Escript and System Agent Reference Guide

iPlanet™ Unified Development Server

Version 5.0

August 2001

Contents

List of Figures

List of Procedures

Preface

This *Escript and System Agent Reference Guide* contains thorough descriptions of the commands of Escript, and the commands and instruments provided by iPlanet UDS system agents. Also covered is information about what commands you can use in Escript to perform system management tasks.

Along with the definitions of the commands and instruments provided by the iPlanet UDS system agents, this book also describes how to navigate to the agents and invoke the commands and access the instruments belonging to the agents.

This manual is organized into two parts:

- Escript
- iPlanet UDS System Agents

This manual assumes that you will have a copy of the *iPlanet UDS System Management Guide* available for information about iPlanet UDS system issues and system management tasks.

If you plan to write custom agents or access the iPlanet UDS system agents using TOOL code, see *Programming with System Agents*.

This preface contains the following sections:

- "Product Name Change" on page 20
- "Audience for This Guide" on page 20
- "Organization of This Guide" on page 20
- "Text Conventions" on page 20
- "Other Documentation Resources" on page 21
- "iPlanet UDS Example Programs" on page 23
- "Viewing and Searching PDF Files" on page 23

Product Name Change

Forte 4GL has been renamed the iPlanet Unified Development Server. You will see full references to this name, as well as the abbreviations iPlanet UDS and UDS.

Audience for This Guide

This manual assumes that you are familiar with the system management tasks described in *iPlanet UDS System Management Guide*.

Organization of This Guide

The following table briefly describes the contents of each chapter:

Part	Chapter	Description
Part 1, "Escript"	Chapter 1, "Using the Escript Utility"	Describes how to use the Escript utility to perform system management tasks.
	Chapter 2, "General Escript Commands"	Describes general Escript commands.
Part 2, "iPlanet UDS System Agents"	Chapter 3, "Using iPlanet UDS System Agents"	Describes how to use iPlanet UDS system agents with Escript and the Environment Console.
	Chapter 4, "iPlanet UDS System Agent Commands and Instruments"	Describes the iPlanet UDS system agents and their states, commands, and instruments.

Text Conventions

This section provides information about the conventions used in this document.

Format	Description
italics	Italicized text is used to designate a document title, for emphasis, or for a word or phrase being introduced.

Format	Description
`monospace`	Monospace text represents example code, commands that you enter on the command line, directory, file, or path names, error message text, class names, method names (including all elements in the signature), package names, reserved words, and URLs.
ALL CAPS	Text in all capitals represents environment variables (FORTE_ROOT) or acronyms (UDS, JSP, iMQ).
	Uppercase text can also represent a constant. Type uppercase text exactly as shown.
Key+Key	Simultaneous keystrokes are joined with a plus sign: Ctrl+A means press both keys simultaneously.
Key-Key	Consecutive keystrokes are joined with a hyphen: Esc-S means press the Esc key, release it, then press the S key.

Other Documentation Resources

In addition to this guide, there are additional documentation resources, which are listed in the following sections. The documentation for all iPlanet UDS products (including Express, WebEnterprise, and WebEnterprise Designer) can be found on the iPlanet UDS Documentation CD. Be sure to read "Viewing and Searching PDF Files" on page 23 to learn how to view and search the documentation on the iPlanet UDS Documentation CD.

iPlanet UDS documentation can also be found online at http://docs.iplanet.com/docs/manuals/uds.html.

The titles of the iPlanet UDS documentation are listed in the following sections.

iPlanet UDS Documentation

- *A Guide to the iPlanet UDS Workshops*
- *Accessing Databases*
- *Building International Applications*
- *Escript and System Agent Reference Guide*
- *Fscript Reference Guide*

- *Getting Started With iPlanet UDS*
- *Integrating with External Systems*
- *iPlanet UDS Java Interoperability Guide*
- *iPlanet UDS Programming Guide*
- *iPlanet UDS System Installation Guide*
- *iPlanet UDS System Management Guide*
- *Programming with System Agents*
- *TOOL Reference Guide*
- *Using iPlanet UDS for OS/390*

Express Documentation

- *A Guide to Express*
- *Customizing Express Applications*
- *Express Installation Guide*

WebEnterprise and WebEnterprise Designer Documentation

- *A Guide to WebEnterprise*
- *Customizing WebEnterprise Designer Applications*
- *Getting Started with WebEnterprise Designer*
- *WebEnterprise Installation Guide*

Online Help

When you are using an iPlanet UDS development application, press the F1 key or use the Help menu to display online help. The help files are also available at the following location in your iPlanet UDS distribution: `FORTE_ROOT/userapp/forte/cl`*`n`*`/*.hlp`.

When you are using a script utility, such as Fscript or Escript, type help from the script shell for a description of all commands, or help <*command*> for help on a specific command.

iPlanet UDS Example Programs

A set of example programs is shipped with the iPlanet UDS product. The examples are located in subdirectories under `$FORTE_ROOT/install/examples`. The files containing the examples have a `.pex` suffix. You can search for TOOL commands or anything of special interest with operating system commands. The `.pex` files are text files, so it is safe to edit them, though you should only change private copies of the files.

Viewing and Searching PDF Files

You can view and search iPlanet UDS documentation PDF files directly from the documentation CD-ROM, store them locally on your computer, or store them on a server for multiuser network access.

NOTE You need Acrobat Reader 4.0+ to view and print the files. Acrobat Reader with Search is recommended and is available as a free download from http://www.adobe.com. If you do not use Acrobat Reader with Search, you can only view and print files; you cannot search across the collection of files.

➤ **To copy the documentation to a client or server**

1. Copy the `doc` directory and its contents from the CD-ROM to the client or server hard disk.

   You can specify any convenient location for the `doc` directory; the location is not dependent on the iPlanet UDS distribution.

2. Set up a directory structure that keeps the `udsdoc.pdf` and the `uds` directory in the same relative location.

   The directory structure must be preserved to use the Acrobat search feature.

NOTE To uninstall the documentation, delete the `doc` directory.

➤ To view and search the documentation

1. Open the file `udsdoc.pdf`, located in the `doc` directory.
2. Click the Search button at the bottom of the page or select Edit > Search > Query.
3. Enter the word or text string you are looking for in the Find Results Containing Text field of the Adobe Acrobat Search dialog box, and click Search.

   A Search Results window displays the documents that contain the desired text. If more than one document from the collection contains the desired text, they are ranked for relevancy.

   NOTE For details on how to expand or limit a search query using wild-card characters and operators, see the Adobe Acrobat Help.

4. Click the document title with the highest relevance (usually the first one in the list or with a solid-filled icon) to display the document.

   All occurrences of the word or phrase on a page are highlighted.

5. Click the buttons on the Acrobat Reader toolbar or use shortcut keys to navigate through the search results, as shown in the following table:

Toolbar Button	Keyboard Command
Next Highlight	Ctrl+]
Previous Highlight	Ctrl+[
Next Document	Ctrl+Shift+]

   To return to the `udsdoc.pdf` file, click the Homepage bookmark at the top of the bookmarks list.

6. To revisit the query results, click the Results button at the bottom of the `udsdoc.pdf` home page or select Edit > Search > Results.

Part 1

Escript

Part 1 of *Escript and System Agent Reference Guide* provides usage and reference information about using the iPlanet UDS Escript utility.

This section contains the following chapters:

Chapter 1

Using the Escript Utility

This chapter describes the Escript utility, the iPlanet UDS command-line interface for managing iPlanet UDS environments and applications.

The Escript utility is the command-line equivalent of the iPlanet UDS Environment Console, except Escript also lets you incorporate environment and application management tasks into scripts. These scripts can then be executed at system startup, for example, or at regular intervals, to collect application statistics, or start and shut down partitions.

This chapter covers the following topics:

- the `escript` command
- using Escript and iPlanet UDS system agent commands to perform system management tasks

For a complete reference listing of all commands that can be used in Escript, see Appendix A, "All Escript and System Agent Commands."

Overview

The Escript utility is the functional equivalent of the Environment Console: you can use it to perform environment tasks, such as creating and modifying environment definitions, and application tasks, such as deploying and managing iPlanet UDS applications.

Like the Environment Console, Escript connects to and communicates with the executing Environment Manager and any active Node Managers in your environment. It lets you perform iPlanet UDS system management tasks by providing you access to your environment repository (environment edit mode) and to the full hierarchy of iPlanet UDS system management agents (agent mode).

Unlike the Environment Console, however, you can use Escript to perform script-based system management by building and executing predefined scripts of Escript commands. For information on building and executing Escript scripts, see "Writing and Running Escript Scripts" on page 35.

General Escript and System Agent Commands

In Escript, you can use two different types of commands: general Escript commands and system agent commands.

General Escript commands modify environment definitions, work with the operating system, and let you create and run scripts. These commands usually affect the operation of the Escript utility, not any particular system agent. General Escript commands are described in Chapter 2, "General Escript Commands."

System agent commands are commands that affect a particular system agent. In the iPlanet UDS runtime system, system agents are objects that manage and monitor parts of the environment, including applications, nodes, and so forth. When you invoke a system agent command, you need to navigate to the appropriate agent before you can invoke the command. This agent is considered the *current agent*. "Navigating around the Agent Hierarchy" on page 115 describes how to navigate to an agent in Escript. iPlanet UDS system agent commands are described with their agents in Chapter 4, "iPlanet UDS System Agent Commands and Instruments."

If you are not sure which type of command you want to use for a particular task, you can check the tables later in this chapter, starting with "Working with the Escript Utility" on page 31. If you know the name of the command you want to use, but are not sure what agent, if any, you must have as the current agent, see Chapter , "."

Escript Help

When using Escript, you can get help at any point by issuing the `Help` command. The `Help` command provides on-line help for all commands currently available. Some commands are associated with a particular agent, so help for those commands is only available when that agent is the current agent.

Starting the Escript Utility

You can start the Escript utility on any node in your iPlanet UDS environment.

➤ **To start the Escript utility on Windows NT platforms**

1. Double-click the Escript icon.

➤ **To start the Escript utility on UNIX, OpenVMS, or Windows NT platforms**

1. Type the `escript` command.

When the Escript utility starts, it gives you an "`escript>`" prompt.

Using the escript Command

As mentioned above, you start the Escript utility on command line-based operating systems by executing the `escript` command.

The syntax of the `escript` command is:

Portable syntax (all platforms)

escript [-**fl** *logger_flags*] [-**fm** *memory_flags*] [-**fst** *integer*] [**-i** *input_file*]
[**-o** *output_file*] [-**fns** *name_server_address]*

OpenVMS syntax

VFORTE ESCRIPT
[/**LOGGER**=*logger_flags*]
[/**MEMORY**=*memory_flags*]
[/**STACK**=*integer*]
[/**INPUT=***input_file*]
[/**OUTPUT=***output_file*]
[/**NAMESERVER=***name_server_address]*

NOTE As in all iPlanet UDS command line specifications, if you use a name that includes a space, you should enclose the name in double quotation marks.

The following table describes the command line flags for the `escript` command.

Flag	Description
-fl **/LOGGER**	Specifies the logger flags to use for the Escript session. See *iPlanet UDS System Management Guide* for information about the syntax for specifying logger flags. Overrides the FORTE_LOGGER_SETUP environment variable setting. On UNIX, you must specify the logger flags in double quotes.
-fm **/MEMORY**	Specifies the memory flags to use for the Escript session. See *iPlanet UDS System Management Guide* for syntax information. Overrides defaults appropriate for the operating system. On UNIX, you must specify the memory flags in double quotes.
-fst **/STACK**	Specifies the thread stack size in bytes for iPlanet UDS and POSIX threads. This specification overrides default stack size allocation. For more information, refer to *iPlanet UDS System Management Guide*.
-i **/INPUT**	Specifies an input file. The file should consist of an Escript script—a set of Escript commands—that you want to execute automatically when the Escript utility starts.
-o **/OUTPUT**	Specifies an alternate output file, in addition to stdout.
-fns *name_server_address* **/NAMESERVER=** *name_server_address*	Specifies the name service address for the environment in which this application will run. This value overrides the value, if any, specified by the FORTE_NS_ADDRESS environment variable. If you want your application to be able to switch to a backup Environment Manager if the primary Environment Manager fails, you can also specify multiple name service addresses, as discussed in *iPlanet UDS System Management Guide*.

Quitting Escript

To exit the Escript utility, invoke the `Quit` or `Exit` command.

This command will prompt you to save any changes before exiting Escript.

Using Escript to Manage Your System

The rest of this chapter describes common system management tasks and contains tables that map system management tasks to Escript commands and system agent commands.

If a command can be invoked anywhere in Escript, the Agent column of the table contains the word "any." Otherwise, the Agent column contains the system agent that must be the current agent when you invoke the command.

If you need to make a particular agent the current agent to invoke a command, use the `FindParentAgent`, and `FindSubAgent` commands to navigate up and down the agent hierarchy. "Navigating around the Agent Hierarchy" on page 115 explains how to locate agents in the agent hierarchy.

Working with the Escript Utility

This section maps tasks involving the Escript utility itself, such as defining the format of file names and getting online help. This table contains tasks involving the Escript utility itself, and describes commands that you can use to perform these tasks.

	Agent	Command	Description	See:
Exit Escript	any	Exit	Exits Escript, prompting you to save if there are any outstanding changes to the environment.	page 72
	any	Quit	Exits Escript, prompting you to save if there are any outstanding changes to the environment.	page 82
Modify Escript log flags	any	ModLogger	Modifies the current logger flag settings for Escript.	page 81
Get help on commands	any	Help	Lists help for general Escript commands and commands belonging to the current agent.	page 77
Define format for file names	any	UseLocal	Sets Escript to expect file names to be specified in local operating system format.	page 100
	any	UsePortable	Sets Escript to expect file names to be specified in iPlanet UDS portable name format.	page 101

	Agent	Command	Description	See:
Define the directory search path	any	ShowPath	Shows the current search path.	page 97
	any	AddPath	Adds the specified directories to the current search path.	page 66
	any	SetPath	Sets the directory search path used by any of the commands that take a file name as an argument.	page 94
Locate a file	any	WhichFile	Searches through the directories in the current directory search path to locate the first directory in which the specified file exists.	page 103
Run memory management	any	CollectMem	Runs memory management on Escript.	page 68
Set Escript's return value	any	ExitStatus	Set the return value for this session of Escript.	page 73

Interacting with the Operating System

This section lists the tasks that you perform to interact with the operating system underlying the iPlanet UDS runtime system on a node. The following table maps the task to a particular Escript command or system agent command.

	Agent	Command	Description	See:
Work with files	any	ListFile	Lists the contents of the specified file onto standard output.	page 78
	any	Ls	Lists the files in a directory.	page 80
	any	Mv	Renames a file in the local file system.	page 82
	any	Rm	Removes a file in the local file system.	page 86

	Agent	Command	Description	See:
Define format for file names	any	UseLocal	Sets Escript to expect file names to be specified in local operating system format.	page 100
	any	UsePortable	Sets Escript to expect file names to be specified in iPlanet UDS portable name format.	page 101
Define the directory search path	any	ShowPath	Shows the current search path.	page 97
	any	AddPath	Adds the specified directories to the current search path.	page 66
	any	SetPath	Sets the directory search path used by any of the commands that take a file name as an argument.	page 94
Define the working directory	any	Pwd	Shows the name of the current working directory.	page 82
	any	Cd	Changes the current working directory.	page 67
Invoke a command directly on the operating system	any	ExecCmd	Executes the specified operating system command.	page 71
	Node	ExecCmdRemote	Executes the specified operating system command from the Node Manager service.	page 252

Locating and Using System Agents

This section maps the tasks for locating system agents, monitoring information about the system agents, and setting instrument values to the commands that you can use in Escript.

Navigating Among System Agents

This section lists the tasks that you perform to navigate among system agents. The following table maps the task to a particular Escript command or system agent command. For more detailed instructions about navigating among system agents, see "Navigating around the Agent Hierarchy" on page 115.

	Agent	Command	Description	See:
Show agent information	any	ShowAgent	Shows the parent agent, subagents, and instruments of the current agent.	page 96
	any	ShowSubAgent	Shows the parent agent, subagents, and instruments of the subagent.	page 98
Make parent the current agent	any, except Environment	FindParentAgent	Makes the parent agent the current agent.	page 74
Show child agent (subagent)	any	FindSubAgent	Makes the subagent the current agent	page 76
Make the Environment agent the current agent	any	FindActEnv	Make the Environment agent the current agent.	page 73
Save and retrieve certain agents by name	any	SaveAgent	Adds the current agent to a pool of saved agents.	page 86
	any	ListSavedAgents	Lists agents in the pool of saved agents.	page 78
	any	FindSavedAgent	Makes the referenced saved agent the current agent.	page 75

Monitoring and Changing Instrument Values

This section lists the tasks that you perform to monitor and change the instrument values of system agents. The following table maps the task to a particular Escript command or system agent command. For more detailed instructions about working with instruments, see "Accessing the Instruments for an Agent" on page 117.

	Agent	Command	Description	See:
Get the value of an instrument	any	ShowAgent	Instrument values are displayed with the agent information.	page 96
	any	ShowInstrument	Returns the instrument value of the instrument.	page 96
Change the value of an instrument	any	UpdateInstrument	Sets the value of an instrument that is not read only.	page 99

Writing and Running Escript Scripts

This section lists the tasks that you perform to write and execute Escript scripts to automate system management tasks.

One of the advantages of using Escript over the Environment Console is that Escript lets you script your routine system management tasks. You can capture a sequence of Escript operations into a script file and then run that file at a subsequent time.

You can run the script by starting Escript with the `-i` flag (and supplying the script file name) or by starting Escript and then issuing the `Include` command at the point at which you want to run the script.

To include comments in your scripts, start the line containing the comment with the # character, as shown in the following example:

```
# Shut down the Banking application
ShutdownSubAgent Banking_cl0
```

The following table maps these tasks to a particular Escript command or system agent command.

	Agent	Command	Description	See:
Run an Escript script	any	Include	Executes the commands in a specified script file.	page 77
Run commands interactively	any	Step	Lets you step through the commands interactively.	page 98
Define the directory search path	any	ShowPath	Shows the current search path.	page 97
	any	AddPath	Adds the specified directories to the current search path.	page 66
	any	SetPath	Sets the directory search path used by any of the commands that take a file name as an argument.	page 94
Define the working directory	any	Pwd	Shows the name of the current working directory.	page 82
	any	Cd	Changes the current working directory.	page 67
Define format for file names	any	UseLocal	Sets Escript to expect file names to be specified in local operating system format.	page 100
	any	UsePortable	Sets Escript to expect file names to be specified in iPlanet UDS portable name format.	page 101
Record entered Escript commands into a file	any	Script	Captures Escript commands and writes them into a specified script file.	page 87
Echo commands to standard output	any	CommentOn	Writes script file commands and output to standard output.	page 68
	any	CommentOff	Stops writing script file commands and output to standard output.	page 68
Handling absent Environment Manager	any	WaitForEnvMgr	Forces scripts to wait for the Environment Manager to start before continuing execution.	page 102
	any	ExitIfNoEnv	Sets Escript to exit when it loses contact with an active Environment Manager.	page 72
Set Escript's return value	any	ExitStatus	Set the return value for this session of Escript.	page 73

Configuring Environment Definitions

This section maps the tasks for configuring an environment definition with the appropriate Escript and system management commands.

In the iPlanet UDS environment, you only have one active environment, which has its own environment definition and is represented by the Environment agent at the top of the agent hierarchy.

However, you can also define other environment definitions, which you can use to simulate other environment configurations. Application developers can then use these environment definitions to test, partition, and make distributions for applications that run in environments described by these environment definitions.

Many Escript commands require that you lock the target environment definition. To lock an environment definition, you use the `LockEnv` command. The lock is exclusive across the environment, meaning that only one user of Escript (or the Environment Console) can obtain the lock at any time. In addition, no developers can actively partition applications with the Partition Workshop while the environment is locked.

You should be careful when you obtain the lock to not leave Escript running for long with an exclusive lock on the active environment.

To improve performance, Escript caches definition information for the active environment in local memory on the machine where it runs. The cached information includes all environment properties and node properties, as well as current installation information for the environment.

Escript does not cache the dynamic runtime status information of an environment, such as the list of active partitions on a node, so that this information is always accurate.

If concurrent users of Escript or the Environment Console make and commit changes to the environment definition while you are in an Escript session, the local Escript cache does not refresh itself. You can manually update your Escript cache by using the `RefreshEnv` command. Escript automatically refreshes its cache when you invoke the `LockEnv` or `ExportEnv` commands.

An Overview of Editing Environment Definitions

➤ **To edit the definition for the active environment**

1. Lock the environment definition using the `LockEnv` command.
2. Enter environment edit mode by invoking the `EditEnv` command.

   You need to enter environment edit mode before you can edit the environment definition. You can see that you are in the environment edit mode of Escript because the command prompt changes to "`envedit>`."
3. Perform your environment editing tasks using Escript environment editing commands.
4. Exit the environment edit mode using the `Exit` or `Quit` command.
5. Save your changes and unlock the environment definition.

   If you have edited the active environment definition, you must invoke either a `Commit` command (to save and unlock) or `UnlockEnv` command (to discard) changes to the environment definition. These commands remove the lock so that other concurrent users can access the environment definition.

➤ **To edit an environment definition for a simulated environment**

1. Lock the environment definition and enter environment edit mode using the `NewEnv` command for new environment definitions or the `FindEnv` command with *is*-`updateable` set to `TRUE` for existing environment definitions. iPlanet UDS automatically places an exclusive lock on the environment definition with these commands.

   You need to enter environment edit mode before you can edit the environment definition. You can see that you are in the environment edit mode of Escript because the command prompt changes to "`envedit>`."
2. Perform your environment editing tasks using Escript environment editing commands.
3. Exit the environment edit mode using the *Exit* or `Quit` command. You will be prompted to commit or discard your changes.

Viewing Environment and Environment Definition Information

The following table lists the commands you can use to view information about the active environment and environment definitions.

	Agent	Command	Description	See:
View information about the active environment	Environment	ShowAgent	Displays information about the active environment.	page 96
List available environment definitions	Environment	ListEnvs	Lists the names of the environments in the environment repository.	page 202
Select an environment definition	Environment	FindEnv	Makes the specified environment definition the current environment.	page 199
View an environment definition	any	• ShowEnv	Shows details of the environment.	page 96
Refresh information about the current environment	any	RefreshEnv	Immediately refreshes information about the current environment.	page 83

Editing an Environment Definition in Environment Editing Mode

This section maps the editing tasks with commands that can be used in the active environment's definition and any simulated environment definitions.

The following table includes tasks you can perform in the environment editing mode to define the contents of an environment definition. The prompt must be "`envedit>`" when you enter any of the commands marked with a dot (•).

	Agent	Command	Description	See:
Show the contents of an environment definition	any	• ShowEnv	Shows details of the environment.	page 96
Lock the environment definition	any	LockEnv	Obtains an exclusive lock on the current environment definition.	page 79
	Environment	FindEnv (with `is-updateable` set to `TRUE`)	Makes the specified environment definition the current environment, locks the environment, and starts the environment editing mode.	page 199
	Environment	NewEnv	Creates a new simulated environment definition with the specified name, locks the environment, and starts environment editing mode.	page 203

	Agent	Command	Description	See:
Start the environment editing mode	any	EditEnv	Puts Escript into the environment editing mode on the active environment definition.	page 70
	Environment	FindEnv	Makes the specified environment definition the current environment and starts environment editing mode.	page 199
Exit environment editing mode	any	• Quit	Exits environment editing mode, prompting you to save if there are outstanding changes.	page 82
	any	• Exit	Exits environment editing mode, prompting you to save if there are outstanding changes.	page 72
Create a new environment definition	Environment	NewEnv	Creates a new simulated environment definition with the specified name, locks the environment, and starts environment editing mode.	page 203
Set a password for the environment	any	• SetPassword	Sets the environment password.	page 93
Define preferred server node	any	• SetEnvPrefNode	Sets the preferred node on which to assign server partitions.	page 88

	Agent	Command	Description	See:
Add a node	any	• AddNode	Adds a node with the specified name to the current environment definition.	page 63
Delete a node	any	• RemoveNode	Deletes the specified node from the environment.	page 85
Save changes to the environment definition	any	Commit	Saves all changes to the environment repository.	page 69
Unlock the environment definition	any	UnlockEnv	Unlocks the exclusive lock on the environment definition, and prompts you to save any outstanding changes.	page 99
Delete an environment definition	Environment	RemoveEnv	Removes a simulated environment definition from the repository.	page 204
Import an environment definition	Environment	ImportEnv	Imports the environment definition from the specified file.	page 201
Export an environment definition	Environment	ExportEnv	Exports the environment definition to the specified file.	page 198

Editing Node Definitions

The following table includes tasks you can perform in the environment editing mode to change the definitions for nodes in an environment definition.

Most of these commands act on the current node, which is defined using the `FindNode` command.

NOTE The current node selected using the `FindNode` command is not the same as a current Node agent that can be selected using the `FindSubAgent` or `FindParentAgent` commands. You cannot navigate to a Node agent and try to change the definition of a node in the running active environment. Similarly, you cannot manage a running node using the current agent in environment edit mode.

To start the environment editing mode, you need to invoke one of the commands described in "Editing an Environment Definition in Environment Editing Mode" on page 40. You can use these commands on the active environment's definition and any simulated environment definitions.

The prompt must be "`envedit>`" when you enter any of the commands marked with a dot (•).

