

**JP1 Version 11**

## **Job Management: Getting Started (Scripting Language)**

**Overview and User's Guide**

**3021-3-B31(E)**

---

## Notices

### ■ Relevant program products

P-8112-B1BL JP1/Advanced Shell 11-00 (for Red Hat Enterprise Linux Server 6 (64-bit x86\_64), Red Hat Enterprise Linux Server 7 (64-bit x86\_64), Oracle Linux 6 (x64), Oracle Linux 7, CentOS 6, CentOS 7, SUSE Linux 12)

P-2A12-B1BL JP1/Advanced Shell 11-00 (for Windows 10, Windows Server 2012, Windows 8, Windows 7, Windows Server 2008)

This manual describes JP1/Advanced Shell, which is included in "Script language" as a subcategory of JP1 products. This manual does not describe JP1/Script.

### ■ Export restrictions

If you export this product, please check all restrictions (for example, Japan's Foreign Exchange and Foreign Trade Law, and USA export control laws and regulations), and carry out all required procedures.

If you require more information or clarification, please contact your Hitachi sales representative.

### ■ Trademarks

HITACHI, Job Management Partner 1, JP1 are either trademarks or registered trademarks of Hitachi, Ltd. in Japan and other countries.

Linux(R) is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

Red Hat is a trademark or a registered trademark of Red Hat Inc. in the United States and other countries.

SUSE is a registered trademark of Novell, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Windows is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Windows Server is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Other company and product names mentioned in this document may be the trademarks of their respective owners.

### ■ Restrictions

Information in this document is subject to change without notice and does not represent a commitment on the part of Hitachi. The software described in this manual is furnished according to a license agreement with Hitachi. The license agreement contains all of the terms and conditions governing your use of the software and documentation, including all warranty rights, limitations of liability, and disclaimers of warranty.

Material contained in this document may describe Hitachi products not available or features not available in your country.

No part of this material may be reproduced in any form or by any means without permission in writing from the publisher.



■ **Issued**

Jan. 2016: 3021-3-B31(E)

■ **Copyright**

All Rights Reserved. Copyright (C) 2016, Hitachi, Ltd.

## Preface

### ■ What you can do with JP1/Advanced Shell

This section describes what can be done with JP1/Advanced Shell.

#### Purposes of JP1/Advanced Shell

JP1/Advanced Shell is a product for improving development productivity and the operational efficiency of batch applications. It enables you to efficiently create and execute job definition scripts (*shell scripts*) for batch jobs.

JP1/Advanced Shell has the features described below.

#### Inheriting assets between the OSs of batch applications

- Using existing assets

You can use shell scripts created in a UNIX environment to develop job definition scripts in a Windows environment.

Because the job definition scripts used in JP1/Advanced Shell employ language specifications that have standard shell compatibility, it is easy to learn the language and migrate from existing shell scripts.

- Cross-platform support

*Cross-platform* means applicability to multiple OS bases. This feature enables you to use cross-platform functions.

- You can execute job definition scripts developed in a Windows environment in both Windows and UNIX environments.
- You can use UNIX-compatible commands in both Windows and UNIX environments.

#### Expediting the configuration of batch applications

- Controlling job execution

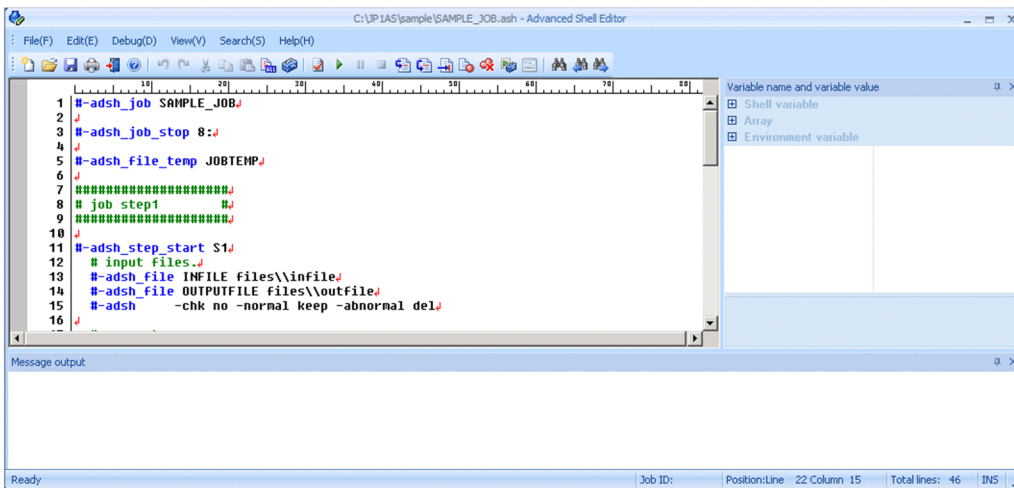
JP1/Advanced Shell extends job definition scripts so that you can automate and concisely code processes that are used repetitiously in batch applications.

You can reduce the volume of coding in job definition scripts and improve readability and maintainability of job definition scripts by doing the following:

- Specifying job step execution conditions
- Using variables that are valid in job steps
- Outputting error messages and setting return codes when batch jobs terminate with errors
- In the event a batch job terminates with an error, automatically terminating child processes forcibly and deleting temporary files used by the batch job
- Using an editor to develop job definition scripts (development environment)  
In the development environment, you can use the JP1/Advanced Shell Editor (a dedicated editor with debugging functions) of the Graphical User Interface (GUI) to develop and debug job definition scripts.
  - You can execute job definition scripts in job steps, and set breakpoints.
  - You can accumulate coverage information for job definition scripts.

The following figure shows the JP1/Advanced Shell Editor window.

## JP1/Advanced Shell Editor window



The screenshot shows the JP1/Advanced Shell Editor window. The main editor area contains a shell script with the following content:

```
1 #-adsh_job SAMPLE_JOB
2
3 #-adsh_job_stop 8:
4
5 #-adsh_file_temp JOBTEMP
6
7 #####
8 # job step1
9 #####
10
11 #-adsh_step_start S1
12 # input files
13 #-adsh_file INFILE files\infile
14 #-adsh_file OUTPUTFILE files\outfile
15 #-adsh -chk no -normal keep -abnormal del
16
```

On the right side, there is a panel titled "Variable name and variable value" with a tree view showing:

- Shell variable
- Array
- Environment variable

At the bottom of the window, there is a "Message output" panel which is currently empty. The status bar at the bottom indicates "Ready", "Job ID:", "Position:Line 22 Column 15", "Total lines: 46", and "INS".

- Efficient file allocation and postprocessing

You can automate and concisely code processes, such as checking for regular files, and allocating and deleting temporary files.

- You can automatically allocate temporary files during batch job execution and delete them once the batch job has terminated.
- You can check for regular files during batch job execution and perform appropriate postprocessing on files depending on job step or job processing results.

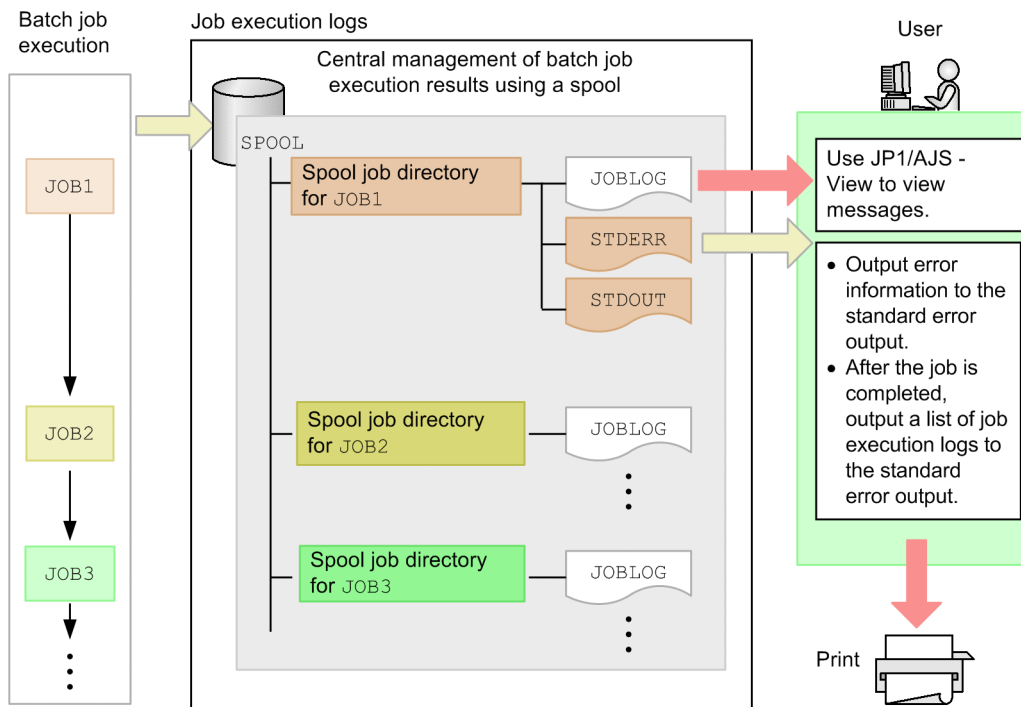
### Improving serviceability and maintainability by central management of batch job execution results

Maintainability of batch applications can be improved by automatically outputting job execution logs in the event of an error and managing such logs centrally.

In conventional open systems, management of batch job execution results is complicated because the results are not stored at one central location. JP1/Advanced Shell enables you to collect batch job execution results on a spool as job execution logs, and to manage them centrally. By using JP1/AJS - View, you can execute batch jobs on a periodic basis and reference the results by automatically executing job definition scripts.

Each job's execution results are output to a spool job directory under the spool directory. The following figure illustrates central management of batch job execution results.

## Central management of batch job execution results



For details about the output contents of the job execution logs, see [4.2.1 Job execution log](#).

### Example of application to a business operation

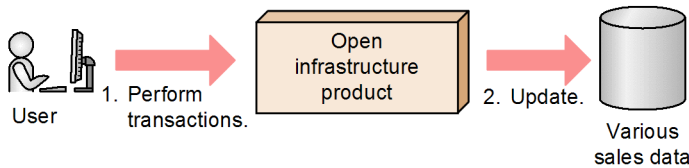
You can apply JP1/Advanced Shell to the following type of business operation.

In the case of an operation that involves many transactions in an online system during the daytime and totaling of the transactions at night, you can develop and execute batch jobs that obtain totals, including sales figures, number of products sold, and inventory updates. You can also develop and execute batch jobs for obtaining rolling totals, such as daily, monthly, and term-end processing, as well as batch jobs that have specific purposes and that are used for special occasions.

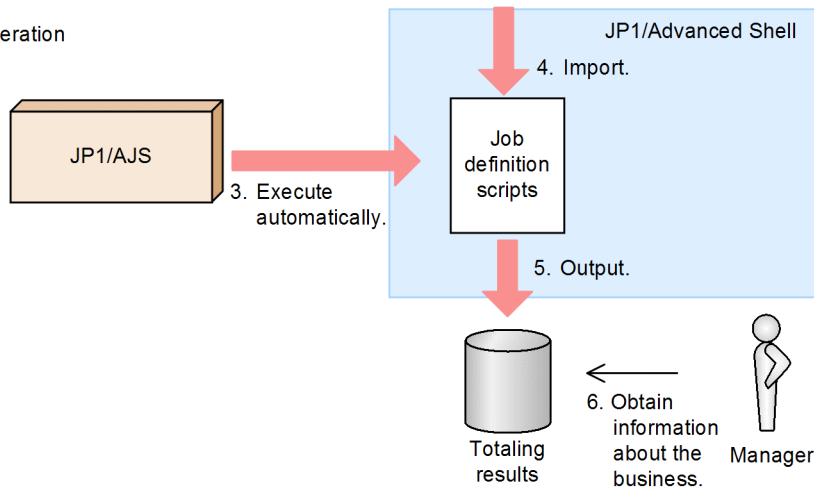
The following figure shows an example of JP1/Advanced Shell operation (for obtaining daily operation totals)

## Example of JP1/Advanced Shell operation (obtaining daily operation totals)

- Start of daily operation



- End of daily operation



To run JP1/Advanced Shell:

1. Start daily operation and perform transactions involving products.
2. The open infrastructure product updates the various sales data.
3. Daily operation ends and JP1/AJS issues instructions to execute job definition scripts automatically at specified times.
4. JP1/Advanced Shell executes job definition scripts to process the various sales data.
5. JP1/Advanced Shell outputs the execution results of the job definition scripts.
6. The manager can obtain information, including totals and changes in product sales, based on the execution results.

### ■ What is explained in this manual

This manual describes the basic way to set up and operate JP1/Advanced Shell. The purpose of this manual is to help the readers understand the concept of JP1/Advanced Shell and the basic way to use it.

This manual is intended for the following persons:

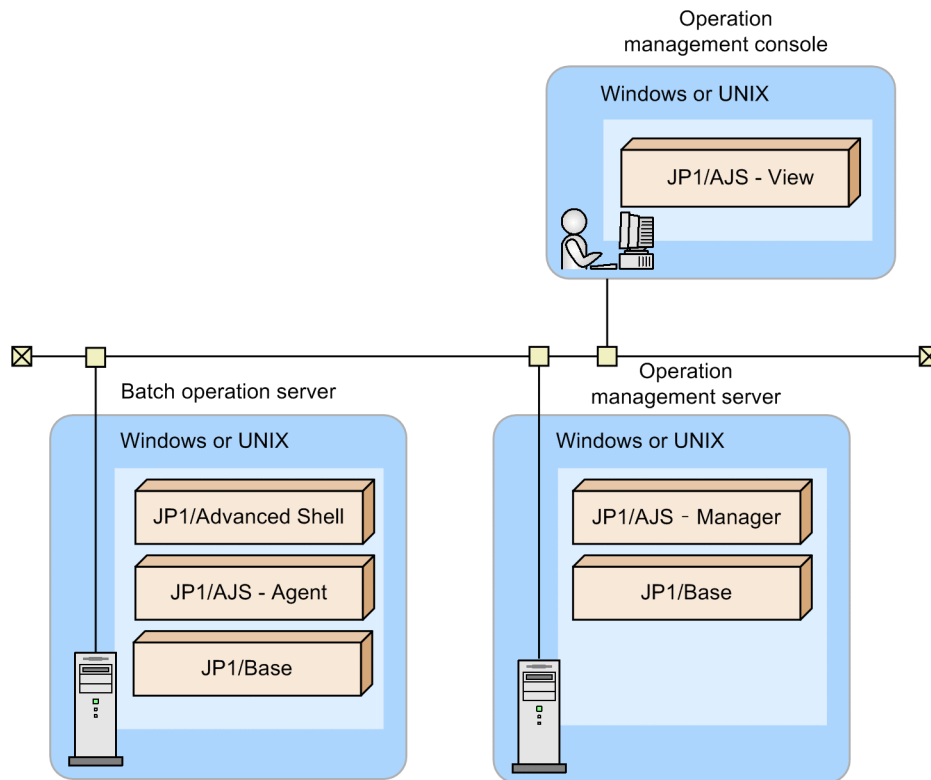
- Those who are considering installation of JP1/Advanced Shell
- Those who want to briefly understand the concept of JP1/Advanced Shell and get a feel for its operation, from setup to operation.

JP1/Advanced Shell consists of the following products:

- JP1/Advanced Shell (script execution base for batch operation)
- JP1/Advanced Shell - Developer (script execution base for batch jobs)

The basic guide provides information about JP1/Advanced Shell.

This manual assumes that the system configuration shown below is used. For operations in any other configuration, see the description of system configurations in the manual *JP1/Advanced Shell*.



The following describes the role of each system component:

- A batch operation server automatically executes job definition scripts, or allows users to manually execute them.
- An operation management server manages executed jobs.
- An operation management terminal displays job execution results by using JP1/AJS - View, and defines job definition scripts that are to be automatically executed.

## ■ How to read this manual

In addition to this basic guide, you can use the JP1/Advanced Shell manual *JP1/Advanced Shell*. This manual provides details about the system configuration setup and operation procedures which are not covered in the basic guide, and reference information such as commands and messages.

