Reboots the host.

Perform Reboot + Reconfigure
A Solaris only profile that reboots the host for post install configuration.

Upgrade All Components
Every installed software component is upgraded if possible.

Use these profiles to create jobs or learn how to define your own in “Profiles” on page 29.

Creating a job

The first thing you will do, is check whether the system the SDS runs on is consistent, and fix conflicts automatically (profile Check System). After this all packages will be updated (profile Upgrade All Components).

Figure 17 on page 21 shows the new job dialog with the task editor dialog open, while completing the following example:

1. Open the new job dialog: **Jobs** → **New**.
2. A job name is automatically created (with a time stamp).
3. Click the radio button **Deploy** to really run this job and not a simulation.
4. To add a task click the new task button  . The task dialog appears and you can add a new task:
   a. Enter a name for the task, for example, `Check System`.
   b. Choose the profile **Check System** from the drop down list.
   c. Click the select button  beside the field Hosts to open the host selection dialog. Add the host the SDS is running on: select **z-sds** and click the right arrow button . Press **OK** to confirm and close the dialog.
   d. Choose the policy **Always ask me** from the drop-down list.
   e. Click **OK** to confirm and close the dialog box.
5. Repeat Step 4 to add another task, but this time choose the profile Upgrade All Components and enter a different name for the task.

> **Task order:** Tasks are run one after another like they are ordered in the task list. You can use the arrow buttons to move a selected task up or down.

6. Click **OK** to run the job. The dialog box closes and a new job appears in the job list.

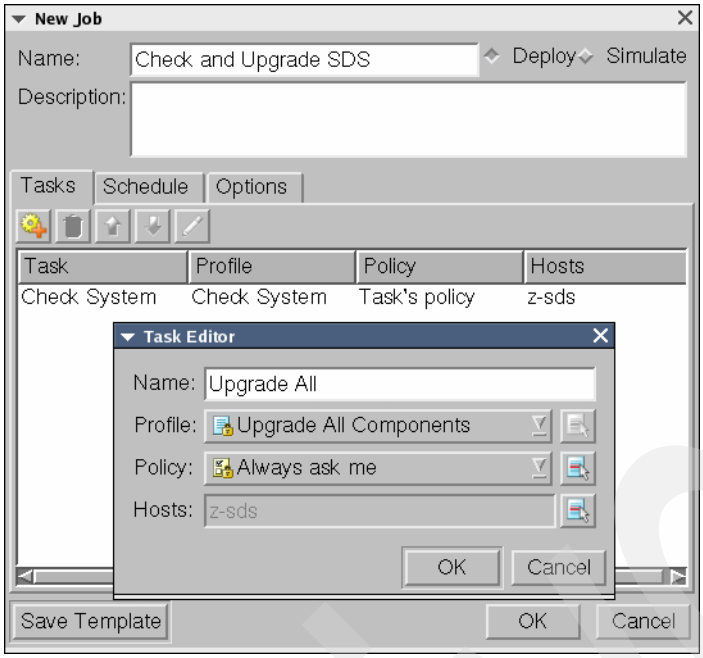

Figure 17 Job to check SDS

The created job starts running immediately. See the next section, “Monitoring jobs”, to find out more about the job and its status.

Monitoring jobs

The job list contains all jobs until you delete them, regardless of whether they are running right now, are scheduled or already have been run. You can right-click a job to open the context menu and rerun or delete the job, or apply a filter to the job list.

Sorting lists: As with all lists, jobs are sorted by clicking on one of the columns. Clicking on that column again reverses the sorting order.

Job Status

The first column in the job list indicates the status of the job represented by an icon. The column Mode shows if the job is in deploy mode or in simulate mode.

Consult the following list of job status icons:

- Finished . The job ran successfully.
- Working . The job is still running.
- Scheduled . The job scheduled to run once.
- Scheduled periodic . The job is scheduled run periodically.
- Waiting for confirmation . The job needs confirmation of some actions.
- Waiting for offline hosts . At least one host the job works with is offline. The job will continue as soon as the hosts are connected to the SDS.
- Failed . The job couldn’t complete or was aborted.
- Deploy mode . The job is in deploy mode.
- Simulate mode . The job is in simulate mode.

You can find more information in the other two lists of the job panel. If you select a job in the job list, the task list displays the tasks the job is made up of. In turn the host in task list shows the hosts the selected task is working with. Figure 18 shows the job panel with its three lists and a running job.

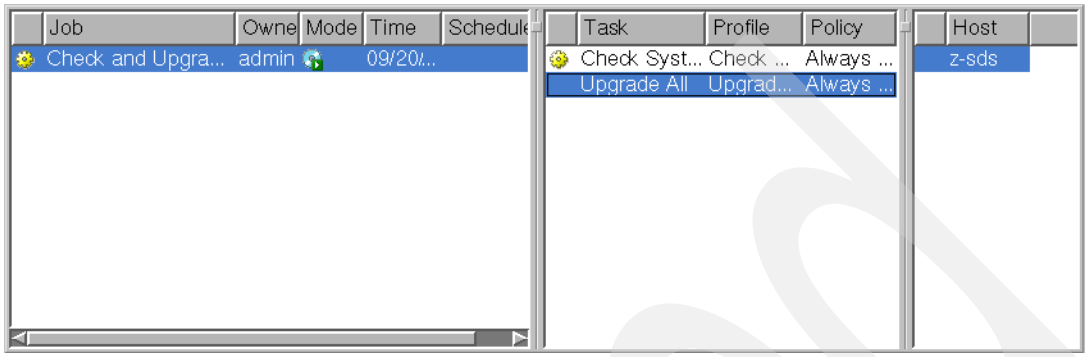

Figure 18 Job panel with running job

Job progress and job log

Open the context menu with a right-click the host in the host in task list and select **Show Progress** to view the detailed progress of the steps the task is taking.

After a jobs is run, open the log by right-clicking on the host in the host in task list and select **Show Log** to see the result and how it was achieved.

Waiting for confirmation

If the job status icon is a question mark, the job has generated a set of actions that will change the system. All actions need a confirmation and since you used the standard policy Always ask me, you need to confirm them manually. See the next section, “Confirming actions”, and find out how to do this. The section “Policies” on page 23 helps you to create your own policies and automate this process.