	Agent	Command	Description	See:
Set the current node	any	• FindNode	Designates the specified node within the current environment as the current node.	page 74
Display information about the current node	any	• ShowNode	Shows details of the current node.	page 97
Define whether the node is a model node	any	• SetNodeModel	Defines whether the node is a model node.	page 92
Define a node as a client node	any	• SetNodeClient	Defines whether a node will be assigned client partitions by default.	page 90
Define a node as a test node	any	• SetSimForNode	Defines whether a node can be used to test an application in a simulated environment.	page 95

	Agent	Command	Description	See:
Add or remove external resources for a node	any	• AddExternalRM	Adds the specified external resource manager name to the current node, and sets the resource manager type to the value specified.	page 62
	any	• RemoveExternalRM	Removes the specified external resource manager from the current node.	page 84
Add or remove communication protocols for a node	any	• AddCommProtocol	Adds the specified communications protocol to the current node.	page 60
	any	• RemoveCommProtocol	Removes the specified communication protocol from the current node.	page 84
Add or remove external libraries for a node	any	• Add3GLProj	Adds the specified restricted external library to the list of those supported by the current node.	page 59
	any	• Remove3GLProj	Removes an external library from the list of those supported by the current node.	page 83

Setting up Simulated Environments

The following table includes tasks you can perform in the environment editing mode to map a simulated environment to a real test environment.

To start the environment editing mode, you need to invoke one of the commands described in "Editing an Environment Definition in Environment Editing Mode" on page 40.

The prompt must be "`envedit>`" when you enter any of the commands marked with a dot (•).

	Agent	Command	Description	See:
Set the test environment	any	• SetEnvForSim	Specifies the name of the test environment that simulates the current environment definition.	page 87
Set the test node	any	• SetNodeForSim	Specifies the name of the node that simulates the current node in the current simulated environment definition.	page 91

Managing Running Environments

This section lists the tasks that you perform when you manage running environments.

For more information about how to manage a running iPlanet UDS environment, see *iPlanet UDS System Management Guide*.

	Agent	Command	Description	See:
Set environment variables	Active Partition	SetEnvRemote	Set an environment variable for the active partition.	page 128
	Installed Partition	SetEnvRemote	Set an environment variable for all the active instances of the installed partition.	page 214
	Node	SetEnvRemote	Set an environment variable for the Node Manager.	page 256
	RepositoryServer	SetEnvRemote	Sets an environment variable for the repository server.	page 311

	Agent	Command	Description	See:
Print status information	Any	DumpStatus	Prints the status of the managed object to Stdout.	page 69
Shut down the environment	Environment	Shutdown	Shuts down the Environment Manager.	page 205
Shut down a node	Node	Shutdown	Shuts down the Node Manager.	page 257
Remove locks held by applications being partitioned	Environment	ListAppConfig	Displays the list of applications currently being partitioned.	page 201
	Environment	ReleaseAppConfig	Releases the configuration lock for the named application.	page 204
Send messages to users in the environment	Environment	GenerateAlert	Sends a message to the Environment Manager, which posts the AlertFromSystem event.	page 200
Start garbage collection	OperatingSystem	RecoverMemory	Attempts to perform a stable garbage collection.	page 272

Managing a Central Repository

This section lists the tasks that you perform when you manage central repositories. For more information about For more information about how to manage central repositories, see *iPlanet UDS System Management Guide*.

The table maps the task to a particular Escript command or system agent command.

	Agent	Command	Description	See:
Shut down the repository server	RepositoryServer	Shutdown	Shuts down the repository server.	page 312
	RepositoryServer	ForceShutdown	Stops the repository server, even if users might still be connected.	page 309
Unlock reserved workspaces	RepositoryServer	UnlockWorkspace	Frees all locks held on the given workspace.	page 312
	RepositoryServer	ForceWorkspaceUnreserved	Removes the reservation a detached shadow holds on the workspace.	page 309

Managing Connected Environments

This section lists the tasks that you perform when you manage connected running environments. For more information about working with connected environments, see "Using the NameService Agent" on page 229 and *iPlanet UDS System Management Guide*.

The table maps the task to a particular Escript command or system agent command.

	Agent	Command	Description	See:
Show information about connected environments	NameService	ShowEnv	Displays information about an environment or all environments known to this environment.	page 241
Show information about partitions	NameService	ShowPart	Shows information about partitions known to this environment.	page 242
Delete information about partitions that cannot be accessed	NameService	RemoveLostParts	Deletes information about partitions that the Environment Manager can no long access.	page 241

	Agent	Command	Description	See:
Add an environment	NameService	ConnectEnv	Connects a target environment to the environment from which the command is invoked.	page 234
Remove an environment	NameService	DisconnectEnv	Separates the current environment from any other environments.	page 237
Show information about the Name Service	NameService	ShowAdmin	Shows information about the Name Service.	page 241
Shut down the Name Service	NameService	Shutdown	Shuts down the NameService agent and the corresponding Name Server.	page 243
Change name space directory	NameService	NsCd	Changes the current name space directory.	page 239
List name space directory	NameService	NsLs	Lists contents of a name space directory.	page 240

The following table maps a task to a particular system agent instrument:

	Agent	Instrument	Description	See:
Change the environment search path	NameService	EnvSearchPath	A list of environments to be used to located named objects in the name space for a group of connected environments.	page 244

To change the environment search path, use the Escript command `UpdateInstrument` command with the `EnvSearchPath` instrument on the NameService agent, as shown in the following example:

```
escript> FindSubAgent NameService
escript> UpdateInstrument EnvSearchPath “@Oakland(a):@NewYork”
```

Logging Information

This section lists the tasks that you perform to set up how to record data about the iPlanet UDS runtime system and running applications.

➤ To view the value of an instrument

1. Navigate to the agent that owns the instrument.

   Use the navigational commands described in "Navigating around the Agent Hierarchy" on page 115.

2. Invoke the `ShowAgent` command to view the agent's instruments.

   You will see all instruments defined for that agent and the current value of each instrument.

   To subsequently view the value of an instrument, invoke the `ShowInstrument` command.

➤ To specify when and where to log instrument data

1. Navigate to the Active Partition agent for which you want to log data.

   Use the navigational commands described in "Navigating around the Agent Hierarchy" on page 115.

2. Invoke the `UpdateInstrument` command for the `LogTimer` instrument, setting the `LogTimer` as active and setting the interval to the desired value (in milliseconds), as shown:

   ```
   escript> UpdateInstrument LogTimer “TRUE 600000”
   ```

   This will turn on the `LogTimer` instrument and specify how often it ticks.

3. Invoke the `UpdateInstrument` command for the `InstrumentLogging` instrument, setting the instrument as active, as shown:

   ```
   escript> UpdateInstrument InstrumentLogging TRUE
   ```

   This turns on the logging of instrument data to the active partition log file every time the `LogTimer` instrument ticks.

4. Invoke the `UpdateInstrument` command for the `LogFile` instrument, setting the active partition log file name, if you want a name different from the default name, as shown:

   ```
   escript> UpdateInstrument LogFile newfile.log
   ```

If the active partition is a standard partition, you can only change where the logged data goes by changing the `LogFile` instrument of the iPlanet UDS executor instance that is running the active partition. Note that the agents for standard active partitions do not have `LogFile` instruments.

➤ **To select an instrument to be logged**

1. Navigate to the agent that owns the instrument.

   Use the navigational commands described in “Navigating around the Agent Hierarchy” on page 115.

2. Invoke the `SetInstrumentLogging` command for an instrument of the DistObjectMgr agent, for example, the `EventsReceived` instrument, to set the instrument to be logged, as shown:

   ```
   escript> SetInstrumentLogging EventsReceived TRUE
   ```

   The `EventsReceived` instrument will now be logged to the active partition log file at the interval previously designated.

Setting up Logging with Agent Commands

The following table maps the task to a particular Escript command or system agent command.

	Agent	Command	Description	See:
Set log flags	Active Partition	ModLoggerRemote	Sets the log flags for the object managed by the agent.	page 127
	NameService	ModLoggerRemote	Sets the log flags for the object managed by the agent.	page 238
	Node	ModLoggerRemote	Sets the log flags for the object managed by the agent.	page 255
	Partition	ModLoggerRemote	Sets the log flags for the object managed by the agent.	page 289
	RepositoryServer	ModLoggerRemote	Sets the log flags for the object managed by the agent.	page 310
	any	ModLogger	Modifies the current logger flag settings for Escript.	page 81
Flush log files	Active Partition	FlushLogFiles	Flushes the log files for this partition.	page 127
	RepositoryServer	FlushLogFiles	Flushes the log files for this partition.	page 309
Log an instrument's values	any	SetInstrumentLogging	Specifies that the instrument be logged for partitions that are being logged.	page 89

Setting up Logging with Agent Instruments

Several agent instruments define settings that affect how data about the iPlanet UDS runtime system and running application is logged.

To view the values of these instruments, use the `ShowAgent` or `ShowInstrument` command. To change the values of these instruments, use the `UpdateInstrument` command for each instrument. For information about using these commands, see "Accessing the Instruments for an Agent" on page 117.

The following table maps the task to a particular system agent instrument:

	Agent	**Instrument**	**Description**	**See:**
Log instrument values	Active Partition	InstrumentLogging	Turns on automatic logging of instruments to the active partition log file.	page 129
	RepositoryServer	InstrumentLogging	Turns on automatic logging of instruments to the active partition log file.	page 313
	Environment	InstrumentLogging	Turns on automatic logging of instrument values to the environment log file.	page 207
Change the log file name	Active Partition	LogFile	(Compiled partitions only) Defines the name of the file to use when logging instruments for the active partition.	page 130
	RepositoryServer	LogFile	(Compiled partitions only) Defines the name of the file to use when logging instruments for the active partition.	page 314
	Environment	EnvironmentLog	Set the name of the file to use when logging important events for the Environment Manager.	page 206
Set interval at which logging occurs	Active Partition	LogTimer	Turns on and sets interval for instrument logging events within the active partition.	page 131
	RepositoryServer	LogTimer	Turns on and sets interval for instrument logging events within the active partition.	page 315

Managing Applications

This section lists Escript and system agent commands by task. The tasks in this section involve configuring, deploying, and managing applications.

Configuring Applications

This section lists the tasks that you perform to set up how to configure application and library distributions before installing them in the environment.

Before you can configure an application, you first need to load the application distribution into your environment.

➤ **To load an application distribution**

1. Make sure the application distribution is in the `FORTE_ROOT/appdist` subdirectory for your environment on a node in your environment. For example, if your environment is named central, and the application is the cl0 version of Auction, then the application distribution must be in the `FORTE_ROOT/appdist/central/auction/cl0` directory.
2. Navigate to the node containing the distribution.
3. Invoke the `ListDistribs` command to confirm that the application distribution resides on that node.
4. Invoke the `LoadDistrib` command to load the distribution into the environment repository.

➤ **To modify a partitioning configuration**

1. Invoke the `LockEnv` command.

   You need to lock the environment definition to make configuration changes.
2. Modify the configuration using commands in the table later in this section.
3. Invoke the `Commit` command to save changes and unlock the environment definition.

➤ **To reconfigure an installed application**

1. Shut down your application by navigating to the Application agent and invoking the `Shutdown` command.
2. Lock your active environment by invoking the `LockEnv` command.

3. Reconfigure the application using the appropriate commands.

   See the table in this section.

4. Save and unlock your environment definition by invoking the `Commit` command.

5. Reinstall the application by navigating to the Application agent and invoking the `Install` command.

When you reinstall the application, the system management services perform an incremental installation, installing only those partitions necessary to update the installed application to reflect the new configuration.

The following table maps the task related to configuring an application to a particular Escript command or system agent command:

	Agent	Command	Description	See:
List available application distributions	Environment	ListDistribs	List the application distributions available locally on the node running Escript.	page 202
	Node	ListDistribs	List the application distributions available on the node.	page 254
Load an application distribution	Environment	LoadDistrib	Loads the specified application distribution into the environment repository from the node that is running the Escript utility.	page 202
	Node	LoadDistrib	Loads the specified application distribution into the environment from the node.	page 255
Assign a component to a node	Application	AssignAppComp	Assigns the specified application component in the current application for installation on the specified node.	page 142
	Partition	Assign	Assigns the partition to the specified node.	page 286

	Agent	Command	Description	See:
Remove a component from a node	Application	UnassignAppComp	Removes the assignment of an application component from a node.	page 149
	Partition	Unassign	Removes the assignment of the partition from a node.	page 294
Define a component as compiled	Application	SetAppCompCompiled	Declares whether a partition or library is to be used in compiled or iPlanet UDS executor form.	page 146
	Partition	SetCompiled	Declares whether the partition is to be used in compiled or iPlanet UDS executor form.	page 291
Set the replication count	Partition	SetRepCount	Sets the replication count for the partition on a particular node.	page 292
Enable a partition for automatic startup	Application	EnableAppComp	Enables automatic startup of a partition on a node.	page 145
	Partition	Enable	Enables automatic startup of the partition on a node.	page 288
Disable a partition for automatic startup	Application	DisableAppComp	Disables automatic startup of a partition on a node.	page 144
	Partition	Disable	Disables automatic startup of the partition on a node.	page 287
Define the startup arguments for a partition	Partition	SetArgs	Sets the argument string used to startup the partition on a particular node.	page 289

Installing Applications

This section lists the tasks that you perform to install application and library distributions.

The environment definition must be unlocked before you can install the application.

➤ **To install an application**

1. Navigate to the Application agent for the loaded application distribution.

2. Invoke the `Install` command to install the assigned application partitions onto all nodes that have Node Managers running in the environment.

   If the installation does not complete, the Application agent has information about the steps that need to be completed in its InstallationSteps instrument, which is described in "InstallationSteps" on page 150.

The following table maps each task related to installing applications to a particular Escript command or system agent command.

	Agent	**Command**	**Description**	**See:**
List available application distributions	Environment	ListDistribs	List the application distributions available locally on the node running Escript.	page 202
	Node	ListDistribs	List the application distributions available on the node.	page 254
Load an application distribution	Environment	LoadDistrib	Loads the specified application distribution into the environment repository from the node that is running the Escript utility.	page 202
	Node	LoadDistrib	Loads the specified application distribution into the environment from the node.	page 255
Install an application	Application	Install	Installs the application into the current environment.	page 146
	Node	ListAppsToInstall	Lists the names of the applications that need to be installed on the node.	page 254
	Node	InstallApp	Installs partitions for an application on the node.	page 253
Uninstall an application	Application	Uninstall	Removes the definition of the current application from the environment.	page 150
Remove outstanding installation locks	Application	ReleaseLock	Release any installation locks on the current application.	page 146

To see the steps that need to be performed to complete the installation of an application, check the InstallationSteps instruments for the application.

Managing Running Applications

This section lists some of the tasks that you perform to manage running applications.

➤ **To start all enabled server partitions in an application**

1. Navigate to the Application agent for the application.
2. Invoke the `Startup` command.

   All enabled installed partitions within the application will start instances until the number of running instances matches the replication count for each installed partition.

➤ **To start a single installed partition**

1. Navigate to the Installed Partition agent that represents the partition you want to start.
2. Invoke the `Startup` command.

   One active partition is started. Starting a single installed partition overrides the start option properties (enabled or disabled and replication count).

The following table maps the task to a particular Escript command or system agent command.

	Agent	Command	Description	See:
Start an application	Application	Startup	Start all server partitions (with all their replicates) for the specified application.	page 149
	Installed Partition	Startup	Starts one instance of the current installed partition on the current node.	page 216
	Node	StartInstPart	Starts one instance of an installed partition on the node.	page 258
	Partition	Startup	Starts all installed server partitions (with all their replicates) represented by the logical partition.	page 294
Shut down an application	Application	Shutdown	Shuts down all server partitions running in the application on all nodes.	page 147
Shut down partitions	Application	ShutdownSubAgent	Shuts down the named subagent and its managed object.	page 148
	Active Partition	Shutdown	Shuts down the active partition.	page 137
	Installed Partition	Shutdown	Shuts down all active instances of the installed partition.	page 215
	Installed Partition	ShutdownSubAgent	Shuts down the named subagent and its managed object.	page 215
	Partition	Shutdown	Shuts down all active instances of the partition.	page 293
	Partition	ShutdownSubAgent	Shuts down the named subagent and its managed object.	page 293
Troubleshoot	any	DumpStatus	Prints the status of the managed object to Stdout.	page 69
	Active Partition	DebugPartition	Places this partition under the control of a C++ debugger.	page 126
	RepositoryServer	DebugPartition	Places this partition under the control of a C++ debugger.	page 309

Chapter 2

General Escript Commands

This chapter is an alphabetically ordered reference for the Escript commands that are not associated with a particular system agent. This chapter includes a description of each of these general Escript commands and instructions for using it.

For information about using system agent commands—commands that are associated with a particular system agent—see Chapter 3, "Using iPlanet UDS System Agents" and Chapter 4, "iPlanet UDS System Agent Commands and Instruments."

A brief introduction summarizes and groups the general Escript commands according to their functions.

For information on using Escript commands and system agent commands to perform system management tasks, see Chapter 1, "Using the Escript Utility."

Escript Commands

The following is a listing of all Escript commands. A dot (•) indicates that the command is available in environment edit mode only.

Add3GLProj

The `Add3GLProj` command adds the specified restricted 3GL project to the list of those supported by the current node.

Add3GLProj *project_name*

Argument	Description
project_name	The name of a C, DCE, or ObjectBroker project that has been defined in the repository.

All access to 3GL routines is through C, DCE, or ObjectBroker projects that have been defined in the repository. You use the steps described in *Integrating with External Systems* to complete the definition, compilation, and linking of a these shared images before using it from TOOL code.

C, DCE, and ObjectBroker projects use a restricted property of `TRUE` to indicate that they can only run on some of the nodes within the environment. Thus, you can only partition that project onto a node that can support it.

The `Add3GLProj` command indicates that the current node supports the C, DCE, or ObjectBroker project specified in the *project_name* argument. Use the `FindNode` command to set the current node. Before invoking the `Add3GLProj` command, you must lock the environment by invoking the `LockEnv` command.

If a C, DCE, or ObjectBroker project has a restricted property of `FALSE`, you do not need to use the `Add3GLProj` command on any nodes, because the partitioning system assumes that the project is available on every node.

• AddCommProtocol

The `AddCommProtocol` command adds the specified communications protocol to the current node.

AddCommProtocol *protocol_name*

Argument	Description
protocol_name	The name of a supported communications protocol for the node type.

The `AddCommProtocol` command indicates that the current node has installed support for a designated communications protocol. Use the `FindNode` command to set the current node.

The `protocol_name` argument gives the name of the communications protocol type that is to be enabled for this node. Valid values are:

Protocol Name	Description
Berkeley Sockets	Standard Berkeley socket library. This is the standard TCP/IP interface on most UNIX systems.
Digital DECnet	Digital DECnet protocol for VMS.
Digital UCX	Digital TCP/IP protocol for VMS.
Pathworks DECnet	Digital Pathworks DECnet protocol.
Pathworks TCP/IP	Digital Pathworks TCP/IP protocol.
PC-NFS	PC/NFS protocol for MS/.
TLI	TCP/IP TLI protocol for UNIX systems.
UNIX Domain Sockets	Berkeley socket library for interprocess communication on a single node. This is available on most UNIX systems.
Windows Sockets	Protocols on MS/Windows that support the Windows Sockets interface.

The protocol that you specify must be supported for the architecture of the current node. See the *Release Notes* for a current list of supported protocols for each architecture type.

If the protocol name has embedded spaces, you must surround it in double quotes. Also, make sure that you maintain the correct upper and lower case letters.

• AddExternalRM

The `AddExternalRM` command adds the specified external resource manager name to the current node, and sets the resource manager type to the value specified.

AddExternalRM *resource_manager_name resource_manager_type*

Argument	Description
resource_manager_name	Any name that you designate to identify the resource manager uniquely within the environment definition.
resource_manager_type	One of the supported resource manager types for the node.

You specify access to relational databases by defining external resource managers on specific nodes in the environment definition. These resources provide the information needed by the partitioning system to provide access to database managers from TOOL code.

Use this command to specify the external resource manager name and type for each of the nodes where you have an accessible installation of one of the supported relational database managers within your environment.

The *resource_manager_name* is a name that you provide to identify the specific installation of a database manager. This name can subsequently be specified in the Project Workshop on the Service Object Property dialog for DBResourceMgr and DBSession service objects. By tying the service object to the external resource manager name, the partitioning system can provide the correct path between your TOOL code and the needed relational database. You can use embedded spaces within the name by surrounding it with double quotes.

You can specify the same resource manager name on more than one node within the environment definition. In this case, the partitioning system provides access to each of the nodes with the named external resource manager. The resource manager name must be of the same resource manager type on all the defining nodes. You cannot use different database manager types on different nodes. You can use simulated environments as a means to provide several different configurations for an application that is portable across database types. In a different simulated environment definition, the same named resource manager can be given a different type. In this case, it is your responsibility to ensure that your TOOL code works correctly on the different database managers.

The *resource_manager_type* is one of the valid relational database types. Valid values are:

Resource Manager Type	Description
DB2	DB2/6000 database system.
Informix	Informix database system.
Ingres	Ingres database system
ODBC	ODBC access to any supported database system.
Oracle	Oracle Version 7 database system.
Rdb	Rdb database system.
Sybase	Sybase database system.

The resource manager that you specify must be supported for the architecture of the current node. See the *iPlanet UDS System Installation Guide* for a current list of supported resource managers for each architecture type.

• AddNode

The `AddNode` command adds a node with the specified name to the current environment definition, using as starting values all values from the specified template node.

AddNode *node_name* [*existing_node_name* | *template_node_name*]

Argument	Description
node_name	A unique name within the environment definition.
existing_node_name	The name of another node with the same architecture type, which sets the starting properties for the node.
template_node_name	The name of a template node, which sets the starting properties for the node.

This command adds a new node to the current environment definition, as specified by the `FindEnv` command. Before invoking the `AddNode` command, you must lock the environment by invoking the `LockEnv` command.

The *node_name* for the new node must be unique within the environment definition.

You can use either the name of an existing node in the environment or the name of a template node (the `template_node_name` argument) to provide a set of starting values for the new node. All properties (except the name) and definitions of external resources, communication protocols, and 3GL projects are taken from the existing or template node, and set for the newly created node. You can then use other Escript commands to modify the definition as necessary.

If any of the names contains spaces, you must surround the name with double quotation marks.

An existing node name is any previously defined node of the same architecture. If you do not have a node of the same architecture type, you can use the following values for the template node name:

Template Node Name	Description
Alpha OpenVMS Client	Alpha client node running OpenVMS.
Alpha OpenVMS Server	Alpha server node running OpenVMS.
Alpha OSF/1 Client	Alpha client node running Digital UNIX.
Alpha OSF/1 Server	Alpha server node running Digital UNIX.
Aviion Intel DGUX Client	Aviion Intel client node running DG/UX.
Aviion Intel DGUX Server	Aviion Intel server node running DG/UX.
HP 9000 HP/UX Client	HP 9000 PA-RISC client node running HP/UX.
HP 9000 HP/UX Server	HP 9000 PA-RISC server node running HP/UX.
Mips SINIX Client	Mips client node running SINIX.
Mips SINIX Server	Mips server node running SINIX.
PC NT Client	Intel PC client running Windows NT.
PC NT Server	Intel PC server running Windows NT.
RS/6000 AIX Client	RS/6000 client node running AIX.
RS/6000 AIX Server	RS/6000 server node running AIX.

Template Node Name	Description
Sequent DYNIX/ptx V4 Client	Sequent client node running DYNIX/ptx Version 4.
Sequent DYNIX/ptx V4 Server	Sequent server node running DYNIX/ptx Version 4.
SPARC Solaris Client	SPARC client node running Solaris.
SPARC Solaris Server	SPARC server node running Solaris.

NOTE The following templates are no longer supported. You should upgrade these templates to the new templates. You should not define new nodes that use these templates.

iPlanet UDS Release 2 Template Node Name	Description	New Template Name
AViiON DGUX Client	Aviion Motorola client node running DG/UX.	AViiON Intel DGUX Client
AViiON DGUX Server	Aviion Motorola server node running DG/UX.	AViiON Intel DGUX Server
Sequent DYNIX/ptx Client	Sequent client node running DYNIX/ptx (Version prior to Version 4).	Sequent DYNIX/ptx V4 Client
Sequent DYNIX/ptx Server	Sequent server node running DYNIX/ptx (Version prior to Version 4).	Sequent DYNIX/ptx V4 Server

The `AddNode` command adds specific nodes within the environment (typically each of your server nodes), and to add “model nodes,” which are node definitions that can be shared for a number of similarly configured machines (typically for a set of clients). Use the `SetNodeModel` command to set the model property for a node.

New nodes that were not added to the environment definition with the `AddNode` command can register themselves within an active environment. When you start a Node Manager on a node connected to an environment, iPlanet UDS automatically adds a node with the proper name and architecture to the active environment's definition.

See the description of the `nodemgr` command in *iPlanet UDS System Management Guide* for information on options for auto-registration of nodes.

AddPath

The `AddPath` command adds the specified directories to the current search path. This search path is used by any of the commands that take a file name as an argument.

AddPath *directory_name* [;*directory_name*...]

Argument	Description
directory_name	The name of a directory in which to look for files which are specified without a path.

Most of the commands that have input file arguments allow you to specify the name of the input file without a full directory specification. If you don't specify a directory, commands use the directory search path, as defined by the `SetPath` and `AddPath` commands, to find the file. Each command checks the directories in the directory search path until it finds a file that matches the unexpanded name.

Use the `AddPath` command to add one or more directories to the end of the current list of directories in the search path. Use the `SetPath` command to reset the entire directory search path list.

Each directory name should be specified as a full directory path name. By default, specify directories in the local operating system directory format. If you have previously invoked the `UsePortable` command, then you should specify the directory name in iPlanet UDS portable format, which is a UNIX style directory format. To specify more than one directory, separate the directory names with semi-colons.

You can embed environment variable names within directory names:

${*environment_variable_name***}**

The dollar sign and brackets indicate that the name inside the brackets is an environment variable, and the current setting of the environment variable replaces the entire specification.