This manual describes the procedures for Windows and for Linux separately. In the description of a UNIX environment, replace "UNIX" with "Linux" except for "UNIX-compatible commands" and "UNIX jobs", which can be used as they are because they are function names.

A reference to another manual is written as follows: For details about something, see *topic-title* in the *manual-name*. Using *topic-title* as a keyword, search for the relevant section in the target manual.

Some windows in this manual might differ from the windows of your product because of improvements made without prior notice.



# Contents

Notices	2
Preface	4

## **1 Installing JP1/Advanced Shell and prerequisite products 11**

1.1	Procedure for installing JP1/Advanced Shell and prerequisite products	12
1.1.1	Installation procedure (for Windows)	12
1.1.2	Installation procedure (for Linux)	12
1.2	Preparations before installation	13
1.2.1	Prerequisite OSs	13
1.2.2	Memory and disk space required for installation	13
1.2.3	Setting the language for the prerequisite OS	13
1.2.4	Confirming the files that can be used in JP1/Advanced Shell	13
1.2.5	Checking the local time setting	15
1.3	Installing the prerequisite products	17
1.3.1	Installing JP1/AJS and related products	17
1.4	Installing JP1/Advanced Shell	18
1.4.1	Installing JP1/Advanced Shell (in Windows)	18
1.4.2	Installing JP1/Advanced Shell(to Linux)	18

## **2 Setting up an environment for JP1/Advanced Shell 20**

2.1	Specifying the environment files	21
2.1.1	Specifying the system environment files	21
2.1.2	Specifying the job environment files	21
2.1.3	Creating required directories	22
2.1.4	Specifying settings for using UNIX-compatible commands	24
2.1.5	Setting up the shell for starting jobs from JP1/AJS (in Linux)	25
2.1.6	Defining files to be started as child jobs	25
2.1.7	Outputting the contents of the job execution log by job type	26
2.1.8	Defining job execution results and log output information	28
2.1.9	Defining the return codes of extended script commands	32
2.1.10	Performing user-specific postprocessing when a job is terminated forcibly	32
2.2	Specifying environment variables	34
2.2.1	Defining the return code in the event of an unresumable error in a job	34

## **3 Defining jobs in JP1/AJS 36**

3.1	Defining jobs in JP1/AJS	37
3.1.1	Defining and executing a jobnet	37

- 3.1.2 Defining batch jobs as PC jobs 39
- 3.1.3 Defining batch jobs as UNIX jobs 41

## **4 Executing batch jobs and checking execution results 44**

- 4.1 Executing batch jobs 45
  - 4.1.1 Procedure for executing batch jobs 45
  - 4.1.2 Executing batch jobs 45
- 4.2 Checking batch job execution results 49
  - 4.2.1 Job execution log 49
  - 4.2.2 Checking batch job execution results 49
  - 4.2.3 Deleting spool jobs 52
- 4.3 Forcibly terminating batch jobs 54
  - 4.3.1 Forcibly terminating child or descendant processes 55
  - 4.3.2 Notes about operations including Ctrl+C (in Linux) 55

## **Appendixes 56**

- A Advanced Use 57
- B Reference Material for This Manual 58
  - B.1 Related publications 58
  - B.2 Abbreviations for Microsoft product names 58
  - B.3 Conventions: Fonts and symbols 59
  - B.4 Conventions: The JP1/Advanced Shell installation folder in Windows 60
  - B.5 Conventions: common application data folder 61
  - B.6 Conventions: Shared documents folder 61
  - B.7 Conventions: Windows menu names used in the manual 61
  - B.8 Conventions: Directory names 61
  - B.9 Abbreviations for product names 61
  - B.10 Conventions: Units (such as KB, MB, GB, and TB) 62
- C Glossary 63

## **Index 71**

# 1

## Installing JP1/Advanced Shell and prerequisite products

This section describes how to install JP1/Advanced Shell and prerequisite products.

## 1.1 Procedure for installing JP1/Advanced Shell and prerequisite products

---

The following describes the procedure for installing JP1/Advanced Shell and prerequisite products.

### 1.1.1 Installation procedure (for Windows)

The following describes the procedure for installing JP1/Advanced Shell and related prerequisite products in a Windows environment.

1. Install and set up the required products on the operation management server.
2. Install and set up the required products on the operation management console.
3. Install and set up the required products on the batch operation server.
4. Install JP1/Advanced Shell on the batch operation server and specify settings such as environment information.

### 1.1.2 Installation procedure (for Linux)

The following describes the procedure for installing JP1/Advanced Shell and related prerequisite products in a Linux environment.

To install the product and related programs:

1. Install and set up the required products on the operation management server.
2. Install and set up the required products on the operation management console in a Windows environment.
3. Install and set up the required products on the batch operation server.
4. Install JP1/Advanced Shell on the batch operation server and specify settings such as environment information.

## 1.2 Preparations before installation

---

The following describes the preparations required before installing JP1/Advanced Shell and related products.

### 1.2.1 Prerequisite OSs

- Red Hat Enterprise Linux Server 6 (64-bit x86\_64)
- Red Hat Enterprise Linux Server 7 (64-bit x86\_64)
- Oracle Linux 6 (x64)
- Oracle Linux 7
- CentOS 6 (x64)
- CentOS 7
- SUSE Linux 12
- Windows Server 2008
- Windows 7
- Windows 8
- Windows Server 2012
- Windows 10

### 1.2.2 Memory and disk space required for installation

A specific formula must be used to calculate the amount of memory and disk space required for installing JP1/Advanced Shell. For details, see the description in *Memory and Disk Space Requirements* in the manual *JP1/Advanced Shell*.

### 1.2.3 Setting the language for the prerequisite OS

The table below lists the language settings and encoding supported by JP1/Advanced Shell for each OS. Make sure that the language and encoding for files used in JP1/Advanced Shell match the language and encoding for the environment in which JP1/Advanced Shell runs. If they do not match, characters might be garbled or I/O data might be invalid.

Table 1-1: Language settings and encoding for each OS

OS	Language setting	Encoding
Windows	The system locale setting is not <b>Japanese (Japan)</b> .	-
Linux	The value of the LANG environment variable is C.	C


### 1.2.4 Confirming the files that can be used in JP1/Advanced Shell

The following lists the files that are used in JP1/Advanced Shell, and provides notes on specifying files and paths.

## (1) List of files used in JP1/Advanced Shell

The table below lists and describes the files that are used in JP1/Advanced Shell. To determine whether a file size can exceed 2 GB, see the description in *Files used in JP1/Advanced Shell* in the manual *JP1/Advanced Shell*.

Table 1-2: Files used in JP1/Advanced Shell

File name (icon)	Extension	File contents
Job definition script file (  )	.ash	A job definition script. The user can assign any file name.
Environment file <sup>#</sup>	.ase	JP1/Advanced Shell environment settings.
System environment file	.ase	System environment settings.
Coverage information file	.asc	Coverage environment information for JP1/Advanced Shell.
Debugging information file	.asd	Debugging information used by the editor (development environment)
System execution log <sup>#</sup>	.log	Log information that provides overall batch job execution logs for the system administrator.
Trace information <sup>#</sup>	.log	JP1/Advanced Shell's internal trace logs.
Temporary file	.tmp	Temporary file used internally by the system.
Coverage display temporary file	.txt	Temporary file used in displaying coverage information. The format of file name is as follows: <code>adshexec_view_job-definition-script-file-name_year-month-date_hour-minute-second.txt</code>
Start log (UNIX only)	.log	Log information that is collected when the user-reply functionality's management daemon is started and stopped.
pid file (UNIX only)	.pid	File used by the user-reply functionality management daemon and <code>adshmdctl</code> command.
Log of the application-execution agent functionality <sup>#</sup> (Windows execution environment only)	.log	Internal log of the application-execution agent functionality.

#

You can collect these files by using the `adshcollect` command. For details about how to collect the files, see the description in *adshcollect command* in the manual *JP1/Advanced Shell*.

### Notes about specifying files and paths

- As the directory delimiter, you can use a backslash (\)<sup>#</sup> for Windows or a forward slash (/) for UNIX. If you use other characters, the operation cannot be guaranteed.
  - If you use a backslash (\) as the directory delimiter for UNIX, the character will not be recognized as the directory delimiter and JP1/Advanced Shell will not operate correctly.
  - If you use a forward slash (/) as the directory delimiter for Windows, the character might be recognized as the directory delimiter. Note that, depending on how the forward slash is used, the character might not be recognized as the directory delimiter and JP1/Advanced Shell might not operate correctly.

#:

A backslash (\) specified in a job definition script is considered an escape character. For this reason, you need to specify two consecutive backslashes (\\) or enclose the character string that includes the backslash in single quotation marks (').

- Do not use a file name that begins with a dot (.).
- The permitted maximum length for path names must comply with the specifications of the OS being used.
- The maximum file name length is 246 bytes (Windows only).
- Do not use reserved device names (such as CON, AUX, and NUL) for file names (Windows only).
- Do not use NTFS streams for file names (Windows only).
- Do not use the junction functionality (Windows only).
- You can use UNC names for file names and path names (example: `\\computer-name\shared-name\file-name`); however, make sure that a path name specified in this format does not end with `shared-name` (or `shared-name\`). The `cd` standard shell command does not support the UNC format. (Windows only)

UNC formats that can be used:

```
\\server\share\dir
```

```
\\10.111.222.33\share\dir
```

UNC formats that cannot be used:

```
\\server\share
```

```
\\10.111.222.33\share
```

- Do not use UNC names for the folder path names for traces, system execution logs, spool, and temporary files (Windows only).

## (2) Notes on file systems

Be careful when using JP1/Advanced Shell with either of the following file systems:

- NFS  
Not supported.

- HSFS

When using HSFS, note the following:

- You cannot install JP1/Advanced Shell on HSFS.
- You cannot create a system execution log or traces on HSFS.
- If you use the user-reply functionality, you cannot specify a directory located on HSFS as the spool job directory.
- If you are using a version earlier than HSFS 07-00 and you want to use a UNIX-compatible command to reference or update the files and directories on HSFS, you will need to specify `NOCACHE` for the HSFS system option `CPFS_CACHE_POLICY` beforehand.
- If you are using HSFS 07-00 or later and you want to use a UNIX-compatible command to reference or update the files and directories on HSFS, you will need to specify `0` for the HSFS system option `CPFS_COMPAT_LINKCNT` beforehand. By default, `0` is specified for `CPFS_COMPAT_LINKCNT`.

### 1.2.5 Checking the local time setting

JP1/Advanced Shell obtains and outputs local time information by referencing environment variables. You must specify the local time settings in the environment variables beforehand.

The commands provided by JP1/Advanced Shell output information according to the OS's time zone setting (Windows) or the `TZ` environment variable (UNIX). Use one of the methods listed below to specify the `TZ` environment variable.

- JP1/AJS's job definition or environment variable definition
- System profile (/etc/profile)
- User profile (\$HOME/.profile)



## 1.3 Installing the prerequisite products

---

This section describes how to install JP1/AJS products (prerequisite products) and related products.

### 1.3.1 Installing JP1/AJS and related products

The following table shows the JP1/AJS products and related products.

Table 1-3: JP1/AJS and related products

Server type	Processing to be performed	Programs
Same batch operation server as for JP1/Advanced Shell	Executes job definition scripts from JP1/AJS	JP1/Base JP1/AJS - Agent <sup>#</sup>
Operation management server	Manages jobs	JP1/Base JP1/AJS - Manager <sup>#</sup>
Operation management console (Windows only)	Displays job execution results	JP1/AJS - View

#

JP1/AJS - Agent is not needed when JP1/AJS - Manager is installed on the same server as JP1/Advanced Shell, because JP1/AJS - Manager provides the JP1/AJS - Agent functions.

For details about how to install JP1/AJS and JP1/Base, see *Job Management: Getting Started (Job Scheduler)*.

## 1.4 Installing JP1/Advanced Shell

---

This subsection describes how to install JP1/Advanced Shell. A user with an administrator role must perform installation.

### 1.4.1 Installing JP1/Advanced Shell (in Windows)

This subsection explains how to perform a new installation of JP1/Advanced Shell.

To perform a new installation:

1. Log on as a user with an administrator role to the Windows machine on which JP1/Advanced Shell is to be installed.
2. Terminate all programs.
3. Place in the CD-ROM drive the CD-ROM that contains JP1/Advanced Shell.
4. Install JP1/Advanced Shell by entering required information as instructed by the installer.

The following information will be requested during installation:

- Product to be installed (JP1/Advanced Shell)
  - Customer Information
  - Destination Folder
5. When the Finish dialog box is displayed, click **Finish**.  
Installation is completed.

### 1.4.2 Installing JP1/Advanced Shell(to Linux)

The following describes how to install JP1/Advanced Shell to Linux from a CD-ROM.

Note that the directory and file names on the CD-ROM might be different from what is shown here, depending on the hardware environment. Use the `ls` command to check the file names and specify file names exactly as displayed.

To install JP1/Advanced Shell:

1. Specify the user permissions.  
Log on as a superuser to the server on which JP1/Advanced Shell is to be installed. Alternatively, use the `su` command to change the user permissions to superuser.
2. Terminate all programs.  
If any existing JP1-series programs and JP1/Advanced Shell program are running, terminate them.
3. Place the medium that contains JP1/Advanced Shell.
4. Mount the CD-ROM device by executing the following command:

```
/bin/mount -r -o mode=0544 /dev/cdrom /cdrom
```

`/cdrom` is the mount point of the CD-ROM device special file. If there is no mount point directory, create one. Note that the device special file name and mount point might differ depending on the environment.

5. Start the Hitachi Program Product Installer by executing the following command:

```
/cdrom/LINUX/setup /cdrom#
```

#: This example assumes `/cdrom` as the mount point.

The Hitachi Program Product Installer starts and the initial window is displayed.

The following is an example of the Hitachi Program Product Installer's initial window:

```
Hitachi PP Installer 05-24

L) List Installed Software.
I) Install Software.
D) Delete Software.
Q) Quit.

Select Procedure ==>
```

6. In the Hitachi Program Product Installer's initial window, enter I.

A list of programs that can be installed is displayed.

7. Select JP1/Advanced Shell, and then enter I.

JP1/Advanced Shell is installed. To select a program, move the cursor to the desired program, and then press the space bar to select it.

The following shows an example of the Hitachi Program Product Installer's installation window:

```
      PP-No.          VR      PP-NAME
<@>001 P-8112-B1BL    1100   Advanced Shell
:
:
F) Forward B) Backward J) Down K) Up Space) Select/Unselect I) Install Q) Quit
```

<@> is displayed to the left of the selected program product. If you enter I following <@>, the following message is displayed on the last line:

```
Install PP? (y: install, n: cancel)==>
```

If you enter y or Y, installation begins. If you enter n or N, installation is cancelled and the program product installation window is displayed again.

8. When installation is completed successfully, enter Q.

The Hitachi Program Product Installer's initial window is displayed again.

Note that the following files are created during installation as installer's logs:

```
/opt/jpllas/instlog/ADSH_INST_LOG
/opt/jpllas/instlog/ADSH_INST_USERLOG
```

If the installer's log files are not created, possible causes are as follows:

- The installer's log files are not regular files.
- The user does not have write permission for the directory in which the installer's log files are to be created.
- A file with the same name already exists at the path of each log file of the installer.

A file with the same name exists in the following cases:

- "/opt" is not a directory.
- "/opt/jpllas" is not a directory.
- "/opt/jpllas/instlog" is not a directory.

# 2

## Setting up an environment for JP1/Advanced Shell

This section describes how to set up an environment for JP1/Advanced Shell.

## 2.1 Specifying the environment files

The two types of environment files are system environment files and job environment files. The supported parameters are the same. The following table explains each type of file.

Table 2-1: Types of environment files

Type of environment file	Description
System environment file	An environment file of this type is common to all systems and is specified by the system administrator. This environment file is used automatically when it is stored in the predefined directory.
Job environment file	This environment file is specified for each job by the developer. Specify this file in the following cases: <ul style="list-style-type: none"><li>• Environment file specified in the <code>ADSH_ENV</code> environment variable</li></ul>

Job controllers use the information defined in system environment files and job environment files.