Confirming actions

A question mark as job status icon indicates that a task needs manual confirmation. To confirm actions do this:

1. Select the job in the job list.
2. Open the context menu by right-clicking on the task that needs confirmation. In the host in task list you can see on which host the task is running at the moment.
3. Choose **Confirm** from the context menu.

The confirmation dialog appears and presents a list of actions. The component information button opens a dialog box, where you can find information about the selected component.

Review the list and then press **Yes to all** to allow all actions. Choosing No for an action forces the job to find a solution without it which usually results in a failed job.

When all actions are set either to yes or no, the OK button becomes active and you can press it to continue. The job continues and tries to find a solution with the choices you made. If that is not possible, the job fails. If this happens, consult the job log for more information.

When you install a fix, this results in an upgrade of the software which has to be confirmed in a second step. When a kernel update is done, you will be asked in another confirmation step whether you want to reboot the affected hosts or not.

Note: The predefined profiles Check Bugs Fix, Check Security and Check System behave different for confirmations. Selecting No for an action just leaves it out, still performing the rest of the actions.

Policies

Manual confirmation can become tedious and it stands in the way of scheduled jobs, which should run with no interaction at all. Use policies to define default behavior for the confirmation of actions.

A policy can be very fine grained down to the level of individual software packages. To maintain this level of detail, the single rules that make up a policy are always specific to a channel. The reason is, that packages can have different names or versions in different distributions or even be in some and not in others.

But again Aduva OnStage can easily hide the complexity and you can set up rules for multi distribution, a feature we will visit in more detail later.

Create a policy

To set up a job that automatically updates components with known security issues and existing bug fixes you first need to create a policy. You also want to the exclude kernel from any changes.

To define the policy:

1. Open the policy dialog with **`Tools`** → **`Policies`**. Use this dialog to add, edit and delete policies.
2. Click the new button  to open the policy editor dialog. See Figure 19 on page 24.
3. Enter a descriptive name for the policy, for example: `Fix all, except kernel`.
4. The drop down list on top of the dialog indicates for which distribution the rules are created. As you will use the multi distribution feature it is not important which distribution you select.
5. Select **Software** in the components list.
6. Select **Yes** in the drop-down lists Install, Uninstall, Upgrade From, Downgrade From, Apply Fix. This adds the rules to the rules list.
7. Now you have to exclude the kernel. Select **kernel-node** under Software in the component list.
8. Select **No** in the drop-down lists Install, Uninstall, Upgrade From, Downgrade From, Apply Fix.
9. Press the multi distribution button  to expand the rules to all distributions.

   A new dialog comes up, which shows the result of the expansion. With such generic rules there shouldn't be any problems. Otherwise see "The multi distributions feature" on page 28. Close and confirm with the **OK** button.
10. Press the **OK** button to create the policy.

Repeat these steps to create any additional policies you need.

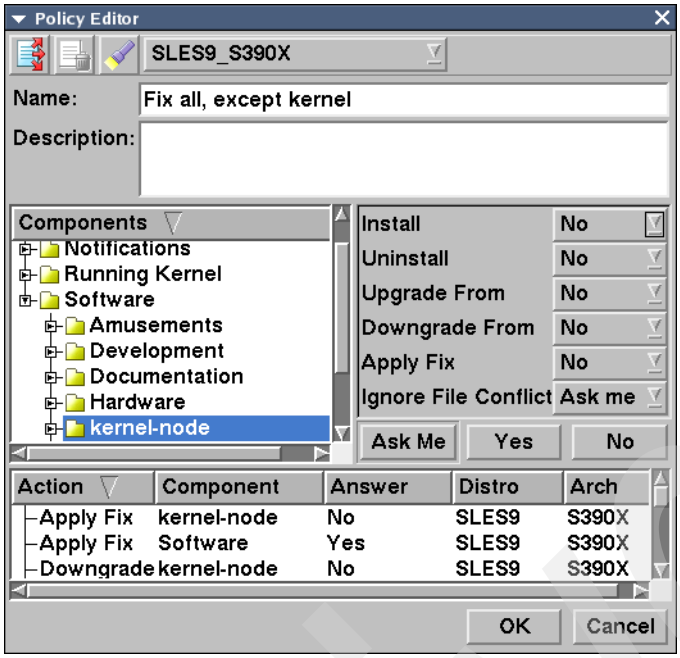

Figure 19 Policy editor dialog

Scheduling a job

You can schedule jobs to run at a specific time, either once or periodically. You will set up a periodic job to check for known security holes and bugs and fix them automatically on all machines.

Follow these steps to set up the periodic job:

1. Open the new job dialog box: **Jobs** → **New**.
2. A job name is automatically created, replace it something more descriptive, for example: `Install patches`.
3. Click the radio button **Deploy** to really run this job and not a simulation.
4. Add two tasks to the job:
   - One with the profile Check Security and the other with Check Bugs Fix.
   - For both tasks choose the All Hosts group and the newly created policy Update All.
5. Click the register card **Schedule** tab to set the scheduling options. See Figure 20 on page 25.
   a. Select the schedule type periodic by setting the appropriate radio button.
   b. Choose the days the job should run. For this job select **Every month** (in the drop down list) and **Every day** (the radio button).
   c. Enter `02:00` in Earliest field. This will run the job at 2 a.m.
6. Click the register card labeled **Options** tab. Select what should be done if a profile or a policy changes that is used by the job.

   Select **Accept changes before next run** to run a job with the new profiles and policies. Otherwise it would run with the old ones. **Notify of changes** sends an e-mail, if you supplied a correct e-mail address for the user and an e-mail server is configured on the SDS.
7. Click **OK** to start the job.

A new job with the periodic scheduled job icon appears in the job list. When the specified time has come, a new job will appear and start running. A “S” enclosed in brackets is appended to the new jobs name.