You can also use the following syntax to expand the environment variable name, but convert it to a portable file format as well:

%{*environment_variable_name*}

The percent sign and brackets indicate that the name inside the brackets is an environment variable, and the current setting of the environment variable replaces the entire specification. You can use this syntax to convert you environment variables to portable format if you have invoked the `UsePortable` command, but have directories specified in environment variables in local format.

```
escript> AddPath /mydisk/mydir;${ENV_VAR}/subdir
escript> AddPath c:\mydir;${ENV_VAR}\subdir
escript> AddPath “Mac HD:Apps:TempFolder”;${ENV_VAR}:Sub
escript> AddPath $dka0:[path];${ENV_VAR}:[otherdisk.otherdir]
escript> UsePortable
escript> AddPath %{FORTE_ROOT}/install/examples
```

Cd

The `Cd` command changes the current working directory.

Cd *directory_name*

Argument	**Description**
directory_name	The name of a directory to make the new working directory.

This command calls the operating system to change the current working directory.

By default, you should specify the `directory_name` argument in local operating system format. If you have previously invoked the `UsePortable` command, specify the directory name in portable file name format. You can embed special syntax in the directory name to have environment variable expansion performed on the specified name. See the `SetPath` command for more details.

CollectMem

The `CollectMem` command runs the automatic memory reclamation on Escript.

CollectMem

Generally, the iPlanet UDS system performs memory reclamation (garbage collection) automatically whenever memory is running low. You can use the `CollectMem` command to explicitly invoke memory reclamation, for more repeatable timings, for example.

CommentOff

The `CommentOff` command tells iPlanet UDS to stop writing the comments in Escript script files to standard output when they are processed by the `Include` command.

CommentOff

Use this command in scripts to stop Escript from writing out the comments to standard output that it reads from files processed by the `Include` command. By default, these comments are not written to standard output. You can use the `CommentOn` and `CommentOff` commands to toggle this setting.

CommentOn

The `CommentOn` command tells iPlanet UDS to start writing the comments in Escript script files to standard output when they are processed by the `Include` command.

CommentOn

Use this command in scripts to make Escript write out the comments to standard output that it reads from files processed by the `Include` command. By default, comments are not written to standard output. You can use the `CommentOn` and `CommentOff` commands to toggle this setting.

Commit

The `Commit` command saves all changes to the environment repository. It also implicitly unlocks the environment definition after the changes are committed.

Commit

Any outstanding changes to the environment definition, such as newly created nodes, modified partition property settings, etc. can be saved by invoking the `Commit` command.

This command implicitly performs an `UnlockEnv` command immediately after the changes are saved. If you intend to make additional changes to the environment definition, you must invoke the `LockEnv` command again before making the changes.

The `Commit` command is only available while Escript is working on one of the agent types that can make changes to the environment repository: Environment agent, Node agent, Application agent, or Installed Partition agent. You cannot invoke the `Commit` command while you are in the environment editing mode of Escript. If you are editing the definition of a simulated environment (after invoking the `FindEnv` or `NewEnv` commands), iPlanet UDS prompts you to commit or discard your changes when you invoke the `Quit` or `Exit` command to get out of the environment editing mode of Escript.

DumpStatus

The `DumpStatus` command writes out detailed troubleshooting information to the log file for the current agent.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

This command prints out detailed troubleshooting information to the log file for the current agent. By default, the `DumpStatus` request passes to the subagents of the current agent. This command can produce a lot of output, so it should be used very sparingly.

Set this argument to 1 if you do not want to pass the `DumpStatus` request to all subagents of the current agent. The default value is `0`, which propagates the request.

The information included in the output is specific to each agent, and is not documented in detail. If you design and build your own agents, you should implement a `DumpStatus` command to print out basic troubleshooting information about your managed object.

The `DumpStatus` command is not available when you are in the environment editing mode of Escript, as it requires a current active agent for its operation.

EditEnv

The `EditEnv` command puts Escript into environment editing mode to edit the active environment's definition.

EditEnv

After you invoke this command, the command prompt for Escript changes to `envedit>`. You can subsequently invoke Escript utility commands and the commands described in "Editing an Environment Definition in Environment Editing Mode" on page 40. After you complete the changes to the environment definition, you should invoke the `Quit` or `Exit` command to get out of the environment editing mode.

You must lock the active environment using the `LockEnv` command before invoking the `EditEnv` command. After you leave the environment editing mode (by invoking `Quit` or `Exit`), you should either commit or abort your changes to the environment definition by invoking the `Commit` or `UnlockEnv` command.

The `EditEnv` command is only available while Escript is working on one of the agent types that can make changes to the environment repository: Environment agent, Node agent, Application agent or (logical) Partition agent.

ExecCmd

The `ExecCmd` command executes the specified operating system command.

ExecCmd *opsys_command* [*bg_flag*] [*in_file*] [*out_file*] [*err_file*]

Argument	Description
opsys_command	A valid operating system command appropriate to the system on which you are running Escript.
bg_flag	Specifies whether you should run the command synchronously with a value of `0` (the default), or asynchronously with a value of `1`.
in_file	An alternate input file for the operating system command.
out_file	An alternate output file for the operating system command.
err_file	An alternate error file for the operating system command.

Enter a valid operating system command for this argument. To include command line arguments, specify the command and its arguments in double quotes.

Special syntax for OpenVMS On OpenVMS, if you want OpenVMS to execute the command, you need to specify the characters "`$ `" (dollar-sign and a space) before the command name so that OpenVMS knows to look for an executable (.com or .exe) file or DCL symbol. If you explicitly specify a path and file extension, OpenVMS tries to execute that particular file in the specified path. You cannot specify both "`$ `" and a path.

The following example shows how you would use the `ExecCmd` command with the "`$ `" syntax. In this example, the `ExecCmd` command invokes the OpenVMS `SHOW DEFAULT` command, which prints the current directory to the `A.OUT` file. This example then invokes the `ListFile` command to display the contents of the `A.OUT` file:

```
ExecCmd "$ SHOW DEFAULT" "" A.OUT A.OUT
fscript > ListFile A.OUT
>>> BEGIN LISTING <<<
 1>   USER:[TOM]
>>> END LISTING <<<
```

The following example shows how you could use the ExecCmd command with a full path name and filename to invoke the iPlanet UDS Corbagen executable. Note that this command should be executed on one line:

```
ExecCmd "FORTE_ROOT:[INSTALL.BIN.ALPHA]CORBAGEN /CORBA_TYPE=OBB
/IDL_FILE=NEW.IDL"
```

By default, the bg_flag argument is set to 0 to indicate that the command is to be run synchronously until it completes. Set it to 1 to indicate that the command is to be started in the background.

Use the in_file, out_file and err_file arguments to redirect the input, output or errors for the command.

Exit

The Exit command exits Escript, prompting you to save if there are outstanding changes to the environment.

Exit

This command is also used to leave the environment editing mode of Escript, which had been entered through the EditEnv, FindEnv or NewEnv commands.

ExitIfNoEnv

The ExitIfNoEnv command tells Escript to exit if it loses contact with an active Environment Manager. Escript, or any iPlanet UDS application, can lose contact with the active Environment Manager if the Environment Manager fails, or if software or hardware communication links to the Environment Manager fail.

ExitIfNoEnv

This command is typically used in batch scripts for detecting error conditions. Use it to stop executing Escript if no Environment Manager is currently running. This prepares Escript to exit the session if it detects that the Environment Manager has become unavailable.

You cannot invoke the ExitIfNoEnv command while you are in the environment editing mode of Escript.

ExitStatus

The `ExitStatus` command sets a return value for this session of Escript. This value is returned to the routine that started this session of Escript when Escript exits.

ExitStatus *integer*

Argument	Description
integer	The value returned to the routine that started Escript. By default, the return value is `0` for if Escript completed without errors or `1` if Escript exited abnormally. You can define other numeric values that are meaningful to you.

The routine that started Escript can check this return value to determine whether Escript completed without errors or exited abnormally.

FindActEnv

The `FindActEnv` command designates the active Environment agent as the current agent.

FindActEnv

When you first bring up Escript, the active Environment agent is considered to be the current agent, so you normally do not invoke the `FindActEnv` command. However, if you change the current agent by using various `FindSubAgent` commands, you can switch the current agent back to the active agent by invoking the `FindActEnv` command.

You cannot invoke the `FindActEnv` command while you are in the environment editing mode of Escript.

• FindNode

The `FindNode` command designates the specified node within the current environment definition as the current node for subsequent editing commands.

FindNode *node_name*

Argument	Description
node_name	The name of a node within the current environment definition.

A number of the environment editing Escript commands operate on the current node, by default. This command designates a node within the environment definition as the current node.

The specified node name must have previously been set up, either by creating a default entry for a node when the Node Manager first is started on that node, or by explicitly invoking the `AddNode` command for the node.

FindParentAgent

The `FindParentAgent` command makes the parent agent the current agent.

FindParentAgent

When Escript first starts, the current agent is the active Environment agent. When you invoke `FindSubAgent`, `FindParentAgent` and `FindSavedAgent` commands, the current agent for Escript changes. To move up the agent hierarchy, invoke the `FindParentAgent` command. See Chapter 3, “Using iPlanet UDS System Agents” or *iPlanet UDS System Management Guide* for a description of the agent hierarchy.

You cannot invoke the `FindParentAgent` command while you are in the environment editing mode of Escript. You also cannot invoke the `FindParentAgent` command while the current agent is the active Environment agent, as it is the root of the agent hierarchy, and has no parent agent.

FindSavedAgent

The `FindSavedAgent` command changes the current agent to one of the saved agents.

FindSavedAgent [*agent_tag*]

Argument	Description
agent_tag	The tag name for an agent as designated by the `SaveAgent` command.

If you are working with several agents, you can quickly move to an agent without having to fully navigate the agent hierarchy. This command is particularly helpful if the agents are not in a simple parent-child relationship. The `FindSavedAgent` command addresses this problem. First you must use the `SaveAgent` command to give an arbitrary tag name to the current agent. After you have changed the current Escript agent, you can invoke the `FindSavedAgent` command to restore the saved agent as the current agent.

The `agent_tag` argument refers to a tag specified in a previous `SaveAgent` command. It is an arbitrary string value. If it contains embedded spaces, you must specify it within quotes.

iPlanet UDS automatically saves the last current agent of each type as a saved agent, so that you can reference the last agent of that type without navigating through the hierarchy again. You can use the following iPlanet UDS tags:

Tag	Object Type
Last Active Partition	Active Partition
Last Application	Application
Last Environment	Environment
Last Installed Partition	Installed Partition
Last Node	Node
Last Partition	Partition

The `FindSavedAgent` command cannot be invoked while Escript is in the environment editing mode.

FindSubAgent

The `FindSubAgent` command moves to a subagent of the current agent. That subagent becomes the current agent.

FindSubAgent *agent_name*

Argument	Description
agent_name	The name of an agent that is a subagent to the current agent.

Use the `FindSubAgent` command to move to one of the subagents of the current agent. After you move to that subagent, all agent commands are directed to the newly-designated agent.

You can see the subagents of an agent by invoking the `ShowAgent` command. You can then use a name shown in the output to designate one of the subagents on a `FindSubAgent` command as the current agent.

The `agent_name` argument can be the full agent name, or can be a special shorthand for some of the iPlanet UDS-supplied agents. For the Installed Partition and Active Partition agents, you can give only the “unique” portion of the agent name to navigate to the subagent. For example, to navigate from a Node agent to an Installed Partition agent, you can drop the node name from the installed partition name (and the underscore). To move from an Installed Partition agent to an Active Partition agent, you can specify only the hexadecimal value of the active partition.

Agent names can also be multi-level, with each level preceded by a “`/`”. This allows you to move directly to a subagent more than one level below the current agent. For example, to move from the Environment agent to the logical (client) Partition agent for the application MyApp_cl0, you can invoke the command:

```
escript> FindSubAgent MyApp_cl0/Client
```

You cannot invoke the `FindSubAgent` command while you are in the environment editing mode of Escript. You also cannot invoke the `FindSubAgent` command if the current agent has no subagents (such as the DistObjectMgr agent).

Help

The `Help` Command lists help for commands.

Help [*command_name* | *match_string*]

Argument	Description
command_name	The name of an Escript command.
match_string	A partial name of a command followed by an asterisk.

If you invoke the `Help` command with no argument, it lists all the Escript commands appropriate in the current context. Only the generally available commands and the agent specific commands are listed. If you are in environment editing mode, only the generally available commands, and the environment editing commands are listed.

If a `command_name` is given, the `Help` command lists the arguments and a short description for the specified command.

If a `match_string` argument is given (an asterisk at the end of the string), the `Help` command lists all matching commands, their arguments, and a short description.

Include

The `Include` command executes the commands in the specified script file.

Include *file_name*

Argument	Description
file_name	The name of a file containing Escript commands to execute.

You can store a commonly executed set of Escript commands in a script file and then execute them by invoking the `Include` command.

The `file_name` argument specifies the name of the file containing a set of Escript commands. The `Include` command uses the current directory search path to determine the location of the included file. For more information, see the `SetPath` command in "SetPath" on page 94. The file name is given in iPlanet UDS portable name syntax if the `UsePortable` command has been invoked. If the `UseLocal` command has been invoked, or if neither has been invoked, then the file name is given in local operating system syntax.

ListFile

The `ListFile` command prints the contents of the specified file to standard output.

ListFile *file_name*

Argument	Description
file_name	The name of the file to print.

Use this command to print the text in a file to standard output.

The `file_name` argument specifies the name of the file to print to standard output. This command uses the current directory search path to determine the location of the file. For more information, see the `SetPath` command in "SetPath" on page 94. The file name is given in iPlanet UDS portable name syntax if the `UsePortable` command has been invoked. If the `UseLocal` command has been invoked, or if neither has been invoked, then the file name is given in local operating system syntax.

ListSavedAgents

The `ListSavedAgents` command lists all the saved agent tags.

ListSavedAgents

The `ListSavedAgents` command lists the tags, names, and agent type of all agents that have been saved using the `SaveAgent` command. You can then use the `FindSavedAgent` command with a tag name, to reset the current agent to one of the agents in the list.

This command also lists the agents that iPlanet UDS automatically saves: the last current agent of each type. iPlanet UDS uses the following tags when storing these agents:

Tag	Object Type
Last Active Partition	Active Partition
Last Application	Application
Last Environment	Environment
Last Installed Partition	Installed Partition
Last Node	Node
Last Partition	Partition

You cannot invoke the `ListSavedAgents` command while you are in the environment editing mode of Escript.

LockEnv

The `LockEnv` command obtains an exclusive lock on the environment until the next `Commit` or `UnlockEnv` command.

LockEnv [*wait_flag*]

Argument	Description
wait_flag	Specify whether Escript should wait (value `1`) or give an error (value `0`) if the lock cannot be obtained immediately.

All commands in Escript that change the definition of the environment and its nodes require that the environment be exclusively locked until the updates are completed. The `LockEnv` command obtains the needed exclusive lock on the environment.

Set the `wait_flag` argument to `0` to give an error if the environment lock is not available; set *it* to `1` to have Escript wait until the environment lock can be obtained, and then resume processing. The default is `0`.

After obtaining the lock, you can invoke any number of environment update commands.

One of the most common commands to invoke after `LockEnv` is the `EditEnv` command, which puts you in the environment editing mode of Escript. After exiting that mode, you can then `Commit` the changes that you made, which implicitly unlocks the environment.

When you invoke the `LockEnv` command, the current definition of the environment is automatically refreshed, so you need not invoke the `RefreshEnv` command first. If the `LockEnv` fails, it displays a list of all current development sessions in the Partition Workshop.

To unlock the environment, you can either invoke a `Commit` command, which saves the changes to the environment and then releases the lock, or the `UnlockEnv` command, which prompts you to save if there are changes, and then releases the lock.

You should be very careful to keep an environment locked for the shortest possible time, as other concurrent users cannot update the environment, or partition applications, for as long as the lock is held.

The `LockEnv` command is only available if you are working on a current agent that can affect the definition of the environment: Environment agent, Node agent, Application agent or (logical) Partition agent.

Ls

The `Ls` command lists the files in a directory.

Ls [*directory_name*]

Argument	Description
directory_name	The name of a directory. The default is the current working directory for Escript.

The `Ls` command lists the full names and permissions for all the files in the specified directory. If no directory is specified, the current working directory for Escript is used. The current working directory is either the directory in which Escript was started, or the last directory set using the `Cd` command.

The `directory_name` argument is specified in local operating system format, by default. If you invoke a `UsePortable` command, however, you must use the portable file format to specify the directory name.

ModLogger

The `ModLogger` command modifies the current logger flag settings for Escript.

ModLogger +(*logger_flags*) | -(*logger_flags*)

Argument	Description
+(*logger_flags*)	Turn on the logger flag settings given in the parentheses.
-(*logger_flags*)	Turn off the logger flag settings given in the parentheses.

To start logging, use the '+' followed by a set of logger settings in parentheses. To stop logging, use the '-' followed by a set of logger settings in parentheses.

The settings specified with the `ModLogger` command modify the logger flag settings that were set when Escript started execution. The original settings for Escript were set either with the `-fl` flag on the Escript command or using the FORTE_LOGGER_SETUP environment variable.

The modified logger settings are only applied to the first file specified in the original logger settings for Escript.

See the `LogMgr` class in the Framework Library online Help for a detailed description of the logger flag syntax. Sample uses of the `ModLogger` command are:

```
escript> ModLogger +(trc:os:1:1 cfg:c4:2-3:1)
escript> ModLogger -(cfg:c4)
```

Mv

The `Mv` command renames a file in the local file system.

Mv *file1_name file2_name*

Argument	Description
file1_name	The name of the file to rename.
file2_name	The new name for the file.

Use this command to rename a file. The two file names are specified relative to the current working directory. This is either the directory in which Escript was started, or the last directory specified in the `Cd` command.

The `file1_name` and `file2_name` arguments are specified in local operating system format, by default. If you have invoked the `UsePortable` command, however, the two arguments must be specified in portable file format.

There should be no existing file with the same name as the `file2_name` argument.

Pwd

The `Pwd` command displays the name of the current working directory.

Pwd

The current working directory is either the directory in which Escript was started, or the last directory specified in the `Cd` command.

Quit

The `Quit` command exits Escript, prompting you to save if there are outstanding changes to the environment definition.

Quit

The `Quit` command is also used to leave the environment editing mode of Escript, which had been entered through the `EditEnv`, `FindEnv` or `NewEnv` commands.

RefreshEnv

The `RefreshEnv` command forces an immediate refresh of the information about the active environment from the Environment Manager.

RefreshEnv

When Escript is running, it caches information about the environment definition so that it can execute more quickly. However, if other concurrent users running Escript or the Environment Console change the environment definition, every executing instance of Escript does not get the set of changes immediately. Use the `RefreshEnv` command to refresh your copy of the definition of the environment. If the information about the environment definition or any installation activity changes, you must use the `RefreshEnv` command to ensure that the information you have is current.

If you invoke the `LockEnv` or `ExportEnv` commands, you automatically gets an up-to-date copy of the environment definition, so you need not invoke the `RefreshEnv` command before invoking those commands.

For all dynamic information about the execution of partitions and applications in the environment, Escript keeps completely up to date with the information in the Environment Manager, and the data does not need to be refreshed manually from the environment.

The `RefreshEnv` command is only available if you are working on a current agent that can affect the definition of the environment: Environment agent, Node agent, Application agent or (logical) Partition agent.

• Remove3GLProj

The `Remove3GLProj` command removes a 3GL project from the list of those supported by the current node and in an environment definition.

Remove3GLProj *project_name*

Argument	Description
project_name	The name of a 3GL project that has been registered for the node, as specified in a previous `Add3GLProj` command.

The `Remove3GLProj` command removes a 3GL project that had previously been registered on the current node by invoking the `Add3GLProj` command. The current node is selected with the `FindNode` command.

Before invoking the `Remove3GLProj` command, you must lock the environment definition.

The `Remove3GLProj` command does not actually remove the 3GL shared library from the installation. You must remove it using standard utilities on your operating system.

Use the `ShowNode` command to list all 3GL projects currently registered for a node.

• RemoveCommProtocol

The `RemoveCommProtocol` command removes the specified communication protocol from the current node.

RemoveCommProtocol *protocol_name*

Argument	Description
protocol_name	The name of a protocol registered for a node, as specified in a previous `AddCommProtocol` command.

Use this command to remove a communications protocol that had previously been registered on the current node using the `AddCommProtocol` command. The current node is designated with the `FindNode` command.

The `protocol_name` argument gives the name of the protocol to remove. (See "• AddCommProtocol" on page 60 for a list of protocols.) To see which protocols are enabled on the current node, use the `ShowNode` command.

• RemoveExternalRM

The `RemoveExternalRM` command removes the specified external resource manager from the current node.

RemoveExternalRM *resource_manager_name*

Argument	Description
resource_manager_name	The name of the external resource manager, as specified in a previous `AddExternalRM` command.

The `RemoveExternalRM` command removes an external resource manager that had previously been registered on the current node by invoking the `AddExternalRM` command. The current node is designated with the `FindNode` command.

The `resource_manager_name` is the name of a resource manager that you have defined as part of the current node. Use the `ShowNode` command to list all external resource managers currently registered for a node.

• RemoveNode

The `RemoveNode` command removes the specified node from the environment definition.

RemoveNode *node_name*

Argument	Description
node_name	The name of a node that is registered in the current environment definition.

Use this command to remove a node from the current environment definition, as designated by the most recent `FindEnv` or `FindActEnv` command.

The `node_name` argument is the name of a node that is registered in the environment definition. Nodes are added to an environment definition either when you invoke the `AddNode` command, or through automatic registration when their Node Managers first connect to the active environment.

You cannot remove the node for an active Node Manager. You must first shut down the Node Manager on that node before invoking the `RemoveNode` command.

Rm

The `Rm` command removes a file from the local file system.

Rm *file_name*

Argument	Description
file_name	The name of the file to remove.

The `file_name` is specified relative to the current working directory. This is either the directory in which Escript was started, or the last directory specified in a `Cd` command.

By default, the `file_name` argument is specified in local operating system format. If you have invoked the `UsePortable` command, however, the `file_name` must be specified in portable file format.

SaveAgent

The `SaveAgent` command saves the current agent under a tag name.

SaveAgent *tag_name*

Argument	Description
tag_name	An arbitrary string to save for later use in `FindSavedAgent`.

The `SaveAgent` command associates the current agent with a tag for later use with the `FindSavedAgent` command. Because you often navigate between agents that are distant from each other in the agent hierarchy, it is convenient to be able to tag one agent, navigate to another agent, and then quickly return to the first tagged agent. You can use `SaveAgent`, `ListSavedAgents` and `FindSavedAgent` commands to navigate quickly.

The tag name is an arbitrary string and is case insensitive. Tag names are available for a single Escript session only; they are not persistently stored.

The `SaveAgent` command is not available if you are in the environment editing mode of Escript.

Script

The `Script` command captures Escript commands as they are input by the interactive user, and writes them to the specified file.

Script *file_name*

Argument	Description
file_name	The name of an operating system file.

The `Script` command directs Escript to write to a specified file all Escript commands subsequently entered. Using this command, you can save a session of Escript commands for later editing and replaying using the `Include` command.

By default, the `file_name` argument is specified in local operating system format. If you have invoked the `UsePortable` command, however, the `file_name` must be specified in portable file format.

• SetEnvForSim

The `SetEnvForSim` command specifies the name of the environment that will be simulated using the active environment.

SetEnvForSim [*environment_name*]

Argument	Description
environment_name	The name of the environment that is being simulated by the active environment.

When you create or import a simulated environment definition in an environment repository, you can designate that the active environment simulate the simulated environment at a given time. Developers running in the Partition Workshop can partition applications for simulated environments and then test the application as if it were running in the other environment.

You invoke the `SetEnvForSim` command to designate the environment definition that will be simulated by the active environment. The `environment_name` argument given is never the name of the active environment. If you do not specify an `environment_name` argument, you cannot simulate an environment.

The following example shows how you could use the `FindEnv` and `SetEnvForSim` commands to specify that the active environment called TestEnv simulate a production environment called RealTimeEnv:

```
escript> FindEnv TestEnv
escript> SetEnvForSim RealTimeEnv
```

After you designate an environment definition to be simulated, you must also specify the mapping of nodes in the simulated environment definition to the current active environment. You use the `SetNodeForSim` command to specify this mapping. After you set up the simulation map, developers can use the partitioning workshop for the simulated environment as though it were the current active environment.

• SetEnvPrefNode

The `SetEnvPrefNode` command sets the preferred node for servers for the current environment definition.

SetEnvPrefNode [*node_name*]

Argument	Description
node_name	The name of a server node to be used as the preferred node for servers when partitioning.

When applications are partitioned within an environment, any partitions that have constraining resource managers, communications protocols, or 3GL libraries are placed automatically on the conforming nodes. However, some partitions may need to be placed on a server node, but do not have constraints which force the system to choose a particular node. The SetEnvPrefNode command allows you to set a specific node within the current environment definition as the default node to use for such unconstrained servers. Use the FindEnv or EditEnv command to designate a simulated or the active environment.

The node_name argument specifies a defined node within the environment definition to use as the preferred node. The node must be a server node, although it can allow client access as well. If no node_name is specified, the current preferred node is removed, and any node can be used by the partitioning system in determining a preferred node.

SetInstrumentLogging

The SetInstrumentLogging command sets the logging status of an instrument in the current agent.

SetInstrumentLogging *instrument_name is_logged*

Argument	Description
instrument_name	The name of an instrument on the current agent.
is_logged	Set to TRUE to turn on logging on the instrument, and FALSE to turn it off.

The system management features of iPlanet UDS provide a mechanism for automatic logging of instruments at defined intervals. See the LogTimer instrument for the Active Partition agent for a detailed description of how to log the values of instruments at regular intervals.