All directories specified in parameters in a system environment file must exist. If you wish to change the default directories, you must create the new directories beforehand.

If you have edited a system environment file in a UNIX environment, check that there are no errors by executing the `adshmdctl` command with the `conftest` option specified.

The following subsections explain how to specify each environment file.

### 2.1.1 Specifying the system environment files

The system administrator creates and specifies the system environment files. The created system environment files take effect when they are stored on the file paths specified in the following table.

Table 2-2: File names of system environment files

Environment	File name of the system environment file
Windows (execution environment)	<b><i>common-application-data-folder</i></b> \HITACHI\JP1AS\JP1ASE\conf\adshrc.ase
UNIX	/opt/jp1as/conf/adshrc.ase

### 2.1.2 Specifying the job environment files

To use a job environment file to execute batch jobs, specify the file path in the `ADSH_ENV` environment variable. Use the procedure described below to create and specify a job environment file.

To create and specify a job environment file:

1. Copy the `sample.ase` environment file sample data from the following directory to a desired directory and file:
  - Windows execution environment  
`installation-folder\JP1ASE\sample\sample.ase`
  - UNIX execution environment  
`/opt/jp1as/sample/sample.ase`

2. Specify the required parameters in the copy of the job environment file.
3. Specify the path of the created job environment file in the `ADSH_ENV` environment variable so that the job environment file can be used during batch job execution.

Use one of the following methods to specify the `ADSH_ENV` environment variable:

- OS setting (Windows only)
- System profile `/etc/profile` (UNIX only)
- User profile (`$HOME/.profile`) (UNIX only)

#

Do not use any of the following characters in a job environment file directory or file name: `& ( ) [ ] { } ^ = ; ! ' + , ` ~ # %`. If any of these characters is used, JP1/Advanced Shell will not function normally.

### 2.1.3 Creating required directories

If you want to change the default settings for the directories required for execution after you have installed JP1/Advanced Shell, you must create new directories, and then specify them in the environment files.

The directories required for JP1/Advanced Shell and the information to be specified are described below. The user who will be running JP1/Advanced Shell must grant the required permissions to these directories.

- Directory for temporary files  
Specify the directory in which the files to be used only within batch jobs are to be created temporarily.
- Directory for the spool  
Specify the directory used to store job execution logs and program output data files.
- Directory for system execution logs  
Specify the directory used to store batch job logs as system execution logs that are used by the system administrator for monitoring execution of batch jobs.
- Directory for traces  
Specify the directory used to store the statuses for troubleshooting purposes in the event of system failure.

The table below lists the directories required in JP1/Advanced Shell.

Table 2-3: Directories required in JP1/Advanced Shell

Directory	Environment setting parameter	Default directory or path	Default permissions
Directory for temporary files	<code>TEMP_FILE_DIR</code>	<ul style="list-style-type: none"> <li>• Execution environment (Windows only) <i>shared-documents-folder</i>\Hitachi\JP1AS\JP1ASE\temp</li> <li>• Execution environment (UNIX only) <code>/var/opt/jplias/temp</code></li> </ul>	CRWD (Windows) 1777 (UNIX)
Directory for the spool	<code>SPOOL_DIR</code>	<ul style="list-style-type: none"> <li>• Execution environment (Windows only) <i>shared-documents-folder</i>\Hitachi\JP1AS\JP1ASE\spool</li> <li>• Execution environment (UNIX only) <code>/var/opt/jplias/spool</code></li> </ul>	CRWD (Windows) 1777 (UNIX)
Directory for system	<code>LOG_DIR</code> <code>LOG_FILE_CNT</code>	<ul style="list-style-type: none"> <li>• Execution environment (Windows only) <i>shared-documents-folder</i>\Hitachi\JP1AS\JP1ASE\log</li> </ul>	CRWD (Windows) 0777 (UNIX)

Directory	Environment setting parameter	Default directory or path	Default permissions
execution logs	LOG_FILE_SIZE	<ul style="list-style-type: none"> <li>Execution environment (UNIX only) /opt/jplab/log</li> </ul>	CRWD (Windows) 0777 (UNIX)
Directory for traces	TRACE_DIR TRACE_FILE_CNT TRACE_FILE_SIZE TRACE_LEVEL	<ul style="list-style-type: none"> <li>Execution environment (Windows only) <i>common-application-data-folder</i>\Hitachi\JP1AS\JP1ASE \trace</li> <li>Execution environment (UNIX only) /opt/jplab/trace</li> </ul>	CRWD (Windows) 1777 (UNIX)

Legend:

The letters shown in the *Default permissions* column indicate the following Windows permissions:

C: Create, R: Read, W: Write, D: Delete

## (1) Required permissions

This subsection describes the permissions required for the users who execute batch jobs.

### (a) In Windows

Grant full control to the users who will be executing batch jobs.

### (b) In UNIX

Grant to the users who will be executing batch jobs the file permissions shown below for each type of directory.

Table 2-4: File permissions for directories

Directory type	Read permission (r)	Write permission (w)	Execution permission (x)	Sticky bit (t)
Directory for temporary files	R	R	R	S
Directory for the spool	R	R	R	S
Directory for system execution logs	R	R	R	N
Directory for traces	R	R	R	S

Legend:

R: Specification is required.

S: Specify according to system operation guidelines.

N: Do not specify.

Specify the sticky bit for directories according to the system operation guidelines.

If no sticky bit is specified for a directory for which a user has write permission, that user can delete any file directly under that directory.

If a sticky bit is specified for a directory, only the owner of the directory or files can delete any file directly under that directory. No other user can delete these files even if the user has write permission for the directory.

## (2) File systems

Because the size of the spool might become large depending on the applications, we recommend that you create and use a dedicated file system.

### 2.1.4 Specifying settings for using UNIX-compatible commands

#### (1) Definitions for using executable UNIX-compatible commands in existing job definition scripts

If you will be using executable UNIX-compatible commands in existing job definition scripts, set the path to the directory in which the UNIX-compatible commands are installed in the `PATH` environment variable. This method eliminates the need for correcting the existing job definition scripts. If there is a command having the same name as a UNIX-compatible command, you can always run the UNIX-compatible command in JP1/Advanced Shell's job definition scripts by specifying the path at the beginning of the `PATH` environment variable value by using the `export` parameter in the environment file.

Before you run your job definition scripts, make sure that the correct paths have been set in each environment in which the job definition scripts are to be run.

#### (2) Preparations for using the script-format UNIX-compatible commands (in Windows)

The script-format UNIX-compatible commands use sample script files provided by JP1/Advanced Shell.

Execute the script-format UNIX-compatible commands (such as `chmod` and `su`) according to the sample script file provided by JP1/Advanced Shell.

To execute script-format UNIX-compatible commands:

1. Copy to a desired folder the files that you will be using of the sample script files stored at the following location:  
**installation-folder** \JP1ASE\sample
2. Rename the copied files to applicable command names.  
For example, rename sample script files `script_chmod1` and `script_su1` as `chmod` and `su`, respectively. If you want to define a command that does nothing, copy sample script file `script_0` and then rename it.
3. To specify only the file name of the sample script, not its absolute or relative path, do either of the following:
  - Store the sample script to be run in the folder defined in the `PATH` environment variable.
  - Add to the `PATH` environment variable the path of the folder containing the sample script that is to be run.
4. If necessary, define `KNAX6831-I` message output suppression.  
If you do not want the `KNAX6831-I` message to be output after the sample script has run, specify the following coding in the job environment file:

```
#-adsh_conf JOBLOG_SUPPRESS_MSG      KNAX6831-I
```

If you want to suppress output of the `KNAX6831-I` message for all job definition scripts in the system, specify the above coding in the system environment file.

5. Run the job definition scripts.



Run the job definition scripts by using the job environment file created in step 4. If you specified the definition in the system environment file, the information specified in step 4 is imported automatically.

## 2.1.5 Setting up the shell for starting jobs from JP1/AJS (in Linux)

The table below shows the login shell used when jobs are started from JP1/AJS. Specify the settings so that the correct login shell can be used.

OS type	Login shell
Linux	bash

*Notes:*

If the `adshexec` command is run as a child process of the login shell when a job is started from JP1/AJS and then forced termination occurs, the login shell's processing might terminate before the `adshexec` command's job execution results are passed to JP1/AJS. If this happens, the job execution results might not be applied to JP1/AJS - View.

To avoid this, first (before starting) check the definitions in the login script file to verify that the login shell's process is overwritten (such as by deleting the `trap` command specification).

For details about the definitions, see the manual *Job Management Partner 1/Automatic Job Management System 3 Configuration Guide 1* or *Job Management Partner 1/Automatic Job Management System 3 Troubleshooting*.

## 2.1.6 Defining files to be started as child jobs

You can specify a job definition script as a command name in another job definition script. This enables you to run a job definition script specified in the `adshexec` command as a JP1/Advanced Shell job. This feature is useful in the following cases:

- Migrating a user's existing asset shell scripts from a UNIX environment to a Windows environment
- Executing an existing shell script that is run in the OS's shell in a UNIX environment as a JP1/Advanced Shell job without rewriting its contents

Of the job definition scripts that are executed as descendant processes, those jobs that are executed by using specific environment setting parameters are called child jobs. A job that is executed from JP1/AJS or a login shell and that is not a child job is called a root job.

If you will be starting job definition script files as child jobs, you must specify in an environment file the conditions for the files to be used. The following provides an overview of the environment setting parameters.

- `CHILDJOB_EXT` parameter  
Defines the extension for a job definition script file that is to be executed as a child job.
- `CHILDJOB_PGM` parameter  
Defines the path to be replaced so that a job definition script file is executed as a child job.
- `CHILDJOB_SHEBANG` parameter  
Defines the path of the executable program of the job definition script file that is to be executed as a child job.

A job definition script file that you create that satisfies the default definition for the `CHILDJOB_SHEBANG` parameter is run as a child job.

## Important note

If you want to run both root and child jobs by using the same environment file parameters, do not change the `ADSH_ENV` environment variable values or the contents of the environment files during job execution.

## 2.1.7 Outputting the contents of the job execution log by job type

What is output to the job execution log depends on the type of job that is executed, as described in the following subsections.

### (1) Destination and output contents of the job execution log when root jobs are executed

This subsection explains the destination and output contents of the job execution log when root jobs are executed in expansion output mode or minimum output mode (specified in the `OUTPUT_MODE_ROOT` parameter).

#### (a) When the expansion output mode is selected

The following table describes the output contents of the job execution log when the expansion output mode is selected:

Message output destination	Description
<code>JOBLOG</code>	Output to spool files.
Script image	Output to spool files.
Destination of the standard output	Output to the destination specified by either of the following methods: <ul style="list-style-type: none"><li>• <code>-s</code> option in the <code>adshexec</code> command</li><li>• <code>OUTPUT_STDOUT</code> parameter in the environment file</li></ul>
Destination of the standard error output	Output to spool files.

A spool job directory is created for each job.

After job execution, the contents of the job execution log, excluding the contents for the standard output, are output to the standard error output.

#### (b) When the minimum output mode is selected

The following table describes the output contents of the job execution log when the minimum output mode is selected:

Message output destination	Description
<code>JOBLOG</code>	Messages that are not subject to output suppression are output to spool files.
Script image	Output of some messages is suppressed in the minimum output mode. For details regarding messages that are suppressed, see <i>Contents of output of job execution log for each job in JP1/Advanced Shell</i> .
Destination of the standard output	Not output to spool files. This information is output to the destination in effect when the process started.
Destination of the standard error output	Messages that are not subject to the suppression of the standard error output and the standard output are output. Also, messages for <code>JOBLOG</code> that are not subject to output suppression are output to the standard error output.

Note that the job execution log is not output to the standard error output when the job terminates.

## (2) Destination and output contents of the job execution log when child jobs are executed

The output destination and the output contents of the job execution log when executing the child jobs are separately described for each the expansion output mode and the minimum output mode (specified with the OUTPUT\_MODE\_CHILD parameter). This command indicates the output contents when the spool job of the child job is being merged into the spool job of the root job. For details regarding output contents when deleting the spool job of a child job, see *Contents of output of job execution log for each job* in JP1/Advanced Shell.

### (a) When the expansion output mode is selected

The following table describes the output contents of the job execution log when the expansion output mode is selected:

Message output destination	Description
JOBLOG	The child job is temporarily output to the file in the spool. JOBLOG of the child job is merged into JOBLOG of root job when the child job finishes. The symbol indicating the start of output ">>>>>> [JOBLOG] path-name" and the symbol indicating the end of output "<<<<<< [JOBLOG] path-name" are output before and after JOBLOG of the child job.
Script image	The child job is temporarily output to the file in the spool. The script image of the child job is merged into the script image of the root job when the child job finishes.
Destination of the standard output	Output to the destination in effect when the process started.
Destination of the standard error output	Output to the destination in effect when the process started. The symbol indicating the start of output ">>>>>> [STDERR] path-name" and the symbol indicating the end of output "<<<<<< [STDERR] path-name" are output before and after the standard error output of a child job.

A spool job directory of the child job is created while a job is being executed, but the directory is deleted after the job is executed.

In addition, the following header lines are not merged.

```

-----
Advanced Shell version-number

[Information]
Job ID           : job-ID
Spool directory : spool-job-directory-path
Date            : execution-date
EnvFile(system) : environment-file-path (System environment file)
EnvFile(job)    : environment-file-path (Job environment file)
Host name       : host-name
[Environment variable from Automatic Job Management System]
environment-variables-passed-from-JP1/AJS
-----
***** JOB CONTROLLER MESSAGE *****

```

### (b) When the minimum output mode is selected

The following table describes the output destination of the job execution log when the minimum output mode is selected:

Message output destination	Description
JOBLOG	<p>Messages that are not subject to the suppression are temporarily output to the file in the spool of the child job.</p> <p>When a message is output, JOBLOG of the child job is merged into JOBLOG of the root job and symbols indicating the start of the output of JOBLOG "&gt;&gt;&gt;&gt;&gt;&gt; [JOBLOG] path name" and symbols indicating the end of output "&lt;&lt;&lt;&lt;&lt;&lt; [JOBLOG] path name" are output before and after JOBLOG of the child job.</p> <p>If no messages have been output, JOBLOG of the child job is not merged into JOBLOG of the root job.</p> <p>Output of some messages is suppressed in the minimum output mode. For details regarding messages that are suppressed, see <i>Contents of output of job execution log for each job</i> in <i>JP1/Advanced Shell</i>.</p>
Script image	Although messages are temporarily output to a file in the spool, these messages are not merged into the script image of the root job.
Destination of the standard output	<p>Output to the destination in effect when the process started.</p> <p>This command outputs messages that are not subject to suppression of the output for the standard output.</p> <p>Messages whose output is suppressed are not output.</p>
Destination of the standard error output	<p>Output to the destination in effect when the process started.</p> <p>This command outputs messages that are not subject to suppression of the output for the standard error output. In addition, this command outputs messages that are not subject to suppression of the output of JOBLOG.</p> <p>Messages whose output is suppressed are not output.</p>

Although a spool job directory of the child job is created, this directory will be deleted after executing the job.

## 2.1.8 Defining job execution results and log output information

Job execution results are output to the spool directory. You can reference some of the output information as job execution logs. In the event of a problem, you can collect logs and investigate the cause of the problem. In the environment file, define the output destination and contents of these logs.

The following table lists the types of log information that are output while JP1/Advanced Shell is running, and where each type is stored.

Table 2-5: Log information output while running JP1/Advanced Shell and the storage locations for the information

Log information	Information that is output	Storage location
Job execution log	Log of batch jobs	Under the spool root directory
System execution log	Comprehensive JP1/Advanced Shell execution log	Directory specified by the LOG_DIR parameter# in the environment file

#

If the parameter is omitted, the default value is used.

The following subsections explain the spool output information and how to define output information for each log.

### (1) Defining the spool output information

This subsection explains the spool-related parameters for each output information to be defined.

## (a) Determining whether the spool job creation suppression functionality is to be used

The spool job creation suppression functionality enables you to prevent the spool directory's disk space usage from increasing continually. It also eliminates the need to delete unneeded directories and files from the spool directory.