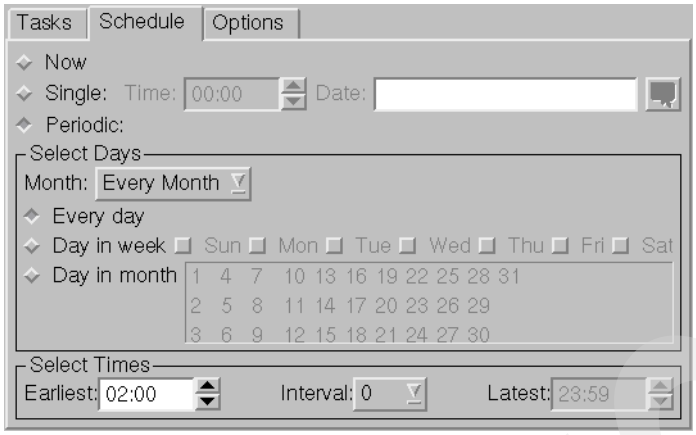

Figure 20 Scheduling a job

To reschedule the job select **Reschedule** in the context menu. Only periodic scheduled jobs can be rescheduled. To stop the job from running periodically delete it, by selecting **Delete** in the context menu.

Running multiple times a day: To run a job multiple times a day, select an interval in the drop down list Interval in the schedule register card. This sets the time in minutes after which the job will run again. The job will start running at the time entered as earliest until the time entered as latest.

Job templates

When you create a new job, you can save a job template by pressing the **Save Template** button in the new job dialog. This creates a new job with the periodic scheduled job icon . This job never runs, but you can use it to create a new job. Choose Copy in the context menu and the new job dialog appears. The tasks and settings are taken from the job template.

Working with the inventory

So far we showed you how to use predefined profiles to check and update a hosts software. But Aduva OnStage has more to offer. This section demonstrates what you can do with the inventory. Here are some topics:

- You can browse the inventory of installed and available packages, and install or remove them, see “The inventory” on page 26 and “Inventory jobs” on page 26.
- You can create new profiles to define which software must be installed on a host, and to define a set of actions to perform. See “Profiles” on page 29.
- You can rollback to earlier states or clone a whole system. See “Rollback” on page 29 and “Cloning a host” on page 30.

The next section “Local components” on page 31 goes even further. You learn how to add you own software packages to the inventory and manage them.

The inventory

The inventory is the list of installed and available components. Each channel has its own inventory.

If you have selected a host in the host list, you will see the components for that host in the component list. Otherwise you will see only the available components for the channel that is selected in the distribution drop down list in the upper tool bar.

Up until now we referred to components as software packages, but they are more than that. For a host you see these default categories of components in the inventory list:

Hardware	Information about the CPU, memory, network adapters and storage.
Local	Additional software packages you add to the inventory.
Notifications	Messages you can send to trigger actions, like rebooting the host.
Running Kernel	The kernel that is running at the moment.
Software	The installed and available software packages certified by Aduva.

If your no host is selected only Local, Notifications and Software are shown.

Components are represented by a browseable tree. A “+” indicates that you can expand the component to see subordinated components. See Figure 21 for an example of the component list and the action list.

The command **Details** in the context menu opens a new window with information about the selected component.

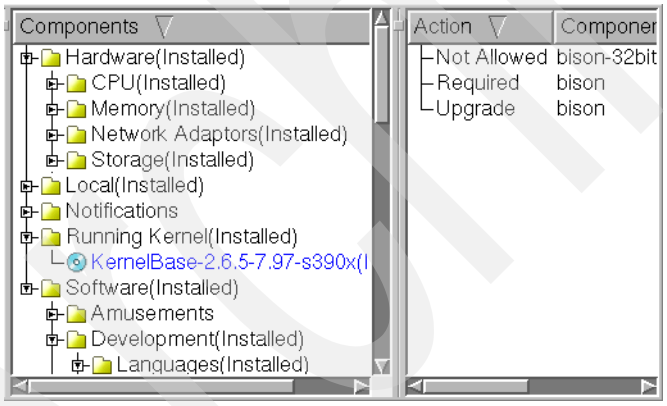

Figure 21 Component list and action list

Note: The component list shows either all components or only the installed components. Select **Components** → **Show Installed** from the menu to switch between the two. If all components are the displayed, the installed ones get “(Installed)” appended to their name.

We will concentrate on software components in “Inventory jobs” on page 26. Local components are discussed in “Local components” on page 31. There is not much to say about the other components, see the *Aduva OnStage Admin Guide* for more information.

Inventory jobs

Software components are divided into three subcomponents; see Figure 22 on page 27.

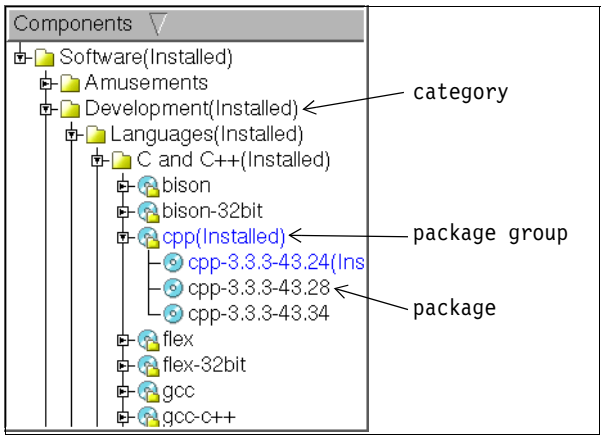

Figure 22 Software components

A package group contains the different versions of a package. A package represents the actual RPM software package.

You can create a job based on the software inventory, when you add actions to the action list. An action is one of the following settings assigned to a component:

- Required → install, if not installed
- Not allowed → uninstall, if installed
- Upgrade → upgrade, if installed

The upgrade setting can be applied to any software component, while the other settings can only be applied to package groups and packages.

The "required" action

This setting installs the package or the newest version of that package group respectively. If the package or at least one package of the group is already installed, no action is taken. Missing dependencies are installed too.

The "not allowed" action

The package or all packages in the group are uninstalled, if any are installed.

The "upgrade" action

The package or all packages in the group or category are upgraded if possible. Only installed packages are upgraded and no new packages are installed, except for new dependencies.

Install a software package

This example shows you how to create a job, that makes sure that x3270, the 3270 terminal emulator, is installed all hosts:

1. Right-click **Software** in the component list to open the context menu.
2. Select **Search**.
3. The search dialog comes up. Enter `x3270` in the Search For field and press **Find**.
4. The component list shows the package group x3270.
5. Open the context menu and select **Required**. This adds the action to the action list.
6. Right-click into the action list, in the context menu choose **Multi Distributions**.
7. The results of the multi distribution process appear in a new dialog. See "The multi distributions feature" on page 28 for more information. Press **OK** to confirm.