The SetInstrumentLogging command toggles on or off the participation of individual instruments in this automatic logging. By default, the instrument logging flag for all instruments is set to FALSE, and must be explicitly turned on by using the SetInstrumentLogging command.

To use the SetInstrumentLogging command, you first navigate to the agent which defines the instrument of interest, using the FindSubAgent and ShowAgent commands. Then invoke the SetInstrumentLogging command on the appropriate instrument, using the name as shown in the ShowAgent command. iPlanet UDS logs the instrument values to the log file of the active partition in which the agent of the instrument resides.

The `instrument_name` is the name of the instrument, as shown in the `ShowAgent` command. If it contains embedded spaces, it must be enclosed in quotes.

The `is_logged` argument is set to `TRUE` or `FALSE` to turn on or turn off the instrument logging, respectively.

The `SetInstrumentLogging` command cannot be used while in the environment editing mode of Escript, or if the current agent has no instruments.

• SetNodeClient

The `SetNodeClient` command sets the current node's client property.

SetNodeClient *client_flag*

Argument	**Description**
client_flag	A flag to indicate whether the node should support client partitions. Value of 0 indicates no support of client partitions; a value of 1 indicates support of client partitions.

Most of the server node types can act as clients as well as servers, allowing the execution of the client portion of the application as a Motif or Windows NT client. The client property allows you to control whether a server node is intended to support client partitions or not. The `SetNodeClient` command sets the client property for the current node, as designated by the `FindNode` command.

Set the value to `0` if the node is not to be used as a client, or to `1` if the node can be used as a client. If the client property for a node is set to `0`, then you cannot run or test client portions of the application on that node, even if you are running the `forte` command on that node. The default partitioning system will not assign client partitions to nodes that do not provide client access; you can manually assign them in the Partition Workshop.

• SetNodeForSim

The `SetNodeForSim` command sets the node to use in simulating the current node.

SetNodeForSim *node_name*

Argument	Description
node_name	The name of a node within the active environment to use to simulate this node.

In the development system developers can test an application using a simulated environment definition. To set up this simulated environment for testing, you need to specify another environment (almost always the active environment) that will act as the simulated environment. You use the `SetEnvForSim` command to designate the environment that stands in for the simulated environment.

After you set up an environment definition to be simulated, you must also specify the mapping of nodes in the simulated environment definition to the active environment. You use the `SetNodeForSim` command to specify this mapping. The `SetNodeForSim` command operates on the current node of a simulated environment definition, and specifies the name of the node in the active environment to be used to simulate that node. Use the `FindEnv` command, followed by the `FindNode` command to designate the current node for this command. After you set up the simulation map, developers can use the partitioning workshop for the simulated environment definition as though it were the active environment.

The `node_name` argument must be a node within the active environment, which is used to simulate this environment definition. The specified node must have the same architecture as the current node.

• SetNodeModel

The `SetNodeModel` command sets the current node's model property.

SetNodeModel *model_flag*

Argument	Description
model_flag	Specifies whether the current node is a model node. Values are 1 if the current node is a model node, or 0 otherwise.

Many sites have many client machines that are very similar, in terms of architecture and communications capabilities, and it can be tedious to define each of these client nodes within the environment. In cases such as this, a single node can be defined in the environment and designated as a model node. Any number of specific machines can then declare themselves as being the same as the model node, and do not need specific node definitions within the environment.

The `SetNodeModel` command sets the model node property for the current node, as designated by the `FindNode` command.

Set the `model_flag` argument to `0` if the node is not a model node, or to 1 if the node is a model node.

After you designate a node in the environment as a model node, any other machines with the same architecture can share the definition of that node by setting the FORTE_MODELNODE environment variable to the name of the model node. On Windows this can be done through the iPlanet UDS control panel.

On PC, a node can define both the FORTE_MODELNODE value and the FORTE_NODENAME value. iPlanet UDS uses the FORTE_MODELNODE value to obtain the definition of the node architecture and capabilities from the environment. The FORTE_NODENAME value is displayed in Escript and the Environment Console for ease of identifying which of several client nodes is actually executing at any given time.

• SetPassword

The `SetPassword` command replaces the active environment's password, needed to start Escript or the Environment Console, with the new password.

SetPassword *old_password new_password*

Argument	Description
old_password	The previous password value. If no password was set, use an empty quoted string ("").
new_password	The new password value. If you want to remove the password, use an empty quoted string ("").

Because both Escript and the Environment Console give users a great deal of control over an executing environment, you can set a password to restrict access to privileged users. The `SetPassword` command allows you to set a single password that afterward must be given to start the Environment Console or start executing Escript commands. By default, environment repositories are created without passwords.

The `old_password` argument specifies the current password, which must be provided in order to change the password. When setting the password for the first time, specify this as an empty string (using double quotes without any characters in between).

The `new_password` argument specifies the new password to use in subsequent sessions. There are no restrictions on the name, but you should follow good password practices in setting the password. The password is case sensitive. To turn off the password, set the new password to an empty string (using double quotes).

SetPath

The `SetPath` command sets the directory search path used by any of the commands that take a file name as an argument.

SetPath *directory_name* [;*directory_name*...]

Argument	Description
directory_name	The name of a directory in which to look for files that are specified without a path.

Most of the commands that have input file arguments, such as the `Include` command, allow you to specify the name of the input file without a full directory specification. In that case, they use the current directory search path, as defined by the `SetPath` and `AddPath` commands, to find the file. The directory search path provides a set of directories which are checked in turn, until a file matching the name is found.

The `SetPath` command resets the entire directory search path. You can use the `AddPath` command to add more directories to a directory search path. By default, the directory search path only includes the current working directory. The current working directory is always considered the last directory in the directory search path, even after the `SetPath` command is invoked. If you want it consulted first, you can specify the current directory first in the list of directories for the `SetPath` command.

Each `directory_name` is specified as a full directory path name. By default, directories should be specified in the local operating system directory format. If you have previously invoked the `UsePortable` command, then the directory name should be specified in iPlanet UDS portable format (UNIX style directory format). To specify more than one directory, separate the directory names with semi-colons.

You can embed environment variable names within the directory names:

${*environment_variable_name*}

The dollar sign and brackets indicate that the name inside the brackets is an environment variable, and the entire specification is replaced with the current setting of the environment variable.

You can also use the following syntax to expand the environment variable name, but convert it to a portable file format as well:

%{*environment_variable_name*}

The percent sign and brackets indicate that the name inside the brackets is an environment variable, and the entire specification is replaced with the current setting of the environment variable. This syntax lets you convert environment variables to portable format if you have invoked the `UsePortable` command, but have directories specified in environment variables in local format.

```
escript> SetPath /mydisk/mydir;${ENV_VAR}/subdir
escript> SetPath c:\mydir;${ENV_VAR}\subdir
escript> SetPath "Mac HD:Apps:TempFolder";${ENV_VAR}:Sub
escript> SetPath $dka0:[path];${ENV_VAR}:[otherdisk.otherdir]
escript> UsePortable
escript> SetPath %{FORTE_ROOT}/install/examples
```

• SetSimForNode

The `SetSimForNode` command sets the current node's simulation property.

SetSimForNode *simulation_flag*

Argument	Description
simulation_flag	A flag indicating whether the current node can simulate a node in a simulated environment definition. Values are '0' if the node is not to be used for simulation, and '1' if the node can be used.

This command defines whether current node can be used when simulating other environment definitions. The current node is selected with the `FindNode` command.

Set the `simulation_flag` argument to `0` to prevent this node from simulating another node, or to `1` to allow this node to simulate other nodes.

By default, a simulated environment definition maps each of its simulated nodes to available nodes in the active environment when developers are in the partitioning workshop. When you define the simulated environment definition, the `SetNodeForSim` command defines the mapping of simulated nodes to active nodes. In the active environment, however, you may want to put certain production nodes off limits when you simulate another environment. Use the `SetSimForNode` command to restrict this usage.

ShowAgent

The `ShowAgent` command lists information about the current agent.

ShowAgent

Use this command to see the name of the agent, its status, the type of object that the agent manages, the parent agent and its type, the instruments of the current agent and their current values, and the list of subagents to this agent and their types.

You cannot invoke the `ShowAgent` command while you are in the environment editing mode of Escript.

• ShowEnv

The `ShowEnv` command shows details of the current environment definition.

ShowEnv

When you invoke the `EditEnv` command, the active environment's definition becomes the current environment definition. You can also use the `FindEnv` command to designate one of the simulated environments as the current environment definition for Escript. In either case, you can invoke the `ShowEnv` command to display information about the environment.

The `ShowEnv` command lists both the basic environment properties, as well as a list of loaded and/or installed applications in the environment, a list of nodes in the environment, and a list of available external resource managers in the environment.

ShowInstrument

The `ShowInstrument` command shows the value of an instrument.

ShowInstrument *instrument_name*

Argument	Description
instrument_name	The name of an instrument in the current agent.

The `ShowInstrument` command shows the value of an instrument in the current agent. See the various sections on specific agents in Chapter 4, "iPlanet UDS System Agent Commands and Instruments" for a description of the instruments that are available for each agent.

The `instrument_name` argument specifies an instrument that is defined for the current agent. The name is case-insensitive. To see a list of the instruments defined on an agent invoke the `ShowAgent` command.

If the instrument is a compound instrument, you can fully qualify the instrument name to show one of the component instruments by using a name in the format "`compound_name.instrument_name`".

The `ShowInstrument` command cannot be used if you are in the environment editing mode of Escript.

• ShowNode

The `ShowNode` command shows details of the current node.

ShowNode

After you designate a current node, by invoking the `FindNode` command, you can display information about the node by invoking the `ShowNode` command.

ShowPath

The `ShowPath` command shows the current search path.

ShowPath

The `ShowPath` command displays the current directory search path for resolving file names in Escript commands that take file names as arguments.

Use the `SetPath` and `AddPath` commands to set up the directory search path.

ShowSubAgent

The `ShowSubAgent` command shows information about one of the subagents to the current agent.

ShowSubAgent *subagent_name*

Argument	Description
subagent_name	The name of one of the subagents to the current agent.

The `ShowSubAgent` command lists properties of one of the subagents to the current agent. It is really just a shorthand way of invoking a `FindSubAgent` followed by a `ShowAgent` command, and provides a way to quickly see information about a subagent without making it the current agent.

The `subagent_name` argument is the full name of a subagent to the current agent. Use the `ShowAgent` command to see the names and types of the subagents to the current agent. You can also use the convention of showing an agent several levels below the current agent by preceding each level by a "`/`" For more information, see the description of `FindSubAgent` in "FindSubAgent" on page 76.

You cannot invoke the `ShowSubAgent` command while you are in environment editing mode.

Step

When an `Include` command is given, the `Step` command allows you to step through the commands interactively before they are executed.

Step

You can use the `Step` command to troubleshoot script files run by the `Include` command. After you invoke the `Step` command, Escript prompts you before invoking each command read from the script file.

UnlockEnv

The `UnlockEnv` command unlocks the exclusive lock on the active environment, and aborts any changes made since the last `LockEnv` command in this session.

UnlockEnv

Many Escript commands require that the active environment be locked by the `LockEnv` command. After you lock the active environment, you can make changes to the active environment. You can commit these changes by invoking the `Commit` command, which also releases the lock on the active environment.

Most of the commands that change the definition of the active environment are made accessible through the use of the `EditEnv` command, which puts Escript in a special environment editing mode. After leaving the environment editing mode (using `Quit` or `Exit`), you can then invoke the `UnlockEnv` command to discard any changes made while in that mode.

If you invoke a `LockEnv` command and make changes that you want to undo, you can invoke the] command to throw away the changes, and then unlock the environment. If there are outstanding changes, Escript prompts you for verification before unlocking the environment.

The `UnlockEnv` command is only available if you are working on a current agent that can affect the definition of the environment: Environment agent, Node agent, Application agent or (logical) Partition agent.

UpdateInstrument

The `UpdateInstrument` command changes the value of an instrument.

UpdateInstrument *instrument_name* [*instrument_data...*]

Argument	Description
instrument_name	The name of an instrument in the current agent.
instrument_data	The new value (or set of values) for the agent. Different instrument types have different types of data.

The `UpdateInstrument` command updates the value of a writable instrument in the current agent. See the various sections on specific agents for a description of the instruments that are available for each agent.

The `instrument_name` is the case-insensitive name of an instrument that is defined for the current agent. You can see a list of the instruments defined on an agent by invoking the `ShowAgent` command. If the instrument is a compound instrument, you can fully qualify the instrument name to update one of the component instruments, by using a name in the format "`compound_name.instrument_name`".

The format of the instrument value(s) is dependent on the type of instrument.

For `ConfigValueInst` instruments, the `instrument_data` value should be valid for whatever type of information is needed. For example, a `ConfigValueInst` instrument that contains a boolean value can be set to either "`TRUE`" or "`FALSE`". An integer value can be set to "`1`" or "`3`", etc. String values are specified with quotes only to include embedded blanks.

For `TimerInst` instruments, the `instrument_data` value contains two parts: a `tick_interval` value (in milliseconds) and an `is_active` flag (set to `TRUE` or `FALSE`). The pair of values must be enclosed in quotes.

The `UpdateInstrument` command cannot be used if you are in the environment editing mode of Escript.

UseLocal

The `UseLocal` command sets up Escript to expect local operating system file name format in commands that represent file names.

UseLocal

Many Escript commands have arguments that require file names. After you invoke the `UseLocal` command, Escript assumes that the file names given in those commands are specified in local operating system format, rather than the iPlanet UDS portable file naming format. The local operating system is determined by the machine where the Escript command was started.

You can use environment variables to specify directory and file names in local file format using the following syntax:

${*environment_variable_name***}**

The dollar sign and brackets indicate that the name inside the brackets is an environment variable, and the entire specification is replaced with the current setting of the environment variable.

When Escript first starts, the default is to use the local operating system file naming format. However, if you invoke a `UsePortable` command in the Escript session, this behavior changes. The `UseLocal` command can be used to revert back to the default naming conventions.

See the `UsePortable` command for details on portable naming conventions.

UsePortable

The `UsePortable` command sets up Escript to expect portable file name format.

UsePortable

Many Escript commands have arguments that require file names. After you invoke the `UsePortable` command, Escript assumes that the file names given in those commands are specified in iPlanet UDS portable file naming format, rather than the local operating system format. This setting gives you a way to build portable scripts of Escript commands.

When Escript first starts, the default is to use the local operating system file naming format for file name arguments. However, if you invoke a `UsePortable` command in the Escript session, this behavior changes. The `UseLocal` command can be used to revert back to the default naming conventions.

iPlanet UDS portable file naming uses the following conventions:

- Directory paths are specified in UNIX format, with slashes to represent the directory hierarchy.
- Directory names are limited to eight characters.
- File names are limited to 8 characters, followed by a maximum of a 3 character extension. Certain extensions provide conventions for file types. See the File class in Framework Library online Help for details.
- The special syntax `%{environment_variable_name}`, can be embedded in a portable file name.

This looks for an environment variable with the given name, and assumes that it represents a directory name in local operating system format. However, it expands it within a portable file name as if it were specified in portable format. This provides a good way to provide local "roots" for directories of files, and then use portable format underneath that root to the tree.

```
escript> UseLocal
escript> Include /mydisk/mydir/myfile.inc
escript> Include c:\mydir\myfile.inc
escript> Include "Mac HD:Apps:TempFolder:myfile.inc"
escript> Include $dska:[mydir]myfile.inc
escript> ListFile ${FORTE_ROOT}/install/examples/tstapps.fsc
escript> ListFile ${FORTE_ROOT}\install\examples\tstapps.fsc
escript> ListFile ${FORTE_ROOT}:install:examples:tstapps.fsc
escript> ListFile ${FORTE_ROOT}:[install.examples]tstapps.fsc
escript> UsePortable
escript> Include /mydisk/mydir/myfile.inc
escript> Include %{FORTE_ROOT}/install/examples/tstapps.fsc
```

WaitForEnvMgr

The `WaitForEnvMgr` command forces Escript scripts to wait for the Environment Manager to start before continuing executing.

WaitForEnvMgr [*number_seconds*]

Argument	Description
number_seconds	The number of seconds to wait before exiting if the Environment Manager does not start, or 0 to wait indefinitely.

If you automate some of your management processes by creating scripts of Escript commands to execute at specific times, you can use the `WaitForEnvMgr` command in those scripts to help ensure that timing problems are minimized between the startup of the Environment Manager partition and the startup of the scripts.

The `number_seconds` argument specifies that if the number of seconds elapses before the Environment Manager starts, then Escript exits. The default for the number of seconds is 0, which means to wait for the Environment Manager to start without timing out.

You cannot invoke the `]` command while you are in the environment editing mode of Escript.

WhichFile

The `WhichFile` command searches through the directories in the current directory search path to locate the first directory in which the specified file exists.

WhichFile *file_name*

Argument	Description
file_name	The simple name of a file to locate.

Use the `WhichFile` command to search each of the directories in the current directory search path to see where a specific file is located. The current directory search path is defined using the `SetPath` and `AddPath` commands. You can use the `ShowPath` command to display the current directory search path.

The `file_name` argument is the name of a file, given without a directory path. Each directory in the current directory search path is checked in turn to see if it contains the named file. When a match is found, the directory name that contained the file is displayed.

Part 2

iPlanet UDS System Agents

Part 2 of *Escript and System Agent Reference Guide* provides usage and reference information for iPlanet UDS system agents.

This section contains the following chapters:

Chapter 3

Using iPlanet UDS System Agents

This chapter explains how to interact with iPlanet UDS system agents using the Environment Console and Escript.

For descriptions of agents and their commands and instruments, see Chapter 4, "iPlanet UDS System Agent Commands and Instruments."

For information about writing TOOL applications that work with iPlanet UDS system agents, see *Programming with System Agents*.

About Using iPlanet UDS System Agents

iPlanet UDS lets you manage and monitor the iPlanet UDS runtime system and your applications using *system agents*. System agents are objects in the runtime system that have commands and instruments that you can use to monitor and adjust how elements of the system are operating. The iPlanet UDS system agents and their commands and instruments are described in Chapter 4, "iPlanet UDS System Agent Commands and Instruments."

You can also define system agents of your own to monitor and manage your applications, as described in *Programming with System Agents*.

How iPlanet UDS System Agents Work

iPlanet UDS system agents are associated with particular parts of applications and the iPlanet UDS runtime system, such as active partitions, the operating system, and load-balancing routers.

Invoking Commands and Accessing Instruments

These system agents have particular commands and instruments associated with them. In general, these commands and instruments operate directly upon the component associated with the agent. Because commands and instruments are associated with a particular agent, you need to find the correct agent and invoke the command or access the instrument on that agent.

For example, if you invoke the Shutdown command on the Environment agent, you will shut down the entire environment, whereas if you invoke the Shutdown command on an Active Partition agent, you will shut down an active partition for a running application. Similarly, if you want to get information about how many repository sessions are open on a central repository, you can check an instrument on the RepositoryServer agent.

In general, you cannot invoke a command on any random agent and have them work. There are some exceptions, such as the `DumpStatus` command, which is defined for all system agents. There are also Escript commands that are not associated with any particular system agent. These commands are documented in Chapter 2, "General Escript Commands."

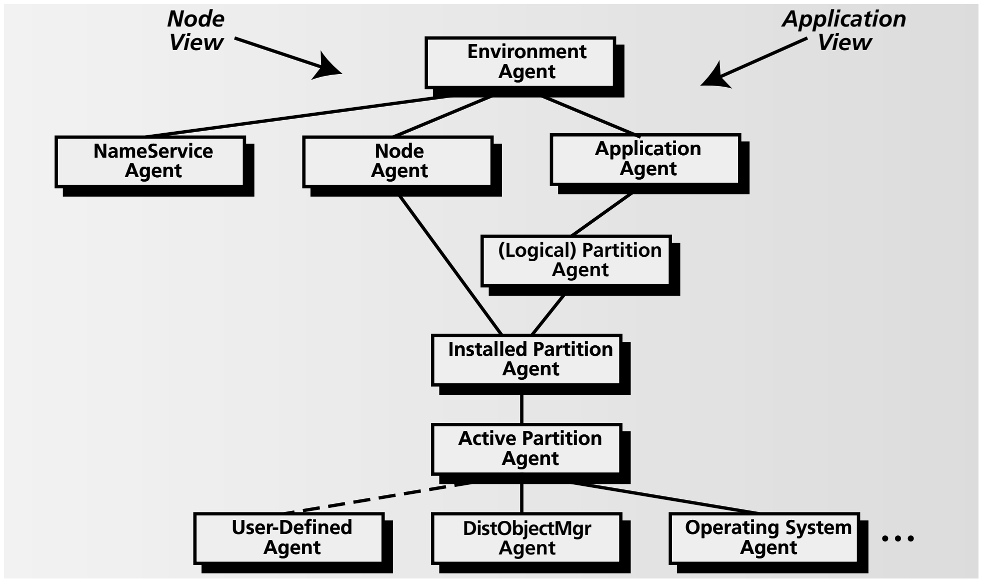
In most cases, however, you need to invoke a system agent command on a particular system agent. Otherwise, the command either won't work, or the results will not be what you intended.

Locating System Agents

The iPlanet UDS system agents are organized into a hierarchy, called the *agent hierarchy*. This hierarchy is a containment hierarchy; that is, the items at the top of the hierarchy contain the items beneath them. You can also think of the items lower in the hierarchy as being components of the items above them.

The following figure shows the general structure of the agent hierarchy:

Figure 3-1 Structure of the Agent Hierarchy



For example, the environment contains one or more nodes, and one or more applications, so the Environment agent is the parent agent (higher in the hierarchy) of one or more Node agents and one or more Applications. The Node and Application agents are subagents of the Environment agents because the nodes and applications they represent are components of the environment.

The diagram in Figure 3-1 indicates a Node View and an Application View. These arrows indicate how the hierarchy splits between the Environment agent and the Installed Partition agent.

Node View If you consider the environment as a set of nodes, then you are thinking in terms of the Node View. In this case, you can navigate down the agent hierarchy from the Environment agent to the Node agent, then to Installed Partition agents for application partitions that are installed on the node.

Application View If you think about the environment in terms of applications, then you are thinking in terms of the Application View. In this case, you can navigate down the agent hierarchy from the Environment agent to the Application agent, then to the Partition agent, and then to the Installed Partition agents for application partitions that are installed on various nodes.

➤ **To locate a specific agent**

1. Start at the Environment agent for the active environment. This is usually the first agent you have access to.
2. Navigate down through the necessary agents to the agent you want. Each application has a specific way of navigating to a particular agent.

iPlanet UDS provides two main applications for managing your system: the Environment Console and Escript, which are described in "Using Agents in the Environment Console" on page 110 and "Using Agents in Escript" on page 114.

To locate an agent in the agent hierarchy, check the information for that iPlanet UDS system agent in Chapter 4, "iPlanet UDS System Agent Commands and Instruments." The description of each agent includes its parent agents and subagents, if any.

Determining Which Commands and Agents to Use

This manual contains tables that list the most common system management tasks. These tables list the associated Escript and system agent commands you need to perform the tasks. These tables are in "Working with the Escript Utility" on page 31.

If you know that name of a command you can use in Escript, but aren't sure what agent, if any, the command belongs to, you can check Appendix A, "All Escript and System Agent Commands."

Using Agents in the Environment Console

The Environment Console provides a graphical user interface that lets you interact with system agents to manage your environment, nodes, applications, and so forth.

The Environment Console provides an Application View, which is useful when you want to consider your iPlanet UDS runtime system as a set of applications. Alternatively, you can use the Node View to work with your environment as a set of defined nodes. To change the setting, choose View > Application View or Node View.

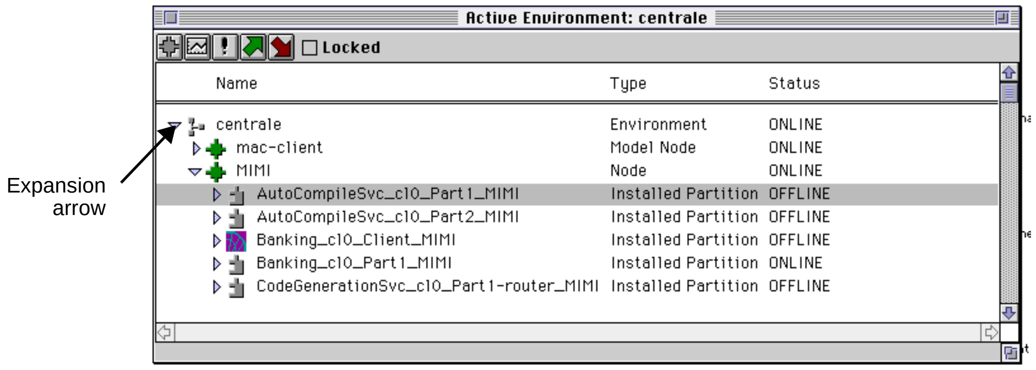
For more general information about using the Environment Console, and for information about performing specific tasks using the Environment Console, see *iPlanet UDS System Management Guide*.

Identifying Parent Agents and Subagents

In the Environment Console, each agent is represented as a component in the containment hierarchy. An agent is the *parent agent* of another agent if you can click the expansion arrow or open the agent and see that the other agent is shown as within the first agent. The other agent is considered the *subagent* of the parent agent.

For example, in Figure 3-2, the Environment agent named Centrale is the parent agent for the Node subagent named Mimi, and the Node agent is the parent of the Installed Partition subagent namedAutoCompileSvc_cl0_Part1_MIMI.