Use the `SPOOLJOB_CREATE` parameter to enable the spool job creation suppression functionality. For details, see the description in *SPOOLJOB\_CREATE parameter* in the manual *JP1/Advanced Shell*.

While using the spool job creation suppression functionality, the root job and child jobs will always operate in the minimum output mode.

The spool directory is necessary even when the spool job creation suppression functionality is used.

## (b) Defining the path name of the spool root directory

The following parameter is used for defining the path name of the spool root directory:

- `SPOOL_DIR` parameter: Defines the path name of the spool root directory.

## (c) Changing the spool job directory or file permissions (UNIX only)

When a job is terminated, its execution results are output to the spool job directory created for that job. You can use the following parameters to change the permissions for the directory or the files under that directory:

- `PERMISSION_SPOOLJOB_DIR` parameter  
Specify this parameter to change permissions for the spool job directory.  
The default is 700.
- `PERMISSION_SPOOLJOB_FILE` parameter  
Specify this parameter to change permissions for the files under the spool job directory.  
The default is 600 (in `.DBG` files, 666).

## (d) Defining the standard output and standard error for spool jobs

When a job is executed, JP1/Advanced Shell's information messages, warning messages, and job execution logs are output in addition to the job execution results. The standard output and the standard error output are output to files under the spool job directory.

Specify the minimum output mode in the following parameter or command option to suppress output of the standard output and the standard error output to files in the spool job directory, in situations such as when you are using only the results of a job with another program.

- `OUTPUT_MODE_ROOT` parameter  
Specify the expansion output mode or the minimum output mode for the root job.
- `OUTPUT_MODE_CHILD` parameter  
Specify the expansion output mode or the minimum output mode for the child jobs.
- `-m` option of the `adshexec` command  
Specify the expansion output mode or the minimum output mode for the jobs.
- `-m` option of the `adshscripttool` command  
Specify the minimum output mode for the child jobs.

If these parameters and option are omitted, expansion output mode is assumed, in which case the standard output and the standard error are output to files under the spool job directory.

In the minimum output mode, information messages and warning messages of JP1/Advanced Shell are not output to the standard output or the standard error output. Also, when jobs are terminated, job execution logs are not output to the standard error output. In addition, in the minimum output mode, messages whose output is suppressed are not output to the job execution logs under spool job directories.

For details regarding the differences in output messages among the expansion output mode and the minimum output mode, see *Suppressing output of information and warning messages to job execution logs* in *JP1/Advanced Shell*.

## **(2) Defining the information to be output to the job execution log**

This subsection explains information related to the job execution log that is to be specified during the environment setup. For details about the information that is output to the job execution log, see [4.2.1 Job execution log](#).

### **(a) Defining the types of job execution logs to be output to the standard error**

When a job is terminated, the information listed below is output as job execution logs to the standard error. The output job execution logs are displayed on the terminal screen used when the `adshexec` command is executed, and in JP1/AJS - View's Execution Results Details dialog box.

- JOBLOG file (Messages indicating the job's execution status, including command execution results and file allocation results)
- Job definition script
- Contents of the standard error during job execution

To output only the contents of the standard error during job execution to the standard error, specify the parameter shown in the following to limit the contents of job execution logs to be output:

- `JOBEXECLOG_PRINT` parameter

When the job has been executed in the minimum output mode, the job execution log will not be output to the standard error output when the job finishes regardless of the specification of the `JOBEXECLOG_PRINT` parameter.

## **(3) Defining the information to be output to the system execution log**

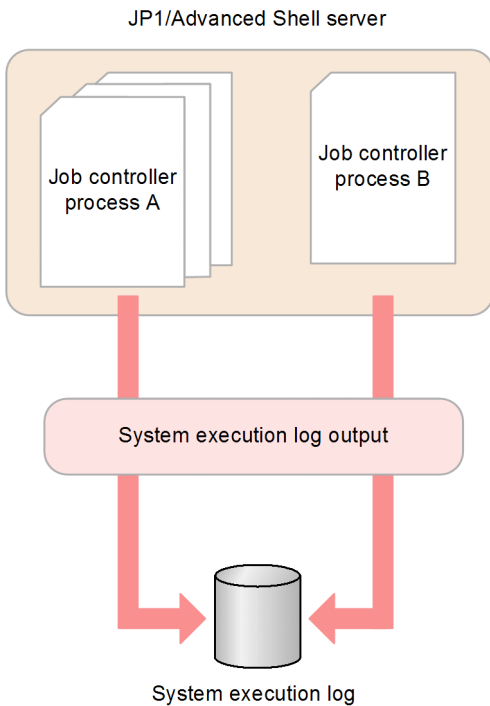
The system execution log provides system administrators with a comprehensive execution history of batch jobs.

The log information is output to `AdshLog.log` under the directory specified by the `LOG_DIR` parameter in the environment file. The files are swapped (`AdshLog_1.log`, `AdshLog_2.log`, ..., `AdshLog_N.log`) according to conditions (such as maximum file size) specified in parameter settings. Because a new system execution log file is created when log files are swapped, the owner of each file will be the user at the time swapping occurs.

### **(a) Flow of output to the system execution log**

The system execution log is the destination for log information about the batch jobs running in each job controller process. You can specify in the environment files the output destination for the system execution log, as well as parameters that control log file swapping (such as maximum file size and number of files). The following figure shows the flow of output to the system execution log.

Figure 2-1: Flow of output to the system execution log



The system execution log is created as follows.

- Messages to be output to the system execution log are collected and output in CSV format.  
For details about the messages that are output, see the description in *Message output destinations* in the manual *JP1/Advanced Shell*.
- In time, log file swapping is performed and a backup is created.
  - Just before it exceeds the file size specified in the `LOG_FILE_SIZE` parameter in the environment file, the current system execution log file is renamed so that it becomes a backup file, and a new system execution log is created and message output continues to it.
  - The file name of the backup will be `AdshLog_N.log` (where *N* is an integer). *N* is assigned a number in ascending order from the newest backup, starting from 1.
  - The maximum number of backups to be created is specified in the `LOG_FILE_CNT` parameter in the environment file. When the number of backup files exceeds this value, the oldest backup file is deleted.

## (b) Parameters required to output the system execution log

The following parameters are used for outputting system execution logs:

- `LOG_DIR` parameter: Defines the path name of the directory to which system execution logs are to be output.
- `LOG_FILE_CNT` parameter: Defines the number of files used for backing up system execution logs.
- `LOG_FILE_SIZE` parameter: Defines the file size for output of system execution logs.

If multiple users output system execution logs to the same file, the `LOG_FILE_CNT` and `LOG_FILE_SIZE` parameter values specified by the last user who starts output of system execution logs take effect. Therefore, we recommend that you use the same value for `LOG_FILE_CNT` and `LOG_FILE_SIZE`.

## (c) Contents of the system execution log

The following shows an example of a message output to the system execution log:

```
seqnum=1, date=2013-12-06T10:41:19.242+09:00, pgmid=adshexec, jobid=6, pid=2720,
msgid=KNAX0004-I, msg="Job ID=000006, JP1NBQSQueueName=, scheduler job ID="
seqnum=2, date=2013-12-06T10:41:19.250+09:00, pgmid=adshexec, jobid=6, pid=2720,
msgid=KNAX0091-I, msg="JOB1 The job started."
seqnum=3, date=2013-12-06T10:41:19.251+09:00, pgmid=adshexec, jobid=6, pid=2720,
msgid=KNAX7901-I, msg="The adshexec command will wait for all asynchronous processes
at the end of the job."
seqnum=4, date=2013-12-06T10:41:19.251+09:00, pgmid=adshexec, jobid=6, pid=2720,
msgid=KNAX7902-I, msg="The adshexec command will run in tty stdin mode."
```

The following table lists and explains the data items that are added in front of the message texts in the system execution log:

Data items output to the system execution log	Meaning
seqnum	Message's serial number
date	Output date and time (in the format <i>yyyy-mm-ddThh:mm:ss.sssTZD</i> )
pgmid	Program ID In a job controller, <i>adshexec</i> is output.
jobid	Job ID
pid	Process ID
msgid	Message ID of the output message
msg	Message text of the output message

### 2.1.9 Defining the return codes of extended script commands

The following parameters are used to change the default values for the return codes that indicate whether extended script commands failed or were successful:

- `ADSHCMD_RC_ERROR` parameter: Defines the return code to be used to indicate that an extended script command failed.
- `ADSHCMD_RC_SUCCESS` parameter: Defines the return code to be used to indicate that an extended script command was successful.

For details, see the description in *Return codes of extended script commands and handling of errors* in the manual *JP1/Advanced Shell*.

### 2.1.10 Performing user-specific postprocessing when a job is terminated forcibly

JP1/Advanced Shell enables you to perform user-specific postprocessing when a forced termination request is received from JP1/AJS by means of the UNIX `SIGTERM` signal or the Windows `taskkill` command (immediate termination of process by a means such as `TerminateProcess`). This feature enables the user to enhance operational flexibility by



performing user-specific termination processing when a forced termination request is received. You must define the `TRAP_ACTION_SIGTERM` environment setting parameter in order to perform user-specific postprocessing when a forced termination request is received.

Note that the operand supported by the `TRAP_ACTION_SIGTERM` environment setting parameter is different between the UNIX edition and the Windows edition. For details about the `TRAP_ACTION_SIGTERM` environment setting parameter, see the description in *TRAP\_ACTION\_SIGTERM parameter* in the manual *JP1/Advanced Shell*.

The following shows an example:

#### Contents of the environment variable

```
#-adsh_conf TRAP_ACTION_SIGTERM TERM
```

#### Contents of the job definition script

```
#-adsh_job JOB01
trap "UAP_TERM" TERM
UAP01
```

➔ If a forced termination request is received while `UAP01` is running, the job controller executes `UAP_TERM`, performs postprocessing (such as deleting allocated files and forcibly terminating descendant processes), and then terminates the job.

## 2.2 Specifying environment variables

The table below lists and describes the environment variables supported by JP1/Advanced Shell.

### Important note

JP1/Advanced Shell sets and references shell and environment variables whose names begin with `ADSH`. Therefore, do not use a shell variable or an environment variable whose name begins with `ADSH` for any purpose other than those described in this manual.

Table 2-6: Environment variables supported by JP1/Advanced Shell

Environment variable name	Information to be specified	Timing of specification when the value is set automatically	Whether a value can be specified
<code>ADSH_ENV</code>	Job environment file name	When the job starts as a custom job	Yes <sup>#</sup>
<code>ADSH_JOBRC_FATAL</code>	Job return code in the event of a fatal error that interrupts job processing such as syntax errors. For details about how to specify the environment variable, see (1) <code>ADSH_JOBRC_FATAL</code> environment variable (specifies the return code in the event of an unresumable error in jobs).	(Not specified automatically)	Yes <sup>#</sup>
<code>AJS_BJEX_STOP</code>	Interface used for forced termination from JP1/AJS. This environment variable must be defined when JP1/Advanced Shell batch jobs are defined in PC or UNIX jobs. Define the environment variable in PC or UNIX job definitions, not in OS settings.	When the job starts as a custom job	Yes (Only <code>TERM</code> is permitted.)

#

If an environment variable is set within a job definition script or an environment file, the value of such an environment variable is valid only for a child job or root job that is started from a job definition script.

For details about environment variables, see the description in *Specifying environment variables* in the manual *JP1/Advanced Shell*.

### 2.2.1 Defining the return code in the event of an unresumable error in a job

If a job is terminated due to an error, such as a memory shortage or a job definition script parsing error, the job controller's return code is set to 1. You can change this return code to any value from 1 to 255 by setting a value in the `ADSH_JOBRC_FATAL` environment variable.

For details about setting method and applicability of the `ADSH_JOBRC_FATAL` environment variable, see *ADSH\_JOBRC\_FATAL environment variable* in *JP1/Advanced Shell*.

## (1) ADSSH\_JOBRC\_FATAL environment variable (specifies the return code in the event of an unresumable error in jobs)

The `ADSSH_JOBRC_FATAL` environment variable is used to specify the job controller's return code in the event a job becomes unresumable and is terminated with an error. The specified end code is applied to jobs that are executed by using the `adshexec` command.

The following shows how to apply the value of this environment variable globally in the entire system:

- Windows  
Define `ADSSH_JOBRC_FATAL` as a system environment variable.
- UNIX  
Specify the `ADSSH_JOBRC_FATAL` environment variable setting in `/etc/profile`.

If this environment variable is not specified and a job terminates with an unresumable error, the controller's return code is set to 1.

### (a) Values permitted in the environment variable

*termination-code* ~<unsigned integer> ((1 to 255))

Specifies the return code to be set when a job cannot be resumed. If the value is padded with leading zeros such as 001, the leading zeros are deleted and the value is treated as being 1.

### (b) Notes

- If the `ADSSH_JOBRC_FATAL` environment variable is defined by using the `export` parameter in the environment file or the `ADSSH_JOBRC_FATAL` environment variable is defined or changed within a file or a job definition script specified in the `ENV` shell variable, this functionality does not take effect within that job. The functionality takes effect on another job that is started from that job.
- The `ADSSH_JOBRC_FATAL` environment variable defines the final return codes for jobs. It does not affect the return codes of individual commands and job steps.
- If any of the following values is set, the job terminates, without being executed, with an error with return code 255:
  - Value consisting of four or more characters (example: 1234)
  - Value outside the permitted range (example: 500)
  - Non-numeric characters (example: 1A4, +8, 8.0)
  - Value consisting of no character (null character string)
- Whether the `ADSSH_JOBRC_FATAL` environment variable is applied in the event of an error depends on each job. If an unresumable error occurs only within a root job or a child job, the `ADSSH_JOBRC_FATAL` environment variable will not be applied to any other root job or child job to change its return code.

# 3

## Defining jobs in JP1/AJS

This section describes how to define jobs in JP1/AJS.

## 3.1 Defining jobs in JP1/AJS

---

This section explains the specification of environment information when JP1/AJS is used.

You automate job execution in JP1/AJS by registering jobs into JP1/AJS - View. In JP1/AJS - View, the commands and batch files that are used for operations are defined as jobs, and system operations are automated by associating the execution order of those jobs.

JP1/AJS - View supports definitions for the following types of jobs:

- Custom jobs
- PC jobs (for Windows)
- UNIX jobs (for UNIX)

The basic guide describes how to define batch jobs as PC jobs or UNIX jobs.

For details about JP1/AJS - View, see the *Job Management Partner 1/Automatic Job Management System 3 Operator's Guide*.

### 3.1.1 Defining and executing a jobnet

To automate job execution in JP1/AJS, you can define registered custom jobs, PC jobs (for Windows), or UNIX jobs (for UNIX) into a jobnet in JP1/AJS - View, and then execute the jobnet. For details about JP1/AJS - View, see the description of job definition in the *Job Management Partner 1/Automatic Job Management System 3 Operator's Guide*.

To define and execute a jobnet in JP1/AJS3 - View:

1. From the Windows **Start** menu, select **All Programs, JP1\_Automatic Job Management System 3 - View**, and then **Job Management System**.

The JP1/AJS3 - View - Login window is displayed.