8. For every channel that was matched there will be an action in the action list, that requires x3270 to be installed on that distribution.
9. Before you run the job, select the hosts in the host list. Click **All Hosts** to select it.
10. Right-click in the action list and choose **Run on Selected Hosts** from the context menu.
11. The run job dialog appears, choose to deploy the job and press **OK**.
12. Your job appears in the job list and starts running.

Creating more complex jobs is just as easy. Repeatedly apply actions to selected components. Set uninstall or upgrade the same way as with require in the example. Actions can be removed from the action list with Delete Selected from the context menu.

By the way, the set of actions in the job list is nearly a profile, except that you didn't save it as one. Before you learn how to do that in "Profiles" on page 29, let's look at the multi distribution feature.

The multi distributions feature

Whatever you do with a component (whether you create a confirmation for a policy or an action for an inventory job), you always work with a component from a specific channel. The current channel is selected in the distribution drop down list in the upper tool bar. When you select a host in the host list it changes to its channel.

The multi distribution feature helps you to generalize your settings. It expands confirmations or actions and tries to match them to the other active distributions.

When you invoke a multi distribution a new dialog appears, see Figure 23. It shows the result of the expansion for every channel. A component is either "Matched" or "Non existent". Expand the tree to the necessary level to view the information you want.

Matched components have a check box which indicates whether they will be adopted or not. Different colors indicate the result for a channel:

- Black: Components are already in the action list.
- Green: Successfully matched all components.
- Dark Yellow: Some components could not be matched (most likely, components exist under a different path or distribution category).
- Red: No component could be matched.

If a distribution is only partially matched, all matches are unchecked.

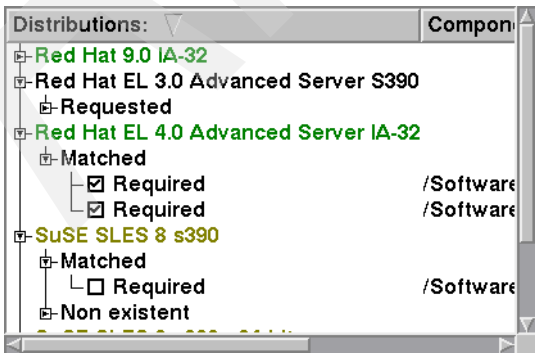

Figure 23 Results of multi distribution

Profiles

Instead of transforming a set of actions into a job you can save them as a profile. You can use it to create tasks or perform a compliancy check.

To save a profile from an action list:

1. Define all your actions like described before.
2. Apply the multi distribution feature if necessary.
3. Open the context menu on the action list and select **Save as profile**.
4. The profile editor dialog appears. Enter a descriptive name and press **OK**.

Profiles are managed with the profile dialog. Open it with **Tools** → **Profile** to add, edit or delete profiles. When you press the new or edit button, the profile editor dialog comes up. It contains a component list and an action list, so you can add and remove actions as described before.

Use a self defined profile the same way as a predefined profile:

1. Create a new job.
2. Add a task, choose the self defined profile, a policy and hosts to run the job on.
3. Deploy or simulate the job.

Profile compliancy

You can attach a profile to a host or a host group. This doesn't restrict the jobs you can run, but you can do a compliancy check based on that.

To attach a profile:

1. Choose **Edit** from the context menu of a host or group.
2. Select the profile from the drop down list Profile.
3. Press **OK** to confirm the changes.

To run a test for compliancy, choose **Profile Compliancy Check** from the context menu of a host. The host is checked against the profile assigned to it and the profiles of the groups it belongs to. A new dialog shows the results and a red X indicates that the host is not compliant with the profile. In this case, use the profile to create a job, which fixes any differences.

Rollback

Aduva OnStage not only helps you to automate updates and deploy new software, you also don't have to worry so much about the consequences anymore. Whether you accidentally applied the wrong profile, ran a job with a weak policy or want to get back to an earlier state of the software configuration, a rollback is as easy as anything else — you just run a job.

With Aduva OnStage you can compare the current inventory to an earlier state and create a set of actions to revert to that state. Every time you deploy a job on a host, its inventory gets saved automatically. This makes it possible to rollback a job or a series of jobs. Besides, you can save the inventory any time you want, or even save all the inventories of a group. Choose Save Inventory in the context menu of the host or host group you want to save.

We now show you how to revert to a previous configuration. In this case, we roll back the last job, a scheduled upgrade of z-ms1. The state was saved as "AU: z-ms1_upgrade...". You need to get the differences between the current and the former state:

1. Right-click the host, in the context menu choose **Compare Inventories**. The inventory compare dialog shown in Figure 24 comes up.
2. Assure that the Target host and the Source are set to the host you want to restore.
3. Set Inventory of the target host must to Current Inventory, otherwise you can compare the inventories, but not create a job out of the differences.
4. Select the state “AU: z-ms1_upgrade...” in the drop down list **Inventory of Source**.
5. Enable the check box **Software** and press **Compare**.
6. The component list gets populated with the components that are only installed in one of the inventory states. They have the same color as the corresponding target or source host name to indicate to which state they belong.
7. Select the radio button **Tasks to make Target like Source**. This changes the component list to show the actions necessary to restore the selected state. With the **Delete** button selected actions can be removed from the list.
8. Verify the actions and press the button **Make Target like Source** when you are done.
9. A run job dialog comes up. Choose the deploy mode and press **OK**.
10. A new job appears. Press **Close** to leave the inventory compare dialog.