Figure 3-2 Parent Agents and Subagents



```
Active Environment: centrale
Locked

Name                                    Type                 Status
centrale                                Environment          ONLINE
  mac-client                            Model Node           ONLINE
  MIMI                                  Node                 ONLINE
    AutoCompileSvc_cl0_Part1_MIMI       Installed Partition  OFFLINE
    AutoCompileSvc_cl0_Part2_MIMI       Installed Partition  OFFLINE
    Banking_cl0_Client_MIMI             Installed Partition  OFFLINE
    Banking_cl0_Part1_MIMI              Installed Partition  ONLINE
    CodeGenerationSvc_cl0_Part1-router_MIMI Installed Partition  OFFLINE
```

Navigating around the Agent Hierarchy

The main Environment Console window is the Active Environment window, which represents the active environment. To navigate around the agent hierarchy, you always start with this agent and this window.

To locate an agent in the agent hierarchy, check the information for each iPlanet UDS system agent in Chapter 4, “iPlanet UDS System Agent Commands and Instruments,” which includes its parent agents and subagents, if any.

You can view the contents of the agent hierarchy two ways:

- Expand the browser list to see the subagents by clicking the expansion arrows for the parent agents of the subagents you are trying to locate, as shown in Figure 3-2

- Open a window for a particular agent by double-clicking on the agent. You can then expand the browser list by clicking the expansion arrow for the agent to see its subagents.

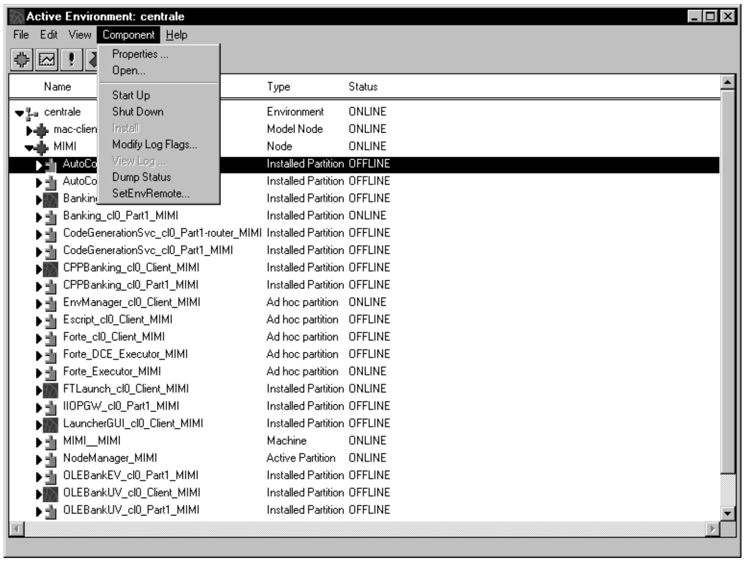
  If an agent has no subagents, then double-clicking the agent opens a window that displays the agent's instruments.

Accessing the Commands for an Agent

The Environment Console provides the commands for each agent in the menu bar when an agent is selected in the window. Most agent commands are included in the bottom section of the Component menu, although some agents also define menus such as Utility and Installation, and place commands in those menus as well.

Figure 3-3 shows the commands that are defined for an Installed Partition agent.

Figure 3-3 Installed Partition Commands



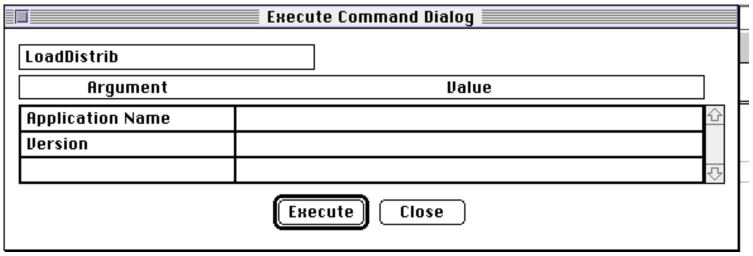
Name	Type	Status
centrale	Environment	ONLINE
mac-clien	Model Node	ONLINE
MIMI	Node	ONLINE
AutoCo	Installed Partition	OFFLINE
AutoCo	Installed Partition	OFFLINE
Bankin	Installed Partition	OFFLINE
Banking_cl0_Part1_MIMI	Installed Partition	ONLINE
CodeGenerationSvc_cl0_Part1-router_MIMI	Installed Partition	OFFLINE
CodeGenerationSvc_cl0_Part1_MIMI	Installed Partition	OFFLINE
CPPBanking_cl0_Client_MIMI	Installed Partition	OFFLINE
CPPBanking_cl0_Part1_MIMI	Installed Partition	OFFLINE
EnvManager_cl0_Client_MIMI	Ad hoc partition	ONLINE
Escript_cl0_Client_MIMI	Ad hoc partition	OFFLINE
Forte_cl0_Client_MIMI	Ad hoc partition	OFFLINE
Forte_DCE_Executor_MIMI	Ad hoc partition	OFFLINE
Forte_Executor_MIMI	Ad hoc partition	ONLINE
FTLaunch_cl0_Client_MIMI	Installed Partition	ONLINE
IIOPGW_cl0_Part1_MIMI	Installed Partition	OFFLINE
LauncherGUI_cl0_Client_MIMI	Installed Partition	OFFLINE
MIMI__MIMI	Machine	ONLINE
NodeManager_MIMI	Active Partition	ONLINE
OLEBankEV_cl0_Part1_MIMI	Installed Partition	OFFLINE
OLEBankUV_cl0_Client_MIMI	Installed Partition	OFFLINE
OLEBankUV_cl0_Part1_MIMI	Installed Partition	OFFLINE

To determine what command you are trying to find and what agent defines the command, see the tables that map system management tasks to Escript and system agent commands, starting in "Working with the Escript Utility" on page 31.

➤ **To invoke a command**

1. Locate the appropriate agent, as described in "Navigating around the Agent Hierarchy" on page 111.
2. Select the agent.
3. Click the command in the menu (usually the Component or Utility menu).
4. If the command has arguments, an Execute Command Dialog appears. Enter the argument values, then click Execute to invoke the command. Figure 3-4 shows one of these dialogs.

Figure 3-4 Execute Command Dialog



LoadDistrib

Argument	Value
Application Name	
Version	

Accessing the Instruments for an Agent

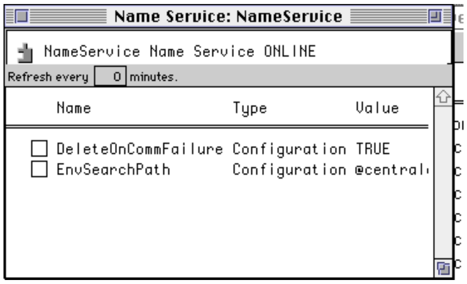
➤ **To access the instruments for an agent**

1. Open a window for a particular agent by double-clicking on the agent.

   If an agent has no subagents, then double-clicking the agent opens a window that displays the agent's instruments, and you do not need to perform the next step.

2. Click the File > Instruments command in the window. This command opens a window that displays the instruments defined by this agent, as shown in Figure 3-5:

Figure 3-5 Instruments of the NameService Agent


Name Service: NameService

NameService Name Service ONLINE

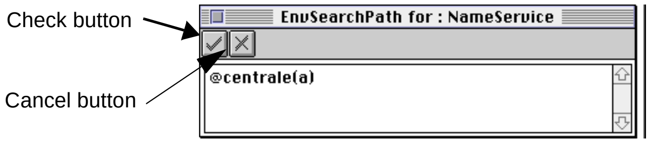
Refresh every 0 minutes.

```
Name                Type          Value
DeleteOnCommFailure Configuration TRUE
EnvSearchPath       Configuration @central
```

➤ To change the value for a settable instrument

1. Make sure that the instrument value is not read only by checking the information for the instrument in Chapter 4, "iPlanet UDS System Agent Commands and Instruments."
2. Double-click on the instrument to open a dialog for the instrument.
3. Change the value of the instrument, then click the check button to save the value, and the cancel button to cancel the change, as shown in Figure 3-6.

Figure 3-6 Changing an Instrument Value



EnvSearchPath for : NameService

```
@centrale(a)
```

You can also log instrument data, which is described in *iPlanet UDS System Management Guide*.

Using Agents in Escript

Escript is a console application that lets you enter commands to manage and access instruments to monitor your iPlanet UDS runtime system and applications.

For more general information about using Escript, and for information about performing specific tasks using Escript, see Chapter 1, "Using the Escript Utility."

Identifying Parent Agents and Subagents

An agent is the *parent* agent of another agent if the other agent is included in the list of subagents for the agent when you invoke the Escript `ShowAgent` command. The other agent is considered the *subagent* of the first agent.

For example, in the following sample Escript output, the Active Partition agent named Banking_cl0_Part1_0x644:0x1 is the parent agent for the NativeLangMgr subagent named NativeLangMgr, and several other subagents as well.

```
escript > showag
Current Agent:
    Type:  Active Partition
    Name:  Banking_cl0_Part1_0x644:0x1
    Status:  ONLINE
    Parent Agent:  Installed Partition Agent -
Banking_cl0_Part1_MIMI  (ONLINE)
    Instruments:
        CanBeActivated       :  FALSE
        LogTimer            :  inactive, 0, 300000
        InstrumentLogging    :  TRUE
        ProcessId            :  247
    Sub Agents:
        NativeLangMgr Agent - NativeLangMgr  (ONLINE)
        EventMgr Agent - EventMgr  (ONLINE)
        TransactionMgr Agent - TransactionMgr  (ONLINE)
        CommMgr Agent - CommMgr  (ONLINE)
        DistObjectMgr Agent - DistObjectMgr  (ONLINE)
        TaskMgr Agent - TaskMgr  (ONLINE)
        OperatingSystem Agent - OperatingSystem  (ONLINE)
        Process Agent - Process  (ONLINE)
escript >
```

Navigating around the Agent Hierarchy

In Escript, the *current agent* is the agent upon which you invoke commands.

When you first start Escript, the current agent is the Environment agent that represents the active environment.

To locate an agent in the agent hierarchy, check the information for each iPlanet UDS system agent in Chapter 4, "iPlanet UDS System Agent Commands and Instruments," which includes its parent agents and subagents, if any.

➤ To navigate down the agent hierarchy

1. Enter the `ShowAgent` command to display the list of subagents for the current agent.

   For a complete description of the `ShowAgent` command, see "ShowAgent" on page 96.

2. Enter the `FindSubAgent` command, as shown in the following example, to navigate down to one of the subagents:

   ```
   escript> FindSubAgent DistObjectMgr
   ```

   For a complete description of the `FindSubAgent` command, see "FindSubAgent" on page 76.

3. If you want to navigate to an agent further down within the hierarchy, repeat the `ShowAgent` and `FindSubAgent` commands until the agent you want is the current agent.

➤ To navigate up the agent hierarchy

1. Enter the `ShowAgent` command to display the parent agent for the current agent.

   For a complete description of the `ShowAgent` command, see "ShowAgent" on page 96.

2. Enter the `FindParentAgent` command, as shown in the following example, to navigate down to one of the subagents:

   ```
   escript> FindParentAgent
   ```

   For a complete description of the `FindParentAgent` command, see "FindParentAgent" on page 74.

3. If you want to navigate to an agent further up within the hierarchy, repeat the `ShowAgent` and `FindParentAgent` commands until the agent you want is the current agent.

Accessing the Commands for an Agent

Escript includes agent commands for the current agent in its online help. If a command is not displayed in the online help when a certain agent is the current agent, then the command is not available for that agent.

To determine what command you are trying to find and what agent defines the command, see the tables that map the typical system management tasks with Escript and system agent commands, starting in "Working with the Escript Utility" on page 31.

➤ **To invoke a command**

1. Make the appropriate agent the current agent, as described in "Navigating around the Agent Hierarchy" on page 115.
2. Enter the command with the appropriate arguments, as shown in the following example. In this example, an Active Partition agent is the current agent:

   ```
   Shutdown
   ```

The commands and instruments for iPlanet UDS system agents are described in Chapter 4, "iPlanet UDS System Agent Commands and Instruments."

Accessing the Instruments for an Agent

➤ **To access the instruments for an agent**

1. Make the appropriate agent the current agent, as described in "Navigating around the Agent Hierarchy" on page 115.
2. Enter the `ShowAgent` command to display the instruments for the current agent and the instruments' values.

   For a complete description of the `ShowAgent` command, see "ShowAgent" on page 96.
3. To show only the value for a single instrument, enter the ShowInstrument command, as in the following example for an instrument on the Node agent. The Node agent is the current agent.

   ```
   escript> ShowInstrument Architecture
   Architecture         :  PC NT
   ```

   The `ShowInstrument` command is described in "ShowAgent" on page 96.

➤ To change the value for a settable instrument

1. Make sure that the instrument value is not read only by checking the information for the instrument in Chapter 4, "iPlanet UDS System Agent Commands and Instruments."

2. Make the appropriate agent the current agent, as described in "Navigating around the Agent Hierarchy" on page 115.

3. Enter the `ShowAgent` command to display the instruments for the current agent and the instruments' values, as shown in the following example:

```
escript > showag
Current Agent:
    Type:  Name Service
    Name:  NameService
    Status:  ONLINE
    Parent Agent:  Environment Agent - centrale  (ONLINE)
    Instruments:
        EnvSearchPath         :  @centrale(a)
        DeleteOnCommFailure   :  TRUE
    Sub Agents:  NONE
escript >
```

   For a complete description of the `ShowAgent` command, see "ShowAgent" on page 96.

4. Enter the `UpdateInstrument` Escript command, with the instrument name and appropriate arguments, as shown in the following example for an instrument in the Name Service. A Name Service agent is the current agent.

```
UpdateInstrument EnvSearchPath @(a):@centrale(a)
```

   The `UpdateInstrument` command is described in "UpdateInstrument" on page 99.

You can also log instrument data, which is described in "Logging Information" on page 49 and *iPlanet UDS System Management Guide*.

Chapter 4

iPlanet UDS System Agent Commands and Instruments

This section is a reference describing all agent commands and instruments accessible through either the Environment Console or the `Escript` utility.

The reference is presented in alphabetical order, grouped by agent, and includes a description of each command and instrument.

About Agent Commands and Instruments

This chapter describes the iPlanet UDS system agents and their commands and instruments in general terms. This chapter describes what the agents represent and the syntax and usage of the commands and instruments associated with the agents. This chapter does not describe the specific usage of the commands or instruments in different applications, such as the Environment Console or Escript.

You can use these agents, commands, and instruments through the Environment Console, Escript, and programmatically through the SystemMonitor library.

For information about using these commands and instruments in the Environment Console, see "Using Agents in the Environment Console" on page 110.

For information about using these commands and instruments in Escript, see "Using Agents in Escript" on page 114.

For information about writing TOOL code that uses these commands and instruments, see *Programming with System Agents*.

Summary of iPlanet UDS System Agents

This chapter covers the commands and instruments of the following agents:

Agent	Manages:	See:
Active Partition	Running partition of an application.	page 121
Ad hoc partition	Acts as the installed partition agent for a running partition that has not been installed using the iPlanet UDS installation process.	page 133
Application	Application that has been loaded into the environment.	page 139
BtreeCache	Cache used with the B-tree central repository to improve performance.	page 151
BtreeRepository	B-tree central repository.	page 158
CommMgr	Communications into and out of an active partition.	page 162
DBSession	Database session for an active partition that accesses a database using a DBSession object.	page 174
DistObjectMgr	Distributed object services for an active partition.	page 191
Environment	iPlanet UDS environment (represented by the Environment Manager).	page 194
EventMgr	Services for receiving and delivering events.	page 208
Installed Partition	Partition of an application that has been installed on a particular machine.	page 210
LoadBalancing Router	Router for a load-balanced service object.	page 217
Machine	Physical machine on which a Node Manager is running.	page 221
Model Node	Definition for a model node.	page 225
NameService	iPlanet UDS Name Service, which manages how objects, services, partitions, and so forth are known within and across environments.	page 227
NativeLangMgr	Services for multinational and multilingual functions.	page 245
Node	Node in the environment.	page 248
ObjectCache	Object cache used by a client repository session.	page 261

Agent	Manages:	See:
OperatingSystem	Operating system services for an active partition, including memory management and other utility functions.	page 265
Partition	Logical partition of an application. This partition is not assigned to any particular node.	page 282
Process	Operating system process on which an active partition runs.	page 295
Repository	Repository managed by the repository server or another iPlanet UDS application.	page 301
RepositoryServer	Running repository server, which manages a central repository.	page 304
RepositoryServerInfo	Provides information about a repository server running in the current environment.	page 317
RepositorySession	Client repository session running on a node that is running a Node Manager.	page 320
TaskMgr	Task management services for an active partition.	page 323
TransactionMgr	Transaction management services, which monitors the state of transactions across partitions.	page 336
Volume	Store device on a particular machine.	page 348

Active Partition Agent

Parent Agent

Installed Partition agent or Ad hoc partition agent

Subagents

All iPlanet UDS runtime system agents, such as DistObjectMgr agent, TaskMgr agent, and TransactionMgr agent.

SystemMonitor Class

ActivePartitionAgent

States

State	Description
BUSY	(Only subagents of Forte_Executor_*nodename*) The iPlanet UDS executor (ftexec) is running a standard iPlanet UDS application partition.
ONLINE	The partition is running.
PENDING	(Only subagents of Forte_Executor_*nodename*) The iPlanet UDS executor (ftexec) has been reserved for a particular standard application partition and is waiting to load the partition.
RUNDOWN	The partition is shutting down.

Command Summary

Command	Arguments	Environment Console menu	Description
DebugPartition	none	Special	Places this partition under the control of a C++ debugger.
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
FlushLogFiles	none	Component	Flushes all of this partition's log files.
ModLoggerRemote	+(*logger_flags*) –(*logger_flags*)	Component > Modify Log Flags	Sets the logger flags for the active partition. If you are invoking this command within TOOL code, use the `ModLogger` command.
SetEnvRemote	*env_variable* *new_value*	Component	Sets an environment variable for the active partition. If you are invoking this command within TOOL code, use the `SetEnvVar` command.

Command	Arguments	Environment Console menu	Description
Shutdown	kill_executors	Component	Shuts down the active partition If the `kill_executors` argument is set to `TRUE` or `1`, any interpreted partitions will shut down their hosting iPlanet UDS executor (ftexec or ftexecd) process as well.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
BuildDate	none	Yes		The date/time at which the distribution containing this partition was made.
CanBeActivated	none	Yes	Configuration	Indicates whether the partition is enabled for startup by the management system.
ExecutingPartition	none	Yes	Configuration	Indicates the partition being executed by this iPlanet UDS executor partition.
InstrumentLogging	*is_active*	No	Configuration	Turns on/off automatic logging of instruments to active partition log file.
IsCompiled	none	Yes	Boolean	`TRUE` if the partition is compiled.
LogFile	*log_file_name*	No	Configuration	The name of the file to use when logging instruments for the active partition.
LogTimer	*is_active* *interval_in_msec*	No	Timer	Turns on/off and sets interval, in milliseconds, for instrument logging events within the active partition.
ProcessID	none	Yes	Configuration	Contains operating system ID for process running the active partition.

Using the Active Partition Agent

The Active Partition agent represents executing partitions within an iPlanet UDS system. These agents can represent compiled partitions or interpreted—or standard—partitions. An Active Partition agent can also represent a running instance of the iPlanet UDS executor, which interprets standard partitions. The Active Partition agent controls how most of the monitoring data is logged for applications.

The Active Partition agent uses different instruments, depending on which type of active partition it represents, as shown in this table:

Type of Active Partition	Instruments
Compiled client partition	CanBeActivated InstrumentLogging LogTimer ProcessId
Compiled server partition	CanBeActivated InstrumentLogging LogFile LogTimer ProcessId
Standard client or server partition	CanBeActivated InstrumentLogging LogTimer ProcessId
iPlanet UDS executor server partition	CanBeActivated ExecutingPartition InstrumentLogging LogFile LogTimer ProcessId

Standard partitions and iPlanet UDS executor partitions When you start a standard partition, the iPlanet UDS runtime system looks for a running iPlanet UDS executor partition with sufficient memory to run the standard partition. If such as partition exists, then the runtime system has that partition load the and interpret the standard partition. Otherwise, the runtime system automatically starts a new iPlanet UDS executor partition, which loads and runs the standard partition.

Therefore, to monitor a standard server partition, you need to check the Active Partition agents for both the application partition and the iPlanet UDS executor partition that is running the partition.

Agents for the iPlanet UDS executor partitions for standard clients do not appear as part of the agent hierarchy, so you can only monitor the Active Partition agent for the standard partition.

➤ To locate a particular iPlanet UDS executor server partition in the Environment Console

1. Locate the Forte_executor_*nodename* agent, which is a subagent of the Node agent for the node where the standard partition is running.
2. Click the expansion arrow next to the Forte_executor_*nodename* agent, then click the Active Partition agent whose name matches the name of the standard partition.

➤ To locate a particular iPlanet UDS executor server partition in Escript

1. Navigate to the Active Partition agent that represents the running standard partition. Note the first or only hexadecimal value at the end of the name of the Active Partition agent.
2. Navigate back to the Node agent, then to the Forte_executor_*nodename* subagent.
3. Navigate to the subagent whose name ends with the same number as the agent for the standard partition.

For example, if the name of the Active Partition of the standard partition is Banking_cl0_Part1_0x66d:0x1, the name of the Active Partition agent for the corresponding iPlanet UDS executor partition would be Forte_executor_Mimi_0x66d.

Parent and subagents The parent agent for an Active Partition agent is an Installed Partition agent or an Ad hoc partition agent. The subagents of the Active Partition agent contain the iPlanet UDS runtime agents (DistObjectMgr agent, TaskMgr agent, and so on), as well as almost all user-defined system agents. Use the `ShowAgent` command for a list of subagents to the Active Partition agent.

States

BUSY

(Only subagents of Forte_Executor_*nodename*) The iPlanet UDS executor (ftexec) is running a standard iPlanet UDS application partition.

You can see what partition the executor partition is running by checking the ExecutingPartition instrument, as described in "ExecutingPartition" on page 129

ONLINE

The active partition is running. If the active partition is not running, the Active Partition agent does not exist in the agent hierarchy.

PENDING

(Only subagents of Forte_Executor_*nodename*) The iPlanet UDS executor (ftexec) has been reserved for a particular standard application partition and is waiting to load the partition.

RUNDOWN

The active partition is shutting down, probably because of a `Shutdown` command on the Active Partition agent or on one of the agent's parent agents: Installed Partition, Partition, Application, Node, Model Node, or Environment. When the active partition has completed any processing it needs to shut down, the Active Partition agent is removed from the agent hierarchy.

Commands

DebugPartition

The `DebugPartition` command places this partition under the control of the C++ debugger for the compiler installed on that node.

DebugPartition

On UNIX platforms, you need to set the FORTE_JIT_DEBUG environment variable before you can use this command.

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

FlushLogFiles

The `FlushLogFiles` command flushes the buffers for all the log files for this partition.

FlushLogFiles

ModLoggerRemote (ModLogger

The `ModLoggerRemote` command sets the logger flags for the active partition being managed by the current agent.

ModLoggerRemote +(*logger_flags*) | -(*logger_flags*)

Argument	Description
+(*logger_flags*)	Turn on the logger flag settings given in the parentheses.
-(*logger_flags*)	Turn off the logger flag settings given in the parentheses.

ModLogger *flags*

To start logging, invoke the `ModLoggerRemote` command using the '+' followed by a set of logger settings in parentheses. To stop logging, use the '-' followed by a set of logger settings in parentheses.

The logger flag settings in the `ModLoggerRemote` command modify any logger flag settings that were specified for the partition, either in the `-fl` startup flag or by the FORTE_LOGGER_SETUP environment variable.

The modified logger settings are only applied to the first file specified in the original logger settings for the Node Manager (or Environment Manager).

See the `LogMgr` class in the Framework Library online Help for a detailed description of the logger flag syntax. The following examples illustrate how to use the `ModLoggerRemote` command:

```
escript> ModLoggerRemote +(trc:os:1:1 cfg:c4:2-3:1)
escript> ModLoggerRemote -(cfg:c4)
```

SetEnvRemote (SetEnvVar)

The `SetEnvRemote` command sets the environment variable for the active partition managed by the current agent.

SetEnvRemote *env_variable new_value*

Argument	Description
env_variable	The name of an environment variable to set.
new_value	The new value of the environment variable to set.

SetEnvVar *env_variable new_value*

The `SetEnvRemote` command changes the setting of the environment variable in the active partition managed by the current agent. Within the TOOL code executing in that partition, any subsequent invocation of the GetEnv method on the OperatingSystem object gets the new setting.

The `env_variable` argument is the name of an environment variable to set in the process running the active partition, and the `new_value` argument is the value for the environment variable.

On UNIX and VMS nodes, the new setting of the environment variable does not remain beyond the current execution of the partition. On Windows NT, the new setting is stored permanently and is picked up in any client partition or iPlanet UDS application started up at a later time, because the values are stored in the registry in Windows NT.

Shutdown

The `Shutdown` command shuts down the active partition managed by the current Active Partition agent.

Shutdown *kill_executors*

The `Shutdown` command shuts down the active partition that is managed by the current agent. This forces an exit of the running partition.

If the `kill_executors` argument is set to `TRUE` or `1`, any interpreted partitions will shut down their hosting iPlanet UDS executor (ftexec or ftexecd) process as well.

When the active partition is stopped with the `Shutdown` command (or from a `Shutdown` propagated from one of the parent agents), the `ForcePartitionExit` method is invoked on the partition. This method cannot be rejected by the running partition (see the `ForcePartitionExit` method on the `Partition` class in the Framework Library online Help for details). Any clients that are connected to the partition respond as if the partition was killed by a machine or software crash, so they respond through their standard recovery mechanisms appropriate to the replication and dialog duration properties for the partition.

Instruments

CanBeActivated

The `CanBeActivated` instrument indicates whether the partition is enabled for start up by the management system. This Configuration instrument is read only.

ExecutingPartition

(Only subagents of Forte_Executor_*nodename*) The `ExecutingPartition` instrument indicates the partition that this partition is executing. This value is an empty string unless the state of the Active Partition agent is `BUSY`. This Configuration instrument is read only.