2. To log in, specify your user name, password, and the host to connect.

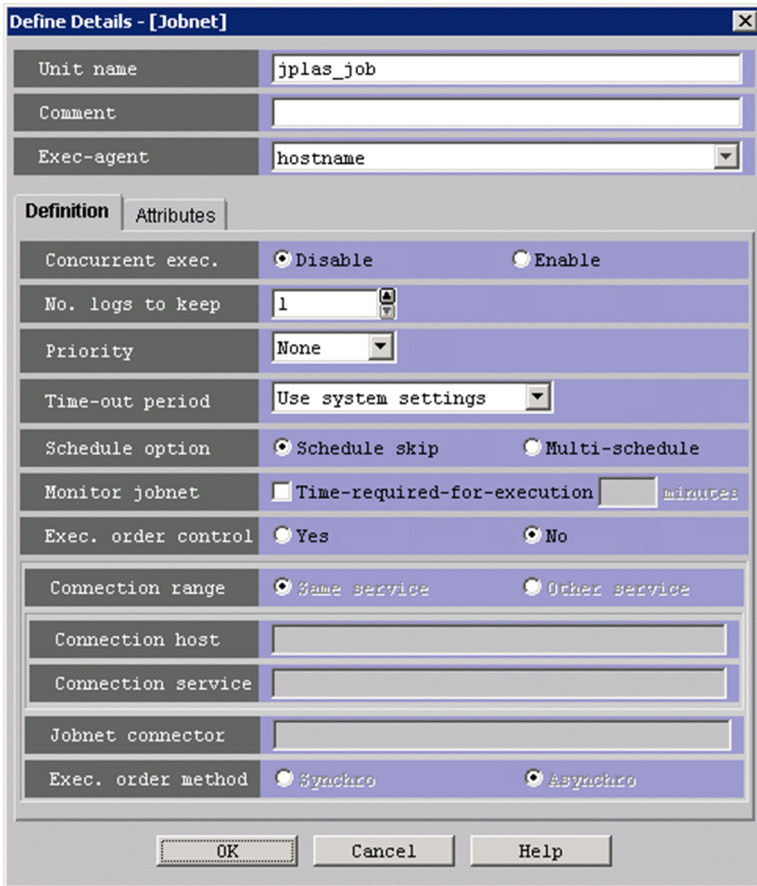
The JP1/AJS3 - View window is displayed.

3. Select **Edit, New**, then **Jobnet**.

The Define Details - [Jobnet] dialog box is displayed.

4. Specify information including attributes of the jobnet, and then click the **OK** button.

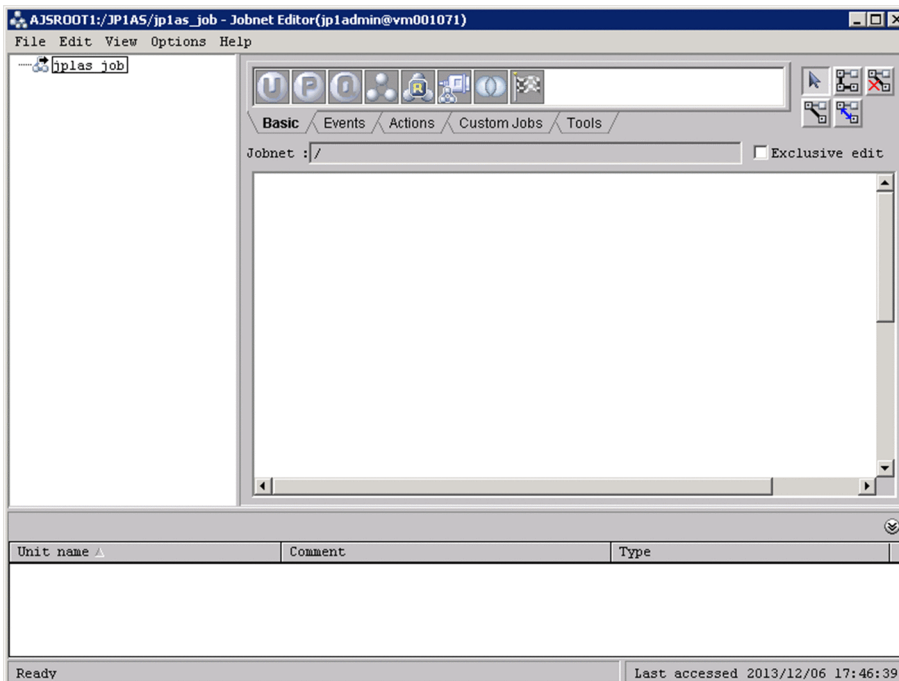
Specify the appropriate information in **Exec-agent** according to the operating environment. This information can be omitted. For details about the JP1/AJS items, see the applicable JP1/AJS manual.



The jobnet is created and displayed in the list area.

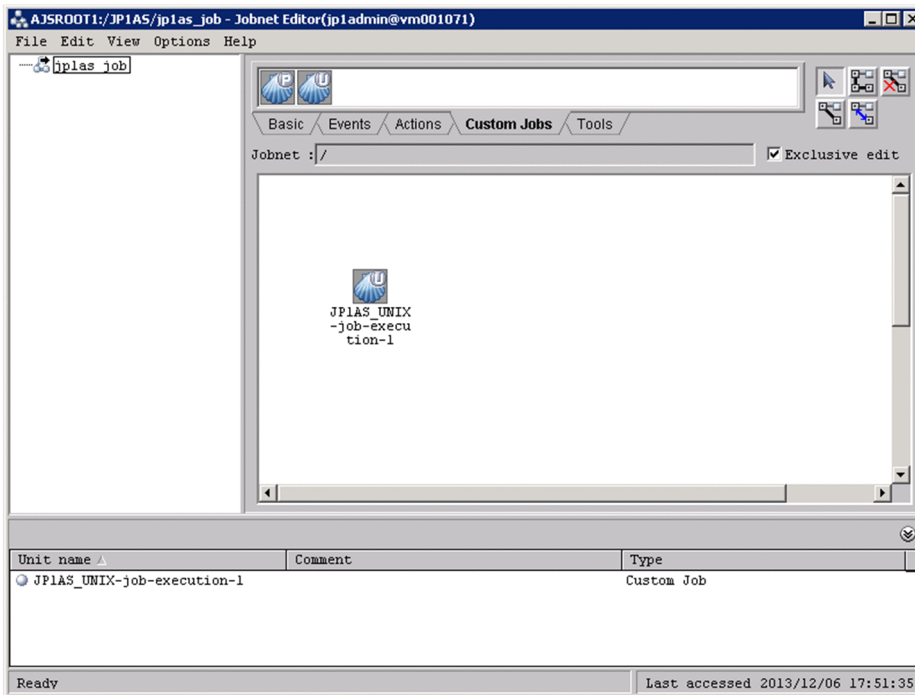
5. Double-click the created jobnet.

The Jobnet Editor window is displayed.



6. Select **Exclusive edit** so that no other user can access the job while you are defining and associating it.

7. Drag the required custom job, PC job, or UNIX job icon from the icon list to the map area. Drag the required custom job, PC job, or UNIX job icon from the icon list to the map area.



The Define Details - [Custom Job], Define Details - [PC Job], or Define Details - [UNIX Job] dialog box is displayed. The subsequent steps vary depending on whether you want to use PC jobs or UNIX jobs.

To use PC jobs, see [3.1.2 Defining batch jobs as PC jobs](#). To use UNIX jobs, see [3.1.3 Defining batch jobs as UNIX jobs](#).

## (1) Notes about jobnet definitions

- Run-time directory to be used when JP1/Advanced Shell's job controller is started from JP1/AJS  
When JP1/Advanced Shell's job controller is started from JP1/AJS, the run-time directory is set to the one that is used when JP1/AJS - Agent (or JP1/AJS - Manager) starts JP1/Advanced Shell's job controller. For details about the run-time directory that is used when JP1/AJS - Agent (or JP1/AJS - Manager) starts JP1/Advanced Shell's job controller, see the manual *Job Management Partner 1/Automatic Job Management System 3 Configuration Guide 1*. In the JP1/AJS manuals, the run-time directory is referred to as a work path (work directory).
- Environment variables to be used when JP1/Advanced Shell's job controller is started from JP1/AJS  
Normally, when JP1/Advanced Shell's job controller for Windows is started from JP1/AJS, the system environment variable settings are enabled when the JP1/AJS services are started and no user environment variables are loaded. For details, see the applicable JP1/AJS manual.
- Connecting to an overseas version of JP1/AJS - Manager whose language is set to English  
When connecting to an overseas version of JP1/AJS - Manager whose language is set to English, in the Define Script Execution dialog box, in the definition information, use only ASCII alphanumeric characters.

## 3.1.2 Defining batch jobs as PC jobs

This subsection explains the items required to define JP1/Advanced Shell batch jobs as PC jobs.

## (1) When defining a batch job

- Executable file name

Specify the path of the `adshexec` command in **Executable file name** in the **Definition** tab in the **Define Details - [PC Job]** dialog box or in `sc="script-file-name"` in the unit definition file.

```
installation-folder\JP1ASE\bin\adshexec.exe
```

- Parameters

Specify the options, job definition script file name, and runtime parameters for the `adshexec` command in **Parameters** in the **Definition** tab in the **Define Details - [PC Job]** dialog box or in `prm="parameter"` in the unit definition file.

- Environment variable

Specify the following details in **Environment variables** in the **Definition** tab in the **Define Details - [PC Job]** dialog box or in `env="environment-variable"` in the unit definition file.

```
AJS_BJEX_STOP=TERM
```

The following figure provides an example specification of a batch job in JP1/Advanced Shell.

Figure 3-1: Specification example of the **Definition** tab in the **Define Details - [PC Job]** dialog box

The screenshot shows the 'Define Details - [PC Job]' dialog box with the 'Definition' tab selected. The fields are as follows:

Unit name	sample
Comment	
Exec-agent	
File name	Hitachi\JP1AS\JP1ASE\bin\adshexec.exe
Parameters	"D:\scripts\sct01.ash" param1 param2
Environment variables	AJS_BJEX_STOP=TERM
Environment file	
Working path	
Priority	None
Standard input	
Standard output	
Standard error	
End judgment	Rule: Judgment by threshold
Warning	
Abnormal	0
Retry on abnormal end	<input checked="" type="radio"/> No <input type="radio"/> Yes
Return code	Greater than or equal to: <input type="text"/> Less than or equal to: <input type="text"/>
Maximum retry times	1 times
Retry interval	1 minutes
User name	

Buttons: OK, Cancel, Help



### 3.1.3 Defining batch jobs as UNIX jobs

This subsection explains the items required to define JP1/Advanced Shell batch jobs as UNIX jobs.

- Script file name

Specify the path of the `adshexec` command in **Script file name** on the **Definition** tab in the **Define Details - [UNIX Job]** dialog box or in `sc="script-file-name"` in the unit definition file:

```
/opt/jpl1as/bin/adshexec
```

Alternatively, you can specify the path of the `adshexec` command following `#!` on the first line (example: `#!/opt/jpl1as/bin/adshexec`), and then specify the path of the job definition script file with execution permissions granted:

```
Path of job definition script file
```

- Command statement

You can specify, as part of command text, the path of the `adshexec` command or path of the job definition script file in the same way as for the script file name. Specify the path in **Command statement** on the **Definition** tab in the **Define Details - [UNIX Job]** dialog box, or in `te="command-text"` in the unit definition file. If a job defined in the command statement is terminated forcibly in JP1/AJS, the following restrictions apply.

- Depending on the timing at which forced termination is performed, you might not be able to reference the contents of the job execution log or standard error output for the job from JP1/AJS - View. In this case, you can check the contents of the job execution log in the spool job directory.
- Jobs with return code 143 are output to the job execution log. However, jobs with return code -1 can be referenced from JP1/AJS - View.

- Parameters

Specify the options, job definition script file name, and runtime parameters for the `adshexec` command in **Parameters** on the **Definition** tab in the **Define Details - [UNIX Job]** dialog box or in `prm="parameter"` in the unit definition file.

If you specified a job definition script file name for the script file name, specify only the runtime parameters.

- Environment variables

Specify the following value in **Environment variables** on the **Definition** tab in the **Define Details - [UNIX Job]** dialog box or in `env="environment-variable"` in the unit definition file:

```
AJS_BJEX_STOP=TERM
```

The following figure shows a specification example of a JP1/Advanced Shell batch job. In this example, the `adshexec` command is specified for "script file name" and job definition script file path is specified.

Figure 3-2: Specification example of the **Definition** tab in the **Define Details - [UNIX Job]** dialog box (when specifying the adshexec command)

The image shows a dialog box titled "Define Details - [UNIX Job]". It has a close button in the top right corner. Below the title bar are three input fields: "Unit name" with the value "sample", "Comment" (empty), and "Exec-agent" (dropdown menu). Below these are three tabs: "Definition" (selected), "Transfer File", and "Attributes". The "Definition" tab contains several fields and controls:

- "Command statement": empty text box with up/down arrows.
- "Script file name": text box containing "/opt/jplas/bin/adshexec".
- "Parameters": text box containing "./user1/scripts/sct02.ash param1 param2".
- "Environment variables": text box containing "AJS\_BJEX\_STOP=TERM" with up/down arrows.
- "Environment file": empty text box.
- "Working path": empty text box.
- "Priority": dropdown menu set to "None".
- "Standard input": empty text box.
- "Standard output": text box with an "Append" checkbox.
- "Standard error": text box with an "Append" checkbox.
- "End judgment": dropdown menu set to "Rule Judgment by threshold", with "Warning" and "Abnormal 0" sub-fields.
- "Retry on abnormal end": radio buttons for "No" (selected) and "Yes".
- "Return code": two text boxes with labels "Greater than or equal to" and "Less than or equal to".
- "Maximum retry times": text box with "1" and "times".
- "Retry interval": text box with "1" and "minutes".
- "User name": empty text box.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

Figure 3-3: Specification example of the **Definition** tab in the **Define Details - [UNIX Job]** dialog box (when specifying a job definition script file path)

The screenshot shows the 'Define Details - [UNIX Job]' dialog box with the 'Definition' tab selected. The fields are as follows:

- Unit name: sample
- Comment: (empty)
- Exec-agent: (dropdown menu)
- Command statement: (empty)
- Script file name: /home/user1/scripts/sct02.ash
- Parameters: param1 param2
- Environment variables: AJS\_BJEX\_STOP=TERM
- Environment file: (empty)
- Working path: (empty)
- Priority: None
- Standard input: (empty)
- Standard output: (empty) Append
- Standard error: (empty) Append
- End judgment: Rule Judgment by threshold
- Warning: (empty) Abnormal: 0
- Retry on abnormal end:  No  Yes
- Return code: (empty) Greater than or equal to (empty) Less than or equal to (empty)
- Maximum retry times: 1 times
- Retry interval: 1 minutes
- User name: (empty)

Buttons at the bottom: OK, Cancel, Help

# 4

## Executing batch jobs and checking execution results

This section describes how to execute batch jobs and how to check batch job execution results.

## 4.1 Executing batch jobs

---

This subsection describes the procedure for executing batch jobs (created in JP1/Advanced Shell) from JP1/AJS, and describes each step.

### 4.1.1 Procedure for executing batch jobs

This subsection explains the general procedure for the operator's tasks when JP1/AJS is used to execute jobs.

#### (1) Defining jobs

To use JP1/AJS to execute jobs, you must define the jobs according to the procedure explained in [3.1.1 Defining and executing a jobnet](#).

#### (2) Executing jobs

The three methods for using JP1/AJS to execute jobs are planned execution, fixed execution, and immediate execution. For details about these three execution methods, see the *Job Management Partner 1/Automatic Job Management System 3 Operator's Guide*.

If you do not use JP1/AJS, you can execute jobs (job definition scripts) by entering commands from the command prompt or shell.

#### (3) Monitoring a jobnet

In JP1/AJS, you start the jobnet monitor to check job execution status.

#### (4) Re-executing jobs

If you need to re-execute jobs, re-execute them from the JP1/AJS - View window.

### 4.1.2 Executing batch jobs

This subsection explains how to start JP1/Advanced Shell's batch job applications by using JP1/AJS from the execution environment.

For details about using JP1/AJS for automation of batch job applications, see the applicable JP1/AJS manual. For details about how to define and execute JP1/Advanced Shell jobs in jobnets, see [3.1.1 Defining and executing a jobnet](#).

When you automate batch job applications, you can reduce costs as well as run your system more securely with a smaller staff. JP1/AJS is a product for automating standard batch job applications. JP1/AJS can also automate a combination of complex batch job applications. Using JP1/Advanced Shell together with JP1/AJS operations provides the following advantages:

- You can use the temporary file function to allocate files that are used temporarily and delete them when the job or job step terminates.
- You can share job definitions among multiple applications by calling external scripts.
- You can achieve flexible job definitions by changing, adding, and deleting coding in job definition scripts.