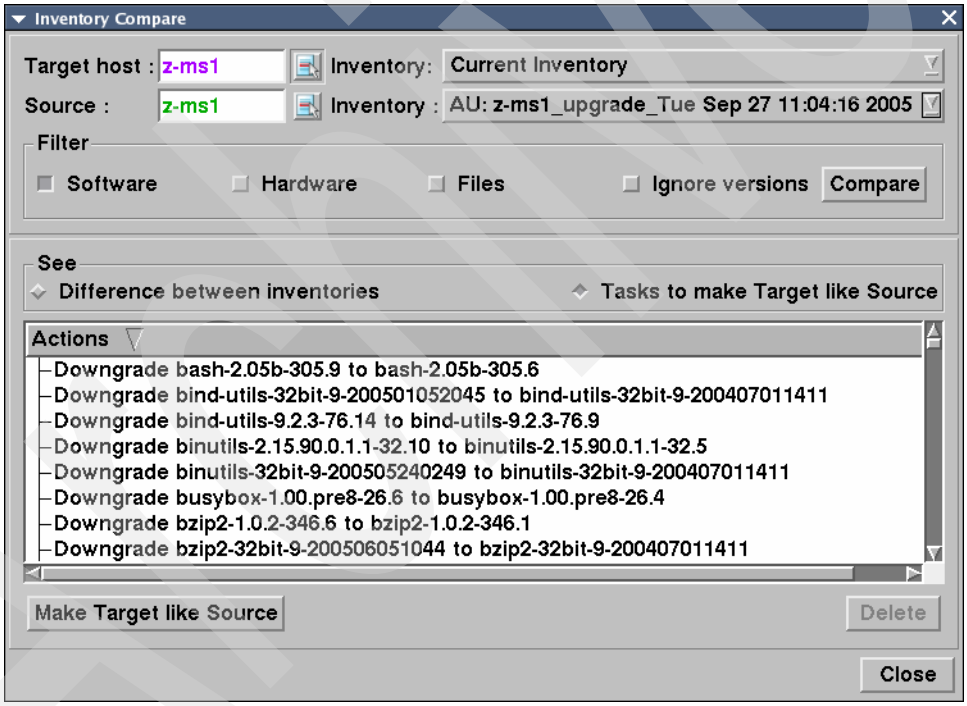

Figure 24 Inventory compare

Cloning a host

An inventory compare is also a great help in finding differences between hosts, as long as they belong to the same channel. You can even compare the hardware that is installed on two hosts.

You can use compare inventory to clone a host in a sophisticated way. Instead of blindly copying a software configuration this feature matches installed components and installs them in a native way.

Cloning a host is done like a rollback, but you select a different host as source host. The inventory state of source and target must be set to current inventory otherwise you cannot create a job from the differences.

Local components

All the software components in the inventory are well tested by the Aduva Certification Lab to ensure a successful and safe deployment. That is the reason why they are also known as Aduva Certified Objects (ACO).

With local components you can manage software that is not certified and distributed by Aduva, so called Non Certified Object (NCO). This can be proprietary or third party products and your own private software. Read more about this in “Local software” on page 31.

The local inventory also holds configuration files and macros or executable files that can be run as pre or post actions in job deployment. More information can be found in “Local files” on page 33.

You upload software and files to the Knowledge Base of the SDS and store them in the local inventory. They are never pushed to the Universal Server and don’t leave your organization.

For Solaris specific information about local components, refer to the *Aduva OnStage User Guide*.

Categories in the local inventory

The default category local is divided in further categories:

- Configuration files
- Local RPMs
- Macros
- Post-actions
- Pre-actions
- Probes

Before you upload any software or files, you should add appropriate sub categories. This example shows you how to add a category for the zfoo package used in “Adding undetected local software” on page 32:

1. Right-click **Local RPMs** in the Local category of the inventory.
2. Choose **`Local`** → **`Add Category`** from the context menu.
3. Enter the name of the category: `zfoo-scripts`.
4. Choose the channels for which you want to add the category in the list. In this case all the distributions for zSeries.
5. Press the **Apply** button to complete the creation.

Local software

Usually Aduva OnStage detects local software by itself and puts it in the Local RPMs group in the category local of the inventory. The local expansion technology generates rules and components for NCOs but of course they are not tested and not as detailed as the rules of ACOs. You can use local software in you profiles and jobs like you use ACOs from the software components.

These icons indicate the status of a local software component:

- Local software that has a packages attached  .

- Local software is detected, but has no packages attached .
- Local software with missing dependencies, either with software attached or not .

Note: The MS Windows version of the Director Console is not capable of uploading local software. You have to use the Linux version or the Command Line Interface.

Adding undetected local software

If you want to deploy local software that isn't installed on any host yet, you have to add the RPM packages to the category local RPMs. We will use the package zfoo-1.3-2.s390x.rpm as an example, a self developed administration script we want to install on all zSeries guests.

See "Categories in the local inventory" on page 31 first, and add a category for zfoo. After this step you can add the local software:

1. Make sure you work with the right distribution. If necessary choose it in the distribution drop down list.
2. Select the category zfoo-scripts you created earlier and choose **Choose Local** → **Add** from the context menu. The add software dialog comes up, see Figure 25 on page 33.
3. Select from where you want to upload the files with the radio buttons Upload File From. Choose either **Console** or **Managed Host** and the corresponding select button becomes active. Proceed with your method of choice:
   a. Upload through Director Console:

      Click the select button and choose the RPM file in the open file dialog. You can select more than one file with the use of Shift and Control while clicking on a file. Press **Open** when you are done.

      The files get added to the list. Press **Apply** to upload the files**.**

   b. Upload from managed host:

      Click the select button. Choose the host in the dialog and press **OK**. Enter the path to the file you want to upload in **File Name** and press **Apply**. Press **Reset** to upload another file.

      Note that the limit for a file is 5 MB when you upload it from a host.
4. When an upload is finished, the **Close** button becomes active. Press it to leave the dialog.

A package group zfoo is created under zfoo-scripts, and the RPM package is placed into it.

It takes about two minutes for the local expansion technology to create the rules. During this time the status indicates missing dependencies which may get resolved during the expansion.

You have to repeat this process for all channels separately.

Note: When you upload an unknown version of a known package, it is added to its package group in the software components.

