Integrating Applications with Linux on zEnterprise

Version 02
## Contents

### About this document
- Who this document is for
- Other Linux on System z publications
- Restrictions and troubleshooting

### Part 1. Installing and configuring

#### Chapter 1. Introduction to the application integration technology
- Attaching a Linux x86 virtual server
- Integrating file systems
- Process monitoring
- Synchronizing RPMs
- Managing users and groups

#### Chapter 2. Installing the technology
- Prerequisites
- Installing the RPMs
- Installing updates

#### Chapter 3. Setting up the technology
- Setting up port forwarding and masquerading
- Integrating logs
- Setting up matching user accounts
- Setting up X11 forwarding
- Attaching a Linux instance running on an x86 virtual server to the mainframe

#### Chapter 4. Managing x86 virtual server packages from the host
- Creating an RPM meta package repository
- Synchronizing RPM packages
- Remotely install and remove x86 packages

### Part 2. Administrator’s Guide

#### Chapter 5. Managing users
- Setting the user management system
- Adding users

#### Chapter 6. Using the attached resources
- Setting up programs or scripts to run on the attached resources
- Running programs or scripts on the attached resources
- Starting a script with ai_exec

#### Chapter 7. Integrating remote processes

### Part 3. Reference

#### Chapter 8. The configuration file
- Title section
- Statements for sharing directories
- Keywords for handling scripts and programs
- Logging statement
- Keywords for networking

© Copyright IBM Corp. 2011
About this document

This document describes how to use the application integration technology in an integrated server environment. It also contains a reference to the technology commands and error messages.

In this document, System z is taken to mean IBM® zEnterprise™ 196 (z196) and zEnterprise 114 (z114) or later. x86 refers to 64-bit IBM System x® servers.

Unless stated otherwise, all z/VM® related information in this document assumes a current z/VM version, see www.ibm.com/vm/techinfo/.

You can find the latest version of this document on developerWorks® at: www.ibm.com/developerworks/linux/linux390/applint.html

Who this document is for

This document is for administrators who want to run and control applications in a zEnterprise System environment.

Other Linux on System z publications

Several Linux on System z publications are available on developerWorks.

You can find the latest versions of these publications at www.ibm.com/developerworks/linux/linux390/documentation_dev.html
• Device Drivers, Features, and Commands, SC33-8411
• Using the Dump Tools, SC33-8412
• How to Improve Performance with PAV, SC33-8414
• How to use FC-attached SCSI devices with Linux on System z, SC33-8413
• How to use Execute-in-Place Technology with Linux on z/VM, SC34-2594
• How to Set up a Terminal Server Environment on z/VM, SC34-2596
• Kernel Messages
• libica Programmer's Reference, SC34-2602

Restrictions and troubleshooting

For restrictions and troubleshooting information, see the respective text files delivered with the application integration technology.
Part 1. Installing and configuring

This part describes how to install the technology and how to configure it. It is assumed that you have an understanding of Linux and System z® terminology.
Chapter 1. Introduction to the application integration technology

This section gives an overview of the concepts and tools constituting the application integration technology. The technology is an infrastructure that integrates x86 Linux applications and System z applications in a hybrid zEnterprise System. The technology manages a hybrid software workload from a Linux on System z instance.

The technology reduces the complexity of such a hybrid environment by presenting all applications as running on a single system. You can then manage and exploit the integrated x86 Linux applications from within Linux on System z. You manage the x86 Linux system resources (such as CPU, memory, binaries, and file systems) through commands from the Linux on System z instance. Conceptually, these resources seem to be part of Linux on System z. This environment retains any x86 Linux certifications while leveraging the operational environment of Linux on System z.

System administration tasks comprise attaching an x86 virtual server, synchronizing RPMs, managing users and groups, and managing applications. Figure 1 illustrates these tasks.

Table 1 on page 4

Figure 1. System administration tasks

A x86 virtual server is a logical construct that consists of processor, memory, and I/O resources. You define a x86 virtual server through the zManager program on the zEnterprise System. The technology supports x86 virtual servers that run Linux. For more details about virtual servers, see the zEnterprise System documentation.

Attaching a Linux x86 virtual server

In the technology, if an x86 operating system image is integrated into a Linux on System z image, it is designated attached.

The technology provides commands to manage attaching x86 operating system images. These commands are summarized in Table 1 on page 4.
Table 1. Attachment commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Environment</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ai_attachable - determine if an x86 virtual server can be attached&quot; on page 51</td>
<td>Linux on x86</td>
<td>Indicates whether an x86 operating system instance can be attached to a Linux on System z instance.</td>
</tr>
<tr>
<td>&quot;ai_att - connect an x86 virtual server&quot; on page 49</td>
<td>Linux on System z</td>
<td>Attaches an x86 operating system instance to Linux on System z and shows status information.</td>
</tr>
<tr>
<td>&quot;ai_usm - manage users&quot; on page 63</td>
<td>Linux on System z</td>
<td>Sets the user management master system.</td>
</tr>
<tr>
<td>&quot;ai_det - detach an x86 virtual server&quot; on page 52</td>
<td>Linux on System z</td>
<td>Detaches x86 operating system instances from Linux on System z.</td>
</tr>
</tbody>
</table>

An x86 Linux instance can be attached to a Linux on System z instance only if the technology is installed on both systems. For additional requirements that both images must satisfy, see Chapter 3, “Setting up the technology,” on page 11.

As soon as an x86 operating system instance is attached to a Linux on System z instance, the users of the Linux on System z instance can use the resources of the x86 Linux instance. Managing the attachment of a virtual server is therefore disruptive. You must be an administrator (root) to change the attachment state. You can only detach an x86 Linux instance if no x86 resources are in use from Linux on System z.

The following sections describe how attached x86 Linux instance resources are represented in Linux on System z and how they are used.

Integrating file systems

When attaching an x86 Linux instance, the file system of the x86 operating system instance is made available to the Linux on System z users.

The attachment is controlled by the /etc/ai/aiz_profiles.conf file. In this file you can configure the System z shares to be exported to the x86 virtual server. See also Chapter 8, “The configuration file,” on page 41.

By default, the root file system of the attached x86 operating system instance is mounted to /x86/<virtual-server-hostname> in the Linux on System z file system, where <virtual-server-hostname> is the x86 virtual server host name as specified in the ai_att command. Linux on System z users can then access all files related to the x86 operating system instance.

Full integration requires that certain parts of the Linux on System z file system are accessible to the attached x86 operating system instance. The choice of which parts of the file system are shared between the two images (for example /home/) can be configured through the /etc/ai/aiz_profiles.conf file. On attachment, the specified directory is shared automatically, and on detachment sharing is stopped automatically (see Figure 2 on page 5). Sharing directories uses NFS as the base technology. NFS servers and clients are required on both the Linux on System z instance and the x86 Linux instance.
In summary, the file system integration between attached x86 operating system instances and Linux on System z images is implemented using NFS and is completely automated through the attachment process (ai_att and ai_det).

**Process monitoring**

Every process started on an attached x86 virtual server is represented on Linux on System z by a *shadow process*.

Shadow processes enable:
- Interaction with the processes running on an attached x86 virtual server as if they were running on the host.
- Monitoring of the processes running on an attached x86 virtual server from the host.

**Note:** The observable memory and CPU load visible in the /proc file system of Linux on System z reflects the CPU and memory consumption of the shadow process; not of the x86 process.

There is a virtual /proc file system for the attached x86 operating system image, accessible through /x86/proc_<virtual-server-hostname>. This contains and mirrors specific x86-related process monitoring information.

Additionally, another /proc file system consolidates the /proc file system information provided by Linux on System z and the attached x86 operating system images. You can access this /proc file system, accessible through /x86/proc, with a command, ai_ps (see "ai_ps - report the status of the current processes" on page 56).

A delay for using the file system elements of x86 nodes is introduced by the network link between Linux on System z and the x86 virtual server.
Synchronizing RPMs

You can manage software packages on the x86 virtual server from the technology host. To do this, synchronize RPM packages between the technology host and the x86 virtual server.

You can set up a repository that mirrors the RPM packages installed on the x86 virtual server. Using a repository helps to resolve dependencies automatically.

For details about synchronizing software packages, see Chapter 4, “Managing x86 virtual server packages from the host,” on page 19.

Managing users and groups

You can manage users centrally from the host or from the attached x86 virtual server.

To manage users, appoint one system in the hybrid environment the user management master. You can change the user management system, but always use the designated system to change or add users. Using the user management master ensures that the users and groups on both systems are synchronized.

For details, see Chapter 5, “Managing users,” on page 27.
Chapter 2. Installing the technology

To set up your environment you need to install an RPM on both the mainframe Linux instance and the Linux instance running on the x86 virtual server.

Prerequisites

The technology packages must be installed on both the System z Linux instance and the x86 virtual server instance. Also, both images must meet a set of requirements concerning the operating systems versions and regarding the connectivity between each other.

System z prerequisites

Before you can use the technology, you must install it on your system. You require:

• An operating system, one of:
  – SUSE Linux Enterprise Server 11 SP 1, service level 2.6.32.23-0.3.1 or later
  – Red Hat Enterprise Linux 6.1

The System z and x86 virtual server Linux distributions must be the same version from the same distributor. For example, if the host Linux instance is SUSE Linux Enterprise Server 11 SP 1, then the x86 virtual server Linux instance must also be SUSE Linux Enterprise Server 11 SP 1.

• The technology RPM for your Linux distribution
• On SUSE Linux Enterprise Server 11:
  – libn curses
  – fuse and libfuse2
• On Red Hat Enterprise Linux 6.1:
  – ncurses-lib
  – fuse-libs
• Network connectivity
• An NFS server and client

x86 virtual server prerequisites

• An operating system, one of:
  – SUSE Linux Enterprise Server 11 SP 1
  – Red Hat Enterprise Linux 6.1

The System z and x86 virtual server Linux distributions must be the same version from the same distributor.

• The technology RPM for your Linux distribution
• Network connectivity
• An NFS server and client

Restriction:

In the first release, the technology supports the attachment of only one x86 virtual server.
Installing the RPMs

Use your package manager to install the RPM packages on the host and on the x86 virtual server.

Before you begin

To install a technology RPM package on the x86 virtual server, you need an SSH session to it.

About this task

System z and x86 virtual server Linux distributions must be the same version from the same distributor. For example, if the host Linux instance is SUSE Linux Enterprise Server 11 SP 1, then the virtual server Linux instance to be attached must also be SUSE Linux Enterprise Server 11 SP 1.

The example in the following section shows an installation using SUSE Linux Enterprise Server 11 SP1. You can install the technology RPMs on the host and the x86 virtual server in any order.

Procedure

Perform the following steps to install the technology RPMs on a virtual server and the host.

1. Obtain the RPMs. You can obtain the technology RPM packages at [http://www.ibm.com/developerworks/linux/linux390/applint.html](http://www.ibm.com/developerworks/linux/linux390/applint.html). They have names of the following form:

   applint-<version>-<distribution>-<platform>.rpm

   For example:

   applint-0.2-33.sles11.1.x86_64.rpm

2. Issue an RPM command to install the package on the virtual server.

   **Example:** The following RPM command installs the SUSE Linux Enterprise Server 11 SP1 RPM applint.

   ```
   [aiatt0]# rpm -ivh applint-0.2-33.sles11.1.x86_64.rpm
   Preparing... ########################################### [100%]
   1:applint ########################################### [100%]
   Starting applint...
   Starting xzdproc:  done
   Starting xzd daemon:  done
   Starting xzdwatchd:  done
   ```

   To install dependencies, note the output of the `rpm` command and install any dependencies, for example, using `zypper`:

   zypper install <dependent RPM>

   **For example:**

   ```
   [aiatt0]# zypper install nfs-kernel-server
   ```

   After installation, a service called `applint` is installed on the x86 virtual server. To check that the installation has completed correctly, ascertain that the `applint` service is running:
If the service was not started automatically, start it manually.