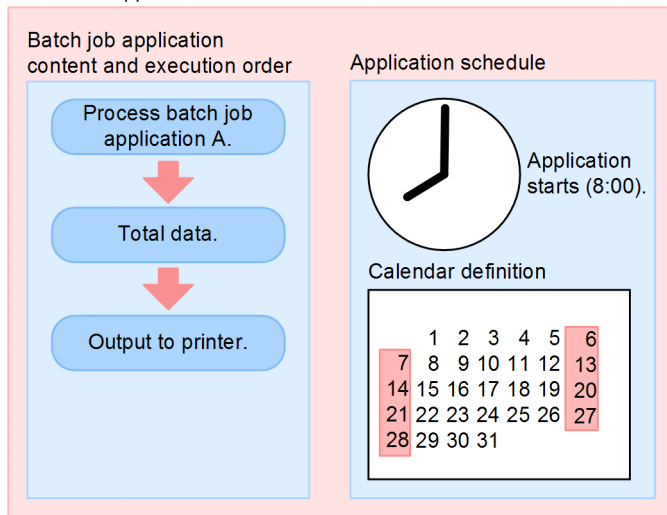
To use JP1/AJS to execute batch job applications automatically, you must define the following:

- Content and processing order of the batch job applications
- Schedule for executing the batch job applications or registration of events that trigger execution of the batch job applications

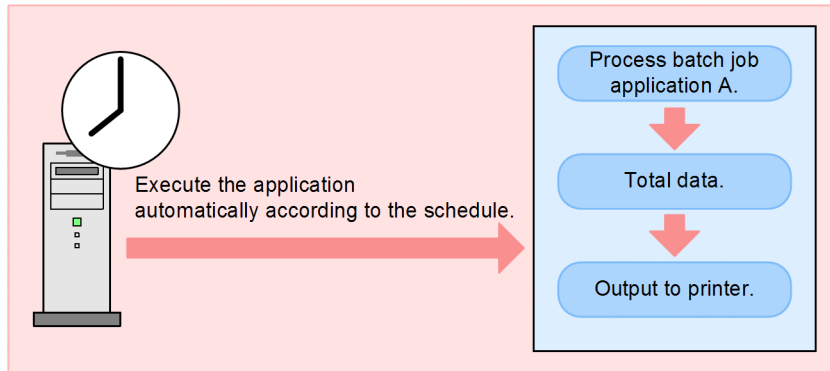
The following figure provides an overview of using JP1/AJS to automate batch job applications. The numbers in the figure correspond to the numbers in the explanation that follows.

**Figure 4-1: Overview of using JP1/AJS to automate batch job applications**

1. Register the batch job application content and execution order, and the application schedule.



2. Execute the batch job application according to the schedule.



1. Registers the batch job application content and execution order, and the application schedule.
2. The batch application is executed automatically according to the registered schedule.

## (1) Defining batch job applications and their execution order

Many applications are executed at a specified time in a specified order.

For example, totaling of sales slips is executed in the following order:

1. Extract data from the database.
2. Sort data.

### 3. Output to printer.

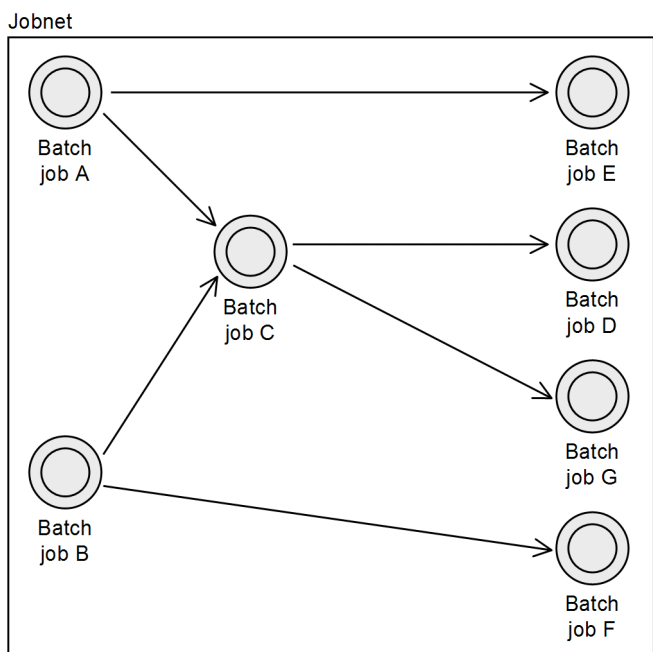
Steps 1 through 3 can be automated as a job controller's job step by defining these steps in a job definition script file, but the task of extracting data from the database at 12:00 cannot be automated. To define batch job applications and their execution order in JP1/Advanced Shell and JP1/AJS, define in the job controller the series of steps that make up the applications and then define the relationships among the definitions of the individual batch job applications and their execution order as the JP1/AJS execution order or execution time.

If batch job applications are broken up into task units, such as commands, application programs, or job definition scripts, JP1/AJS alone can achieve jobs equivalent to those that can be achieved by JP1/Advanced Shell. They are also called jobs in JP1/AJS.

When batch job applications and execution orders are defined in JP1/Advanced Shell and JP1/AJS, the batch job execution orders are defined by using jobnets in JP1/AJS.

The following figure shows a jobnet used when batch job applications and their execution order are defined in JP1/Advanced Shell and JP1/AJS.

**Figure 4-2: Jobnet used to define batch job applications and their execution order in JP1/Advanced Shell and JP1/AJS**



#### **Explanation**

The following explains the execution order of the batch jobs that are defined by using JP1/AJS jobnet.

- When batch job A terminates, batch job E is executed.
- When batch jobs A and B terminate, batch job C is executed.
- When batch job C terminates, batch jobs D and G are executed.
- When batch job B terminates, batch job F is executed.

## **(2) Defining the definition schedule of batch job applications and their execution order**

To automatically define a definition schedule for multiple batch job applications and their execution order, you need a schedule definition that determines when this definition is to be executed.

JP1/AJS's schedule definition contains such information as a calendar that specifies the company's business days and holidays, the date and time execution is to begin, and an execution interval. Based on this definition, JP1/AJS determines the execution schedule and automatically starts JP1/Advanced Shell's job execution on the specified date and time.

### **(3) Registering the timing of starting batch job applications**

You can register an event, such as when a file is created or when some specific event occurs, as the timing for starting a batch job application. If you have registered the required information, you can start a batch job application at a specified time as well as whenever some specified event (such as creation of a file) occurs.



## 4.2 Checking batch job execution results

---

You can check batch job execution results in the job execution log. This subsection describes what the job execution log is, how to check the job execution log, and how to delete the job execution log.

### 4.2.1 Job execution log

The purpose of a job execution log is to notify users of the results of executing batch jobs. This log information, excluding the contents for the standard output for user programs, is output to the files under the spool job directory and to the standard error output when a job terminates. You can use JPI/AJS - View, among other methods, to view job execution logs.

The following information is output to the job execution log:

- Start and end messages for batch jobs
- Start and end messages for job steps
- Contents of job definition scripts
- Results of executing commands
- Status and postprocessing results of files that have been prepared
- Standard output from user programs (`stdout`)<sup>#1</sup>
- Standard error output from user programs (`stderr`)<sup>#2</sup>
- Messages related to acquiring coverage information

#1

Output while the job is running to the standard output in effect at the time the job started if either of the conditions listed below is satisfied.

- The `-s` option is specified in the `adshexec` command or `PARENT` is specified in the `OUTPUT_STDOUT` parameter in the environment file.
- When the root job is the minimum output mode

#2

A root job running in minimum output mode outputs data to the standard error output in effect at the time the job started, rather than to a file under the spool job directory.

The `JOBLOG_SUPPRESS_MSG` parameter can be set to suppress output to the `JOBLOG` file of some information messages. For details about messages that can be specified, see the description in *JOBLOG\_SUPPRESS\_MSG parameter* in the manual *JPI/Advanced Shell*.

### 4.2.2 Checking batch job execution results

In the spool root directory specified in the environment file, create a directory for each job and output job execution results to that directory. Job execution logs and the files output by programs in job steps are output to the directory for each job.

The following shows the structure of the spool directory:

The structure of the spool directory is shown below. Note that the description below only covers the directories and files used in this manual. For details about the entire structure of the spool directory, see *Outputting job execution results to spool* in the manual *JP1/Advanced Shell*.

```

spool-root-directory
|- lock-file
+- spool-job-directory
   +-JOBLOG#1
   +-JOBLOG job-ID sequence-number
   +-JOBLOG number-giving-the-order-in-which-a-child-job-starts#2
   +-SCRIPT#1
   +-SCRIPT number-giving-the-order-in-which-a-child-job-starts#2
   +-STDERR#1
   +-STDOUT#1
   +-step-number step-name_STDOUT#1
   +-step-number step-name_STDERR#1

```

#1

The contents of this file are also output to the job execution log. For details about what is output to the job execution log, see [4.2.1 Job execution log](#).

#2

This is a temporary file created during job execution. The following explains the contents of such temporary files.

File name	Description
JOBLOG <b>number-giving-the-order-in-which-a-child-job-starts</b>	Job execution log for a child job for merging that is output when MERGE (merging the child job's spool job into the root job's spool job) is specified in the SPOOLJOB_CHILDJOB parameter in the environment file
SCRIPT <b>number-giving-the-order-in-which-a-child-job-starts</b>	

If a job is terminated immediately by SIGKILL in UNIX or `TerminateProcess` in Windows, these files might remain in the spool job directory. When you delete spool directories, also delete these files.

The following subsections explain the files and directories that are not temporary files.

## (1) spool-root-directory

The directory name is specified in the `SPOOL_DIR` parameter in the environment file.

## (2) spool-job-directory

This directory has the job sequence number as its name and is created for each job. When the job terminates, the directory is renamed to *job-ID-job-name*.

You can use the `adshhk` command to delete accumulated spool jobs. For details about the `adshhk` command, see [4.2.3 Deleting spool jobs](#).

When a job terminates, the spool job directory named with the job ID is renamed. If a directory exists with the same name as the new directory, renaming will fail and the name of the spool job directory will remain as the job ID. Because the job execution has been completed and the succeeding job can be executed, the job returns 0 as the return code. While a directory named with the job ID remains, that job ID cannot be used and the directory cannot be deleted by using the `adshhk` command.

### (3) JOBLOG

This is for job execution messages. Messages indicating the job's execution status, including command execution results and file allocation results, are output to this directory.

### (4) JOBLOG\_job-ID\_sequence-number

This is the job execution log for a child job.

This file is created only when a child job is specified with the minimum output mode by using one of the following methods when starting the child job:

- Specifying MINIMUM with the `-m` option in the `adshexec` command
- Specifying MINIMUM with the `-m` option in the `adshscripttool` command
- Specifying MINIMUM in the `OUTPUT_MODE_CHILD` parameter

This file is not created when `MERGE` (merging the child job's spool job into the root job's spool job) is specified in the `SPOOLJOB_CHILDJOB` parameter.

### (5) SCRIPT

This is for script image files. The contents of the first job definition script started and the contents of external job definition script files specified in the `#+adsh_script` command are output to this directory. External job definition script files specified using other methods, such as the `.` (dot) command, are not output to this directory. When you want to output the contents of job definition scripts as logs, you must use the `#+adsh_script` command.

If `MERGE` is specified in the `SPOOLJOB_CHILDJOB` parameter when the root job is run in the expansion output mode and the child job is run in the minimum output mode, the child job's `SCRIPT` is not merged into the root job's `SCRIPT`. For details, see the description in *Merging a child job's spool job into the root job's spool job* in the manual *JPI/Advanced Shell*.

### (6) STDERR

This is the standard error output for the job. This file is not created when the root job is specified with the minimum output mode using one of the following methods when starting the root job:

- Specifying MINIMUM with the `-m` option in the `adshexec` command
- Specifying MINIMUM in the `OUTPUT_MODE_ROOT` parameter

The following header is output at the beginning of the file:

```
***** JOB SCOPE STDERR *****
```

### (7) STDOUT

This is the standard output for the job. It is created when the `-s` option is specified in the `adshexec` command or `SPOOL` is specified in the `OUTPUT_STDOUT` parameter in the environment file. This file is not created when the root job is specified with the minimum output mode using one of the following methods when starting the root job:

- Specifying MINIMUM with the `-m` option in the `adshexec` command
- Specifying MINIMUM in the `OUTPUT_MODE_ROOT` parameter

The following header is output at the beginning of the file:

```
***** JOB SCOPE STDOUT *****
```

## (8) step-number\_step-name\_STDOUT

If job steps are defined, this is the standard output within the corresponding job step. If the job step name consists of more than eight bytes, only the first eight bytes of the job step name are used for *step-name*.

This standard output is created when the `-s` option is specified in the `adshexec` command or `SPOOL` is specified in the `OUTPUT_STDOUT` parameter in the environment file. This file is not created when the root job is specified with the minimum output mode using one of the following methods when starting the root job:

- Specifying `MINIMUM` with the `-m` option in the `adshexec` command
- Specifying `MINIMUM` in the `OUTPUT_MODE_ROOT` parameter

## (9) step-number\_step-name\_STDERR

If job steps are defined, this is the standard error output within the corresponding job step. If the job step name consists of more than eight bytes, only the first eight bytes of the job step name are used for *step-name*.

This file is not created when the root job is specified with the minimum output mode using one of the following methods when starting the root job:

- Specifying `MINIMUM` with the `-m` option in the `adshexec` command
- Specifying `MINIMUM` in the `OUTPUT_MODE_ROOT` parameter

### 4.2.3 Deleting spool jobs

Spool jobs stored on a spool increase in size while they are stored in a spool directory. Therefore, make sure that you periodically delete old spool jobs to free up disk space.

- How to delete spool jobs

To delete spool jobs, enter the `adshhk` command below. For details about how to specify the `adshhk` command, see the description in *adshhk command* in the manual *JPI/Advanced Shell*.

```
adshhk target-list-file-name report-file-name log-file-name [number-of-days]
```

Before you execute the `adshhk` command, specify in the file indicated as *target-list-file-name* the necessary information, including the name of the spool directory that contains the spool jobs to be deleted.

The `adshhk` command's execution results are output to the file whose name is specified in *report-file-name*. The execution results are also output to trace logs.

Error messages are output to the file whose name is specified in *log-file-name*.

Spool jobs that have existed for more than the number of days specified in *number-of-days* are deleted. For example, if you specify 2, spool jobs that have existed for 2 or more days are deleted.

- Report file created by the `adshhk` command

When the `adshhk` command has executed, the execution results are output to a report file. In the following example report file, the header information is output on the first line:

```
"jobid","jobname","rc","start date","end date","act","info","spool","target
days","execute date"
"000056","JOB001","1","2011/06/13 09:03:31","2011/06/13 09:03:31","delete","","C:
\Documents and Settings\All Users\Documents\Hitachi\jpl\jplase
\spool","15","2011/06/30 18:19:58"
:
```

**Legend:**

The first line of the execution results contains the headers listed below. The subsequent lines display the values corresponding to the header items.

Header	Meaning
jobid	Job ID
jobname	Job name
rc	Job's return code
start date	Job's execution start date and time (in the format <i>yyyy/mm/dd hh:mm:ss</i> ). The spool allocated to the debugger itself when the job was started in the debugger mode is used to output the debugger's logs, not the job execution results. Therefore, the debugging start date and time are output.
end date	Job's termination date and time (in the format <i>yyyy/mm/dd hh:mm:ss</i> ). The spool allocated to the debugger itself when the job was started in the debugger mode is used to output the debugger's logs, not the job execution results. Therefore, the job termination date and time are not output.
act	Applied action (keep: save, delete: delete, error: an error occurred during deletion processing)
info	Detailed error information
spool	Spool directory
target days	Target days
execute date	Command execution start date and time (in the format <i>yyyy/mm/dd hh:mm:ss</i> )



**Important note**

If the spool directory contains any user-specified file or directory that was not created by JP1/Advanced Shell, the `adshhk` command outputs a message such as `KNAX4419-E`, and then terminates.

## 4.3 Forcibly terminating batch jobs

---

There are two ways to forcibly terminate a job:

- If the job was started from JP1/AJS, use JP1/AJS's forced termination procedure.  
To be able to forcibly terminate from JP1/AJS a job for a job icon that was executed in Windows or UNIX, you must have specified the `AJS_BJEX_STOP=TERM` environment variable beforehand. For details about jobs for job icons that are run in Windows or UNIX, see [3.1.1 Defining and executing a jobnet](#).
- Send a termination request signal to the `adshexec` command's process. In Windows, you can use a command such as `taskkill` to terminate the `adshexec` process.