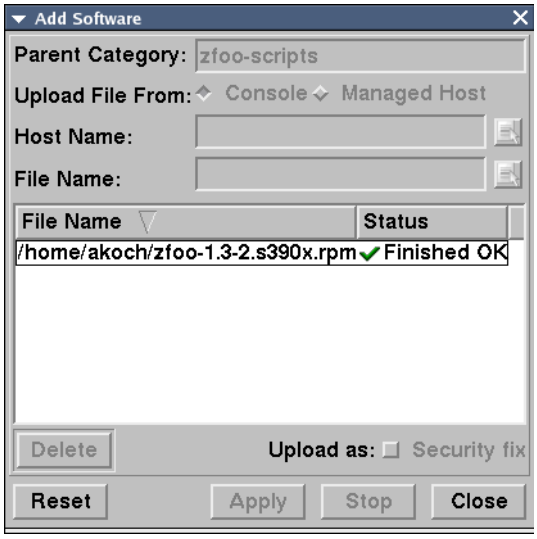

Figure 25 Add local software

Attaching software to detected local components

In some cases you need to attach local software manually:

- You want to replace a detected local software with a different version.
- A local component was detected, but the SDS couldn't upload it.
- There are missing dependencies with a local component.

To attach software select the detected component and choose **`Local`** → **`Load`** from the context menu. In the dialog select the distributions and upload the file either from the console or a managed host as described in "Adding undetected local software" on page 32.

Mark local software a security fix

You can mark local software as a security fix, and use the Check Security profile to update all installed versions.

Do this when you upload the component by activating the check box **Security fix** or later on, when you choose **`Local`** → **`Edit`** from the context menu.

Local files

With local files Aduva OnStage gives you the tools to automate nearly every aspect of administration. You can deploy configuration files and use scripts to perform all kinds of tasks.

Configuration files

You can upload configuration files to the local inventory to install them on hosts. Use Macros to customize them to a hosts environment.

Our zfoo package from a previous examples stores a configuration file under /etc/zfoo/zfoo.conf. We assume, that lately this file has changed and we want to update it on all hosts.

We can upload the new configuration file and deploy it with a job:

1. Add a category zfoo under configuration files in local components, see "Categories in the local inventory" on page 31.

2. Create a file declaration: Choose **Local** → **Add** from the context menu of the category you just created. Enter the path were the file should be installed under **Target File Pathname**. Make sure that the directory exists before you use the file in a job.

   Select all S/390 channels in the list. Press **Apply** to commit the declaration.

3. Attach a file: A new category with the files' path name is created. In its context menu choose **Local** → **Add**. The add dialog comes up.

   You can upload multiple version of one file. To distinguish between them, enter a text under Version, it is appended to the file name in the inventory.

   Select the S/390 channels and upload the file either from the console or a managed host as described in "Adding undetected local software" on page 32. Press **Close** when you are done.

4. Now you can add the configuration file to an action list as required and deploy it with a job, see "Install a software package" on page 27.

The next thing we want to show you are macros. Let's suppose time has passed and new versions of the zfoo scripts were developed. Some functions need the information whether they run on SLES 9 or RHEL 3 and we want to store this information in a config file. Of course we can create two different versions of the zfoo.conf file and deploy them with profiles. But you can imagine where that ends when more and more localization needs to be done on the config file.

Macros are a way out of this. A macro is a script that sends a single line to standard out. When you install a configuration file that contains a macro placeholder, the Agent downloads the macro, executes it and its output replaces the placeholder.

Follow these steps to make the installation of the zfoo configuration more flexible:

1. Prepare the configuration file and put the placeholder in it:

   ```
   distribution: <^AM^>get_distribution<^AM^>
   ```

   A macro placeholder is a string enclosed in "<^AM^>".

2. Upload the new version of the configuration file to the local inventory.
3. Create a macro file called get_distribution.sh, see Example 1.
4. Upload the macro file: In the macros category choose **Local** → **Add** from the context menu. Enter `get_distribution` in **Macro Name**. The name must correspond to the placeholder.

   Select the distributions and upload the file from the Director Console or a host, see "Adding undetected local software" on page 32. Press **Close** when you are done.

Whenever you install the new configuration file, the macro is executed to replace the placeholders.

Example 1 Macro get_distribution.sh

```
#!/bin/bash
#
# simple macro to detect the distribution
# depends on unmodified /etc/issue

SLES9="SUSE LINUX Enterprise Server 9"
RHEL4="Red Hat Enterprise Linux AS release 4"
UNKNOWN="^"

for DISTRO in SLES9 RHEL4 UNKNOWN; do
        eval SIGNATURE=\$$DISTRO
```

```
        cat /etc/issue | grep "$SIGNATURE" > /dev/null 2>&1
        if [ $? = 0 ] ; then
                echo $DISTRO
                exit 0
        fi
done
exit 1
```

Requirements for executable files

Executable files that you upload as a macro, action or probe must conform to these requirements:

1. For scripts make sure the shebang line is in place: The first line of a script must be the magic cookie “#!” followed by the path to the interpreter.
2. The exit value is checked and as long as it is zero, which means successful, the job goes on with the next task. Otherwise it would fail.

Actions

We used the term action before, when we talked about assigning settings like “required” or “not allowed” to components. In this section we refer to them as *software actions*. When we use the term *action*, we mean an executable file, either a binary or a script, that is uploaded to the local inventory.

There a two types of action: A pre- action runs before the software actions of a task, while a post-action runs after them.

Let’s make the update of the configuration file more user friendly. Maybe someone has done some changes to the configuration file you don’t want to overwrite without making a backup. Add the pre-action the following way:

1. Create a script file called backup_zfoo.sh, see Example 2.
2. Upload the file: In the Pre-actions category choose **`Local`** → **`Add`** from the context menu. Enter `backup_zfoo` in **Pre Action Name**.

   Select the distributions and upload the file from the Director Console or a host, see “Adding undetected local software” on page 32. Press **Close** when you are done.

You can add actions to the action list just like software actions by making them “required”. Setting “upgrade” or “not allowed makes” no sense for an action and is not possible. When an action is required in the action list, the Agent executes it when the job is deployed.

Example 2 Pre action backup_zfoo.sh

```
#!/bin/bash
#
# backup a possibly existing zfoo.conf

ZFOO_CONF=/etc/zfoo/zfoo.conf

if [ -f $ZFOO_CONF ]; then
        DATE=`date +%Y%m%d%H%M`
        mv $ZFOO_CONF ${ZFOO_CONF}.${DATE}.bak
        exit $?
fi
exit 0
```

Probes

Like an action, a probe is an executable file. It purpose is not to initialize or set up conditions for a job, but to check whether prerequisites are fulfilled. Again a exit value of zero indicates that the job can go on. Probes are executed before pre actions and even before the Dependency Resolver analyzes the task list.