InstrumentLogging

The `InstrumentLogging` instrument sets the automatic logging of instruments to the active partition log file. For compiled partitions, the log file is the file specified by the `LogFile` instrument of this Active Partition agent. For interpreted partitions, the log file is the log file for the instance of the iPlanet UDS executor (ftexec) that is running this partition. The `InstrumentLogging` instrument is a Configuration instrument.

InstrumentLogging [*is_active*]

Argument	Description
is_active	Indicates whether automatic logging of instrument logging is currently active. Set to the string `TRUE` to make the logging active or `FALSE` to make it inactive.

The `InstrumentLogging` instrument turns on the automatic logging of active instruments to the active partition's log file each time the timing interval for the `LogTimer` instrument in the active partition expires. By default, automatic logging is disabled for active partitions. The `InstrumentLogging` instrument is used in conjunction with the `LogTimer` instrument, also defined on the active partition. See "LogTimer" on page 131 for more information on how the process of automatic logging works.

The `is_active` argument is a boolean value set to `FALSE` by default. If `is_active` is `TRUE`, then the values of the current set of instruments being logged in the Active Partition agent, or any of its subagents, are automatically logged to the active partition's log file. If `is_active` is `FALSE`, no logging takes place. Note that even if you turn off the `InstrumentLogging` instrument, the detailed data is still collected, which could be a significant performance drain. Therefore, you need to disable the `LogTimer` instrument as well (unless you want to log to the environment log file).

LogFile

The `LogFile` instrument indicates the name of the file to use when logging events for the compiled active partition. The `LogFile` instrument is a Configuration instrument.

NOTE This instrument is available only for compiled partitions. For interpreted partitions, the logged information is always logged to the log file for the instance of the iPlanet UDS executor (ftexec) that is running this partition.

LogFile *log_file_name*

Argument	Description
log_file_name	Indicates the name of the file to use for logging the active partition logging events.

The `LogFile` instrument specifies the name of the log file to use for logging instrument events for the Active Partition agent. The information that is logged includes instrument logging events, as well as an audit trail of all important operations performed by the active partition, such as starting. The Active Partition agent log file is independent of the log files specified using the `-fl` flag for logging messages generated by an application or the iPlanet UDS runtime system.

The `log_file_name` argument indicates the name of the log file to use for logging. This file name should be given in one of two ways: relative or absolute. In either case, however, it uses iPlanet UDS portable file name syntax (UNIX style). If a relative name is given for `log_file_name` (it does not start with a `/`), then the file is given relative to the `FORTE_ROOT/log` directory on the node on which the active partition is executing. If an absolute path is given in the `log_file_name`, it is an absolute path on the machine on which the active partition is executing.

If you change the logging file name after the active partition has already started logging to another file, that file is closed, and the new file is opened.

The following examples show how you can set the `LogFile` instrument in Escript:

```
escript> UpdateInstrument LogFile vdir:/onvms/ap.log
escript> UpdateInstrument LogFile /udir/sparc/ap.log
```

LogTimer

The `LogTimer` instrument turns on or off logging events and sets the interval for instrument logging events within the active partition. The `LogTimer` instrument is a Timer instrument.

LogTimer [**“***is_active interval_in_msec* **”**]

Argument	Description
is_active	Indicates whether the timer is currently active. Set to the string “`TRUE`” to make the timer active or “`FALSE`” to make it inactive. The default value is `FALSE`.
interval_in_msec	The number of milliseconds between log timer events. The default value is `300000`.

The `LogTimer` instrument sets the timer interval and active status for instrument logging events in the active partition. These `LogTimer` settings apply to any instruments that have been added to the instrument logging list for the active partition, through use of the `SetInstrumentLogging` command. By default, no instruments are on the instrument logging list for the active partition, so you must explicitly turn them on.

To turn on the `LogTimer` instrument, set the `is_active` argument to the string `TRUE` and the `interval_in_msec` to the interval, in milliseconds, between timer logging events. In Escript, you must specify quotes around the pair of arguments, as shown in the following example:

```
escript> UpdateInstrument LogTimer “TRUE 10000”
```

To turn off the `LogTimer` instrument, set the `is_active` argument to the string `FALSE`. You must also set the value for the `interval_in_msec` argument to some value, but it will not matter because the timer is not active. Again, you must specify the argument pair in quotes, or you can omit the arguments and Escript prompts for their values. For example, the following command turns off the `LogTimer` instrument:

```
escript> UpdateInstrument LogTimer “FALSE 10000”
```

After you have turned on the `LogTimer` instrument, and have invoked a `SetInstrumentLogging` command on one or more instruments in the Active Partition agent or any of its subagents (DistObjectMgr agent, TaskMgr agent, any user-defined agents, and so on), the current values of these instruments are collected whenever the `LogTimer` timing interval expires. However, by default, the values of the instruments are still not automatically logged to any files. You can request that the instruments be automatically logged, either to the environment or the active partition log (or both), by turning on the `InstrumentLogging` instrument on either of those agents. Once the `InstrumentLogging` instrument has been enabled, the logging of instruments occurs at each expiration of the timing interval.

The values of the instruments are also posted with the LogInstruments event on the Active Partition and Environment agents. See *Programming with System Agents* for information on how to process this event programmatically.

The following set of Escript commands shows how to log the value of the number of messages sent and received within the Auction server partition to both the partition’s log file and to the environment log file. The first set of commands shows how to navigate to the instruments for messages sent and received (DistObjectMgr agent instruments), and how to add them to the instruments being logged:

```
escript> FindSubAgent mynode
escript> FindSubAgent auction_part1_cl0
escript> ShowAgent
...this lists active partitions, with partition ids in hexadecimal...
escript> FindSubAgent 0x1234
escript> FindSubAgent DistObjectMgr
escript> SetInstrumentLogging MessagesSent 1
escript> SetInstrumentLogging MessagesReceived 1
```

The next set of commands follows immediately and sets up the `LogTimer` instrument on the active partition for 100-second logging, and then enables the automatic logging of the instruments into both the active partition log and the environment log:

```
escript> FindParentAgent
escript> UpdateInstrument LogTimer “TRUE 100000”
escript> UpdateInstrument InstrumentLogging TRUE
escript> FindActEnv
escript> UpdateInstrument InstrumentLogging TRUE
```

At this point, you are logging the `MessagesSent` and `MessagesReceived` instruments from the DistObjectMgr agent in the log files for both the active partition for auction_part1_cl0 on mynode and for the log file for the environment as a whole.

ProcessID

The `ProcessID` instrument contains the operating system ID for the process running the active partition. This Configuration instrument is read only.

The `ProcessID` instrument is system-specific, and can be used to help troubleshoot problems using other system management tools.

Ad hoc partition Agent

Parent Agent

Node agent

Subagents

Active Partition agent

SystemMonitor Class

GenericPartitionAgent

States

State	Description
OFFLINE	No instances of this partition are running.
ONLINE	An instance of this partition is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
ModLoggerRemote	+(*logger_flags*) -(*logger_flags*)	Component > Modify Log Flags	Sets the logger flags for all of the active partitions that are represented by this Installed Partition agent.
SetEnvRemote	*env_variable new_value*	Component	Sets the environment variable for all of the active instances of the installed partition managed by the current agent.
Shutdown	none	Component	Shuts down all active instances of the installed partition represented by the current agent.
ShutdownSubAgent	*subagent*	none	Shuts down the named subagent and its managed object.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
CanBeActivated	none	Yes	Configuration	Indicates whether the partition is enabled for startup by the management system.

Using the Ad hoc partition Agent

The Ad hoc partition agent is a temporary agent that appears when a partition that has not been installed using the iPlanet UDS installation process runs. When such a partition starts up, an Active Partition agent is created, and an Ad hoc partition agent also appears. This Ad hoc partition agent becomes the subagent of the Node where the partition is running, and has the Active Partition agent for the partition as its subagent. The Ad hoc partition agent is known to the Node Manager until the Node Manager shuts down and restarts.

In the Environment Console, the Ad hoc partition agent is labelled an Ad hoc partition agent. In Escript, the Ad hoc partition agent is labelled an Installed Partition agent.

The main differences between Ad hoc partition agents and Installed Partition agents is that you cannot start up additional partitions using the `Startup` command on the Ad hoc partition agent, and the Ad hoc partition has only a Node agent as a parent.

Parent and subagents The parent agent to the Ad hoc partition is the Node agent. The subagents to an Ad hoc partition agent are the Active Partition agents, which represent executing instances of the partition.

States

OFFLINE

No instances of this uninstalled partition are running.

ONLINE

An instance of this uninstalled partition is running.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

ModLoggerRemote

The `ModLoggerRemote` command sets the logger flags for all of the active partitions that are represented by this Installed Partition agent.

ModLoggerRemote +(*logger_flags*) | -(*logger_flags*)

Argument	Description
+(*logger_flags*)	Turn on the logger flag settings given in the parentheses.
-(*logger_flags*)	Turn off the logger flag settings given in the parentheses.

To start logging, invoke the `ModLoggerRemote` command using the '+' followed by a set of logger settings in parentheses. To stop logging, use the '-' followed by a set of logger settings in parentheses.

The logger flag settings in the `ModLoggerRemote` command modify any logger flag settings that were specified for the partition either in the Partition Workshop, the `-fl` startup flag, or by the FORTE_LOGGER_SETUP environment variable.

The modified logger settings are only applied to the first file specified in the original logger settings for the Node Manager (or Environment Manager).

See the `LogMgr` class in the Framework Library online Help for a detailed description of the logger flag syntax.

```
escript> ModLoggerRemote +(trc:os:1:1 cfg:c4:2-3:1)
escript> ModLoggerRemote -(cfg:c4)
```

SetEnvRemote

The `SetEnvRemote` command sets the environment variable for all of the active instances of the installed partition managed by the current agent.

SetEnvRemote *env_variable new_value*

Argument	Description
env_variable	The name of an environment variable to set.
new_value	The new value of the environment variable.

The `SetEnvRemote` command changes the setting of the environment variable in all instances of the active partition managed by the current Installed Partition agent. Within the TOOL code executing in that partition, any subsequent invocation of the `GetEnv` method on the OperatingSystem object gets the new setting.

The `env_variable` argument is the name of an environment variable to set in the process running the active partition, and the `new_value` argument is the value for the environment variable.

On UNIX and VMS nodes, the new setting of the environment variable does not remain beyond the current execution of the partition. On Windows NT, the new setting is stored permanently and is picked up in any client partition or iPlanet UDS application started at a later time, because the value is stored in the registry in Windows NT.

Shutdown

The `Shutdown` command shuts down all active instances of the installed partition represented by the current agent.

Shutdown

The `Shutdown` command shuts down the Installed Partition agent. The shutdown request is propagated to all of the subagents of the Installed Partition agent. The most important of these are the Active Partition agents that are running the partition on this node.

ShutdownSubAgent

The `ShutdownSubAgent` command shuts down the named active partition.

ShutdownSubAgent *subagent*

Argument	Description
subagent	The name of a subagent to be shut down with its managed object.

If the named subagent does not exist, the `ShutdownSubAgent` command does nothing.

The `ShutdownSubAgent` command performs the same function as the following command sequence in an Escript script:

```
escript> FindSubAgent AutoCompileSvc_cl0_Part1_0x4d7:0x1
escript> Shutdown
escript> FindParentAgent
```

The difference between this sequence of commands and using the `ShutdownSubAgent` command is that the `Shutdown` command is not invoked on the current agent if the subagent does not exist.

Instruments

CanBeActivated

The `CanBeActivated` instrument indicates whether the partition is enabled for startup by the management system. This Configuration instrument is read only and is always false for ad hoc partitions.

Application Agent

Parent Agent

Environment agent

Subagents

Partition agents for the application

SystemMonitor Class

ApplicationAgent

States

State	Description
DEGRADED	Some parts of the application are active, but at least one of the application's shared server partitions is either OFFLINE or DEGRADED.
IN-PROGRESS	The application is partially installed, but should not yet be run. An important server partition is not yet installed, for example.
LOADED	The application's definition is loaded into the environment, but no one has tried to install it.
OFFLINE	No server partitions of this application are running.
ONLINE	All of the application's server partitions are running.
RELEASED	A distribution of this application exists, but it has not been loaded into the environment.

Command Summary

Command	Arguments	Environment Console menu	Description
AssignAppComp	*node_name* *component_name*	none	Assigns the specified application component in the current application for installation on the specified node.
DisableAppComp	*node_name* *partition_name*	none	Disables startup of the specified partition on the specified node in the current application.
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
EnableAppComp	*node_name* *partition_name*	none	Enables startup for the specified partition on the specified node, in the current application.
Install	none	Component	Installs the current application into the environment.
ReleaseLock	none	none	Releases any installation locks on the current application.
SetAppCompCompiled	*node_name* *compiled_flag* *component_name*	none	Declares whether a partition (or library) is to be used in compiled or iPlanet UDS executor form.
Shutdown	*kill_executors*	Component	Shuts down all server partitions running in the application on all nodes. If the argument is set to `TRUE` or `1`, the interpreted partitions will shut down their hosting iPlanet UDS executor (ftexec or ftexecd) process as well.
ShutdownSubAgent	*subagent*	none	Shuts down the named subagent and its managed object.
Startup	none	Component	Starts all server partitions (with all their replicates) for the specified application.
UnassignAppComp	*node_name* *component_name*	none	Removes the assignment of an application component from a node.
Uninstall	none	Component	Removes the definition of the current application from the environment.

Instrument Summary

Instrument	Arguments	Read Only?	Type	Description
InstallationSteps	none	Yes	SubObject	Shows the steps still needed to install an application.

Programmatic Command Summary

Command	Arguments	Returns	Description
FindPartByService	*service_name*	Object	Returns the agent for the (logical) Partition that contains the named service object.

Using the Application Agent

The Application agent represents an application that has been loaded into an environment repository.

Parent and subagents The parent agent for an application is the Environment agent, and the subagents are the Logical Partition agents for the application.

The states for the Application agent indicate the general state of an application. However, the state might not completely reflect what parts of the application are installed or waiting to be installed. For more information about whether the application is completely installed or whether you need to take further action to complete the installation process, check the `InstallationSteps` instrument for the application, which is described in "InstallationSteps" on page 150.

States

DEGRADED

Some parts of the application are active, but at least one of the application's shared server partitions is either OFFLINE or DEGRADED.

IN-PROGRESS

The application is partially installed, but should not yet be run. An important server partition is not yet installed, for example. For specific information about the status of an application in terms of its installation, check the `InstallationSteps` instrument for the application, which is described in "InstallationSteps" on page 150.

LOADED

The application's definition is loaded into the environment, but no one has tried to install it. To install the entire application, use the Application agent's `Install` command, which is described in "Install" on page 146. You can also install specific parts of the application on a node using the `InstallApp` command on the Node agent, which is described in "InstallApp" on page 253.

OFFLINE

No server partitions of this application are running.

ONLINE

All of the application's server partitions are running. If an application contains only client partitions, then ONLINE means that at least one client partition is running.

RELEASED

A distribution of this application exists, but it has not been loaded into the environment. To load the distribution in Escript, use the Environment agent's `LoadDistrib` command, which is described in "LoadDistrib" on page 202. In the Environment Console, you can use the File > Load Distribution command to load the distribution.

Commands

AssignAppComp

The `AssignAppComp` command assigns the specified application component in the current application for installation on the specified node.

AssignAppComp *node_name component_name*

Argument	Description
node_name	The name of the node where the component is being assigned for installation.
component_name	The optional name of the component that is being assigned to the node.

Use the `AssignAppComp` command to assign an application component within the current application for installation on a node within the environment. Before invoking the `AssignAppComp` command, you must lock the environment by invoking the `LockEnv` command. An application component can either be one of the partitions within the application, or one of the libraries within a library configuration.

The `node_name` argument must be a valid node defined in the environment. The node must have all of the external resource managers, communications protocols, and libraries needed to support the component.

The `component_name` argument is the name of the partition (or library in a library configuration) to be assigned to the node. You can specify the unique trailer portion of the name (such as "client") as shorthand for the partition name.

When an application is partitioned, each partition (or library in a library configuration) is designated for future installation on one or more nodes in the environment, based on matching the needed properties of the partition and the actual properties of each node. When a subsequent `MakeAppDistrib` command is given for the application from Fscript or the Partition Workshop, support for installing the component on the designated nodes is provided in the distribution. The `AssignAppComp` command can be used to designate additional nodes for future installation of a component.

If you assign a partition component that is not a replicated partition to more than one node, the partition starts on only one of the assigned nodes. You can use the `EnableAppComp` command to designate on which node a partition component will start, once installed. However, if you assign the partition to more than one node, you can provide manual failover if a partition fails. Manual failover means using the Environment Console or the `Startup` command of the Installed Partition agent to start the partition on any of the nodes on which the partition has been assigned and successfully installed.

DisableAppComp

The `DisableAppComp` command disables the startup of the specified partition on the specified node in the current application.

DisableAppComp *node_name partition_name*

Argument	Description
node_name	The node on which the partition is to be disabled.
partition_name	The name of the partition to be disabled.

The `DisableAppComp` command removes the automatic startup capabilities of a partition component on a particular node. You use the `DisableAppComp` command when you want to assign a partition to a node, but do not want the server on that node to start up when the `Startup` command on the node or Application agent is invoked.

You can only invoke the `DisableAppComp` command for partition components.

Before invoking the `DisableAppComp` command, you must lock the environment by invoking the `LockEnv` command.

The `node_name` argument is the name of a node in the environment to which the partition is assigned. You can assign the partition to the node either as part of the default partitioning or through the `AssignAppComp` command.

The optional `partition_name` argument is the name of the partition component to be disabled on the node. You can specify the unique trailer portion of the name (such as “client”) as shorthand for the partition name.

You can assign a partition within an application to one or more nodes in the environment. For non-replicated partitions, you can designate only one of the assigned nodes as the node on which the partition will be automatically started and managed. This is done by the default partitioning, or by invoking the `EnableAppComp` command for the partition on that node. For replicated partitions, any number of assigned nodes can be enabled using the `EnableAppComp` command, and servers are automatically started on all of the enabled nodes.

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

EnableAppComp

The `EnableAppComp` command enables automatic startup for the specified partition on the specified node in the current application.

EnableAppComp *node_name partition_name*

Argument	Description
node_name	The node on which the partition is be enabled.
partition_name	The name of a partition in the current application for which a node is to be enabled.

The `EnableAppComp` command specifies that a partition component, be enabled for startup on its assigned node. Before invoking the `EnableAppComp` command, you must lock the environment by invoking the `LockEnv` command.

You can only invoke the `EnableAppComp` command on partition components.

The `node_name` argument is the name of a node in the environment to which the partition is assigned.

The optional `partition_name` argument is the name of the partition component to be enabled on the node. You can specify the unique trailer portion of the name (such as “client”) as shorthand for the partition name.

You can assign a partition within an application to one or more nodes in the environment. For non-replicated partitions, you can designate only one of the assigned nodes as the node on which the partition will be automatically started and managed. This is done by the default partitioning, or by invoking the `EnableAppComp` command for the partition on that node. For replicated partitions, any number of assigned nodes can be enabled using the `EnableAppComp` command, and servers are automatically started on all of the enabled nodes.

Install

The `Install` command installs the current application into the environment.

Install

The `Install` command takes the current application and starts to install it on the nodes in the environment. `Install` uses the currently executing Node Managers in the environment to install all partitions that can be installed on the appropriate nodes. The environment cannot be locked to invoke the `Install` command.

If some Node Managers are not accessible, `Install` may install only part of the application. Once the Node Managers are again accessible, you can invoke the `Install` command again to complete the installation.

If the Install command does not complete the installation, you can use the `ShowInstrument` command on the InstallationSteps instrument on the Application agent to see what remains to be done.

For complete instructions about installing an iPlanet UDS application using the Environment Console, see *iPlanet UDS System Management Guide*. For information about the commands to use when installing an iPlanet UDS application using Escript, see "Installing Applications" on page 55.

ReleaseLock

The `ReleaseLock` command releases any installation locks on the current application.

ReleaseLock

An installation lock might be placed on an application while it is being installed, and if it is left there by the system, it can be removed by invoking the `ReleaseLock` command.

CAUTION This command should be used only if the lock cannot be removed through normal `Install`, `Uninstall` and `LoadDistrib` commands.

SetAppCompCompiled

The `SetAppCompCompiled` command declares whether a partition (or library) is to be used in compiled or standard iPlanet UDS executor form.

SetAppCompCompiled *node_name compiled_flag component_name*

Argument	Description
node_name	The name of a node to which the component is assigned.
compiled_flag	Set to `TRUE` to set the component to compiled form, or `FALSE` to set it to iPlanet UDS executor form.
component_name	The name of a component in the current application or library configuration.

If a component has been distributed and installed as a compiled component on one or more nodes in an environment, the compiled form of the component is automatically used. However, the iPlanet UDS distribution package also includes the information needed to use the same component in iPlanet UDS executor form. You can use the `SetAppCompCompiled` to tell the system to use the iPlanet UDS executor form of the component, or to convert back to the compiled form.

If a component, say a partition, was not designated as compiled before the distribution was made, you can only run the partition in standard iPlanet UDS executor form, so you cannot use the `SetAppCompCompiled` command for that partition.

Before invoking the `SetAppCompCompiled` command, you must lock the environment by invoking the `LockEnv` command.

The `node_name` argument designates one of the nodes to which the component has been assigned. If the component is not yet assigned to a node, you can invoke the `AssignAppComp` command to assign it to that node.

The `compiled_flag` argument specifies that the component is to be used in compiled form by specifying a value of `TRUE`, or to be used in standard iPlanet UDS executor form by specifying a value of `FALSE`.

The `component_name` argument is the name of one of the partitions in the current application (or libraries in the current library configuration). The component must have been distributed in compiled form in order to invoke the `SetAppCompCompiled` command.

Shutdown

The `Shutdown` command shuts down all server partitions running in the application on all nodes.

Shutdown *kill_executors*

The `Shutdown` command shuts down the Application agent and all active instances of the partitions running for the application. The shutdown request is propagated to all of the subagents of the Application agent, which shuts down all active partitions in the application, running on all nodes. The Shutdown command therefore provides a simple way to shut down all iPlanet UDS partitions running in an environment.

When the `Shutdown` request results in stopping any of the active partitions in the application, it invokes the `ForcePartitionExit` method on each of the partitions, which cannot reject the request. (See documentation for the `ForcePartitionExit` method on the `Partition` class in the Framework Library online Help for details.)

If the `kill_executors` argument is set to `TRUE` or `1`, any interpreted partitions will shut down their hosting iPlanet UDS executor (ftexec or ftexecd) process as well.

ShutdownSubAgent

The `ShutdownSubAgent` command shuts down all running instances of the named logical partition.

ShutdownSubAgent *subagent*

Argument	Description
subagent	The name of a subagent to be shut down with its managed object.

If the named subagent does not exist, the `ShutdownSubAgent` command does nothing.

The `ShutdownSubAgent` command performs the same function as the following command sequence in an Escript script:

```
escript> FindSubAgent AutoCompileSvc_cl0_Part1
escript> Shutdown
escript> FindParentAgent
```

The difference between this sequence of commands and using the `ShutdownSubAgent` command is that the `Shutdown` command is not invoked on the current agent if the subagent does not exist.

Startup

The `Startup` command starts all server partitions (with all their replicates) for the specified application.

Startup

The `Startup` command directs the Node Managers to start all server partitions for the current application. A partition is automatically started for each node that contains a partition within the installed application. If an installed partition has a replication count, iPlanet UDS starts that number of instances of the partition on the node. If some of the partitions in the application are already running, this command brings up as many new partitions as are necessary to bring it back to its defined level of partitions and replicates. If all partitions are already running at their defined levels, this command does nothing.

You do not need to invoke the `Startup` command to start the servers for an installed application, because the first client attempting to connect to a server triggers the auto-startup of the minimum number of partitions needed for the application to run. However, this step can be time consuming and does not provide for load balancing or failover protection. You should start the partitions for an application when you start the runtime system.

You can invoke the `Startup` command of the Installed Partition agent to start one installed partition.

UnassignAppComp

The `UnassignAppComp` command removes the assignment of an application component from a node.

UnassignAppComp *node_name component_name*

Argument	Description
node_name	The name of the node for which the component is to be unassigned.
component_name	The name of a component in the current application.

The `UnassignAppComp` command removes a component from a future installation on a specific node in the environment.

Before invoking the `UnassignAppComp` command, you must lock the environment by invoking the `LockEnv` command, described in "LockEnv" on page 79.

The `node_name` argument must be a valid node defined in the environment, and must have the partition assigned to it.

The `component_name` argument is the name of the partition (or a library in a library configuration) to be unassigned from the node. You can specify the unique trailer portion of the name (such as "client") as shorthand for the partition name.

See the `AssignAppComp` command for more information on assigning components to nodes (see "AssignAppComp" on page 142).

Uninstall

The `Uninstall` command removes the definition of the current application from the environment.

Uninstall

Any application that has been loaded into the environment using the `LoadDistrib` command on the Environment or Node agent is considered registered in the environment. To remove the application, use the `Uninstall` command.

You cannot have the active environment locked before invoking the `Uninstall` command.

You do not need to use the `Commit` command after using the `Uninstall` command, because the changes are immediately saved to the environment repository.

NOTE The `Uninstall` command does not remove the actual files in the installation associated with the application. You must use standard operating system utilities to do that. However, if you re-install an application, the existing files are overwritten.

Instruments

InstallationSteps

The `InstallationSteps` instrument shows the steps still needed to install an application. This `SubObject` instrument is read only.

The `InstallationSteps` instrument shows the steps needed to complete the full installation of the application in the environment. This instrument is initialized when you load the application into the environment, using the `LoadDistrib` command in the Environment or Node agent. At that point, you can navigate to the Application agent, and display the `InstallationSteps` instrument to see what steps are needed.