When a job is forcibly terminated, the job controller forcibly terminates its child or descendant process that are executing. For details, see [4.3.1 Forcibly terminating child or descendant processes](#).

After forcibly terminating the child or descendant process, the job controller performs postprocessing on the allocated files, and then terminates the job without executing any subsequent job steps or commands. The job controller does not execute a subsequent job step even if `abnormal` or `always` is specified in its `run` attribute. In UNIX, when a job is forcibly terminated, the `adshexec` command terminates with an error by signal. For details about the job processing in UNIX when `SIGTERM` is received, see the description in *Processing when signals are received* in the manual *JP1/Advanced Shell*. For details about the job processing in Windows when jobs are forcibly terminated, see the description in *Job processing during forced termination* in the manual *JP1/Advanced Shell*.

### Important note

In Windows, when the `adshexec` command is started, the `adshexecsub` command is also started, and when the `adshexec` command is forcibly terminated, the `adshexecsub` command is also terminated. Therefore, do not forcibly terminate the `adshexecsub` command. If an attempt is made to forcibly terminate the `adshexecsub` command, the following events might occur:

- A descendant process that is executing might not terminate.
- Temporary files might remain in the system.

If these events occur, use the `taskkill` command or the task manager to forcibly terminate the descendant process and delete the temporary files manually.

### Important note

Because JP1/Advanced Shell in a Windows environment uses job objects to forcibly terminate descendant processes, note the following:

- A descendant process generated from JP1/Advanced Shell cannot be associated with a job object.
- If a process of JP1/Advanced Shell has already been associated with a job object, forced termination of the job will not terminate the process generated by the child process of JP1/Advanced Shell.

### Important note

In Windows, if a job that executes an external command and generates a child process is terminated forcibly and more than 255 processes that are at its grandchild or lower levels exist concurrently, the `KNAX6381-E`

message might be issued and renaming of the spool job directory might fail. Note the following three points about this:

- To reference a spool job directory that has failed, use the directory name displayed in the immediately following `KNAX6382-I` message that is issued.
- A spool job directory whose renaming has failed cannot be deleted by the `adshhk` command. If necessary, delete it manually.
- In the case of a job that has failed in renaming a spool job directory in the execution environment, job definition script operation information is not output by the `adshevtout` command.

### 4.3.1 Forcibly terminating child or descendant processes

If a job is forcibly terminated, the job controller forcibly terminates its child or descendant processes, and then terminates the job.

### 4.3.2 Notes about operations including Ctrl+C (in Linux)

If a job is executed in the non-terminal input mode, operation such as **Ctrl+C** and **Ctrl+\** might not be able to terminate simultaneously the root job, child jobs, and other external commands that were started.<sup>#</sup> If you wish to forcibly terminate these jobs and commands all at once, use the `kill` command to send a termination request signal such as `SIGTERM` to the root job immediately under the login shell.

#

If a job is executed in the non-terminal input mode, the `adshexec` command's process and its child processes belong to separate process groups. Therefore, if an operation such as **Ctrl+C** or **Ctrl+\** is performed from the login shell while the job is executing, `SIGINT` or `SIGQUIT` is sent only to the process group currently running in the foreground.

The jobs and external commands running as descendant processes of the job that received the signal are forcibly terminated, but those jobs and external commands running as higher processes, including the parent process, are not forcibly terminated.

# Appendixes



## A. Advanced Use

---

To use JPI/Advanced Shell more efficiently, see the manual *JPI/Advanced Shell*.

## B. Reference Material for This Manual

---

This appendix provides reference information for reading the manual.

### B.1 Related publications

This manual is part of a related set of manuals. The manuals in the set are listed below (with the manual numbers):

#### JP1/Advanced Shell

- *JP1 Version 11 JP1/Advanced Shell Description, User's Guide, Reference, and Operator's Guide (3021-3-B32(E))*

#### JP1/AJS

- *JP1 Version 11 Job Management: Getting Started (Job Scheduler) (3021-3-B11(E))*
- *JP1 Version 11 JP1/Automatic Job Management System 3 System Design (Configuration) Guide (3021-3-B13(E))*
- *JP1 Version 11 JP1/Automatic Job Management System 3 Configuration Guide (3021-3-B15(E))*
- *JP1 Version 11 JP1/Automatic Job Management System 3 Troubleshooting (3021-3-B17(E))*
- *JP1 Version 11 JP1/Automatic Job Management System 3 Operator's Guide (3021-3-B18(E))*

### B.2 Abbreviations for Microsoft product names

This manual uses the following abbreviations for Microsoft product names.

Abbreviation		Full name or meaning
Windows Server <sup>#</sup>	Windows Server 2012	Microsoft(R) Windows Server(R) 2012 Standard
		Microsoft(R) Windows Server(R) 2012 Datacenter
		Microsoft(R) Windows Server(R) 2012 R2 Standard
		Microsoft(R) Windows Server(R) 2012 R2 Datacenter
	Windows Server 2008	Microsoft(R) Windows Server(R) 2008 R2 Datacenter
		Microsoft(R) Windows Server(R) 2008 R2 Enterprise
Microsoft(R) Windows Server(R) 2008 R2 Standard		
Windows <sup>#</sup>	Windows 10	Windows(R) 10 Home(32-bit version)
		Windows(R) 10 Pro(32-bit version)
		Windows(R) 10 Enterprise(32-bit version)
		Windows(R) 10 Home(64-bit version)
		Windows(R) 10 Pro(64-bit version)
		Windows(R) 10 Enterprise(64-bit version)
	Windows 8	Windows(R) 8.1(32-bit version)
		Windows(R) 8.1 Pro (32-bit version)

Abbreviation		Full name or meaning
Windows <sup>#</sup>	Windows 8	Windows(R) 8.1 Enterprise (32-bit version)
		Windows(R) 8.1 (64-bit version)
		Windows(R) 8.1 Pro (64-bit version)
		Windows(R) 8.1 Enterprise (64-bit version)
		Windows(R) 8 (32-bit version)
		Windows(R) 8 Pro (32-bit version)
		Windows(R) 8 Enterprise (32-bit version)
		Windows(R) 8 (64-bit version)
		Windows(R) 8 Pro (64-bit version)
		Windows(R) 8 Enterprise (64-bit version)
	Windows 7	Microsoft(R) Windows(R) 7 Enterprise
		Microsoft(R) Windows(R) 7 Professional
		Microsoft(R) Windows(R) 7 Ultimate

#: Windows Server and Windows are sometimes referred to collectively as *Windows*.

## B.3 Conventions: Fonts and symbols

The following table explains the text formatting conventions used in this manual:

Text formatting	Convention
<b>Bold</b>	<p>Bold characters indicate text in a window, other than the window title. Such text includes menus, menu options, buttons, radio box options, or explanatory labels. For example:</p> <ul style="list-style-type: none"> <li>From the <b>File</b> menu, choose <b>Open</b>.</li> <li>Click the <b>Cancel</b> button.</li> <li>In the <b>Enter name</b> entry box, type your name.</li> </ul>
<i>Italic</i>	<p>Italic characters indicate a placeholder for some actual text to be provided by the user or system. For example:</p> <ul style="list-style-type: none"> <li>Write the command as follows: <code>copy source-file target-file</code></li> <li>The following message appears: A file was not found. (file = <i>file-name</i>)</li> </ul> <p>Italic characters are also used for emphasis. For example:</p> <ul style="list-style-type: none"> <li>Do <i>not</i> delete the configuration file.</li> </ul>
Monospace	<p>Monospace characters indicate text that the user enters without change, or text (such as messages) output by the system. For example:</p> <ul style="list-style-type: none"> <li>At the prompt, enter <code>dir</code>.</li> <li>Use the <code>send</code> command to send mail.</li> <li>The following message is displayed: <code>The password is incorrect.</code></li> </ul>
<u>underline</u>	The underline indicates the default value among two or more values enclosed in selection symbols.

The following table explains the symbols used by this manual in syntax explanations:

Symbol	Convention
	A vertical bar separates multiple items, and has the meaning of OR. For example: A   B   C means A, or B, or C.
{ }	Curly brackets indicate that only one of the enclosed items is to be selected. For example: { A   B   C } means only one of A, or B, or C.
[ ]	Square brackets indicate that the enclosed item or items are optional. A vertical bar is used to delimit multiple items. For example: <b>Examples:</b> [ A ] means that you can specify A or nothing. [ B   C ] means that you can specify B, or C, or nothing.
< >	Single angle brackets enclose the syntax element that must be used to specify an item.
+	The plus sign indicates that the immediately preceding item can be specified multiple times. It is also used to indicate that the items before and after it are specified together. <b>Examples:</b> { A   B } + Indicates that A or B can be specified multiple times in any order. CR+LF Indicates that the carriage return character (CR) and the linefeed character (LF) are specified together.
*	The asterisk indicates that the immediately preceding item can be omitted or that it can be specified one or more times. <b>Example:</b> { A   B } * Indicates that A or B can be specified one or more times in any order or that A and B can both be omitted.
~	A swung dash indicates that the syntax element enclosed by the single angle brackets (< >), double angle brackets (<< >>), or double parentheses ( ( ( ) ) ) that follow must be used to specify the item that precedes the swung dash.
<< >>	Double angle brackets enclose the default value for an item.
( ( ) )	Double parentheses enclose the permissible range of values that can be specified.
. . .	An ellipsis indicates that the immediately preceding item can be repeated as many times as necessary. For example: A, B, B, . . . means that, after you specify A, B, you can specify B as many times as necessary.
Δ	Denotes a single-byte space. Δ <sub>0</sub> : Denotes zero or more spaces (spaces can be omitted). Δ <sub>1</sub> : Denotes one or more spaces (at least one space is required).

The following table explains the syntax elements used in this manual:

Syntax element	Characters that can be specified
<unsigned integer>	<numeric characters> +

## B.4 Conventions: The JP1/Advanced Shell installation folder in Windows

In this manual, *installation folder* refers to the folder in which JP1/Advanced Shell has been installed, unless otherwise stated. The following shows the installation folder when the product is installed with the default settings.

x86 environment:

*system-drive*: \Program Files\Hitachi\JP1AS

x64 environment:

*system-drive*: \Program Files (x86)\Hitachi\JP1AS

## B.5 Conventions: common application data folder

The following shows the **common application data folder** used in this manual.

Windows 10, Windows Server 2012, Windows 8, Windows 7, or Windows Server 2008:

*system drive*: \ProgramData

## B.6 Conventions: Shared documents folder

The following shows the **shared documents folder** used in this manual.

Windows 10, Windows Server 2012, Windows 8, Windows 7, or Windows Server 2008:

*system drive*: \Users\Public\Documents

## B.7 Conventions: Windows menu names used in the manual

The Windows menu names used in this manual assume that you are using one of the following OSs:

Windows 7, Windows Server 2008

In Windows 10, Windows Server 2012, or Windows 8, no **Start** menu is displayed. Instead, you must use the Start window, which can be opened from the lower left corner of the window.

## B.8 Conventions: Directory names

This manual uses the term *directory* wherever possible as a generic term for what Windows calls a *folder* and UNIX calls a *directory*.

In connection with this convention, this manual uses / as the directory delimiter. In Windows-specific cases, \ is used as the folder delimiter.

## B.9 Abbreviations for product names

This manual uses the following abbreviations for product names:

Abbreviation		Full name or meaning
JP1/Advanced Shell		JP1/Advanced Shell
JP1/AJS	JP1/AJS3	JP1/Automatic Job Management System 3 - Agent

Abbreviation		Full name or meaning	
JP1/AJS	JP1/AJS3	JP1/Automatic Job Management System 3 - Manager JP1/Automatic Job Management System 3 - View	
JP1/AJS - Agent	JP1/AJS3 - Agent	JP1/Automatic Job Management System 3 - Agent	
JP1/AJS - Manager	JP1/AJS3 - Manager	JP1/Automatic Job Management System 3 - Manager	
JP1/AJS - View	JP1/AJS3 - View	JP1/Automatic Job Management System 3 - View	
UNIX	Linux	CentOS 6	CentOS 6
		CentOS 7	CentOS 7
		Oracle Linux 6	Oracle Linux(R) 6
		Oracle Linux 7	Oracle Linux(R) 7
		Red Hat Enterprise Linux 6 (x64)	Red Hat Enterprise Linux Server 6(64-bit x86_64)
		Red Hat Enterprise Linux 7	Red Hat Enterprise Linux Server 7(64-bit x86_64)
		SUSE Linux 12	SUSE Linux(R) Enterprise Server 12(x64)

## B.10 Conventions: Units (such as KB, MB, GB, and TB)

This manual uses the following conventions:

- 1 KB (kilobyte) is 1,024 bytes.
- 1 MB (megabyte) is 1,024<sup>2</sup> bytes
- 1 GB (gigabyte) is 1,024<sup>3</sup> bytes.
- 1 TB (terabyte) is 1,024<sup>4</sup> bytes.

## C. Glossary

---

This glossary defines the terminology used in this manual.

### .env file

A file in which are set the path names to the ENV environment variables and that is loaded when the shell starts. You can use the `KSH_ENV_READ` environment setting parameter to specify whether this file is to be loaded.

### argument

A generic term for an item that is specified following a command name. Multiple arguments are separated by a delimiter on the command line or in a job definition script.

### arithmetic operation

Any of the calculations performed using arithmetic operators in a job definition script. In an arithmetic operation, the values assigned to variables are handled as numeric values.

### base name

The portion of a file name excluding the *.extension* portion. For example, the base name of `adshexec.exe`, the program for executing batch jobs, is `adshexec`.

### batch job

A job executed by batch processing.

### batch operation server

A server on which JP1/Advanced Shell is installed that is used to execute batch jobs. JP1/AJS - Agent or JP1/AJS - Manager must be installed when JP1/AJS is used.

### batch processing

The process of gathering collected data and transactions and processing them in bulk on a regular schedule, such as every day, week, or month.

### breakpoint

Coding that forces execution to stop and that is inserted into a job definition script in order to pause the processing during development in order to check the operational status of the job definition script. The debugger interrupts the processing at a breakpoint so that the developer can check the values of variables and registers at the time of the interruption.

### built-in command

This is a command built into the shell and can be executed by the shell itself. JP1/Advanced Shell provides standard shell commands (standard shell commands and regular built-in commands) and extended shell commands. Special built-in commands have the characteristic of terminating the shell executing the command when the syntax of the command is wrong. Regular built-in commands continue processing without terminating the shell executing the command even when the syntax of the command is wrong.

### child job

A job whose job definition script is executed as a descendant process of the root job. Child jobs are executed in accordance with one of the following parameter specifications or their default settings:

- `CHILDJOB_EXT` parameter
- `CHILDJOB_PGM` parameter
- `CHILDJOB_SHEBANG` parameter

### child job execution log output file

The output file for the job execution log of a child job, which is created by the child job and output in the spool job directory of the root job.

## command

Generic name for any instruction that can be used in JP1/Advanced Shell. Commands are executed from the shell or the command prompt, as well as from job definition scripts.

## command grouping

Facility for executing multiple commands as a unit in JP1/Advanced Shell.

## command line

The line displayed to the user for entering commands. In the Windows command prompt, input is entered after the > on the command line. In the UNIX shell, input is entered after the % on the command line.

## command prompt

The window in a Windows environment that requests that a command be entered.

## command separator

The functionality that allows developers to write more than one command on a single line of a job definition script in JP1/Advanced Shell.

## conditional

A test that controls the processing branch that is to execute based on the results of a conditional expression in a control statement in a job definition script.

## conditional expression

A formula used in a job definition script that expresses a calculation using numeric comparisons, string comparisons, file attributes, logical operators, and the ternary operator.

## conditional parameter

Any of the parameters that are set in an environment file and that are specified in order to configure the environment setting parameters and export parameters that are valid only in the physical host or in a specific logical host.

## console

The terminal screen.