Probes are uploaded and used in the same way as actions.

The command line interface

With the Command Line Interface you can use the power of Aduva OnStage from the command line and write scripts or interfaces for more automatization.

The program's name is **osc** and if you run it without parameters you get an overview of the commands you can use, see Example 3.

Example 3 Command overview

```
[akoch@z-ms4 ~]# osc
Usage: osc -command [Options]
Available commands:
aca:    Add Confirmation Policy Attribute
afd:    Add Target File Declaration
ag:     Add new Group of hosts
ah:     Add new Host
ahg:    Add Host to Group
alc:    Add Local Category
apa:    Add Profile Attribute
asp:    Add Local Software Package
atl:    Upload a local configuration file, macro, probe, pre-action, or post-action
cc:     Copy Confirmation Policy
chi:    Compare Hosts/Snapshots
cip:    Convert Inventory to Profile
cp:     Copy Profile
csp:    Convert Snapshot to Profile
dg:     Delete Group
dh:     Delete Host
dj:     Delete Job
dss:    Delete Saved Snapshot
exp:    Export command
fc:     Find Component
imp:    Import command
lah:    List All Hosts
lc:     List Confirmation Policies
lca:    List Confirmation Policy Attributes
ld:     List Distributions
lg:     List Groups
lgh:    List Group's Hosts
lh:     List Host Properties
lhi:    List Host Inventory
ljs:    List Jobs Status
ljsa:   List Job Status Attributes
ll:     List Logs
lp:     List Profiles
lpa:    List Profile Attributes
lss:    List Saved Snapshots for host/group
rg:     Rename Group
rh:     Rename Host
scj:    Submit Compare Job to make Target like Source
```

```
sgi:    Save Group Snapshot
shi:    Save Host Snapshot
sj:     Submit Job
```

Every command has options that tell Aduva OnStage what to do with which objects, for example, jobs, hosts, or profiles. To get information about options, simply run the command. See Example 4.

Example 4 Command options

```
[akoch@z-ms4 ~]# osc -lh
Failed preparing osc argument. Required h (Host name) missing.
lh:     List Host Properties

Command Line Arguments:
h:      Host name
[p]:    Admin password (if not given, will be requested interactively)
[u]:    Admin username (if not given, will be requested interactively)
```

Example 5 shows a command to get information about z-sds. You have to enter a user name and password to authenticate, but you can also supply them on the command line:

```
osc -lh -h z-ms1 -u admin -p ********
```

Example 5 Host information

```
[akoch@z-ms4 ~]# osc -lh -h z-sds
Initializing osc
Username: admin
Password:
Host Data :
Host Name = z-sds
Unique String = z-sds
Host IP = 9.12.4.231
Host Type is: standard Host
Host is Connected
Host registration time = Tue Sep 27 10:15:37 2005
Host inventory time stamp = Tue Sep 27 10:29:32 2005
```

With the background from the previous section you should be able to construct other commands. For more examples refer to the *Aduva OnStage Admin Guide*.

Note: The Command Line Interface logs into the System Dependency Server and if you are already logged in the Director Console with the same user name, your session is closed.

We recommend you create a separate user if you regularly run scripts that use **osc**. A user needs full privileges to run the Command Line Interface.

Restarting applications and reconfiguration

The installation process takes care of configuring and setting up all Aduva OnStage components. Nevertheless, you may want to change some of settings later.

Start scripts for the components

All applications that run as daemons start at boot time. To restart or stop them, use the init scripts in /etc/init.d:

- director_agent for the Agent.
- director_engine for the Dependency Manager.
- director_scheduler for the Aduva Server scheduler.
- director_server for the Aduva Server.

For example, if you want stop an Agent issue the following command as root at the command line:

```
/etc/init.d/director_agent stop
```

Replace “stop” with start, restart or status in order to start, restart or get a status information.

Reconfiguration of server and ports

All Aduva OnStage applications use a file called director.rc which contains the default configuration. Never change it directly. Copy the option you want to adjust to the .director.rc file and change it there.

You can reconfigure the proxy settings of the Aduva Server and the default ports. You must restart an application to make changes effective.

Configuration of the proxy settings

The path to the configuration file is /usr/local/aduva/director_server/cgi-bin/.director.rc

The parameters in Example 6 control settings that are used to contact the Universal Server via a HTTPS capable Web proxy.

Example 6 Proxy configuration

```
( all ) ( proxy_server_name , “proxy” );
( all ) ( proxy_server_port , 8080 );
( all ) ( proxy_user_name , “user” );
( all ) ( proxy_user_password , “password” );
```

To unset a proxy, replace the host name of the proxy server with a dash.

When you need to change authentication settings, refer to the *Aduva OnStage Admin Guide*, because you also need to change some files in /etc/director_server.

Configuration of the Aduva Server port

The listening port of the Aduva Server, which is an Apache based application, is configured in a separate file, /etc/director_server/httpd.conf. See Example 7 for the lines that define the listening port.

If you change the port in the Listen setting, you must also change the port associated with the VirtualHost setting. Don’t change these settings unless you really understand what you are doing and have some experience with Apache configuration. Keep a backup of the original file in case you run into problems.

Example 7 Listening port of the Aduva Server

```
...
Listen 8002
...
<VirtualHost _default_:8002>
...
```

If you have changed the port of the Aduva Server you must also adjust the server_port setting in all client configurations! See “Configuration of the clients” on page 39.

Configuration of the Dependency Manager port

To change the listening port of the Dependency Manager adjust the distrizor_port setting in /usr/local/aduva/director_engine/bin/.director.rc. See Example 8 for an extract from that file.

Example 8 Dependency Manager port settings

```
( all ) ( distrizor_host , "z-sds" );
( all ) ( distrizor_port , 8100 );
...
```

If you have changed the port of the Dependency Manager you must also change the setting distrizor_port in all client configurations! See “Configuration of the clients” on page 39.