After you invoke the `Install` command on the Application agent, some of the steps are likely to be completed, but you can show any that are still to be done by re-displaying the `InstallationSteps` instrument. This might refer to nodes that were offline when the last installation was done, and can give you hints as to what might need to be done to complete the installation. Once all components have been installed successfully, the `InstallationSteps` instrument is empty. After that, if any changes are made, the instrument shows the steps necessary to propagate those changes.

Programmatic Commands

FindPartByService

The `FindPartByService` command returns the agent for the (logical) partition that contains the named service object.

FindPartByService *service_name*

Argument	**Description**
service_name	Name of the service object whose partition you want to access.

The `service_name` argument can be either a simple service object name or fully-qualified name, which includes both the project name and the service object name, as shown:

```
FindPartByService Banking.AccountServer
```

If you specify only the simple service object name, and you have more than one service object with the same name, you need to use the fully-qualified name. Otherwise, iPlanet UDS will return the first service object it finds with the specified name.

BtreeCache Agent

Parent Agent

RepositoryServer agent or Active Partition agent

Subagents

none

SystemMonitor Class

SystemAgent

States

State	Description
ONLINE	The B-tree cache exists and is in use.

Commands

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instruments

Instrument	Argument	Read Only?	Class	Description
MaxRORepos	*max_number*	No	Configuration	Maximum number of read-only repositories for which the system keeps files opened at any one time.

Instrument	Argument	Read Only?	Class	Description
NumCommits	none	Yes	Counter	Total number of transactions committed by all B-tree repositories in this partition.
NumFlushes	none	Yes	Counter	Total number of times all repositories have flushed this cache.
PagesFlushed	none	Yes	Counter	Total number of pages written from the cache to the disk.
PagesInUse	none	Yes	Counter	Number of pages currently used by the cache.
ReadHits	none	Yes	Counter	Total number of times a page being read was found in the cache.
ReadMisses	none	Yes	Counter	Total number of times a page being read was not found in the cache.
ROReposClosed	none	Yes	Counter	Total number of times a read-only repository has been closed by the system to limit the number of files open concurrently.
ROReposInUse	none	Yes	Counter	Number of read-only repositories in use.
TotalPages	*number*	No	Configuration	Number of 4k pages that are available to the cache.
WriteHits	none	Yes	Counter	Total number of times a page being written was found in the cache.
WriteMisses	none	Yes	Counter	Total number of time a page being written was not found in the cache.

Using the BtreeCache Agent

The BtreeCache agent represents the cache used with the B-tree central repository to improve repository performance. The B-tree cache stores repository data in memory, to reduce the number of times that iPlanet UDS needs to read repository data from or write repository data to a storage device. The B-tree cache stores the repository data in 4 KB pages.

You can improve performance by making the B-tree cache larger. To improve the rate at which iPlanet UDS retrieves repository data, increase the number of 4 KB pages that the B-tree cache can contain by setting the `TotalPages` instrument of the BtreeCache agent.

Parent and subagents The parent agent for a BtreeCache agent is either a RepositoryServer agent, or an Active Partition agent for the iPlanet UDS Executor that is executing this interpreted partition or library. A BtreeCache agent has no subagents.

States

ONLINE

The B-tree cache exists and is in use.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

MaxRORepos

The `MaxRORepos` instrument specifies the maximum number of read-only repositories for which the system keeps files opened at any one time. The `MaxRORepos` instrument is a Configuration instrument.

If the number of repositories in use, as indicated by the `ROReposInUse` instrument, is greater than the `MaxRORepos` value, you should consider increasing this value.

MaxRORepos [*max_number*]

Argument	Description
max_number	Specifies the maximum number of read-only repositories whose files can be opened at any one time. The default value of this instrument is system-dependent. The smallest value this instrument can have is `2`. The largest value this instrument can have depends on the machine. You can determine the largest possible value by entering an invalid value, such as `zero`.

NumCommits

The `NumCommits` instrument records the total number of transactions that have been committed by all B-tree repositories in the partition. This Counter instrument is read only.

NumFlushes

The `NumFlushes` instrument records the total number of times all repositories have flushed this cache. This Counter instrument is read only.

Generally, the closer this value is to the value of the `NumCommits` instrument, the better the performance of the repository.

If this value is larger than the value of the `NumCommits` instrument, then the transaction sizes exceed the cache size, which means that the operating system needs to read or write to the disk in the middle of transactions. You should consider increasing the value of the `TotalPages` instrument to increase the size of the cache to improve performance.

PagesFlushed

The `PagesFlushed` instrument records the total number of pages written from the cache to the disk. This Counter instrument is read only.

PagesInUse

The `PagesInUse` instrument counts the number of pages currently used by the cache. This Counter instrument is read only.

This value should be the same as the value of the `TotalPages` instrument after extended use. If the value is smaller, the value of the `TotalPages` instrument (the cache size) is larger than needed.

ReadHits

The `ReadHits` instrument records the total number of times a page being read was found in the cache. This Counter instrument is read only.

When a page is found in the cache, it does not need to read the page from the disk itself.

ReadMisses

The `ReadMisses` instrument records the total number of times a page being read was not found in the cache, and therefore needed to be read from the disk. This Counter instrument is read only.

If the ratio of misses (ReadMisses) to the total number of reads (ReadHits + ReadMisses) is more than 20%, consider increasing the size of the cache by increasing the `TotalPages` instrument value.

ROReposClosed

The `ROReposClosed` instrument records the total number of times a read-only repository has been closed by the system to limit the number of files open concurrently. This Counter instrument is read only.

ROReposInUse

The `ROReposInUse` instrument counts the number of read-only repositories currently in use. This Counter instrument is read only.

If this value is greater than the value of the `MaxRORepos` instrument, some files are being closed when other files are being read to keep the total number of open files below the value of `MaxRORepos` instrument.

TotalPages

The `TotalPages` instrument specifies the number of 4k pages that are available to the cache. The `TotalPages` instrument is a Configuration instrument.

TotalPages [*number*]

Argument	Description
number	Specifies the number of 4k pages that are available to the cache. The default value for this cache depends on the application being run. The `rpclean` command has a default of `150`. The default value for a repository server is also `150`. Default values for other applications is `25`. The minimum value is `5`, and the maximum value is `2048`.

NOTE You can improve performance of the repository by increasing the value of the `TotalPages` instrument to increase the size of the cache. If the value of the `NumFlush` instrument is greater than the value of the `NumCommits` instrument, the transaction sizes exceed the cache size, meaning that the repository server needs to read or write to the disk in the middle of transactions, which can reduce performance.

Increasing the cache size also increases the memory used by the partition, which means that you might need to increase the memory size set by the memory flags for the partition as well. However, if the increased memory settings are too high for the available physical memory, the performance might be reduced.

WriteHits

The `WriteHits` instrument records the total number of times a page being written has been found in the cache. This Counter instrument is read only.

WriteMisses

The `WriteMisses` instrument records the total number of times a page being written has not been found in the cache, which requires another page to be written to disk to make space for it. This Counter instrument is read only.

BtreeRepository Agent

Parent Agent

Repository agent

Subagents

none

SystemMonitor Class

SystemAgent

States

State	**Description**
ONLINE	The B-tree repository is open and in use.
FAULT	The B-tree repository switched to read-only mode because of an error.
RUNDOWN	The B-tree repository has closed, but the agent has not yet been removed.

Commands

Command	**Arguments**	**Environment Console menu**	**Description**
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instruments

Instrument	Argument	Read Only?	Class	Description
AvgTransactionSizeKB	none	Yes	Average	Average size of transactions committed to the repository since it was opened, in kilobytes.
DataFileSizeKB	none	Yes	Counter	Size of data file in kilobytes
DeletedSpaceKB	none	Yes	Counter	Approximate amount of delete space, in kilobytes, in the data file that could be recovered using the `rpclean -q` command on the repository.
IndexFileSizeKB	none	Yes	Counter	Size of index file in kilobytes.
NumTransactions	none	Yes	Counter	Number of transactions committed to this repository since it was opened.
RepositoryName	none	Yes	Configuration	The name of the B-tree repository.
VolumeFreeSpaceKB	none	Yes	Counter	Space available on the volume on which the repository resides, in kilobytes.

Using the BtreeRepository Agent

The BtreeRepository agent represents the cache used with the B-tree central repository to improve repository performance.

Parent and subagents The parent agent for a BtreeRepository agent is a Repository agent. A BtreeRepository agent has no subagents.

States

ONLINE

The B-tree repository is open and in use.

FAULT

The B-tree repository switched to read-only mode because of an error. Check the log file for the partition using the repository to determine what error occurred.

On a central repository, check the .rop file for error and historical information. You probably need to restart the repository server after you fix the problem.

RUNDOWN

The B-tree repository has closed, but the agent has not yet been removed from the agent hierarchy.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

AvgTransactionSizeKB

The `AvgTransactionSizeKB` instrument records the average size of transactions, in kilobytes, committed to the repository since it was opened. This Average instrument is read only.

The transaction size is the amount of data written to disk in unique locations. For example, if you write twice to the same region of the data file, the writes to that region are only counted once in determining the size of the transaction.

If the value of this instrument is more than the value of the `VolumeFreeSpaceKB` instrument, you should free up disk space immediately.

DataFileSizeKB

The `DataFileSizeKB` instrument indicates the size of the data file (.btd) in kilobytes. This Counter instrument is read only.

DeletedSpaceKB

The `DeletedSpaceKB` instrument records the approximate amount of delete space, in kilobytes, in the data file that could be recovered using the `rpclean -q` command on the repository. This Counter instrument is read only.

This value does not include space that could be reclaimed by using the full `rpclean` command to delete unused objects.

IndexFileSizeKB

The `IndexFileSizeKB` instrument indicates the size of the index file (.btx) in kilobytes. This Counter instrument is read only.

NumTransactions

The `NumTransactions` instrument counts the number of transactions that have been committed to this repository since it was opened. This Counter instrument is read only.

RepositoryName

The `RepositoryName` instrument contains the name of the B-tree repository, including the path if the path is not `FORTE_ROOT/repos`. This Configuration instrument is read only.

VolumeFreeSpaceKB

The `VolumeFreeSpaceKB` instrument indicates the amount of space available on the volume on which the repository resides, in kilobytes. This Counter instrument is read only.

Try to maintain at least 5000 to 10000 kilobytes of free space to avoid have transactions aborted because of lack of space.

If the value of this instrument is less than the value of the instrument, you should free up disk space immediately.

CommMgr Agent

Parent Agent

Active Partition agent

Subagents

None

SystemMonitor Class

CommMgrAgent

States

State	Description
ONLINE	The communication service is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
ActiveRecvTasks	none	Yes	Counter	Number of running tasks that can receive incoming communication messages (receiver pool).
ActiveSendTasks	none	Yes	Counter	Number of running tasks that can send outgoing communication messages (sender pool).
AvgRecvPacketSize	none	Yes	Counter	Average size of the packets that this partition has received during the life of this partition.
AvgSendPacketSize	none	Yes	Counter	Average size of the packets that this partition has sent during the life of this partition.
BlockedSends	none	Yes	Counter	Number of attempts to send outgoing communication messages that were returned because network machine or protocol resources were busy.
BusyRecvTasks	none	Yes	Counter	Number of tasks that are busy receiving incoming communication messages.
BytesReceived	none	Yes	Counter	Number of bytes that the current partition has read from the network.
BytesSent	none	Yes	Counter	Number of bytes that the current partition has sent across the network.
CommMgrDispatches	none	Yes	Counter	Number of dispatchable internal Communication Manager objects that have reached the partition from the network.

Instrument	Argument	Read Only?	Type	Description
ConfiguredRcvTasks	*number_tasks*	No	Configuration	Target number of tasks that can receive incoming communication messages (receiver pool).
ConfiguredSendTasks	*number_tasks*	No	Configuration	Target number of tasks that can send outgoing communication messages (sender pool).
CurrentConnections	none	Yes	Counter	Number of connections currently held by this partition.
CurrentPacketsAllocated	none	Yes	Counter	Number of iPlanet UDS packets currently allocated from the main iPlanet UDS heap.
DistObjectMgrDispatches	none	Yes	Counter	Number of dispatchable internal objects that have reached the partition from the network.
InboundCloses	none	Yes	Counter	Total number of connections that have been held by this partition that have been closed by other partitions.
InboundConnects	none	Yes	Counter	Total number of connections that have been held by this partition that were initiated by other partitions.
KeepAliveCloses	none	Yes	Counter	Total number of connections closed by keepalive in a partition.
KeepAliveCount	*count*	No	Configuration	Number of pings before connection is closed
KeepAliveCycle	*interval*	No	Configuration	Length of time, in seconds, that a connection can be inactive before keepalive processing starts. A `0` value turns off the keepalive function.
KeepAliveInterval	*interval*	No	Configuration	Interval, in seconds, in which a ping message is expected to reply.

Instrument	Argument	Read Only?	Type	Description
MaxPacketPoolSize	none	Yes	Counter	Largest size of the free pool for iPlanet UDS network packets.
MaxRecvQueueDepth	none	Yes	Counter	Maximum number of iPlanet UDS network packets that have been received and not immediately dissembled.
MaxSendQueueDepth	none	Yes	Counter	Maximum number of iPlanet UDS network packets that have been assembled but not immediately sent.
OutboundCloses	none	Yes	Counter	Number of connections that have been held by this partition that this partition closed.
OutboundConnects	none	Yes	Counter	Number of connections that have been held by this partition that this partition opened.
PacketPoolGrows	none	Yes	Counter	Number of times iPlanet UDS expanded the space required for packets (packet pool).
PacketPoolShrinks	none	Yes	Counter	Number of times iPlanet UDS reduced the space required for packets (packet pool).
PacketsReceived	none	Yes	Counter	Number of iPlanet UDS packets received by this partition
PacketsSent	none	Yes	Counter	Number of iPlanet UDS packets sent by this partition.
Recvs	none	Yes	Counter	Number of times the Communication Manager read data from the network.
Sends	none	Yes	Counter	Number of times the Communication Manager sent data over the network.
SendToLocationCalls	none	Yes	Counter	Number of times the partition tried to send a method message or event to a remote partition.

Using the CommMgr Agent

The CommMgr agent is an agent that manages the communications service for an active partition. The communications service provides access to low-level network connections in and out of a partition.

Parent and subagents The parent agent for the CommMgr agent is an Active Partition Agent. There are no subagents to the CommMgr agent.

Setting Keepalive Threshold Values

iPlanet UDS provides a keepalive feature that helps you and your applications to quickly detect network failures. This feature is very similar to TCP Keepalive, and it works in the following way:

1. The Communication Manager keeps track of how long a connection has been inactive.
2. If a connection has been inactive longer than a specified amount of time (the *keepalive cycle*), the Communication Manager pings the remote partition on this connection.
3. If the Communication Manager receives a reply from the remote partition, the Communication Manager allows the connection to continue and resets its record of how long the connection has been inactive.

   If the Communication Manager does not receive a reply from the remote partition within the specified amount of time (the *keepalive interval*), the Communication Manager either pings the remote partition again, or closes the connection. The number of times the Communication Manager pings a remote partition before it closes the connection is the *keepalive count*.
4. When the Communication Manager closes a connection, the Distributed Object Manager raises a DistributedAccessException object, which the application should handle to recover state information before failing over to another service.

The CommMgr agent has the following instruments that let you monitor and tune the iPlanet UDS keepalive feature:

KeepAliveCloses Represents the total number of connections that have been closed by keepalive processing in a partition.

KeepAliveCount Specifies the number of pings that the keepalive feature attempts before it closes the connection.

KeepAliveCycle Specifies the length of time, in seconds, that a connection can be inactive before performing keepalive processing. Setting this value to `0` turns off the keepalive feature.

KeepAliveInterval Specifies the interval, in seconds, after a ping message is sent, during which a reply is expected.

States

ONLINE

The communication service for the active partition is running. The communication service is a part of the runtime system, and cannot run unless an active partition is running.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

ActiveRecvTasks

The `ActiveRecvTasks` instrument represents the number of running tasks that can receive incoming communication messages (receiver pool). This Counter instrument is read only.

ActiveSendTasks

The `ActiveSendTasks` instrument represents the number of running tasks that can send outgoing communication messages (sender pool). This Counter instrument is read only.

AvgRecvPacketSize

The `AvgRecvPacketSize` instrument represents the average size of the packets that this partition has received during the life of this partition. This Counter instrument is read only.

The average packet size is the number of bytes received by this partition divided by the number of packets received by this partition (BytesReceived / PacketsReceived).

AvgSendPacketSize

The `AvgSendPacketSize` instrument represents the average size of the packets that this partition has sent during the life of this partition. This Counter instrument is read only.

The average packet size is the number of bytes sent by this partition divided by the number of packets sent by this partition (BytesSent / PacketsSent).

BlockedSends

The `BlockedSends` instrument represents the number of attempts to send outgoing communication messages that were returned because network machine or protocol resources were busy. iPlanet UDS tries to send these outgoing messages again later. This Counter instrument is read only.

BusyRecvTasks

The `BusyRecvTasks` instrument represents the number of tasks that are busy receiving incoming communication messages. This Counter instrument is read only.

BytesReceived

The `BytesReceived` instrument represents the number of bytes that the current partition has read from the network. This Counter instrument is read only.

BytesSent

The `BytesSent` instrument represents the number of bytes that the current partition has sent across the network. This Counter instrument is read only.

CommMgrDispatches

The `CommMgrDispatches` instrument represents the number of dispatchable internal Communication Manager objects that have reached the partition from the network. This Counter instrument is read only.

ConfiguredRcvTasks

The `ConfiguredRcvTasks` instrument specifies a target number of tasks that can receive incoming communication messages (receiver pool). The `ConfiguredRcvTasks` instrument is a Configuration instrument.

At times, the iPlanet UDS system might need to start more tasks than the specified number; however, the system returns to the target number when it is able.

ConfiguredRcvTasks *number_tasks*

Argument	Description
number_tasks	Number of tasks requested for the receiver pool.

The `number_tasks` argument in the `ConfiguredRcvTasks` command indicates the number of tasks to set for the receiver pool.

ConfiguredSendTasks

The `ConfiguredSendTasks` instrument specifies a target number of tasks that can send outgoing communication messages (sender pool). The ConfiguredSendTasks instrument is a Configuration instrument.

ConfiguredSendTasks *number_tasks*

Argument	Description
number_tasks	Number of tasks requested for the sender pool.

The `number_tasks` argument in the `ConfiguredSendTasks` command indicates the number of tasks to set for the sender pool.

CurrentConnections

The `CurrentConnections` instrument represents the number of connections currently held by this partition. This Counter instrument is read only.

CurrentPacketsAllocated

The `CurrentPacketsAllocated` instrument represents the number of iPlanet UDS network packets that are currently allocated from the main garbage-collected heap. This Counter instrument is read only.

This instrument, along with the `MaxPacketPoolSize` instrument, can indicate the amount of memory that is being used for communications between this partition and other partitions.

DistObjectMgrDispatches

The `DistObjectMgrDispatches` instrument represents the number of dispatchable internal objects that have reached the partition from the network. This Counter instrument is read only.

InboundCloses

The `InboundCloses` instrument represents the total number of connections that have been held by this partition and closed by other partitions. This Counter instrument is read only.

InboundConnects

The `InboundConnects` instrument represents the total number of connections that have been held by this partition and initiated by other partitions. This Counter instrument is read only.

KeepAliveCloses

The `KeepAliveCloses` instrument represents the total number of connections that have been closed by keepalive processing in a partition. This Counter instrument is read only.

KeepAliveCount

The `KeepAliveCount` instrument specifies the number of pings that the keepalive feature performs before it closes the connection. The `KeepAliveCount` instrument is a Configuration instrument.

KeepAliveCount *count*

Argument	Description
count	Number of pings that the keepalive processing attempts before closing the connection. The default value is `3`.

After the keepalive feature has pinged the remote partition the specified number of times and failed to get a response, the keepalive feature closes the connection.

To close unavailable connections more quickly, set this value to a lower number.

KeepAliveCycle

The `KeepAliveCycle` instrument specifies the length of time, in seconds, that a connection can be inactive before performing keepalive processing. The `KeepAliveCycle` instrument is a Configuration instrument.

Setting this value to `0` turns off the keepalive feature. This value must be a positive integer.

KeepAliveCycle *interval*

Argument	Description
interval	Length of time, in seconds, that a connection can be inactive before performing keepalive processing. Setting this value to `0` turns off the keepalive function. The default value is `0`.

After a connection has been inactive for the specified period of time, the keepalive feature starts pinging the remote partition, as specified by the `KeepAliveCount` and `KeepAliveTimer` instruments, to determine whether to close the connection or not.

To detect unavailable connections more quickly, set this value to a lower number.

KeepAliveInterval

The `KeepAliveInterval` instrument specifies the interval, in seconds, after a ping message is sent, during which a reply is expected. The `KeepAliveInterval` instrument is a Configuration instrument.

KeepAliveInterval *interval*

Argument	Description
interval	Interval, in seconds, during which a ping message is expected to reply. The default value is `10`.

If the Communication Manager does not receive a reply to its ping before the specified interval elapses, it does one of the following:

- If the number of pings attempted is equal to the value of the `KeepAliveCount` instrument, the Communication Manager closes the connection.

  The Distributed Object Manager raises a DistributedAccessException object, which the application should handle to recover state information before failing over to another service.

- If the number of pings attempted is less than the value of the `KeepAliveCount` instrument, the Communication Manager attempts to ping the remote partition again.

For example, if the keepalive feature has pinged the remote partition four times without receiving a reply and the `KeepAliveCount` is `4`, then the Communication Manager closes the connection.

To detect unavailable connections more quickly, set this value to a lower number.

MaxPacketPoolSize

The `MaxPacketPoolSize` instrument represents the largest number of packets that have been in the free pool of iPlanet UDS network packets before shrinking or deallocating memory back to the main garbage-collected heap. This Counter instrument is read only.

This instrument, along with the `CurrentPacketsAllocated` instrument, can indicate the amount of memory that is being used for communications between this partition and other partitions.

MaxRecvQueueDepth

The `MaxRecvQueueDepth` instrument represents the peak number of iPlanet UDS network packets that have been received and not immediately processed. This Counter instrument is read only.

This instrument can indicate that the network is delivering network packets faster than the partition can immediately process them.

If the value of this instrument seems high, you might be able to improve performance by increasing the value of the `ConfiguredRcvTasks` instruments. The higher value will use more memory, but the performance increase for big applications can be worth the additional memory.

MaxSendQueueDepth

The `MaxSendQueueDepth` instrument represents the peak number of iPlanet UDS network packets that have been assembled but not immediately sent. This Counter instrument is read only.

This instrument can indicate that the partition is preparing network packets faster than the network can accept them.

If the value of this instrument seems high, you might be able to improve performance by increasing the value of the `ConfiguredSendTasks` instruments. The higher value will use more memory, but the performance increase for big applications can be worth the additional memory.

OutboundCloses

The `OutboundCloses` instrument represents the number of connections that have been held and closed by this partition. This Counter instrument is read only.

OutboundConnects

The `OutboundConnects` instrument represents the number of connections that have been held and opened by this partition. This Counter instrument is read only.

PacketPoolGrows

The `PacketPoolGrows` instrument represents the number of times iPlanet UDS expanded the space required for packets (packet pool). This Counter instrument is read only.

PacketPoolShrinks

The instrument represents the number of times iPlanet UDS reduced the space required for packets (packet pool). This Counter instrument is read only.

PacketsReceived

The `PacketsReceived` instrument represents the number of iPlanet UDS packets received by this partition. This Counter instrument is read only.

An iPlanet UDS packet can contain one object or a partial object, depending on the sizes of the objects being sent and the way the data is serialized for network transfer.

PacketsSent

The `PacketsSent` instrument represents the number of iPlanet UDS packets sent by this partition. This Counter instrument is read only.

An iPlanet UDS packet can contain one object or a partial object, depending on the sizes of the objects being sent and the way the data is serialized for network transfer.

Recvs

The `Recvs` instrument represents the number of times the Communication Manager read data from the network. This Counter instrument is read only.

Sends

The `Sends` instrument represents the number of times the Communication Manager sent data over the network. This Counter instrument is read only.

SendToLocationCalls

The `SendToLocationCalls` instrument represents the number of times the partition tried to send a method message or event to a remote partition. This Counter instrument is read only.

DBSession Agent

Parent Agent

Active Partition agent

Subagents

None

GenericDBMS class

DBSessionAgent

States

State	Description
ONLINE	The database session is running.

Command Summary

Command	Arguments	Description
DumpStatus	*no_propagate*	Prints the status of the managed object to Stdout.
PrintStmtQueue	none	Displays the prepared SQL statements.
Reconnect	none	Reconnects the database session.
Shutdown	none	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
CloseCursorError	none	Yes	Counter	Number of close cursor attempts that failed.
CloseCursorSuccess	none	Yes	Counter	Number of times a cursor was closed.
CloseExtentError	none	Yes	Counter	Number of close cursor extent attempts that failed.
CloseExtentSuccess	none	Yes	Counter	Successful executions of close cursor extent.
DeleteStmtExecuted	none	Yes	Counter	Successful executions of Delete statements.
DescribeTableDone	none	Yes	Counter	Describe database tables executed.