3. Install the corresponding RPM on the host.

   Example: The following RPM command installs the SUSE Linux Enterprise Server 11 SP1 applint RPM on the host.

   ```bash
   [aihost]\# rpm -ivh applint-0.2-33.sles11.1.s390x.rpm
  Preparing... ########################################### [100%]
   1:applint ########################################### [100%]
   applint 0:off 1:off 2:off 3:off 4:off 5:off 6:off
   ``)

   After installation the `applint` service is available, but it is not started.

   ```bash
   [aihost]\# service applint status
   Checking for applint service:
   xzdhbd unused
   xzdpsfd unused
   xzdwatchd unused
   ``)

   Do not start the service. The `applint` service starts automatically when a x86 virtual server is attached, and stops when the x86 virtual server is detached.

4. On the x86 virtual server, issue the `ai_attachable` command to check if the Linux instance is ready for attachment. For example:

   ```bash
   [aiatt0]\# ai_attachable
   ai_attachable: x86 virtual server can be attached to IBM System z.
   ``)

Results

After installation, a service called `applint` is installed on both the host and the x86 virtual server. The service is running on the x86 virtual server and is stopped on the host. An NFS server and client are installed on both the host and the x86 virtual server.

This is shown in Figure 3.
Installing updates

Keep your installation up-to-date by installing updates on both the host and the x86 virtual server.

About this task

Try to keep the operating system on the host and the virtual server at the same level.

Procedure
1. Detach the x86 virtual server.
2. Install the updates on both the host and the virtual server separately
3. Attach the x86 virtual server again.
Chapter 3. Setting up the technology

After installation you can prepare the environment for the technology.

Set up entails:

- Setting up port forwarding and masquerading.
- Integrating logs on page 13
- Setting up matching user accounts on page 13.
- Setting up X11 forwarding on page 16.
- Attaching a Linux instance running on an x86 virtual server to the mainframe on page 16.

Setting up port forwarding and masquerading

Optionally, if a network address translation setup is advantageous to your workload, you can configure port forwarding and masquerading.

Before you begin

- A network connection must be set up between host and the x86 virtual server.
- You must have installed an NFS client and an NFS server on both the System z host and on the x86 virtual server.
- Your firewall settings must allow NFS server ports on the mainframe and the x86 virtual server.
- The firewall on the x86 virtual server must allow the technology daemon connections. The default ports are 23100/tcp and 23101/tcp.
- The x86 virtual server and the host must be configured in the same IP subnet.

About this task

In this task you will first work on the x86 virtual server to set up a network. Then you will work on the host, using the /etc/ai/aiz_profiles.conf file to configure the connection.

See Chapter 8, “The configuration file,” on page 41 for more details about the configuration file.

The example used in the following assumes that the x86 virtual server will communicate with the host through a private network, see Figure 4 on page 12.

Procedure

Perform the following steps to set up port forwarding and masquerading.

1. Specify the default gateway for the x86 virtual server. On the x86 virtual server, issue a command of the form:

   [aiatt0]# route add default gw <aihost-IP> <if-name>

   where <aihost-IP> is the IP address of a network interface at the host, and <if-name> is the interface name. For example:

   [aiatt0]# route add default gw 1.2.3.4 eth1
2. Activate port forwarding and masquerading. On the host, edit
/etc/ai/aiz_profiles.conf and specify network address translation (NAT).

Change the following line
NETWORK_CONFIG="none"

to
NETWORK_CONFIG="nat"

Specifying “nat” activates the network address translation from the external
interface to the internal interface and vice versa on the System z host.

3. Disclose the external LAN interface on the host. On the host, edit
/etc/ai/aiz_profiles.conf and specify the interface name. For example:
NETWORK_NAT_IF_HOST_EXT="eth0"

4. To disclose the internal LAN interface on the host. On the host, edit
/etc/ai/aiz_profiles.conf and specify the interface name. For example:
NETWORK_NAT_IF_HOST_INT="eth1"

5. To make the host aware of the internal LAN interface of the virtual server. On
the host, edit /etc/ai/aiz_profiles.conf and specify the interface name. For
example:
NETWORK_NAT_IF_BLADE="eth1"

6. For each service or port to be forwarded, add (using a unique suffix \( n \)) a
statement of this form: NETWORK_NAT_FW_PORT_\( n \)="<host port or
service>/<transport> <blade port or service>" where

<host port or service>
Is the port or service at the host.

<transport>
Is the optional transport protocol with the default “tcp”.

<blade port or service>
Is the port or service at the x86 virtual server.

For example:
NETWORK_NAT_FW_PORT_1="2222/tcp 22"

**Results**

From the statements in the configuration file, firewall rules are generated and
installed during x86 virtual server attachment.

![Network setup result](image)
Integrating logs

You can forward log messages issued on the x86 virtual server to a log file on the host.

About this task

Messages are kept on the x86 virtual server to enable problem determination and debugging at the x86 virtual server. The message forwarding is active for the duration of the attachment.

Procedure

1. Edit the /etc/ai/aiz_profiles.conf configuration file.
2. Find the statement for converged logging:
   
   ```
   # Collect messages from virtual servers on the host /var/log/...
   ConvergedLogging = "no"
   ```
3. Change the entry for converged logging to "yes":
   
   ```
   ConvergedLogging = "yes"
   ```
4. Attach the x86 virtual server to activate the changes.

Results

A message log file called /var/log/<virtual_server>_logfile is created, where <virtual_server> is the x86 virtual server host name as specified in the ai_att command.

Setting up matching user accounts

The user accounts for the host and the x86 virtual server must correspond.

Before you begin

- The x86 virtual server must not be attached. This is to ensure that the locking mechanism of the /etc/passwd and /etc/group files is not active.
About this task

Attaching an x86 virtual server to a host requires that the user and group attributes are identical on both systems. This is best done on the System z Linux instance before attaching an x86 virtual server.

Procedure

Use the Python script `aizchketc.py` on the host to verify the synchronization of the user and group attributes. Perform the following steps:

1. Invoke the script to compare the `/etc/passwd` file, using the remote file as base, and checking all account attributes.
   Issue a command of the following form:
   ```bash
   [aihost]# aizchketc.py -t <tcpip-addr> -p -rd
   ```
   where `<tcpip-addr>` is the IP address of the x86 virtual server.

2. Compare the `/etc/passwd` file, using the local file as base, and checking all account attributes.
   Issue a command of the following form:
   ```bash
   [aihost]# aizchketc.py -t <tcpip-addr> -p -ld
   ```
   where `<tcpip-addr>` is the IP address of the x86 virtual server.

3. Compare the `/etc/group` file, using the remote file as base, and checking all group attributes.
   Issue a command of the following form:
   ```bash
   [aihost]# aizchketc.py -t <tcpip-addr> -g -rd
   ```
   where `<tcpip-addr>` is the IP address of the x86 virtual server.

4. Compare the `/etc/group` file, using the local file as base, and checking all group attributes.
   Issue a command of the following form:
   ```bash
   [aihost]# aizchketc.py -t <tcpip-addr> -g -ld
   ```
   where `<tcpip-addr>` is the IP address of the x86 virtual server. Any conflicting user IDs and group IDs must be manually changed.

5. Change all user IDs (`<u_old>`) and group IDs in the passwd and group files of the x86 virtual server to new values that match the values of the host user ID, `<u_z>`.
   a. Issue a command of the following form to change the user IDs:
   ```bash
   [aihost]# find / -uid <u_old> > /tmp/<tempfile>
   ```

   b. Issue a command of the following form to change the group IDs:
   ```bash
   [aihost]# find / -gid <g_old> > /tmp/<tempfile>
   ```
You can now check the list of files returned by the find command before changing the ownership of them in the next step.

6. Change file ownership.
   a. Issue a command of the following form to change ownership of the user ID:
   
   ```
   [aihost]# cat /tmp/<tempfile> | xargs chown <u_z>
   ```

   b. Issue a command of the following form to change ownership of the group ID:
   
   ```
   [aihost]# cat /tmp/<tempfile> | xargs chown <g_z>
   ```

Example

The following sample shows how to synchronize the user lists when a user ID testusz initially exists only on the host and a user ID testusx exists on an x86 virtual server named aiatt0, both with the same uid value 1006.

```
[aihost]# aizchketc.py -t 1.2.3.4 -p -rd
Password:
--- Start verification of local and remote passwd and group files ---
using target: 1.2.3.4 file1: /etc/passwd file2: /tmp/passwd
--- check conflicting entries:
testusx testusz have id: 1006
--- conflicting entries: 1
invoke: adduser testusx -M -u 1006 -g 100 -d /home/testusx -c "User mismatch" -s /bin/bash
--- 1 item(s) are missing: testusx, on local system
--- different entries:
--- different entries: 0
```

The aizchketc script identifies testusx and testusz as conflicting entries. To correct this, do the following:

1. Make the testusz user ID unique and mark files owned by testusz with the new uid.
   
   ```
   [aihost]# usermod -u 1007 testusz
   [aihost]# find / -uid 1006 > /tmp/uid.1006
   [aihost]# cat /tmp/uid.1006 | xargs chown 1007
   ```

2. Define the missing user ID on the host (System z):
   
   ```
   [aihost]# useradd testusx -M -u 1006 -g 100 -d /home/testusx -c "User mismatch" -s /bin/bash
   ```

3. Run the aizchketc script again to identify the missing user ID on the x86 virtual server userlist:
Setting up X11 forwarding

If the applications you want to run on the x86 virtual server are X11 clients, you must configure the host to handle remote X11 requests.

Procedure
1. Edit the configuration file /etc/ssh/sshd_config to configure the host. Set the following variables:
   a. X11Forwarding to “yes”
   b. X11UseLocalhost to “no”
2. Restart the SSH deamon.

Attaching a Linux instance running on an x86 virtual server to the mainframe

After successful installation on both the mainframe and the x86 virtual server, a Linux instance running on an x86 virtual server can be attached to a Linux instance running on the mainframe. As long as an x86 virtual server is attached to the mainframe, the technology packages cannot be removed from either system.

Before you begin
- You must have installed the technology service applint on System z and on the x86 virtual server. After installation, the service is in a stopped state on System z, and running on the x86 virtual server.
- The Linux instance running on System z and the instance running on the x86 virtual server must be the same Linux distribution and version.
- A network connection is required between System z and the x86 virtual server. The firewall settings have to be set to allow for NFS server ports on both hosts. In addition, the x86 firewall must allow for daemon connections. Default ports are 23100/tcp and 23101/tcp.
About this task

After successful installation on both architectures, an x86 virtual server can be attached to a System z system.

Procedure

Perform the following steps to attach an x86 virtual server.

On the Linux instance running on System z, issue the `ai_att` command:

```
[aihost]# ai_att <virtual-server-hostname>
```

where `<virtual-server-hostname>` can be an IP address or a resolvable host name.

The attach process is controlled by the `aiz_profiles.conf` file. For details about the `ai_att` command see the man page.

**Example:** To attach virtual server `aiatt0` to a host named `aihost`, issue:

```
[aihost]# ai_att aiatt0
Mounting remote procs to /x86/proc_aiatt0... Done.
Mounting combined procs to /x86/proc... Done.
Starting applinet service:
  Starting xzdpsd: done
  Starting xzdwatchd: done
  Attaching... done
x86 virtual server successfully attached.
```

Results

You can inspect the x86 virtual server file system that was mounted during the attachment process:

```
[aihost]# cd /x86/
[aihost:/x86]# ls -al
```

```
total 28
drwxr-xr-x 22 root root 4096 2011-04-01 13:47 ..
drwxr-xr-x 2 root root 4096 2011-03-25 09:42 bin
drwxr-xr-x 10 root root 4096 2010-09-30 13:35 lib
drwxr-xr-x 2 root root 4096 2011-03-25 09:42 proc
```

You can also use the `ai_att` command with the `--details` option to display the resulting configuration, for example:

```
[aihost]# cd /x86/
[aihost:/x86]# ai_att --details
```

```
Mounting remote procs to /x86/proc_aiatt0... Done.
Mounting combined procs to /x86/proc... Done.
Starting applinet service:
  Starting xzdpsd: done
  Starting xzdwatchd: done
  Attaching... done
x86 virtual server successfully attached.
```
The /x86 folder is the base folder for the x86 virtual server file system and contains the following:

- aiatt0 is the x86 virtual server root file system mounted on the host.
- proc_aiatt0 contains process monitoring information from the x86 virtual server provided by the technology environment. It can be viewed with the technology tools, for example ai_top.
- proc contains consolidated process monitoring information of the attached x86 virtual server and the host.
- aihost is a folder provided by the host for exchange of data with the attached x86 virtual server.
- Other folders are reserved by the technology.

As soon as a Linux instance running on an x86 virtual server is attached to a particular Linux on System z instance, the users of the Linux on System z instance can use the resources that belong to the x86 virtual server Linux instance.

To detach an attached x86 virtual server, use the ai_det command. See "ai_det - detach an x86 virtual server" on page 52 for details.
Chapter 4. Managing x86 virtual server packages from the host

Synchronizing the RPM packages enables you to manage virtual server packages from the host. First create a repository of RPM meta packages on the host that mirrors the RPMs installed on the x86 virtual server. Meta packages can be installed on the System z host and contain x86 packages that are remotely installed on the attached virtual server. Then synchronize the installed x86 virtual server packages with the host. Finally, you can remotely install and remove packages on the attached x86 virtual server from the host with automatic dependency resolution.

About this task

The synchronization entails:
1. "Creating an RPM meta package repository"
2. "Synchronizing RPM packages" on page 21
3. "Remotely install and remove x86 packages" on page 21

Creating an RPM meta package repository

Create a repository of RPM meta packages that mirrors the RPM packages installed on the attached x86 virtual server.

Before you begin

- The repository must include all x86 virtual server RPM packages in one or more directories (for example /path/to/X86_RPMS). You can create the repository without attaching an x86 virtual server.
- The package "createrepo" needs to be installed to create a new repository on the host.

About this task

You can set up the host to mirror the attached x86 virtual server package configuration. Installation and removal of packages on the attached x86 virtual server can then be kept in sync with meta packages on the System z host.

Procedure

1. Create a directory for the meta package repository on the host. For example, /path/to/repo).
2. Convert the x86 RPM packages to meta packages.
   a. Create the subdirectory /path/to/repo/s390x
   b. Use the following command to create meta packages for all x86 RPMs:

   ```bash
   [aihost]# ls /path/to/X86_RPMS/*.rpm | xargs ai_rpm -p -o /path/to/repo/s390x
   ```

   The x86 RPMs in /path/to/X86_RPMS must be accessible from the Linux instance running on the host. Depending on the number of x86 RPM files, creating the meta packages can take some time. After completion, the directory /path/to/repo/s390x contains one meta package for each x86 RPM.
When creating meta packages for the installation media specific to your distribution, you must use all x86 RPM files provided by your distributor. Repeat step 2 on page 19 for every architecture-specific subdirectory, for example, noarch/, i686/, or x86_64/. This ensures that all meta RPMs are created in the output directory </path/to/repo>/s390x.

3. Write repository metadata to the </path/to/repo>/repodata directory for each meta package.

   [aihost]# createrepo </path/to/repo>

4. Add the newly created repository to the package management tool specific to your distribution.

   • On SUSE Linux Enterprise Server 11 SP1:
     a. Run the zypper command to add the repository:

        [aihost]# zypper addrepo -t rpm-md -n "Meta repo" </path/to/repo> AI-metarepo

     b. Refresh the status of the newly added repository. Issue a command as follows:

        [aihost]# zypper refresh

        When refreshing the repository, you will be notified that its metadata is not signed. You can continue anyway, or you can sign it yourself. To check if the new repository is available within zypper, run the following command and examine the output:

        [aihost]# zypper repos

        # | Alias | Name | Enabled | Refresh
        +------------------------+-----------------+---------+--------
        1 | AI-metarepo | Meta repo | Yes | No

   • On Red Hat Enterprise Linux 6.1:
     a. Create a file named, for example, /etc/yum.repos.d/AI-metarepo.repo with the following content:

        [AI-metarepo]
        name=Meta repo
        baseurl=file://</path/to/repo>
        enabled=1
        gpgcheck=0

     b. To update the status of the new repository, run the following command and examine the output for the name of the repository:

        [aihost]# yum repolist enabled

        repo id repo name status
        AI-metarepo Meta repo 3,542

        When the meta package repository is known to the package management tool, new packages can be installed on the attached x86 virtual server by starting the installation of the corresponding meta package on the host. The deinstallation of meta packages on the host removes the corresponding packages from the attached x86 virtual server.

        To manage the technology meta packages, use the package management tools specific to your distribution, for example:

        • zypper for SUSE Linux Enterprise Server 11 SP1.
Synchronizing RPM packages

Retrieve a list of installed RPMs from an x86 virtual server, then install the corresponding meta RPM packages on the host.

About this task

You can keep the packages installed on the virtual server and the meta packages synchronized. Do this before installing new packages on the x86 virtual server from the meta-RPM package repository. This way dependencies are resolved automatically based on the meta-RPM package repository.

Procedure

1. Attach the x86 virtual server with which you synchronize the host (see “ai_connect an x86 virtual server” on page 49).
2. Edit the file /etc/ai/ai_syncrpms.exclude and add one x86 package name per line for all packages that shall not be mirrored by meta packages on the host.
   
   **Note:** The file already contains the name of the application integration RPM, as well as a placeholder for GPG public keys that cannot be mirrored. These entries must not be deleted.
   The package names already include the AI-<architecture> identifier, so the returned list can be used immediately with the xargs command.
3. Now execute the synchronization script ai_syncrpms. The script retrieves a list of all installed x86 packages from the attached virtual server and installs the corresponding meta packages on the System z host. For example:

   ```
   [aihost]# ai_syncrpms
   <<> meta packages are going to be installed.
   [S]how list of packages, [c]ontinue with installation, or [a]bort?
   ```

   By pressing the corresponding key, you can view a list of all packages that will be installed, continue with the synchronization, or end the process. Depending on the number of installed x86 packages, the synchronization process can take some time.

Results

After completion, every package already installed on the attached x86 virtual server is represented by an installed meta package on the host. During the installation you will see messages about the remote installation failing. These are to be expected, and confirm that you are indeed installing meta packages for the existing RPMs.

Remotely install and remove x86 packages

After setting up the meta package repository, you can remotely install and remove x86 packages on the attached virtual server.

Before you begin

When installing or removing packages that creates or modifies user or group accounts, remember to appoint the attached virtual server the user management system before starting. Otherwise the action fails or does not complete correctly.
**About this task**

During remote installation of new packages on the attached virtual server, dependencies between the meta packages are resolved automatically and get installed together with the requested package. When removing x86 packages from the host, dependencies are also recognized and get uninstalled together with the requested package. Use the package management tools specific to your distribution, for example:

- **zypper** for SUSE Linux Enterprise Server 11 SP1.
- **yum** for Red Hat Enterprise Linux 6.1.

**Procedure**

To remotely install an x86 package including its dependencies on the attached virtual server:

- **On SUSE Linux Enterprise Server 11 SP1** use the following command:

```
$ zypper install soprano-AI-x86_64
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW packages are going to be installed:
  clucene-core-AI-x86_64 libraptor1-AI-x86_64 librasqal1-AI-x86_64
  libredland0-AI-x86_64 libsoprano4-AI-x86_64 soprano-AI-x86_64

The following packages are not supported by their vendor:
  clucene-core-AI-x86_64 libraptor1-AI-x86_64 librasqal1-AI-x86_64
  libredland0-AI-x86_64 libsoprano4-AI-x86_64 soprano-AI-x86_64

6 new packages to install.
Overall download size: 1.2 MiB. After the operation, additional 1.1 MiB will be used.
Continue? [y/n/?] (y): 

... 
Retrieving package soprano-AI-x86_64-2.3.0-4.1.92.s390x (6/6), 104.0 KiB (101.0 KiB unpacked)
Installing: soprano-AI-x86_64-2.3.0-4.1.92 [done]
Additional rpm output:
  Installing soprano-2.3.0-4.1.92.x86_64.rpm on attached virtual server... Done.
```

- **On Red Hat Enterprise Linux 6.1** use the following command:
$ yum install soprano-AI-x86_64

... Dependencies Resolved

Package Arch Version Repository Size
==============================================================================
Installing:
soprano-AI-x86_64 s390x 2.3.1-1.2.el6 AI-metarepo 546 k

Installing for dependencies:
clucene-core-AI-x86_64 s390x 0.9.21b-1.el6 AI-metarepo 284 k
libxml-t-AI-x86_64 s390x 1.1.26-2.el6 AI-metarepo 459 k
postgresql-libs-AI-x86_64 s390x 8.4.7-2.el6 AI-metarepo 188 k
qt-AI-x86_64 s390x 4.6.2-17.el6 AI-metarepo 4.0 M
raptor-AI-x86_64 s390x 1.4.18-5.el6 AI-metarepo 203 k
rasqal-AI-x86_64 s390x 0.9.15-6.1.el6 AI-metarepo 178 k
redland-AI-x86_64 s390x 1.0.7-11.el6 AI-metarepo 197 k

Transaction Summary
==============================================================================
Install 8 Package(s)

Total download size: 6.0 M
Installed size: 6.0 M
Is this ok [y/N]: y

... Installed:
soprano-AI-x86_64.s390x 0:2.3.1-1.2.el6

Dependency Installed:
clucene-core-AI-x86_64.s390x 0:0.9.21b-1.el6
libxml-t-AI-x86_64.s390x 0:1.1.26-2.el6
postgresql-libs-AI-x86_64.s390x 0:8.4.7-2.el6
qt-AI-x86_64.s390x 0:4.6.2-17.el6
raptor-AI-x86_64.s390x 0:1.4.18-5.el6
rasqal-AI-x86_64.s390x 0:0.9.15-6.1.el6
redland-AI-x86_64.s390x 0:1.0.7-11.el6

Complete!

You might see messages about meta packages not being supported. For either
distribution, you can safely ignore them.

Results

After completion, the requested x86 package including its dependencies is installed
on the attached virtual server, and each of the packages is represented by a meta
package on the System z host. During removal of meta packages, dependencies
are also recognized.
Administration tasks for the technology include attaching and detaching an x86 virtual server, installing and uninstalling RPMs on the virtual server, and managing user and group accounts.
Chapter 5. Managing users

After installing the RPMs and attachment of an x86 virtual server to the host, you can define a user management system and add, delete, and change user IDs. Using a user management system ensures that user IDs are kept synchronized on the host and the attached x86 virtual server. The application integration technology uses the `/etc/passwd` file for user management.

Managing users includes:
- "Setting the user management system" on page 13
- "Setting up matching user accounts" on page 13
- "Adding users" on page 28

Setting the user management system

In the technology environment, user and group entries must be the same on the host and the x86 virtual server. The existing users and groups and their attributes must be identical. Use the `ai_usm` command to appoint a system in the hybrid environment that controls user management. By default the user management system is the host, unless another system is appointed with the `ai_usm` command.

About this task

See "Setting up matching user accounts" on page 13 for information about how to set up an environment in which the users and groups match each other on all involved systems. Only one system is allowed to change the user or group repository at a time. The `ai_usm` command is valid only after an x86 virtual server is successfully attached. The user management is controlled from the host, thus the `ai_usm` command is only available on the mainframe.

Procedure

1. To set the user management master, specify:
   ```
   # ai_usm <hostname>
   ```
   Example:
   ```
   [aihost]# ai_usm aihost
   ```

2. To query the status of the user management master, issue:
   ```
   # ai_usm
   ```
   Example:
   ```
   [aihost]# ai_usm
   User management master: aihost
   ```

Results

You can now use Linux user management commands, such as `useradd`, `userdel`, and `vipw` on the designated user management system.
Adding users

Use the `useradd` command to add users. Use only the designated user management system to add users.

Before you begin

- Adding users requires that a user management system has been defined, see [“Setting the user management system” on page 27](#).
- An x86 virtual server must be attached to the host.

About this task

Tasks described here for users also work for groups. The corresponding mechanisms work for deleting users, see the `userdel` man page for details.

Procedure

1. Ensure that you are working from the user management system.
   
   Issue the `ai_usm` command:

   
   ```
   [aihost]# ai_usm
   User management master: aihost
   ```

2. Add the user.
   
   Issue a command of the following form:

   ```
   [aihost]# useradd -m -c "<optional comment>" <user_ID>
   ```

   where `<user_ID>` is the user ID to be added.

Example

The following sample shows how to check that the current system is the user management system, add the user testuser, and check that the user definition on the virtual server has changed:

1. Check that the current system is the user management system:

   ```
   [aihost]# ai_usm
   User management master: aihost
   ```

2. Add the user testuser:

   ```
   [aihost]# useradd -m -c "Defined on z" testuser
   ```

3. Check that the user definition on the virtual server (aiatt0) has changed:

   ```
   [aihost]# cat /x86/aiatt0/etc/passwd
   at:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/bash
   ...
   testuser:x:1008:100:Defined on z:/home/testuser:/bin/bash
   ```

This task is illustrated in [Figure 6 on page 29](#).
If you were to try to add a user on the x86 virtual server it would fail because aiatt0 is not the user management master:

```bash
[aiatt0]# useradd testuser2
Cannot lock password file: already locked.
```

You can establish the x86 virtual server as the user management master and define a user there.

```bash
[aihost]# ai_usm aiatt0
[aihost]# ai_usm
User management master: aiatt0
```

Now you can switch to the virtual server and define a user there:

```bash
[aiatt0]# useradd -c "Defined on x" testxxxx
```

Then check that the user exists on the host:

```bash
[aihost]# cat /etc/passwd
at:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/bash
...
testuser:x:1008:100:Defined on z:/home/testuser:/bin/bash
testxxxx:x:1009:100:Defined on x:/home/testxxxx:/bin/bash
```

User home directories are created only on the user management system. If it is useful to have the same content on both the virtual server and the host, share the `/home` directory, see "Statements for sharing directories" on page 42.
Chapter 6. Using the attached resources

You can use the attached x86 virtual server resources, for example to run programs or scripts.

About this task

Using the resources entails:

- "Setting up programs or scripts to run on the attached resources" on page 32
- "Running programs or scripts on the attached resources" on page 32
- "Starting a script with ai_exec" on page 34

Setting up programs or scripts to run on the attached resources

Most programs and scripts run on the attached resources without any setup. Others might require environment setup in the form of environment variables or resource definitions. X11 clients require an SSH session and many programs require a pseudo terminal.

About this task

The technology recognizes scripts to be run on the attached x86 virtual server based on entries in the configuration file. Administrators can change or add these entries to define magic strings or file extensions for scripts to be run on the x86 virtual server.

The technology automatically sets up various binfmt_misc handlers on System z during attachment. The binfmt_misc handlers control the execution of x86 binaries on the attached x86 virtual server. You can also set up handling of specific script types in the /etc/ai/aiz_profiles.conf file. See "Keywords for handling scripts and programs" on page 42 for details about magic strings, file extensions, and script handlers.

Scripts are started either transparently by the binfmt_misc script handler or explicitly by using the ai_exec command on the host. Use the ai_exec command to run scripts that are neither handled by the the technology binfmt_misc handler nor located in the mounted /x86 path, see "Starting a script with ai_exec" on page 34.

Some programs and shell scripts need special setup. You might need to do one or more of the following:

Procedure

1. Set environment variables. Set the required environment variables on the host. The technology mirrors these variables to the x86 virtual server.

2. Define resources with ulimit. Resource definitions, for example, for memory, on the host is transferred to the x86 virtual server. You might need to change any transferred values to higher values on the x86 virtual server. You can use the ulimit command to query and change your resource settings. Issue ulimit -a for a list of resources.

3. Set a pseudo terminal. Pseudo terminals can be set by administrators or other users:
Administrators: Either set the PTY_DEFAULT variable to "yes", or add the real path name of the application to the list of GUIAPPS in the /etc/ai/aiz_profiles.conf configuration file.

Other users: Set the AI_USER_GUIAPPS=<filename> environment variable, where <filename> denotes the name of the file with the paths to the programs. Specify one application path per line. Paths cannot contain blanks. Empty lines are ignored.

The following example shows how to make a program called myprogram use a pseudo terminal:

a. As a non-root user, on the virtual server, create a file called /home/<non-root user>/guiapps.lst with the following content:
"/path/to/myprogram"

b. As the same user, on the host, export the environment variable AI_USER_GUIAPPS="guiapps.lst".

c. Run /x86/<virtual server>/path/to/myprogram.

As a result, myprogram uses a pseudo terminal.

Consider X11 clients. If the application is an X11 client, you must start an SSH terminal session with the host.

a. To do this, issue a command of the following form on the workstation that is attached to the host:
ssh -X <z-hostname>

where <z-hostname> is the System z host name.

b. The host must have been configured for X11 forwarding. Use the configuration file /etc/ssh/sshd_config to configure the host. Set X11Forwarding to "yes" and X11UseLocalhost to "no".

What to do next

You can now run your program or shell script:

- If your program or script is handled by the technology binfmt_misc handler and located in the mounted /x86 path, see "Running programs or scripts on the attached resources."

- If your program or script is neither handled by the technology binfmt_misc handler nor located in the mounted /x86 path, see "Starting a script with ai_exec" on page 34.

Running programs or scripts on the attached resources

You can call programs and most scripts conveniently from the host. The program or script call is transferred to the x86 virtual server, where the script or program is run. Output from the x86 virtual server processes is forwarded to the host (stdout) and input is passed from the host to the remote x86 virtual server process.

Before you begin

- An x86 virtual server must be attached.
- The directory you start the program from (the current working directory) must exist on the x86 virtual server.
About this task

On attachment the file system of the x86 virtual server is mounted on the host. The directory has a name of the form /x86/<virtual server>/, where <virtual server> is the name of the x86 virtual server as used on the attach command.

Procedure

Perform the following steps to run a program or script on an attached x86 virtual server.

1. Log in to your Linux on System z instance.
2. Locate the program you want to run by either:
   - Going to the mounted directory of the attached x86 virtual server Linux instance
   - Alternatively, specifying the fully qualified path to the program.

   The program must reside in the /x86/<virtual server>/ directory tree. For example, the sort program on the attached virtual server might be located in /x86/aiatt0/bin/sort

3. Start the x86 program. For example, using the hostname command:

   ![Example of running hostname command]

   Any non-quoted meta-characters and functionality such as piping (|), output redirection (>), and expansion of parameters or path names are run on the host. All catchable signals are forwarded from the host to the x86 worker process.

   **Note:** Should a script start running on the host instead of the x86 virtual server, ensure that the setup is correct, in particular, that the aiz_profiles.conf has an entry for the script. Also ensure that all referenced resources are mounted and can be found. All path information is interpreted according to the host file system.

Example

Assume that the x86 virtual server aiatt0 is attached to the host aihost and that the root file system of aiatt0 is accessible as /x86/aiatt0.

To pipe the contents of /etc/passwd to a sort program running on the attached x86 virtual server and redirect the output to the file mypasswd.txt on System z, issue:

![Example of piping and redirecting output]

What to do next

The running x86 program is represented on Linux on System z as a shadow process. You can monitor the process of programs that run over an extended period of time (see Chapter 7, "Integrating remote processes," on page 35).

For example, the /x86/aiatt0/bin/sort program will be visible on the host as a shadow process. In contrast, the cat /etc/passwd process runs on the host and does not spawn a shadow process.
Starting a script with ai_exec

You can force a script to run on the x86 virtual server even if the script is neither handled by the the technology binfmt_misc handler nor located in the mounted /x86 path.

Before you begin

If the script on the x86 virtual server does not reside in the mounted /x86 directories you need to specify the full path of the script.

Procedure

Use the ai_exec command to start the script. On the host, issue a command of the following form:

```
ai_exec <path/myscript>
```

where `<path/myscript>` is the path to the script. For example:

```
[aiuser@aihost:]# ai_exec home/user/myscript
```

The ai_exec command sends the request to run the script to the x86 virtual server. The script then runs on the x86 virtual server and the output is returned to the host.

Example

This example illustrates how the technology handles symbolic links. You can run, for example, `lsmod` in different ways:

- The `lsmod` command residing in `/sbin/lsmod` is a symbolic link to `/bin/lsmod` on the host.
- The `lsmod` command residing in `/x86/aiatt0/sbin/lsmod` is a symbolic link to `/bin/lsmod` on the attached virtual server.

If you were to enter:

```
[aiuser@aihost:]# /x86/aiatt0/sbin/lsmod
```

you would run `lsmod` on the host, because the symbolic link `/sbin/lsmod` is resolved by the kernel, and `/bin/lsmod` is an s390x executable.

If you were to enter:

```
[aiuser@aihost:]# /x86/aiatt0/bin/lsmod
```

you would run `lsmod` on the attached virtual server, because `/x86/aiatt0/bin/lsmod` is an x86 executable.

If you were to enter:

```
[aiuser@aihost:]# ai_exec lsmod
```

you would run `lsmod` on the attached virtual server.
Chapter 7. Integrating remote processes

This complete scenario shows the life cycle of a program from installation to usage. You can observe the status of such programs that you have started on the attached x86 virtual server using the `ai_ps`, `ai_top`, and `ai_lsof` commands.

About this task

You can observe the status of programs that you started on the attached x86 virtual server without logging in to the attached x86 virtual server.

In the following example, the virtual server aiatt0 is attached to the host aihost. The binary executable `foo` will be installed and used to demonstrate the execution of a program from the host on the attached virtual server. After installation, the file is located in `/usr/bin/` on the attached x86 virtual server. During remote installation of the x86 package, a new user named "foo-user" will be created.

Procedure

1. Install the RPM package. The example assumes that the x86 application `foo` is contained in an RPM package named `foo-0.1-1.x86_64.rpm`. The RPM package must be accessible from the host.
   a. Convert the x86 RPM into a meta RPM that can be remotely installed on the attached virtual server. Use the `ai_rpm` command to convert:

   ```
   [aihost:]# ls -l foo*.rpm
   -rw-r--r-- 1 root root 5118 2011-08-02 11:10 foo-0.1-1.x86_64.rpm
   [aihost:]# ai_rpm foo-0.1-1.x86_64.rpm
   [aihost:]# ls -l foo*.rpm
   -rw-r--r-- 1 root root 5118 2011-08-02 11:10 foo-0.1-1.x86_64.rpm
   -rw-r--r-- 1 root root 7562 2011-08-02 11:11 foo-AI-x86_64-0.1-1.s390x.rpm
   ```

   A new RPM file has been created in the same directory named `foo-AI-x86_64-0.1-1.s390x.rpm`. This is the meta RPM for the System z platform that contains the x86 RPM.
   b. Declare the x86 virtual server the user management master. Without giving the x86 virtual server the right to create users, the installation of the "foo" package would fail. Use the `ai_usm` command to change the user management master:

   ```
   [aihost:]# ai_usm
   User management master: aihost
   [aihost:]# ai_usm aiatt0
   [aihost:]# ai_usm
   User management master: aiatt0
   ```

2. Installing the meta RPM.
   a. To install the previously created meta package use the following command:

   ```
   [aihost:]# rpm -ivh foo-AI-x86_64-0.1-1.s390x.rpm
   Preparing... ########################################### [100%]
   1:foo-AI-x86_64################################################################### [100%]
   Installing foo-0.1-1.x86_64.rpm on attached virtual server... Done.
   ```

   After completion, two packages are installed, on the host the meta package "foo-AI-x86_64" and on the attached x86 virtual server the package "foo".
Furthermore, the application "foo" is located in /usr/bin/ and the new user "foo-user" has been created on the attached x86 virtual server.

b. To confirm, check with the following commands:

```
[aihost:]# ls -l /x86/aiatt0/usr/bin/foo
-rwxr-xr-x 1 root root 8819 2011-08-02 11:10 /x86/aiatt0/usr/bin/foo
[aihost:]# grep "foo-user" /x86/aiatt0/etc/passwd
foo-user:x:1001:100::/home/foo-user:/bin/bash
```

c. When the user management permissions are no longer required by the attached virtual server, use the `ai_usm` command to declare the host the user management master again:

```
[aihost:]# ai_usm aihost
[aihost:]# ai_usm
User management master: aihost
```

3. Start remote processes. Now you can remotely start the newly installed program "foo".

a. Change to the directory /usr/bin/ of the attached virtual server, which is accessible through the mount point /x86/<virtual server hostname>.

```
[aihost:]# cd /x86/aiatt0/usr/bin
```

b. Run "foo" in the background:

```
[aihost: /x86/aiatt0/usr/bin]# ./foo &
[1] 12345
```

The executable is started on the attached virtual server and the process ID shown in the output belongs to the shadow process that has been created on the host to represent the process on the attached x86 virtual server. This process ID can be used in subsequent calls to the technology commands in order to retrieve information about the remotely started process.

4. Use `ai_ps` to display information about the process. The output of the `ps` command shows one entry for each shadow process on the host, whereas the `ai_ps` command substitutes the shadow process entries with the real process information from the attached x86 virtual server. To see the difference, compare the values for CPU usage and CPU time in the output of the two commands for a CPU-intensive process that was started remotely on the attached x86 virtual server:

```
[aihost:]# ps -ef
UID PID PPID C STIME TTY TIME CMD
... 
root 12345 10000 0 14:36 pts/0 00:00:00 1234@aiatt0 ./foo
[aihost:]# ai_ps -ef
UID PID PPID C STIME TTY TIME CMD
... 
root 12345 10000 99 14:36 ? 00:00:17 1234@aiatt0 ./foo
```

In addition to the name of the started program, the name of the shadow process contains the process ID of the remotely started process and the host name of the attached x86 virtual server. In the previous example the system with the host name aiatt0 is attached to the host. The remote process with the x86 process ID 1234 is represented by a shadow process with the System z process ID 12345.
5. Use the **ai_top** command to show a dynamic, real-time list of processes currently running on the system. The output of the **top** command shows one entry for each shadow process on the host, whereas the **ai_top** command substitutes the shadow process entries with the real process information from the attached x86 virtual server.

a. To see the difference compare the values for CPU usage and used CPU time in the output of the two commands for a CPU-intensive process that was started remotely on the attached virtual server:

```
[aihost:]# top -p 12345
   PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
12345 root 20 0 3788 1572 1340 S 0 0.2 0:00.01 foo

[aihost:]# ai_top -p 12345
   PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
12345 root 20 0 3700 356 276 R 100 0.0 0:57.57 foo
```

Use the **-p** command-line option to limit the output to a set of process IDs. In the previous example this is used to only show the remotely started process. If the output is not restricted to particular process IDs, the list of processes generally exceeds the maximum lines available in a console. In contrast to **top**, the **ai_top** command lets you scroll the process list with the Up Arrow and the Down Arrow keys.

b. Other useful options can be used in interactive mode, while running the **ai_top** command. Press C to change the COMMAND column to show the whole command line of the program:

```
[aihost:]# ai_top -p 12345
   PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
12345 root 20 0 3700 356 276 R 100 0.0 0:57.57 1234@aiatt0 ./foo
```

The command line shows that the process is remotely started on the attached virtual server aiatt0 with the remote process ID 1234.

c. Assuming that the remotely started program "foo" is a threaded application, press Shift + H to switch to thread mode. The **ai_top** command shows all threads of a process:

```
[aihost:]# ai_top -p 12345
   PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
12352 root 20 0 3700 356 276 R 50 0.0 0:23.78 1234@aiatt0 ./foo
12353 root 20 0 3700 356 276 R 50 0.0 0:23.78 1234@aiatt0 ./foo
12345 root 20 0 3700 356 276 R 0 0.0 0:00.00 1234@aiatt0 ./foo
```

You can see that the remotely started process consists of three threads. The main thread is not doing any CPU-intensive work, but the sub-threads are each using 50% of a CPU.

6. Use the **ai_lsof** command to list files currently opened by running processes. Use the command in one of the following modes:

**List all open files regardless of which process is responsible for opening it**

The output for shadow processes is substituted by the **lsof** output for the remotely started processes on the attached virtual server. To see the files opened by the shadow processes use the **lsof** command.
**List all open files for a specific set of processes**

Use the `-p` command-line option to restrict the output to specific processes. For example, use the following command to only show the open files of the remotely started foo process with the System z process ID 12345:

```
[aihost:]# ai_lsof -p 12345
```

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>PID</th>
<th>USER</th>
<th>FD TYPE</th>
<th>DEVICE</th>
<th>SIZE/OFF</th>
<th>NODE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo</td>
<td>12345</td>
<td>root</td>
<td>cwd DIR</td>
<td>8,3</td>
<td>69632</td>
<td>/x86/aiatt0/usr/bin</td>
</tr>
<tr>
<td>foo</td>
<td>12345</td>
<td>root</td>
<td>rtd DIR</td>
<td>8,3</td>
<td>4096</td>
<td>/</td>
</tr>
<tr>
<td>foo</td>
<td>12345</td>
<td>root</td>
<td>txt REG</td>
<td>8,3</td>
<td>11781</td>
<td>/x86/aiatt0/usr/bin/foo</td>
</tr>
</tbody>
</table>

**List all processes that have opened a specific file**

To list all processes that have opened a specific file, add the path to the file as a parameter to the `ai_lsof` command. For example, use the following command to only show those processes that have opened the binary executable of the foo program on the attached x86 virtual server:

```
[aihost:]# ai_lsof /x86/aiatt0/usr/bin/foo
```

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>PID</th>
<th>USER</th>
<th>FD TYPE</th>
<th>DEVICE</th>
<th>SIZE/OFF</th>
<th>NODE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo</td>
<td>12345</td>
<td>root</td>
<td>txt REG</td>
<td>8,3</td>
<td>11781</td>
<td>/x86/aiatt0/usr/bin/foo</td>
</tr>
</tbody>
</table>
Part 3. Reference

This part contains reference information.
Chapter 8. The configuration file

The main configuration file for the technology is /etc/ai/aiz_profiles.conf. You require administrator rights to change settings in the configuration file. Use this file to tailor the technology infrastructure to your environment. For an easy setup, accept the defaults unless they collide with other server configurations in your environment.

The configuration file contains:

- Title section
- "Statements for sharing directories" on page 42
- "Keywords for handling scripts and programs" on page 42
- "Logging statement" on page 44
- "Keywords for networking" on page 45
- "Tuning statement" on page 45
- "Session control characters" on page 46

Generally, the following rules apply to the configuration file:

- The section labeled [applint] is mandatory. The keywords ACTIVE and HOST are filled in automatically.
- Lines that begin with a number sign (#) are comment lines. A number sign in the middle of a line makes the remaining line a comment.
- Empty lines are permitted.
- The specifications are not case sensitive.

Note: You must re-attach the x86 virtual server to activate any changes made to the configuration file.

Title section

The title section contains two keywords:

HOST = <host name>
ACTIVE = true | false

These keywords are filled automatically when a virtual server is attached; there is no need for you to do anything.

An example is shown in Figure 7

[applint]
HOST = "localhost"
ACTIVE = "false"

Figure 7. Title section
Statements for sharing directories

You can optionally give the attached x86 virtual server access to directories on the host. Such directory sharing requires an NFS server on the host and an NFS client on the x86 virtual server. Define one SHARE statement of the following form for each directory you want to share:

```
SHARE_<n>="<shared directory>:<NFS mount options>"
```

Defines the directories on the host to be shared with the x86 virtual server, where:

<n> is the number of the shared directory.

<shared directory:> is the path to the directory.

<NFS mount options> is the NFS mount option for the directory. If omitted, the technology uses the mount options \texttt{rw, no_root_squash, no_subtree_check, crossmnt}. See the \texttt{exports} man page for supported options. The technology passes all options “as is” to NFS.

When attaching the virtual server, the NFS share is mounted over any file system already mounted at the specified mount point.

\textbf{Figure 8} shows an example statement for sharing the /home directory.

\begin{verbatim}
# Shares to be exported to blade
SHARE_1 = "/home:"
\end{verbatim}

\textbf{Figure 8. Statement for sharing directories}

For statements in the configuration file pertaining to the network, see “Setting up port forwarding and masquerading” on page 11. See also the \texttt{aiz_profiles.conf} man page.

Keywords for handling scripts and programs

The \texttt{binfmt_misc} handler recognizes programs and scripts to be run on an attached x86 virtual server.

- Scripts are recognized by a magic string within the first few bytes of a file or the file extension. You can use the \texttt{BINFMT\_MAGIC} and \texttt{BINFMT\_EXT} keywords in the configuration file to specify a magic string or file extension to fit your scripts.

- Programs are handled by \texttt{binfmt_misc}. The setup for handling programs is determined automatically at attachment time. You can use the \texttt{BINFMT\_EXT} and the \texttt{GUIAPPS} keywords to specify the extension of files that should be handled by \texttt{binfmt_misc} and whether a pseudo terminal should be started.

The keywords used to handle scripts and programs are:

\textbf{BINFMT\_DEFAULT\_HANDLER}

Defines the handler for automatic script handling. The default is /sbin/xzdbinfmt.

Example:

```
BINFMT\_DEFAULT\_HANDLER = "/sbin/xzdbinfmt"
```
**BINFM'T MAGIC**

Specifies entries to be registered with the binfmt_misc handler during attachment. An entry contains the following fields (separate fields with colons):

- **Suffix**: The filename suffix to be used in `/proc/sys/fs/binfmt_misc` to represent this entry.
- **Magic**: The magic string for recognition of this type.
- **Offset**: The magic string offset in the first line of the script.
- **Handler**: The interpreter to be used for this script type. The default handler is used when the field is empty.

Example:

```
BINFM'T_MAGIC = "bash:/bin/bash:2:/sbin/xzdbinfmt"
```

According to the example, files with the suffix bash and containing the string `/bin/bash` starting in the third position of the first line are to be interpreted using the handler located in `/sbin/xzdbinfmt`.

**BINFM'T_EXT**

Specifies file extensions to be registered at the binfmt_misc handler during attachment (separate fields with colons).

- **Ext**: The file extension for recognition of this type.
- **Handler**: The interpreter to be used for this script type. The default handler is used when the field is empty. The x86 virtual server might require an appropriate binfmt_misc handler to match this specification.

Example:

```
BINFM'T_EXT = "php:/sbin/xzdlauncher"
```

According to the example, files with the extension php are handled by the handler located in `/sbin/xzdlauncher`.

**PTY_DEFAULT** = "no | yes"

Specifies whether a pseudo terminal should be created when an application is started on an x86 virtual server. The default is "no".

**GUIAPPS**

Creates a pseudo terminal for selected applications when the PTY_DEFAULT is set to "no". The technology provides a predefined list of applications requiring this setting. You can extend this list with applications you commonly use. Separate list items with colons.

Example:

```
GUIAPPS="/bin/bash:/bin/sh:/bin/vi"
```

A list entry may not contain blanks.

Users with no administrator authorization can set the AI_USER_GUIAPPS environment variable to extend the list of applications for which a pseudo-terminal is created. See step 3 on page 31 for details and an example.

**XZDPFSD_LISTEN**

Sets the listen queue length for xzdpfsd. The listen queue length is governed by the XZDPFSD_LISTEN variable in the `/etc/ai/aiz_profiles.conf` configuration file. The default is 500. When shadow processes are missing due to a high shadow-process creation rate, increase the value.
**XZDPMNTD_CACHE**

Specifies the time in milliseconds the internal data cache of the xzdpmntd remote procs daemon is used before the data is updated from the attached virtual server. The valid range is between 1 and 10000 ms. Specify 0 to disable the internal cache. This results in remote updates for every read operation that is performed on the remote procs for an attached x86 virtual server.

The configuration file sample in Figure 9 has been simplified to show only the specifications for starting programs.

```
# BINFMFT entries for script processing
# BASH scripts must be handled by the xzdbinfmt handler.
BINFMT_DEFAULT_HANDLER = "/sbin/xzdbinfmt"
# a) register binfmt_misc handlers for scripts - multiple entries of
# "suffix:magic:offset:handler:"
BINFMT_MAGIC = "bash:/bin/bash:2:/sbin/xzdbinfmt:binsh:/bin/sh:2:\
 :csh:/usr/bin/csh:2:" 
# b) register binfmt_misc handlers for scripts - multiple entries of "ext:handler:"
BINFMT_EXT = "php:/sbin/xzdlauncher:tk:/sbin/xzlalouncer:"

# Create X86 process with or without a pseudo terminal
# if PTY_DEFAULT is omitted "no" is the default
# PTY_DEFAULT="no"
# GUI APPS (separate applications with a colon ":")
# if PTY_DEFAULT="no"
# the following applications are started on x86 using a pseudo terminal:
GUIAPPS="/bin/bash:/bin/sh:/bin/vi:/bin/ex:/usr/bin/mc:/bin/ls:/usr/bin/less:\
 /usr/bin/man:/bin/tcsh:/bin/sash:/bin/ksh:/usr/bin/ssh:\
 /usr/bin/slogin:/usr/bin/python:/usr/bin/pstree.x11:/usr/bin/sqlite3"

# Control listen queue length of xzdpfsd.
XZDPFSD_LISTEN="500"

# Cache timeout for xzdpmntd in msec (0 to disable)
XZDPMNTD_CACHE="100"
```

*Figure 9. Statements for starting programs*

**Logging statement**

By default log messages issued on the x86 virtual server are written to log files on the virtual server. Use the logging statement to forward log messages to the host.

Use the following keyword for logging:

**ConvergedLogging = “no | yes”**

ConvergedLogging = “yes” causes log messages to be forwarded to the host. A message log file `/var/log/<virtual_server>_logfile` is created where `<virtual_server>` is the x86 virtual server host name as specified in the `ai_att` command.

ConvergedLogging = “no” does not forward log messages.

The configuration file sample in Figure 10 on page 45 has shows the logging statement.
Keywords for networking

Use the following keywords for the network setup:

**NETWORK_CONFIG**
Specifies port forwarding. To activate port forwarding, specify “nat”. For no port forwarding, specify “none”. Specifying “nat” activates the network address translation from the external interface to the internal interface and vice versa on the System z host.

**NETWORK_NAT_IF_HOST_EXT**
Specifies the external network interface on the host.

**NETWORK_NAT_IF_HOST_INT**
Specifies the internal network interface to the x86 virtual server on the host.

**NETWORK_NAT_IF_BLADE**
Specifies the internal network interface to the host on the x86 virtual server.

**NETWORK_NAT_FW_PORT_<n>**
Specifies a list of ports to be forwarded. For each port to be forwarded specify NETWORK_NAT_FW_PORT_<n> = "<zport> <xport>", where <n>=1,2... and where <zport> specifies the service offered at the host and <xport> specifies the x86 virtual server port.

The configuration file sample in [Figure 11] shows the network statements.

Tuning statement

Use the following keywords for tuning:

**HB1INTERVAL**
Specifies the send interval of the heartbeat in seconds. When the interval expires, a heartbeat signal is sent (in both directions). Typical values are in the range 1 - 5 seconds.

**HB2INTERVAL**
Specifies the heartbeat observer interval in seconds. When the interval expires, the heartbeat function checks that a heartbeat signal arrived during the interval. Otherwise the system assumes that connection to host or to
the virtual server is broken and a cleanup function is initiated. Typical values are in the range 15 - 60 seconds with an upper limit of 60 - 75 seconds.

The configuration file sample in Figure 12 shows the tuning statement.

```
# Heartbeat configuration
HB1INTERVAL="1"
HB2INTERVAL="15"
```

Figure 12. Tuning statement

**Session control characters**

Use the following keywords to control pseudo-terminal sessions:

**AI_TTY_ESC_CHAR**

Sets the escape character that lets application integration recognize particular signals used for session control. The character should not be the same as the escape character of the remote login program, for example, "~" for SSH. A subsequent Ctrl+Z stops the corresponding shadow process on the application integration host. An escaped AI_CONN_RESET_CHAR resets the current session.

Non-root users can set an escape character by setting the AI_TTY_ESC_CHAR environment variable.

**AI_CONN_RESET_CHAR**

Sets the character for session reset.

The configuration file sample in Figure 13 shows the session control characters statement.

```
# Escape character to let ai recognize certain signals for session control.
AI_TTY_ESC_CHAR = "^"

# Session reset character (used after AI_TTY_ESC_CHAR).
AI_CONN_RESET_CHAR = "."
```

Figure 13. Session control characters
Complete configuration file example

The following example shows what a configuration file could look like. For a current version, see the aiz_profiles.conf file delivered with the technology.

# Example configuration file

[applint]
HOST = "aiatt0"
ACTIVE = "true"

# Sharing directories

SHARE_1 = "/home:

# Programs and scripts

BINFMT_DEFAULT_HANDLER = "/sbin/xzdbinfmt"
BINFMT_DEFAULT_HANDLER = "/sbin/xzdbinfmt"
BINFMT_MAGIC = "bash:/bin/bash:2:/sbin/xzdbinfmt:binsh:/bin/sh:2:/usr/bin/sh:2::sh:/usr/bin/csh:2::"
BINFMT_EXT = "php:/sbin/xzdlauncher:

Create virtual server processes with or without a pseudo terminal
if PTY_DEFAULT is omitted "no" is the default.
PTY_DEFAULT="no"

GUI APPS (separate applications with a colon ":")
if PTY_DEFAULT="no" the following applications are
started on the virtual server using a pseudo terminal:
GUIAPPS="/bin/bash:/bin/sh:/bin/vi:/bin/ex:/usr/bin/mc:/bin/ls:/usr/bin/less:
/usr/bin/man:/bin/tcsh:/bin/bash:/bin/ksh:/bin/sh:/bin/stdout::
/usr/bin/login:/usr/bin/python:/usr/bin/pstree.x11:/usr/bin/sqlite3"

Control listen queue length of xzpfsd:
XZDPFSDD_LISTEN="500"

Cache timeout for xzpmntd in msec (0 to disable):
XZDPMNTD_CACHE="100"

Converged logging

Set return code of ai_att to EX_OSFILE (72) when user or group file not in sync
ErrorOnUsersOutOfSync = "on"
Collect messages from virtual servers on the host /var/log/...
ConvergedLogging = "no"
# Networking

# To enable the network and firewall configuration set NETWORK_CONFIG = "nat"
NETWORK_CONFIG = "none"
NETWORK_NAT_IF_HOST_EXT = "eth0"
NETWORK_NAT_IF_HOST_INT = "eth1"
NETWORK_NAT_IF_BLADE = "eth1"
NETWORK_NAT_FW_PORT_1 = "23 23"

# Tuning

# A heartbeat is sent every n seconds (both directions):
HB1INTERVAL="1"
# The heartbeat daemon checks every m seconds for missing heartbeats:
HB2INTERVAL="15"

# Session control characters

# Escape character to access particular functions for session control.
AI_TTY_ESC_CHAR = "^"
# Session reset character (used after AI_TTY_ESC_CHAR).
AI_CONN_RESET_CHAR = "."
Use technology commands to manage RPMs, attach and detach an x86 virtual server, monitor processes, and control user and group accounts. Use all commands on System z except `ai_attachable`.

**ai_att - connect an x86 virtual server**

Connects an x86 virtual server to the host.

**Purpose**

`ai_att` connects an x86 virtual server to the host. `ai_att` expects the application integration technology service to be running on the x86 virtual server. If necessary, `ai_att` starts the technology service on the host (System z).

The command mounts the root file system of the x86 virtual server under `/x86/<x86 virtual server hostname>/` and provides system information of the x86 under `/x86/proc_<x86 virtual server hostname>/`. The host name is registered at the x86 virtual server in the `/var/run/ai.hostid` file.

After successfully attaching an x86 virtual server, you can issue a command to start an x86 program. The program call is transferred to the x86 virtual server, where the program is run.

Specify any files to be exported from the host to the x86 virtual server in the configuration file `/etc/ai/aiz_profile.conf`.

For more information about this command, review the following sections:

- "Authorization"
- "Environment"
- "Prerequisites"
- "ai_att syntax" on page 50
- "Option descriptions" on page 50
- "Examples" on page 50

**Authorization**

To run this command, you must be root on the host.

**Environment**

This command can be issued from a Linux instance running on System z.

**Prerequisites**

The service `applint` must be installed on both the mainframe and the x86 virtual server. It must be running on the x86 virtual server. To check if an x86 virtual server is ready for attachment, run `ai_attachable` on the x86 virtual server.
Option descriptions

The meaning of the options is:

<x86 virtual server>
Is the host name or IP address of the Linux instance running on the virtual server.

-d or --details
Displays configuration details for an attached x86 virtual server. The details include:
• The host name or IP address of the Linux instance running on the attached x86 virtual server.
• A list of mount points for file systems that are shared between the host and the attached x86 virtual server.
• A list of file-system mount-points that are related to the attached x86 virtual server.
• A list of daemons that are started on behalf of the attached virtual server, including their process ID and the port number to which they are listening, if any.

-q or --query
Displays the host name of the attached x86 virtual server.

-r or --refresh
Rereads the profile data and, if changes are found, updates the attachment configuration. For example, if new shared directories are found, these are mounted on the x86 virtual server.

-h or --help
Displays the help information for the command.

-v or --version
Displays the version of the command.

See the ai_att man page for the return codes.

Examples

Using ai_att to attach an x86 virtual server

Assuming that the host name of the x86 virtual server is aiatt0, issue the following command to attach it:

[aihost]# ai_att aiatt0
ai_attachable - determine if an x86 virtual server can be attached

Checks if an x86 virtual server can be connected to the host.

**Purpose**

ai_attachable determines if an x86 virtual server is ready for attachment to the host. ai_attachable expects the application integration technology service to be running on the x86 virtual server.

For more information about this command, review the following sections:

- "Authorization"
- "Environment"
- "Prerequisites"
- "ai_attachable syntax"
- "Option descriptions"
- "Examples" on page 52

**Authorization**

To run this command, you must be root on the Linux instance running on the x86 virtual server.

**Environment**

This command can be issued from a Linux instance running on an x86 virtual server.

**Prerequisites**

- The service applint must be installed on both the mainframe and the x86 virtual server.
- An SSH session is required.

**ai_attachable syntax**