Configuration of the clients

Consult the following list for the location of the configuration file for a client application:

Agent /opt/local/aduva/director_agent/bin/.director.rc

Command Line Interface
~/.director_cli/bin/.director.rc (user specific)
/usr/local/aduva/director_cli/bin/.director.rc (global)

Director Console
~/.director_console/bin/.director.rc (user specific)
/usr/local/aduva/director_console/bin/.director.rc (global)

Director Console MS Windows version
C:\director_console\bin\.director.rc (as long as the default path was used in the installation)

Example 9 Client configuration details

```
# settings for contacting the Aduva Server
( all ) ( server_name , "z-sds" );
( all ) ( server_port , 8002 );

# settings for contacting the Dependency Manager
( all ) ( distrizor_host , "z-sds" );
( all ) ( distrizor_port , 8100 );

# listening port for Agent
( all ) ( general.__a.agent_port, 8200 );
```

When you change the listening port of an Agent, the new settings are communicated to the Dependency Manager the next time the Agent registers itself. No further adjustment to the configuration of the Dependency Manager is necessary.

Conclusion

This Redpaper introduces you to the concepts and basic usage of Aduva OnStage and its advantage over traditional software configuration management. The power of Aduva OnStage becomes even more evident when you project the small examples we presented into a big and complex environment. In short, Aduva OnStage brings Linux administration to the next level and eases the processes of:

- Application deployment
- Configuration management
- Change management

- Patch control

Several topics are not discussed or only mentioned briefly:

- Reports provide information about distribution specific security incidents, extended job history and package compliance information.
- To extend Aduva OnStages capabilities up to whole lifecycle management you can use the additional Bare Metal Installation (BMI) plug-in. With BMI you can install predefined Linux systems on PXE boot capable Intel machines from scratch. Contact Aduva or one of its business partners for more information.
- Aduva distributes a driver for the integration with IBM Tivoli® Orchestrator, for more information visit:

  `http://www.aduva.com/index.php?page=integration_tivoli`
- The *Aduva OnStage User Guide* contains examples for deploying both WebSphere® and DB2® or implementing on demand computing with floating servers and load management. It also gives detailed information about the usage of the Command Line Interface and the specialties for managing Solaris hosts.

References

Aduva OnStage Admin Guide

Aduva OnStage User Guide

Glossary

Action When an Agent picks up a job the Dependency Resolver creates actions from the individual task. For example install, upgrade or uninstall software packages.

Aduva Certified Object (ACO)
A software package that is certified by the Aduva Certification Lab, which means it has been tested to ensure a successful and safe deployment.

Aduva Server A modified Apache Web server that runs as part of the System Dependency Server. It communicates with the Universal Server and fetches software packages and rules from it.

Agent Application running on a host. Provides information about the software configuration and manages the deployment of software packages on a host.

Channel A specific version of a distribution on a specific architecture supported by Aduva OnStage. The term distribution is used interchangeably in this Redpaper.

Component A logical unit in the inventory that can be managed, like software packages, files or hardware information.

Conflict Sometimes two software packages mustn't be installed together, because they overwrite files from each other or introduce bugs. This is called a conflict.

Director Console
The graphical user interface to manage Aduva OnStage.

Command Line Interface
The command line interface to manage Aduva OnStage.

Dependency A software package often needs other packages to be installed in order to work. This is called a dependency.

Dependency Manager
An application that is part of the System Dependency Server. It prepares and initiates jobs on Agents and monitors their status.

Dependency Resolver
When an Agent receives a job, the Dependency Resolver runs a set of patented algorithms to calculate the most effective way to fulfill the request. Rules from the Knowledge Base are used to be aware of dependencies and conflicts.

Host Aduva OnStage term for a managed server running the Agent.

Inventory The representation of components from a channel or host.

Job A set of tasks that is distributed to Agents in order to change the software configuration and work with components.

Knowledge Base
A part of the System Dependency Server that holds rules and ACOs fetched from the Universal Server. The Local Knowledge Base additionally contains the local components.

Local components
Local software and local files that are stored in the Local Knowledge Base.

Local expansion technology
A mechanism that creates rules for local software.

Local files Configuration files and macros, or probes and actions that are uploaded to the Local Knowledge Base.

Local software Third party or self developed software packages. They are stored in the Local Knowledge Base and can be managed with Aduva OnStage.

Non Certified Object (NCO)
Another name for local software. Indicates that they are not tested by the Aduva Certification Lab and use only the rules embedded in the RPM package.

Profile A definition of software packages that are required, should be upgraded or are not allowed. Used in a task to install, upgrade or uninstall software.

Policy Confirmation rules for the actions the Dependency Resolver creates out of a profile.

Rules Also referred to as dependency rules or deployment rules. Individually created by Aduva Certification Lab for every software package they represented extended knowledge about dependencies and conflicts.

Software package
For Linux hosts usually an RPM package that incorporates an application, library or documentation that can be installed on a host.

System Dependency Server
The server application that consists of the Aduva Server, the Dependency Manager and the Knowledge Base.

Task A combination of a profile, a set of hosts and the policy that sets the confirmation rules for the resulting actions.

Universal Server

Located at knowledge.aduva.com it holds the master Knowledge Base against the System Dependency Server is synchronized.

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Alexander Koch is an IT-Consultant at the Millenux GmbH, the Aduva business partner for Germany. He is finishing his diploma in Computer Science with a major in Media at the University of Applied Sciences, "Hochschule der Medien" in Stuttgart.

Erich Amrehn is a Certified Senior IT Specialist at the EMEA Technical Marketing Competence Center (TMCC), Boeblingen, Germany. Before joining the TMCC, he worked as a Project Leader at the International Technical Support Organization, Poughkeepsie Center. During that time, he wrote Redbooks™ and taught Linux topics worldwide. Before joining the ITSO in 1998, he worked as a Technical Consultant to the IBM S/390® division for e-commerce on S/390 in Europe, the Middle East, and Africa. He also has 13 years of VM experience in various technical positions in Germany and other areas in Europe and worldwide.

Thanks to the following people for their contributions to this project:

Gregory Geiselhart
International Technical Support Organization, Poughkeepsie Center

Michael Schrenk
Aduva

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.

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 may 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.

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 illustrates 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.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®	IBM®	System z9™
iSeries™	OpenPower™	System/390®
pSeries®	PowerPC®	Tivoli®
zSeries®	Redbooks™	WebSphere®
z9™	Redbooks (logo) ™	
DB2®	S/390®	

The following terms are trademarks of other companies:

Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

i386, Intel, Pentium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.