## control statement

Same meaning as *script control statement*.

## core dump

A source of maintenance information collected by a trace program and consisting of core files and dump files. When a problem occurs, the contents of memory are saved to a file that can be used to assist with troubleshooting.

## coverage information

Information that provides measurements in tests of the extent of coverage. The two types of coverage information are C0, which is statement coverage information, and C1, which is branch coverage information. C0 measures the ratio (%) of commands in a job definition script that execute, while C1 measures the ratio (%) of branches in a job definition script that execute.

## custom job

A predefined job for executing a task with a specific purpose in JP1/AJS. The custom job component for JP1/Advanced Shell is required in order to take advantage of JP1/AJS's custom job functionality in JP1/Advanced Shell.

## debug

The process of testing a job definition script created in the development environment or of investigating errors in a script. To debug, you must launch the debugger.

## debugger

A program for testing a job definition script created in the development environment and for investigating errors in a script. In the Windows environment, the debugging functions of the JP1/Advanced Shell editor are used. In the UNIX environment, the debugger is started by specifying the -d option in the `adshexec` command.



#### definition file

A file that defines the directories into which data for troubleshooting is to be collected.

#### development environment

An environment provided by JP1/Advanced Shell - Developer that supports development of job definition scripts for batch processing.

#### dialog box

A window that asks the user to enter a response.

#### editor

A program for creating job definition scripts efficiently by taking advantage of a variety of features provided in the development environment.

#### environment file

A file that contains environment information.

#### environment information

Information, such as environment variables and environment file parameters, that must be set before JP1/Advanced Shell starts.

#### environment setting parameter

Any of the parameters that are set in an environment file for the purpose of defining the JP1/Advanced Shell execution environment. These parameters are specified in the format `#-adsh_conf parameter value`.

#### environment variable

Any of the variables that contain various system settings that can be set by the user.

#### execution environment

The environment provided by JP1/Advanced Shell for execution of batch operations. JP1/Advanced Shell refers to the execution environment in its narrow sense.

#### export parameter

A parameter that is set in the environment file and whose function is to set an environment variable when a command starts.

#### extended script command

A command that is executed in a job definition script. Compared to normal shell script commands, these commands offer the additional capability to control batch job execution. They are also referred to as job execution control commands. In JP1/Advanced Shell, these commands start with `#-adsh`.

#### extended shell command

A built-in command that is internal to the shell and executed by the shell itself. Extended shell commands can be used in job definition scripts.

#### extended shell variable

A shell variable with a special meaning that is provided by JP1/Advanced Shell.

#### external command

Any of the UNIX-compatible commands, OS-provided commands, user-created executable files, and other programs that are not shell built-in commands.

#### fault injection mode

A mode used during debugging to simulate the occurrence of an error.

In UNIX, you enable or disable the fault injection mode with the `joberrmode` command. In Windows, you choose the **Fault Injection Mode** menu item in the JP1/Advanced Shell editor.

#### file allocation

In JP1/Advanced Shell, such operations as registering postprocessing of files are referred to as *file allocation*.

#### file descriptor

A numeric identifier for distinguishing the different types of input and output in JP1/Advanced Shell. In JP1/Advanced Shell, the standard output is assigned 1, the standard error output is assigned 2, and 3 through 9 can be allocated and used for other purposes.

#### flow control

Functionality for controlling the event-issuance interval for JP1 events that are issued during execution of the `adshread` and `adshecho` commands.

#### here document

A redirection functionality used in a job definition script by which standard input is generated with the job definition script.

#### job controller

A program for controlling a job while the job is running. The `adshexec` command is the job controller.

#### job definition script file

A program file that defines a job that has been prepared as a job definition script.

#### job execution log

A collection of messages output by a job, including the start and end messages for the job and job steps. At the end of a job, the contents of the job execution log are sent to the standard error output by the job controller.

#### job ID

An identification number (sequentially generated between 000001 and 999999) that is assigned to a job by JP1/Advanced Shell at the time the job is executed. Each job is assigned a unique identifier, so that each job can be identified individually on the basis of its job ID. Once job ID 999999 has been assigned, the next job is assigned job ID 000001.

#### job information

Information associated with a job, such as the job name, job ID, and job step names.

#### jobnet

A set of jobs whose execution order is defined. Jobs within the jobnet execute automatically in the predefined order. The jobnet is a functionality provided by JP1/AJS.

#### job scheduler

A product that performs job scheduling. It is part of a suite of products in JP1/Advanced Shell used to link to JP1/AJS.

#### job step

A range of processing within a job defined in a job definition script that demarcates a unit of specific processing. The job step is the smallest unit for performing a specific operation (task) in JP1/Advanced Shell. A job is made up of a collection of job steps. Job steps are defined with the  `#-adsh_step_start`,  `#-adsh_step_error` (optional), and  `#-adsh_step_end` commands.

#### JP1/Advanced Shell

A product used to create and execute batch jobs from job definition scripts. JP1/Advanced Shell can be divided into JP1/Advanced Shell and JP1/Advanced Shell - Developer. In this narrow sense, JP1/Advanced Shell refers to the execution environment in which batch jobs are executed from job definition scripts. Batch jobs in both Windows and UNIX can be run from the same job definition script.

#### JP1/Advanced Shell - Custom Job

A program for creating jobs that are custom-defined in the operation management console in JP1/Advanced Shell.

## JP1/Advanced Shell - Developer

A product used for developing job definition scripts for batch jobs. This term also refers to the development environment in which job definition scripts are developed.

## JP1/AJS3

Abbreviation for JP1/Automatic Job Management System 3, which is the successor product to JP1/AJS2. By linking JP1/Advanced Shell to JP1/AJS3, you can achieve distributed processing among multiple PCs.

## log

Historical information that is output by the computer. Timestamps, messages, and similar items are output as logs.

## long option

A type of option specified in command arguments. A long option begins with two consecutive hyphens (--) followed by a character string.

## metacharacter

A character (or character string) that has a special meaning in a job definition script.

## operand

A type of command argument specified on the command line. An operand is a default command argument that is specified in addition to option names and option values. Parameter values are also called operands.

## option

In general, a pre-selected capability that is added to the instructions provided by a computer input device.

In JP1/Advanced Shell, a command argument consisting of one hyphen (-) followed by one character is called a short option, and a command argument consisting of two consecutive hyphens (--) followed by a character string is called a long option.

An argument specified immediately following an option is the option's *value*.

## pipe

A functionality for linking the standard output of a previous command to the standard input of a subsequent command.

## program output data file

A file to which a user program can output its execution results. JP1/Advanced Shell creates the file name automatically in order to consolidate the user program's output results with the system execution log.

## quotation

Either the single quotation mark (') or the double quotation mark (").

## redirection

Capability before a command in a job definition script is executed to change the input source for the information needed to execute the command or the output destination for the execution results. Typically, the keyboard is assigned as the standard input and the screen is assigned as the standard output, but redirection enables these assignments to be changed.

## regular built-in command

Any of a set of the built-in commands among the standard shell commands. In the case of a regular built-in command, even if its command syntax is invalid, it does not exit the shell that is executing the command (see also *special built-in command*).

## regular file

A file used for input or output by a job definition script. Regular files might remain after the job finishes, or regular files might be deleted during execution of the job. Regular files can be defined with the #-adsh\_file command or the adshfile command.

reply-request message

A message that asks the operator to enter a reply.

reply-waiting event

A JP1 event that provides notification of a reply-request message.

reserved script command

A command that can be used as a reserved word in a job definition script. An example is the `time` command.

return code

A code that is returned to report the execution result of a job definition script or a command.

root job

A job executed from JP1/AJS or a login shell that is not a child job.

script

A text file into which is assembled a series of commands that can be executed sequentially from the shell. A script in JP1/Advanced Shell is called a *job definition script*, and they can be executed in both the Windows and UNIX environments.

script control statement

A statement for managing commands in a job definition script. Examples include the `if`, `for`, `while`, `until`, and `case` statements.

script file

A file in which a script that has been created is saved.

shell

A program that interprets instructions provided by a computer input device and passes them to the OS.

shell command

Generic name for any command used in JP1/Advanced Shell that is executed in the shell or from the command prompt.

shell operation command

A command that is provided as an executable binary file or a shell script. The two types of shell operation commands are those that can be used only in job definition scripts and those that can be used not only in job definition scripts but from OS shells and the command prompt. The shell operation commands include the `adshexec` command (executes batch jobs).

shell option

Any of the pre-selected capabilities that are added to the instructions provided by a computer input device to the shell.

shell script

A text file into which you assemble a series of commands so that you can then execute those commands sequentially from the shell. A shell script in JP1/Advanced Shell is referred to as a *job definition script*, and it can be executed in both the Windows and UNIX environments.

shell variable

An area of memory assigned as a value in a job definition script. You can reference the value of a created variable.

short option

A type of option specified in command arguments. A short option begins with a hyphen (-) followed by one character.

signal

A mechanism in UNIX by which processes report to each other the occurrence of asynchronous events. For example, a signal is sent when a job is forcibly terminated in JP1/Advanced Shell.

### special built-in command

Any of a set of the built-in commands among the standard shell commands. In the case of a special built-in command, if its command syntax is invalid, it exits the shell that is executing the command (see also *regular built-in command*).

### spool

The location where JP1/Advanced Shell stores the execution results of jobs and job execution logs.

### spool job

The execution results for each job created in the spool directory.

### standard error output (stderr)

A stream to which a program outputs its error messages and other messages.

### standard input (stdin)

A stream from which a program receives its input data.

### standard output (stdout)

A stream to which a program outputs its data.

### standard shell command

A built-in command that is internal to the shell and executed in a process in the shell itself. Standard shell commands can be used in job definition scripts.

### subshell

In a UNIX environment, a child process that has the same name as the job controller, is neither a root job nor a child job, and is created temporarily automatically when an external command or a specific syntax is executed in a job definition script.

### symbolic link

A link that is implemented as a file that contains the actual file path.

### system execution log

A log output by a job controller in JP1/Advanced Shell in order to facilitate integrated management of job execution status by the system administrators. Log information from multiple job controllers can be output to a single log.

### temporary file

A file whose use is transient during job execution. Temporary files are created by a job or job step, and they are deleted automatically when the job terminates. Temporary files are defined with the `#-adsh_file_temp` command.

### trace log

Information collected to assist in investigating and resolving problems that occur in JP1/Advanced Shell.

### trap action

An action that is defined in the `trap` command's *action* argument.

### UNIX-compatible command

Any of the standard UNIX commands, such as the `ls` command, that can be used in JP1/Advanced Shell. These commands can also be used in a Windows environment, which facilitates interoperability between UNIX and Windows.

### variable

A location or array in memory that is used to handle values in a job definition script. Examples of variables include shell variables and environment variables.

### watchpoint

A special breakpoint that stops a job definition script when the value of a variable or expression changes. A watchpoint can be managed in the same way as any other type of breakpoint.

## wildcard

A character, such as the asterisk (\*) or the question mark (?), that can be specified as a stand-in for any character or character string. The asterisk (\*) represents any character string, and the question mark (?) represents any single character.

In addition, you can use square brackets ( [ ] ) to obtain a match with any of the characters in the character string enclosed in the square brackets. You can also use the hyphen ( - ) to separate values constituting a range, or the exclamation mark ( ! ) for a condition to be true when none of the characters enclosed in square brackets results in a match. You can also use the comma ( , ) to assemble a comma-separated list of character strings, any one of which can be selected.

# Index

## Symbols

.env file 63

## A

ADSH\_ENV [environment variable] 34  
ADSH\_JOBRC\_FATAL (environment variable) 35  
    specifying return code in event of unresumable error  
    in jobs 35  
ADSH\_JOBRC\_FATAL [environment variable] 34  
AJS\_BJEX\_STOP [environment variable] 34  
allocation management file 49  
argument 63  
arithmetic operation 63

## B

base name 63  
batch application  
    expediting configuration of 4  
    inheriting asset between OSs of 4  
batch job 63  
batch job application  
    defining 46  
    defining definition schedule of 47  
    defining definition schedule of execution order of 47  
    defining execution order of 46  
    overview of using JP1/AJS to automate 45  
    registering timing of starting 48  
    starting, by using JP1/AJS from execution  
    environment 45  
batch job execution results  
    central management of 5  
    improving serviceability and maintainability by  
    central management of 5  
batch operation server 63  
batch processing 63  
breakpoint 63  
built-in command 63

## C

checking the local time setting 15  
child job 63  
    defining file to be started as 25  
child job execution log output file 63

command 63  
    grouping 63  
command line 63  
command prompt 63  
command separator 63  
common application data folder 61  
conditional 63  
conditional expression 63  
conditional parameter 63  
confirming files that can be used in JP1/Advanced  
Shell 13  
console 63  
control statement 63  
conventions  
    abbreviations 61  
    directory names 61  
    fonts and symbols 59, 61  
    KB, MB, GB, and TB 62  
core dump 63  
coverage information 63  
creating required directories 22  
custom job 37, 63

## D

debug 63  
debugger 63  
definition file 63  
determining whether to use spool job creation  
suppression functionality 29  
development environment 63  
dialog box 63

## E

editor 63  
environment file 63  
    specifying 21  
environment information 63  
environment setting parameter 63  
environment variable 63  
execution environment 63  
expansion output mode 29  
export parameter 63  
extended script command 63  
    defining return code of 32

extended shell command 63

extended shell variable 63

external command 63

## F

fault injection mode 63

file allocation 63

file descriptor 63

flow control 63

font conventions 59

## G

GB meaning 62

## H

here document 63

## I

installation procedure (for Linux) 12

installation procedure (for Windows) 12

installing JP1/Advanced Shell 18

installing JP1/Advanced Shell (in Windows) 18

installing JP1/Advanced Shell(to Linux) 18

installing prerequisite products 17

## J

job

defining 45

executing 45

re-executing 45

job controller 63

job definition script 4

job definition script file 63

job environment file 21

job execution log 49, 63

outputting contents by job type 26

job ID 63

job information 63

job scheduler 63

job step 63

jobnet 46, 63

defining and executing 37

monitoring 45

used for defining batch job application and their execution order in JP1/Advanced Shell and JP1/AJS 46

JP1/Advanced Shell 63

example of application to business operation 6

purposes of 4

JP1/Advanced Shell - Custom Job 63

JP1/Advanced Shell - Developer 63

JP1/AJS, specifying environment information for (applicable when JP1/AJS is used) 37

JP1/AJS3 63

## K

KB meaning 62

## L

log 63

long option 63

## M

MB meaning 62

memory and disk space requirements for installation 13

metacharacter 63

minimum output mode 29

## N

notes on file systems 15

## O

operand 63

operator

tasks of 45

option 63

## P

PC job 37

Performing user-specific postprocessing when a job is terminated forcibly 32

pipe 63

preparation before installation 13

preparations for using script-format UNIX-compatible commands (in Windows) 24

prerequisite OS 13

program output data file 63

## Q

quotation 63



## R

- redirection 63
- reference material 58
- regular built-in command 63
- regular file 63
- related publications 58
- reply-request message 63
- reply-waiting event 63
- reserved script command 63
- return code 63
  - in event of unresumable error in job, defining 34
  - in event of unresumable error in job, specifying 35
- root job 63

## S

- script 63
- script control statement 63
- script file 63
- Setting environment variables 34
- setting language for prerequisite OS 13
- shared documents folder 61
- shell 63
  - setting up 25
- shell command 63
- shell operation command 63
- shell option 63
- shell script 4, 63
- shell variable 63
- short option 63
- signal 63
- special built-in command 63
- specifying settings for using UNIX-compatible commands 24
- spool 22, 63
  - defining output information 28
  - outputting job execution result to 49
- spool job 63
  - deleting 52
- standard error output
  - stderr 63
- standard input
  - stdin 63
- standard output
  - stdout 63
- standard shell command 63
- subshell 63

- symbol conventions 59
- symbolic link 63
- system environment file 21
- system execution log 22, 63

## T

- TB meaning 62
- temporary file 22, 63
- trace 22
- trace log 63
- trap action 63

## U

- UNIX job 37
- UNIX-compatible command 63

## V

- variable 63

## W

- watchpoint 63
- wildcard 63