```plaintext
ai_attachable [-h -v]
```

**Option descriptions**

The meaning of the options is:

- `-h` or `--help`
  - Displays the help information for the command.
- `-v` or `--version`
  - Displays the version of the command.
Return codes

The **ai_attachable** returns a message indicating success or failure of the attachment test. Possible return codes are:

0  Virtual server can be attached to IBM System z.
64 Virtual server already attached to: <ai_host>
69 Service applint not installed, or service applint not running. Virtual server cannot be attached.

Examples

Using **ai_attachable** to check if an x86 virtual server can be attached.

Assuming that the x86 virtual server is called aiatt0, issue the following command on the x86 virtual server:

```
[aiatt0]# ai_attachable
ai_attachable: x86 virtual server can be attached to IBM System z.
```

ai_det - detach an x86 virtual server

Disconnects an x86 virtual server from the host.

Purpose

For more information about this command, review the following sections:

- "Authorization"
- "Environment"
- "ai_det syntax" on page 53
- "Option descriptions" on page 53
- "Examples" on page 53

Authorization

To run this command, you must be an administrator of the Linux instance that is running on the mainframe.

Environment

This command can be issued from a Linux instance running on System z.

Prerequisites

The service **applint** must be installed on both the mainframe and the x86 virtual server.
ai_det syntax

```
ai_det [-x86 virtual server] [-q] [-h] [-v]
```

Option descriptions

The meaning of the options is:

<x86 virtual server>
Is the host name of the Linux instance running on the virtual server.

-q or --query
Displays the host name of the attached x86 virtual server.

-h or --help
Displays the help information for the command.

-v or --version
Displays the version of the command.

See the ai_det man page for the return codes.

Examples

Using ai_det to detach an x86 virtual server

Assuming that the host name of the x86 virtual server is aiatt0, issue the following command to detach it:

```
[aihost]# ai_det aiatt0
```

Specifying the x86 virtual server is optional. For example:

```
[aihost]# ai_det
Detaching...
Shutting down applint service:
Stopping xzdhd: done
Stopping xzdpfsd: done
Stopping xzdwatchd: done
Unmounting combined procfs from /x86/proc... Done.
Unmounting remote procfs from /x86/proc_aiatt0... Done.
x86 virtual server successfully detached.
```

If processes of the attached x86 virtual server are still visible as shadow processes on the host, the detach operation fails.

Using ai_det to list the host name of the attached x86 virtual server

```
# ai_det -q
Systems attached:
aiatt0 [applint]
```
ai_exec - run scripts on an x86 virtual server

Runs a script on an x86 virtual server remotely from the host.

Purpose

Use ai_exec on the host to run scripts that are neither handled by the technology binfmt_misc handler nor located in the mounted /x86 path. The scripts can be for any type of interpreter that is available to Linux on the x86 virtual server, for example, Perl, Python, or a Linux shell such as bash.

Scripts specified in a shared directory are also accessible to the host. ai_exec ends with the return code of the executed script.

For more information about this command, review the following sections:

- "Environment"
- "Prerequisites"
- "ai_exec syntax"
- "Option descriptions"
- "Examples"

Environment

This command can be issued from a Linux instance running on System z.

Prerequisites

- An x86 virtual server must be attached to the host.

ai_exec syntax

```
ai_exec <virtual server path to script> [arguments]
-f [arguments]
```

Option descriptions

The meaning of the options is:

- `<virtual server path to script>`
  Location of the script. The script must be accessible to the virtual server.

- `-h` or `--help`
  Displays the help information for the command.

- `-v` or `--version`
  Displays the version of the command.

Examples

Using ai_exec to remotely install software on an x86 virtual server
Assume that x86 virtual server aiatt0 is attached to the host. Attaching mounts the x86 virtual server's root file system to /x86/aiatt0. Further, assume that the installation script is /tmp/product/install.sh on the x86 virtual server.

Issue the following command to run the script:

```
[aihost]# ai_exec /tmp/product/install.sh
```

### ai_lsof - list open files

**ai_lsof** lists information about files opened by processes running on the host and on the Linux instance that runs on the attached x86 virtual server. x86 virtual server processes are shown with the PID of their shadow process. Shadow processes are omitted from the output. To see shadow process information, run **lsof** on the host.

Unlike **lsof**, **ai_lsof** does not support the **-F** option. **ai_lsof** always displays its output as a formatted table with fixed column width.

For more information about this command, review the following sections:
- “Environment”
- “ai_lsof syntax”
- “Option descriptions”
- “Examples” on page 56

### Environment

This command can be issued from a Linux instance running on System z.

### ai_lsof syntax

```
ai_lsof [-p <pid-set>] [-h] [-v] [-<file>
```

### Option descriptions

**ai_lsof** is based on the **lsof** command, but with a restricted set of features. The following options are supported:

- **-p or --pid <pid-set>**
  
  Includes or excludes the files for the processes whose process identification (PID) numbers are in the set <pid-set>. The set <pid-set> is a comma-separated list of PIDs of the form:
  
  "<PID1>,<PID2>,...
  
  For example:
  
  "123"
The output for shadow processes is substituted by the `lsof` output for the remotely started processes on the attached virtual server. To see all files opened by the shadow processes use the `lsof` command.

```
[aihost:]# ai_lsof
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
init 1 root cwd DIR 94,1 4096 2 /
init 1 root rtd DIR 94,1 4096 2 /
init 1 root txt REG 94,1 44744 48449 /sbin/init
init 1 root mem REG 94,1 152819 249986 /lib64/ld-2.11.1.so
init 1 root mem REG 94,1 126176 250043 /lib64/libselinux.so.1
init 1 root mem REG 94,1 1809336 249993 /lib64/libc-2.11.1.so
init 1 root mem REG 94,1 19161 249999 /lib64/libdl-2.11.1.so
init 1 root 10u FIFO 0,15 0t0 1305 /dev/initctl
[...]
```

List all processes that have opened a specific file

To list all processes that have opened a specific file, add the path to the file as a parameter to the `ai_lsof` command. For example, use the following command to only show those processes that have opened the binary executable of the `foo` program on the attached x86 virtual server:

```
[aihost:]# ai_lsof /x86/aiatt0/usr/bin/foo
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
foo 12345 root txt REG 8,3 11781 106808 /x86/aiatt0/usr/bin/foo
```

**ai_ps - report the status of the current processes**

Displays process information from the attached x86 virtual server and the host.

**Purpose**

The `ai_ps` command displays a combined view of processes running on the host and the Linux instance that runs on the attached x86 virtual server.
The output for processes that run on the attached x86 virtual server has the following format:

```
<Remote PID>@<x86 virtual server> <command>
```

where `<x86 virtual server>` is the IP address if the IP address was specified when attaching the x86 virtual server or the host name if the host name was specified. For example:

```
1234@1.2.3.4 vi
```

Or:

```
1234@aiatt0 vi
```

The `ai_ps` command output includes the real resource usage on the attached virtual server for processes that run there. See Chapter 7, “Integrating remote processes,” on page 35 for an example.

In contrast, the `ps` command shows processes running on the host. These include the shadow processes for processes running on the attached x86 virtual server. The shadow processes do not reflect the resource consumption of their counterparts on the attached x86 virtual server.

For more information about this command, review the following sections:

- “Authorization”
- “Environment”
- “ai_ps syntax”
- “Option descriptions” on page 58
- “Examples” on page 58

**Authorization**

To run this command, you must have a user ID on the host.

**Environment**

This command can be issued from the host.

**Prerequisites**

The technology RPM packages must be installed.

**ai_ps syntax**

```
ai_ps <options>
   -h
   -v
```
Option descriptions

All options that are valid for the ps command can be used for ai_ps.

<options>

See the ps command for available options and their meanings.

-h or --help

Displays the help information for the command.

-v or --version

Displays the version of the command.

Examples

Using ai_ps to display a full list of all processes

Assuming that a program called my_program is running on the attached x86 virtual server aiatt0, you can use ai_ps to display process information:

```
[aihost:]# ai_ps -ef
UID    PID    PPID   C   STIME   TTY    TIME    CMD
root   1      0       0  Oct25   ?        0:00:02  init [3]
...    
root   54174  54172  0      09:48  pts/0   00:00:00  -bash
root   54697  54174  0      09:57  pts/0   00:00:00  22643@aiatt0 ./my_program
```

ai_rpm - create meta RPM packages

You can manage RPM packages on the attached x86 virtual server through the use of meta RPM packages. Use the ai_rpm command to create an RPM in the Linux on System z environment that contains an RPM for installation on an attached x86 virtual server.

Purpose

The intention of the meta RPM packages is to help you keep RPM packages on the host and the attached virtual server in synch. You can use the meta packages to manage RPMs on the x86 virtual server from the host.

First create a repository of meta RPM packages on the host. You can then install or remove packages on the x86 virtual server by installing or removing the corresponding meta package on the host.

To install meta packages on the host, use the package management tool specific to your distribution for example, zypper for SUSE Linux Enterprise Server, or yum for Red Hat Enterprise Linux.

For more information about this command, review the following sections:

- "Authorization"
- "ai_rpm syntax" on page 59
- "Option descriptions" on page 59
- "Example" on page 59

Authorization

You must be root on the host to use this command.
ai_rpm syntax

The meaning of the options is:

<x86_RPM>
Is the RPM file to be converted.

-o <path> or --outputdir <path>
Specifies an output directory <path> for the converted meta RPM package files. If not specified, the current working directory is used.

-p or --preserve-dependencies
Maps the dependencies of the x86 RPMs in the meta packages. If not specified, the resulting meta RPM package does not include dependencies on other meta RPM packages.

-f or --force
Forces the creation of an output file. An existing file with the same name will be overwitten without warning.

-h or --help
Displays the help information for the command.

-v or --version
Displays the version of the command.

Example

Using ai_rpm to convert an RPM
Assuming that you have an RPM named vim-7.2-8.8.x86_64.rpm and that you want to convert it:

```
# ai_rpm vim-7.2-8.8.x86_64.rpm
```

This produces a meta RPM named vim-AI-x86_64-7.2-8.8.s390x.rpm

Using ai_rpm to convert multiple RPMs
This examples assumes that there is a directory named /path/to/x86_RPMs that contains multiple x86 RPM packages. To create meta packages for them all, list them and pipe the list to the ai_rpm command. Processing multiple RPM files generates status messages for each file that has been successfully processed, for example:
Depending on the number of x86 RPM files, this can take a while. The `xargs` command tries to put all RPM files returned by the `ls` command into one `ai_rpm` call. If the maximum command line length is exceeded, the `ai_rpm` call is split into multiple calls.

---

**ai_top - display CPU-intensive processes**

Displays the most CPU-intensive processes on the system.

**Purpose**

`ai_top` provides processor activity in real time. It displays a listing of the most CPU-intensive tasks on the system, and provides an interactive interface for manipulating processes. You can sort the tasks by CPU usage, memory usage, and run time.

For more information about this command, review the following sections:

- "Authorization"
- "Environment"
- "Prerequisites"
- "ai_top syntax" on page 61
- "Option descriptions" on page 61
- "Examples" on page 62

**Authorization**

To run this command, you must have a user ID on the host.

**Environment**

This command can be issued from a Linux instance running on System z.

**Prerequisites**

The service `applint` must be installed and running on both the mainframe and the x86 virtual server.
ai_top syntax

```
```

Option descriptions

The meaning of the options is:

- **-b** or **--batch-mode**
  Starts ai_top in batch mode, which can be useful for sending the output to other programs or to a file. In this mode, ai_top will not accept input and runs until the iterations limit set with the **-n** command-line option, or until killed.

- **-d** or **--delay** `<seconds>`
  Specifies the delay between screen updates in seconds. The default is three seconds.

- **-n** or **--iterations** `<iterations>`
  Specifies the maximum number of screen updates ai_top runs before ending. If omitted, the command runs indefinitely. You can then stop it, for example with Ctrl+C.

- **-p** or **--pid** `<pid>`
  Specifies the process IDs to be monitored. Specify multiple processes in a comma-separated list of process IDs.

- **-s** or **--summary-line**
  Shows aggregated values at the end of the task list.

- **-h** or **--help**
  Displays the help information for the command.

- **-v** or **--version**
  Displays the version of the command.

Interactive options

When ai_top is running, you can use the following keys to control the program:

- **C**
  Press C to toggle between the display of only the command name of a process, or the whole command line, including the program's arguments.

- **D**
  Press D to get a prompt for specifying the delay between screen updates in seconds. Valid values are integers or decimal numbers in the range 0.1 - 3600 seconds. You can specify one decimal place; longer input is shortened to one decimal place, for example 2.99999 is shortened to 2.9. The default is 3 seconds.

- **F**
  Press F to show a menu for selecting columns to display. By pressing the corresponding hot key, a column can be enabled or disabled. A selected column is marked by an asterisk, and printed in bold letters.
**Shift+F**
Press Shift+F to display a list of column hot keys for sorting. Press the corresponding hot key to select a column. If a column is already selected as the sort column, the hot key toggles between descending and ascending sort order. The current sort column is shown in bold letters and is marked by the sort order symbol (">" descending, or "<" ascending).

**Shift+H**
Press Shift+H to enable or disable the monitoring of threads. By default, threads are not visible in the list of processes.

**O**
Press O to display a list of column hot keys for rearranging the column display order. Press the corresponding hot key to change the position of a column. Each column can be moved up in the list by pressing the upper-case hot key letter, and down by pressing the lower-case letter. Moving a column up in the list means that the column is shifted one position to the left in the task list window, by moving it down it is shifted to the right.

**S**
Press S to toggle a row of aggregated values, such as the total amount of CPU time for all processes, at the end of the task list.

**Up Arrow**
Press the Up Arrow key to scroll the list of processes up by one line.

**Down Arrow**
Press the Down Arrow key to scroll the list of processes down by one line.

**Page Up**
Scrolls the list of processes up by one page.

**Page Down**
Scrolls the list of processes down by one page.

**Home**
Scrolls the list of processes up to the top.

**End**
Press the End key to scroll the list of processes down to the bottom.

**Q**
Press Q to quit the program.

**Examples**

**Using ai_top to find the most CPU-intensive tasks on an x86 virtual server**

Issue the following command to display running processes:

```bash
# ai_top <options>
```

The output might look like the following:
**ai_usm - manage users**

Sets the right to change user and group authority and manages users and groups in the Linux on System z environment.

**Purpose**

The `ai_usm` command sets the user management system in the Linux on System z environment.

For more information about this command, review the following sections:

- [Authorization](#)
- [Environment](#)
- [Prerequisites](#)
- [ai_usm syntax](#)
- [Option descriptions](#)
- [Examples](#)

**Authorization**

To run this command, you must be root on the Linux instance that is set as the user and group management server.

**Environment**

To set the user and group management server, issue this command from a Linux on System z instance.

**Prerequisites**

Adding or modifying users is best done on the host after x86 virtual server attachment.
ai_usm syntax

```
ai_usm
  <host name>
  -h
  -v
```

Option descriptions

The meaning of the options is:

- `<host name>`
  specifies the name of the user management host.

- `-h` or `--help`
  displays the help information for the command.

- `-v` or `--version`
  Displays the version of the command.

See the `ai_usm` man page for the return codes.

Examples

Using ai_usm to set the user and group management server

Assuming the x86 virtual server you want to use as user management server is called "aiatt0":

```
# ai_usm aiatt0
```

Using ai_usm to create a new user

Follow these steps to create a new user:

1. Set the x86 virtual server you want to use as user management server, for example:
   ```
   # ai_usm aiatt0
   ```

2. On myblad01, issue a `useradd` command to add the new user, for example:
   ```
   # useradd -c "This is my first user" firstuser
   ```

3. Optionally, check that the user is defined in the `/etc/passwd` file:
   ```
   # cat /etc/passwd
   at:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/bash
   ...
   firstuser:x:1009:100:Defined on x:/home/firstuser:/bin/bash
   ...
Part 4. Appendixes
Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

Documentation accessibility

The Linux on System z publications are in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when you use the PDF file and want to request a Web-based format for this publication, use the Reader Comment Form in the back of this publication, send an email to eservdoc@de.ibm.com, or write to:

IBM Deutschland Research & Development GmbH
Information Development
Department 3248
Schoenaicher Strasse 220
71032 Boeblingen
Germany

In the request, be sure to include the publication number and title.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility at

www.ibm.com/able
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.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.
This information is for planning purposes only. The information herein is subject to change before the products described become available.

**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at


Adobe is either a registered trademark or trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
determine if can be attached  51
directories
    sharing  42
disconnecting  52
display  60
display process information  56
displaying CPU usage  60
displaying usage for processes  60

E
environment variable
    AI_USER_GUIAPPS  31
environment variables  31
exports
    NFS  42

F
file systems
    integrating  4
    sharing  4
files
    list open  55

G
GUIAPPS
    example of setting  31
    keyword in configuration file  42

H
HB1INTERVAL
    keyword in configuration file  45
HB2INTERVAL
    keyword in configuration file  45
HOST
    keyword in configuration file  41
    host meta RPM  19

I
ia_usm command
    example  27
install
    x86 RPMs  21
installation  3
installing  7, 35
    meta RPM packages  21
    RPM package on the System z host  8
    RPM package on the x86 virtual server  8
    updates  10
integrating file systems  4
integrating logs  13

K
keyword in configuration file
    AI_CONN_RESET_CHAR  46

keyword in configuration file (continued)
    AI_TTY_ESC_CHAR  46
    BINFMT_DEFAULT_HANDLER  42
    BINFMT_EXT  42
    BINFMT_MAGIC  42
    ConvergedLogging  44
    GUIAPPS  42
    HB1INTERVAL  45
    HB2INTERVAL  45
    NETWORK_CONFIG  45
    NETWORK_NAT_FW_PORT_n  45
    NETWORK_NAT_IF_BLADE  45
    NETWORK_NAT_IF_HOST_EXT  45
    NETWORK_NAT_IF_HOST_INT  45
    PTY_DEFAULT  42
    SHARE_n  42
    XZDPFS_LISTEN  42
    XZDPMNNTD_CACHE  42

L
list open files  55
logging
    statement in configuration file  44
logs
    integrating  13

M
manage  63
manage RPM packages  58
managing  63
    x86 RPMs  21
managing users  27
managing users and groups  6
masquerading  11
matching  13
meta package repository
    creating  19
meta RPM packages  21
monitoring processes  5
mount options
    for shared directories  42

N
NETWORK_CONFIG
    keyword in configuration file  45
NETWORK_NAT_FW_PORT_n
    keyword in configuration file  45
NETWORK_NAT_IF_BLADE
    keyword in configuration file  45
NETWORK_NAT_IF_HOST_EXT
    keyword in configuration file  45
NETWORK_NAT_IF_HOST_INT
    keyword in configuration file  45
    NFS
    exports  42
    NFS mount options  42
O
observe status 35
on virtual server 54
open files
 list 55
overview
 application integration technology 3

P
piping 32
port forwarding 11
port forwarding and masquerading 11
prerequisites 7
process
 remote 35
process monitoring 5
process status
 reporting 56
processes 60
processor activity 60
programs
 setup 31
pseudo terminal
 setup for programs that need 31
pseudo-terminal session control
 configuration file 46
PTY_DEFAULT
 example of setting 31
 keyword in configuration file 42

R
Red Hat Enterprise Linux, user accounts 13
reference 41
remote process
 displaying information 56
remote processes 35
remove
 x86 RPMs 21
restrictions i
route add 11
RPM
 managing 21
 meta package repository 19
 synchronizing virtual server with host 19
rpm command 8
RPM packages
 creating meta packages 58
 installing meta 21
RPMs
 synchronizing 6
run scripts on 54
running on virtual server 54
running programs
 on an attached x86 virtual server 32
running programs on the attached x86 virtual server 31
running scripts 34, 54

S
scripts 54
 setup 31
 starting with ai_exec 34
set up 11, 13
 attaching a virtual server 16
user accounts 28
setting 63
 user management system 27
setting up
 application integration 11
setting up scripts 31
setup 3, 11
shadow process 32
SHARE_n
 keyword in configuration file 42
shell scripts
 starting on an attached x86 virtual server 32
show 60
signals 32
software packages
 managing from System z 6
software repository
 creating 19
specifying 11
starting programs
 on an attached x86 virtual server 32
starting scripts 34
starting shell scripts
 on an attached x86 virtual server 32
synchronizing RPMs 6
System z 7

T
title section
 aiz_profiles.conf 41
troubleshooting i
tuning
 statement in configuration file 45

U
ulimit 31
ulimit command 31
updates
 installing 10
user accounts 13
 adding 28
user management system 63
 setting 27
useradd command 28
userdel command 28
users
 managing 27
users and groups 63
 managing 6
using the attached resources 31
using the x86 virtual server 31
V
virtual server 49, 51, 52, 54
  attaching 16
  processes running on 35
  processes, monitoring 5

X
X11 clients
  setup for 31
X11 forwarding
  setup for 31
x86 RPMs
  managing 21
x86 virtual server 7, 49, 52, 54
  attaching 16
  setting up for running programs or scripts 31
  using the attached 31
XZDPFSD_LISTEN
  keyword in configuration file 42
XZDPMNTD_CACHE
  keyword in configuration file 42
Readers’ Comments — We’d Like to Hear from You

Linux on System z
Integrating Applications with Linux on zEnterprise
Version 0.2

Publication No. SC34-2608-00

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.
Submit your comments using one of these channels:
• Send your comments to the address on the reverse side of this form.
• Send your comments via email to: eservdoc@de.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

Email address