Instrument	Argument	Read Only?	Type	Description
DynamicStmtNotPrepare	none	Yes	Counter	Dynamic SQL statements not prepared due to errors.
DynamicStmtPrepare	none	Yes	Counter	Dynamic SQL statements successfully prepared.
ErrorExecuteDeleteStmt	none	Yes	Counter	Failures of delete statements.
ErrorExecuteInsertStmt	none	Yes	Counter	Failures of insert statements.
ErrorExecuteStmt	none	Yes	Counter	Failures while processing any types of execute statements.
ErrorExecuteUpdateStmt	none	Yes	Counter	Failures of update statements.
ExecuteImmediateError	none	Yes	Counter	Failures of execute immediate statements.
ExecuteImmediateSuccess	none	Yes	Counter	Successful executions of execute immediate statements.
ExtendCursorError	none	Yes	Counter	Number of extend cursor attempts that failed.
ExtendCursorSuccess	none	Yes	Counter	Successful executions of extend cursor.
FetchCursorError	none	Yes	Counter	Failures of fetch cursor statements.
FetchCursorSuccess	none	Yes	Counter	Successful executions of fetch cursor.
GetDatabaseDescError	none	Yes	Counter	Times that `GetDatabaseDesc` method encountered errors
GetDatabaseDescSuccess	none	Yes	Counter	Successful executions of `GetDatabaseDesc` method.

Instrument	Argument	Read Only?	Type	Description
GetResultValuesError	none	Yes	Counter	Times that an internal method encountered errors.
GetResultValuesSuccess	none	Yes	Counter	Successful executions of an internal method.
GetTableListError	none	Yes	Counter	Times that `GetTableList` method encountered errors.
GetTableListSuccess	none	Yes	Counter	Successful executions of `GetTableList` method.
InsertStmtExecuted	none	Yes	Counter	Insert statements successfully executed.
IsConnected	none	Yes	Configuration	Whether the database session is connected.
NumberOfAbort	none	Yes	Counter	Database transactions aborted.
NumberOfCommit	none	Yes	Counter	Database transactions committed.
NumberOfCursorsBuffered	none	Yes	Counter	Cumulative number of cursors buffered.
NumberOfErrorsBuilt	none	Yes	Counter	Exceptions raised within the database session.
NumberOfExplicitTransaction	none	Yes	Counter	Explicit database transactions.
NumberOfGetInputValues	none	Yes	Counter	Invocations of an internal method.
NumberOfGetMaxDataRows	none	Yes	Counter	Invocations of an internal method.
NumberOfGetOutputParams	none	Yes	Counter	Invocations of an internal method.
NumberOfImplicitTransaction	none	Yes	Counter	Implicit database transactions.

Instrument	Argument	Read Only?	Type	Description
NumberOfPrepareToCommit	none	Yes	Counter	Times that iPlanet UDS tried to synchronize a transaction before committing.
NumberOfRowsBuffered	none	Yes	Counter	Total number of rows buffered.
NumberOfRowsFetched	none	Yes	Counter	Total number of rows fetched.
OpenCursorEphemeralError	none	Yes	Counter	Errors encountered while opening reentrant cursors.
OpenCursorEphemeralSuccess	none	Yes	Counter	Successful opens of reentrant cursors.
OpenCursorProcedureError	none	Yes	Counter	Failures to open a cursor for a stored procedure.
OpenCursorProcedureSuccess	none	Yes	Counter	Successful executions of open cursor for a stored procedure.
OpenCursorSelectError	none	Yes	Counter	Failures to select data using a cursor.
OpenCursorSelectSuccess	none	Yes	Counter	Successful executions of select using a cursor.
RemoveStatementError	none	Yes	Counter	Failures while removing a prepared SQL statement.
RemoveStatementSuccess	none	Yes	Counter	Successful removal of a prepared SQL statement.
RemoveTOOLStatementError	none	Yes	Counter	Failures to remove statement of a prepared SQL statement on behalf of TOOL.
RemoveTOOLStatementSuccess	none	Yes	Counter	TImes that prepared SQL statements were removed on behalf of TOOL.

Instrument	Argument	Read Only?	Type	Description
RowsAffectedThroughExecute	none	Yes	Counter	Rows changed by any SQL statement.
RowsInserted	none	Yes	Counter	Total number of rows inserted.
SelectError	none	Yes	Counter	Number of times errors were encountered while executing a select statement.
SelectRowCount	none	Yes	Counter	Total number of rows selected.
SelectSuccess	none	Yes	Counter	Number of select statements that succeeded.
SetResultSizeError	none	Yes	Counter	Times that the `SetResultSizes` method encountered errors.
SetResultSizeSuccess	none	Yes	Counter	Successful executions of `SetResultSizes` method.
StmtExecuted	none	Yes	Counter	SQL statements successfully executed.
TOOLCloseCursorError	none	Yes	Counter	Failures of TOOL `sql close cursor` statements.
TOOLCloseCursorSuccess	none	Yes	Counter	Successful executions of TOOL `sql close cursor` statements.
TOOLFetchCursorError	none	Yes	Counter	Failures of TOOL `sql fetch cursor` statements.
TOOLFetchCursorSuccess	none	Yes	Counter	Successful executions of TOOL `sql fetch cursor` statements.
TOOLSelectError	none	Yes	Counter	Failures of TOOL `sql select` statements.

Instrument	Argument	Read Only?	Type	Description
TOOLSelectSuccess	none	Yes	Counter	Successful executions of TOOL `sql select` statements.
TOOLStmtNotPrepare	none	Yes	Counter	Times that TOOL SQL statements were not prepared due to errors.
TOOLStmtPrepare	none	Yes	Counter	Times that TOOL SQL statements were successfully prepared.
UpdateStmtExecuted	none	Yes	Counter	Successful executions of `sql update` statements.

Using the DBSession Agent

The DBSession agent is an agent that manages the database session (DBSession) for an active partition that accesses a database using a DBSession object.

Parent and subagents The parent agent for the DBSession agent is an Active Partition agent. There are no subagents to the DBSession agent.

The DBSessionAgent supports a number of instruments, all of which are read only. Each instrument indicates the number of conditions that have occurred thus far in the current database session. You can use multiple instruments to calculate useful information. For example, to find the total number of transactions thus far, you would add the `NumberOfExplicitTransaction` instrument and the `NumberOfImplicitTransaction` instrument.

States

ONLINE

The database session (DB session) for the active partition is running.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to standard output.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

PrintStmtQueue

The `PrintStmtQueue` command prints the prepared SQL statements to standard output.

PrintStmtQueue

Reconnect

The `Reconnect` command reconnects the database session to the database. You can use this command when the database session has disconnected from the database because of a network failure or other problem.

Reconnect

You cannot use this command if your database session has disconnected from the database because the `DBSession.Disconnect` method was invoked.

Instruments

CloseCursorError

The `CloseCursorError` instrument indicates the number of times an attempt to close a cursor has failed since this database session started. This Counter instrument is read only.

For information about the cause of these failures, check the exceptions returned to the application that tried to close the cursor.

CloseCursorSuccess

The `CloseCursorSuccess` instrument indicates the number of times a cursor associated with this database session was closed without errors. This Counter instrument is read only.

CloseExtentError

The `CloseExtentError` instrument indicates the number of times an attempt to close a result set has failed since this database session started. This Counter instrument is read only.

For information about the cause of these failures, check the exceptions returned to the application that tried to close the result set.

CloseExtentSuccess

The `CloseExtentSuccess` instrument indicates the number of times a result set was closed without errors. This Counter instrument is read only.

DeleteStmtExecuted

The `DeleteStmtExecuted` instrument indicates the number of times data was successfully deleted using the `sql delete` statement since this database session started. This Counter instrument is read only.

DescribeTableDone

The `DescribeTableDone` instrument indicates the number of times the `DBSession.DescribeTable` method was invoked since this database session started. This Counter instrument is read only.

DynamicStmtNotPrepare

The `DynamicStmtNotPrepare` instrument indicates the number of times, since the database session started, that a dynamic SQL statement has not been prepared because of errors. An application tried to prepare the SQL statement using the `DBSession.Prepare` or `DBSession.PreparePositioned` methods. This Counter instrument is read only.

For information about the cause of the errors, check the exceptions returned to the application that tried to prepare the SQL statement.

DynamicStmtPrepare

The `DynamicStmtPrepare` instrument indicates the number of times, since the database session started, that a dynamic SQL statement has been prepared using the `DBSession.Prepare` or `DBSession.PreparePositioned` methods. This Counter instrument is read only.

ErrorExecuteDeleteStmt

The `ErrorExecuteDeleteStmt` instrument indicates the number of times `sql delete` statements failed because of errors since this database session started. This Counter instrument is read only.

For information about the cause of the errors, check the exceptions returned to the application that tried to execute the `sql delete` statement.

ErrorExecuteInsertStmt

The `ErrorExecuteInsertStmt` instrument indicates the number of times `sql insert` statements failed because of errors since this database session started. This Counter instrument is read only.

For information about the cause of these failures, check the exceptions returned to the application that tried to execute the `sql insert` statement.

ErrorExecuteStmt

The `ErrorExecuteStmt` instrument indicates the number of time an attempt to execute a SQL statement failed since this database session started. This Counter instrument is read only.

For information about the cause of these failures, check the exceptions returned to the application that tried to execute a SQL statement.

ErrorExecuteUpdateStmt

The `ErrorExecuteUpdateStmt` instrument indicates the number of times `sql update` statements failed because of errors since this database session started. This Counter instrument is read only.

For information about the cause of these failures, check the exceptions returned to the application that tried to execute the `sql update` statement.

ExecuteImmediateError

The `ExecuteImmediateError` instrument indicates the number of times since this database session started that `sql execute immediate` statements or invocations of the `DBSession.ExecuteImmediate` method failed. This Counter instrument is read only.

For information about the cause of these failures, check the exceptions returned to the application that tried to execute the `sql execute immediate` statement or invoke the `DBSession.ExecuteImmediate` method.

ExecuteImmediateSuccess

The `ExecuteImmediateSuccess` instrument indicates the number of times since this database session started that `sql execute immediate` statements and `DBSession.ExecuteImmediate` invocations were successful. This Counter instrument is read only.

ExtendCursorError

The `ExtendCursorError` instrument indicates the number of times a new result set could not be opened by invoking the `DBSession.ExtendCursor` method because of errors. This Counter instrument is read only.

For information about the cause of the errors, check the exceptions returned to the application that invoked the `DBSession.ExtendCursor` method.

ExtendCursorSuccess

The `ExtendCursorSuccess` instrument indicates the number of times a new result set was opened by invoking the `DBSession.ExtendCursor` method since this database session started. This Counter instrument is read only.

FetchCursorError

The `FetchCursorError` instrument indicates the number of times an attempt to fetch data failed since this database session started. This Counter instrument is read only.

For information about the cause of these failures, check the exceptions returned to the application that tried to execute the `sql fetch cursor` statement or invoke the `DBSession.FetchCursor` method.

FetchCursorSuccess

The `FetchCursorSuccess` instrument indicates the number of times data was fetched using the `sql fetch` cursor statement or `DBSession.FetchCursor` method. This Counter instrument is read only.

GetDatabaseDescError

The `GetDatabaseDescError` instrument indicates the number of times errors occurred when the `DBSession.GetDatabaseDesc` method was invoked. This Counter instrument is read only.

For information about the cause of the errors, check the exceptions returned to the application that invoked the DBSession.GetDatabaseDesc method.

GetDatabaseDescSuccess

The `GetDatabaseDescSuccess` instrument indicates the number of times the `DBSession.GetDatabaseDesc` method was invoked and completed without errors. This Counter instrument is read only.

GetResultValuesError

The `GetResultValuesError` instrument indicates the number of times results could not be retrieved while fetching data because of errors.

For information about the cause of the errors, check the exceptions returned to the application that tried to fetch the data.

GetResultValuesSuccess

The `GetResultValuesSuccess` instrument indicates the number of times results were successfully retrieved while fetching data. This Counter instrument is read only.

GetTableListError

The `GetTableListError` instrument indicates the number of times the `DBSession.GetTableList` method encountered errors. This Counter instrument is read only.

For information about the cause of the errors, check the exceptions returned to the application that invoked the `DBSession.GetTableList` method.

GetTableListSuccess

The `GetTableListSuccess` instrument indicates the number of times `DBSession.GetTableListSuccess` method was invoked and completed without errors. This Counter instrument is read only.

InsertStmtExecuted

The `InsertStmtExecuted` instrument indicates the number of times `sql insert` statements were executed without errors. This Counter instrument is read only.

IsConnected

The `IsConnected` instrument indicates whether the database session is connected to the database. Possible values for this instrument are `TRUE` or `FALSE`. This Configuration instrument is read only.

If this value is false because of a network or database failure, you can try to reconnect the database session to the database using the `Reconnect` command, described in "Reconnect" on page 181.

NumberOfAbort

The `NumberOfAbort` instrument indicates the number of times database transactions were aborted. This Counter instrument is read only.

For information about the why the database transactions were aborted, check the exceptions returned to the application.

NumberOfCommit

The `NumberOfCommit` instrument indicates the number of database transactions that have committed. This Counter instrument is read only.

NumberOfCursorsBuffered

The `NumberOfCursorsBuffered` instrument indicates the number of times buffers have been allocated for `sql select` statements executed as implicit transactions. This Counter instrument is read only.

If you find that you are running low on memory, and the value of this instrument is high, you should consider replacing some of the implicit transactions in your application with explicit transactions.

NumberOfErrorsBuilt

The `NumbersOfErrorsBuilt` instrument indicates the number of iPlanet UDS exceptions that have been raised since the database session started. The exceptions counted by this instrument include exceptions that have been handled by the database session, as well as exceptions that have been raised by the database session itself. This Counter instrument is read only.

NumberOfExplicitTransaction

The `NumberOfExplicitTransaction` instrument indicates the number of explicit iPlanet UDS transactions that used this database session. Explicit transactions are enclosed by the TOOL `begin transaction` and `end transaction` statements. This Counter instrument is read only.

NumberOfGetInputValues

The `NumberOfGetInputValues` instrument indicates the number of times an internal method was invoked when a variable for a SQL statement is being processed. This Counter instrument is read only.

NumberOfGetMaxDataRows

The `NumberOfGetMaxDataRows` instrument indicates the number of times an internal method was invoked as the result of an `sql select` statement. This Counter instrument is read only.

NumberOfGetOutputParams

The `NumberOfGetOutputParams` instrument indicates the number of times an internal method was invoked when procedures were executed. This Counter instrument is read only.

NumberOfImplicitTransaction

The `NumberOfImplicitTransaction` instrument indicates the number of implicit database transactions that used this database session. Implicit transactions occur on individual SQL statements that are executed outside of TOOL `begin transaction` and `end transaction` statements. This Counter instrument is read only.

NumberOfPrepareToCommit

The `NumberOfPrepareToCommit` instrument indicates the number of times the iPlanet UDS runtime system tried to synchronize elements of a transaction before committing a transaction. This Counter instrument is read only.

NumberOfRowsBuffered

The `NumberOfRowsBuffered` instrument indicates the total number of rows that have been buffered since the database session started. This Counter instrument is read only.

NumberOfRowsFetched

The `NumberOfRowsFetched` instrument indicates the total number of rows that have been fetched since the database session started. This Counter instrument is read only.

OpenCursorEphemeralError

The `OpenCursorEphemeralError` instrument indicates the number of times that an attempt to open a reentrant cursor failed. This Counter instrument is read only.

These reentrant cursors are used by the iPlanet UDS runtime system to retrieve intermediate results when select statements are imbedded within other select statements. These cursors are included in the count of open cursors.

OpenCursorEphemeralSuccess

The `OpenCursorEphemeralSuccess` instrument indicates the number of times this database session successfully opened reentrant cursors. This Counter instrument is read only.

These reentrant cursors are used by the iPlanet UDS runtime system to retrieve intermediate results when select statements are imbedded within other select statements. These cursors are included in the count of open cursors.

OpenCursorProcedureError

The `OpenCursorProcedureError` instrument indicates the number of times the database session failed when trying to open a cursor for a stored procedure. This Counter instrument is read only.

For information about the cause of the failure, check the exceptions returned to the application that attempted to open the cursor.

OpenCursorProcedureSuccess

The `OpenCursorProcedureSuccess` instrument indicates the number of times the database session successfully opened a cursor for a stored procedure. This Counter instrument is read only.

OpenCursorSelectError

The `OpenCursorSelectError` instrument indicates the number of times that an attempt to open a cursor for a `sql select` statement failed. This Counter instrument is read only.

For information about the cause of the failure, check the exceptions returned to the application that attempted to open the cursor.

OpenCursorSelectSuccess

The `OpenCursorSelectSuccess` instrument indicates the number of times that a cursor was successfully opened for a `sql select` statement. This Counter instrument is read only.

RemoveStatementError

The `RemoveStatementError` instrument indicates the number of times an attempt to remove a prepared SQL statement failed. This Counter instrument is read only.

For information about the cause of the failure, check the exceptions returned to the application.

RemoveStatementSuccess

The `RemoveStatementSuccess` instrument indicates the number of times a prepared SQL statement was removed. This Counter instrument is read only.

RemoveTOOLStatementError

The `RemoveTOOLStatementError` instrument indicates the number of times an attempt to remove a prepared SQL statement failed. This Counter instrument is read only.

The iPlanet UDS runtime system attempted to remove a prepared SQL statement on behalf of a TOOL SQL statement because the system needs to recover some memory used to store prepared SQL statements.

RemoveTOOLStatementSuccess

The `RemoveTOOLStatementSuccess` instrument indicates the number of times the iPlanet UDS runtime system removed a prepared SQL statement on behalf of a TOOL SQL statement because the memory allocated for prepared statements is full. This Counter instrument is read only.

RowsAffectedThroughExecute

The `RowsAffectedThroughExecute` instrument contains a running count of database rows that have been altered by insert, update, or delete SQL statements or procedures. This Counter instrument is read only.

RowsInserted

The `RowsInserted` instrument indicates the number of database rows that have been inserted during this database session. This Counter instrument is read only.

SelectError

The `SelectError` instrument indicates the number of times an attempt to execute a select statement failed. This instrument counts failures that occurred when using either a `sql select` statement or the `DBSession.Select` method. This Counter instrument is read only.

SelectRowCount

The `SelectRowCount` instrument indicates the total number of rows that have been selected during this database session. This Counter instrument is read only.

SelectSuccess

The `SelectSuccess` instrument indicates the number of times `sql select` statements executed without errors. This Counter instrument is read only.

SetResultSizeError

The `SetResultSizeError` instrument indicates the number of times errors occurred when the `DBSession.SetResultSizes` method was invoked. This Counter instrument is read only.

For information about the cause of the errors, check the exceptions returned to the application that invoked the `DBSession.SetResultSizes` method.

SetResultSizeSuccess

The `SetResultSizeSuccess` instrument indicates the number of times the `DBSession.SetResultSizes` method completed without errors. This Counter instrument is read only.

StmtExecuted

The `StmtExecuted` instrument indicates the total number of SQL statements that have been successfully executed during the database session. This Counter instrument is read only.

TOOLCloseCursorError

The `TOOLCloseCursorError` instrument indicates the number of times the TOOL `sql close cursor` statement failed. This Counter instrument is read only.

For information about the cause of the failure, check the exceptions returned to the application that used the `sql close cursor` statement.

TOOLCloseCursorSuccess

The `TOOLCloseCursorSuccess` instrument indicates the number of times the TOOL `sql close cursor` statement completed without errors. This Counter instrument is read only.

TOOLFetchCursorError

The `TOOLFetchCursorError` instrument indicates the number of times the TOOL `sql fetch cursor` statement failed. This Counter instrument is read only.

For information about the cause of the failure, check the exceptions returned to the application that used the `sql fetch cursor` statement.

TOOLFetchCursorSuccess

The `TOOLFetchCursorSuccess` instrument indicates the number of times the TOOL `sql fetch cursor` statement completed without errors. This Counter instrument is read only.

TOOLSelectError

The `TOOLSelectError` instrument indicates the number of times the TOOL `sql select` statement failed. This Counter instrument is read only.

For information about the cause of the failure, check the exceptions returned to the application that used the `sql select` statement.

TOOLSelectSuccess

The `TOOLSelectSuccess` instrument indicates the number of times the TOOL `sql select` statement completed without errors. This Counter instrument is read only.

TOOLStmtNotPrepare

The `TOOLStmtNotPrepare` instrument indicates the number of times iPlanet UDS could not prepare a TOOL SQL statement because of errors. This Counter instrument is read only.

For information about the cause of the errors, check the exceptions returned to the application that used the TOOL SQL statement.

TOOLStmtPrepare

The `TOOLStmtPrepare` instrument indicates the number of times iPlanet UDS prepared a TOOL SQL statement without errors. This Counter instrument is read only.

UpdateStmtExecuted

The `UpdateStmtExecuted` instrument indicates the number of times `sql update` statements were executed successfully since this database session started. This Counter instrument is read only.

DistObjectMgr Agent

Parent Agent

Active Partition or RepositoryServer agent

Subagents

None

SystemMonitor Class

DistObjectMgrAgent

States

State	Description
ONLINE	The distributed object manager is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
EventsReceived	none	Yes	Counter	Number of remote events received from this partition since it started.
EventsSent	none	Yes	Counter	Number of remote events sent from this partition since it started.
MessagesReceived	none	Yes	Counter	Number of methods in this partition that have been invoked from other partitions since it started.
MessagesSent	none	Yes	Counter	Number of times objects in this partition have invoked methods on remote objects.

Using the DistObjectMgr Agent

The DistObjectMgr agent is an agent that manages the distributed object services for an active partition. The distributed object services provide access to logical communications into and out of a partition.

Parent and subagents The parent agent for the DistObjectMgr agent is an Active Partition agent, as it runs in all active partitions. There are no subagents to the DistObjectMgr agent.

The DistObjectMgr agent has no defined commands.

States

ONLINE

The distributed object services for the active partition are running. The distributed object services are part of the runtime system, and cannot run unless an active partition is running.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

EventsReceived

The `EventsReceived` instrument represents the number of remote events received from this partition since it started. This includes system management events (such as LogTimer events carrying instrument values). This Counter instrument is read only.

EventsSent

The `EventsSent` instrument represents the number of remote events sent from this partition since it started. This Counter instrument is read only.

MessagesReceived

The `MessagesReceived` instrument represents the number of methods in this partition that have been invoked by other partitions since this partition started. This number includes system management messages (such as those sent between system management agents). This Counter instrument is read only.

MessagesSent

The `MessagesSent` instrument represents the number of times objects in this partition have invoked methods on remote objects. This number includes any system management messages. This Counter instrument is read only.

Environment Agent

Parent Agent

None

Subagents

Node agent, Application agent, and Name Service agent

SystemMonitor Class

EnvironmentAgent

States

State	Description
ONLINE	The Environment Manager is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
ExportEnv	*file_name* *environment_name*	File > Export Environment	Exports an environment definition into a specified file.
FindEnv	*env_name* *is_updateable*	none	Selects the specified environment definition as the current environment, and puts Escript in environment editing mode.
GenerateAlert	*subject_text* *message_text*	Utility	Sends an alert message to the Environment agent.
ImportEnv	*file_name*	File > Import Environment	Imports the definition of an environment from the specified file.
ListAppConfig	none	none	Displays the list of applications currently being partitioned in the active environment.
ListDistribs	none	File > Load Distribution	Lists the application distributions available locally on the node running Escript.
ListEnvs	none	File > Open	Lists the names of the environments in the Environment Manager repository.

Command	Arguments	Environment Console menu	Description
LoadDistrib	*application_name compatibility_level*	File > Load Distribution	Loads the specified application distribution into the environment repository from the node that is running the Escript utility.
NewEnv	*environment_name*	File > New	Creates a new simulated environment definition with the specified name.
ReleaseAppConfig	*client_id*	none	Forces a release of the configuration lock for the named application being configured in the named environment.
RemoveEnv	*env_name*	none	Removes a simulated environment definition from the repository.
Shutdown	*no_propagate*	Component	Shuts down the Environment Manager and its agent.
ShutdownSubAgent	*subagent*	none	Shuts down the named subagent and its managed object.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
EnvironmentLog	*log_file_name*	No	Configuration	Name of the file to use when logging important events for the Environment Manager.
InstrumentLogging	*is_active*	No	Configuration	Turns on or off the automatic logging of instruments to the environment log file.

Instrument	Argument	Read Only?	Type	Description
InstallTaskCount	none	Yes	Configuration	Indicates the number of tasks devoted to installing applications.

Using the Environment Agent

The Environment agent manages the environment as a whole (and is represented by the Environment Manager service).

The Environment agent is the topmost agent in the system management agent hierarchy. To navigate to the Environment agent, you can invoke the following command:

```
escript> FindActEnv
```

Parents and subagents The Environment agent is the parent of two main types of subagents: Node agents and Application Agents. These agents represent, respectively, the various nodes within the environment, and the various applications loaded or installed in the environment. There is also a NameService agent, which manages the Name Service, and a Library Configuration agent, which represents a loaded library configuration.

States

ONLINE

The Environment Manager that uses the name service address (`-fns` flag or FORTE_NS_ADDRESS environment variable setting) is running. If no Environment Manager is running that uses that name service address, then you cannot access the agent hierarchy.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

ExportEnv

The `ExportEnv` command exports the environment definition for the active environment into a specified file.

ExportEnv [*file_name*]

Argument	Description
file_name	The name of a file in which to write the definition of the environment. See below for default value.

The `ExportEnv` command stores the definition of an environment in a file. This command can be used to transfer an environment definition to another iPlanet UDS environment for use as a simulated environment. The exported environment definition can be subsequently loaded with the `ImportEnv` command.

Recovering an environment repository You can also use the exported file to recover an environment repository by using the `nodemgr -e` command, specifying the file name with the `-b` flag, as shown in the following example:

```
nodemgr -e -b c:\forte\envdist\docenv.edf
```

The `ExportEnv` command automatically refreshes the definition of the environment before executing, so you do not need to invoke the `RefreshEnv` command first.

The `file_name` argument specifies the name of the file into which to write the definition. You should use iPlanet UDS portable file name syntax if you have invoked the `UsePortable` command and use native file name syntax if you have invoked the `UseLocal` command or have not invoked either `UsePortable` or `UseLocal`. If no file_name is specified, `FORTE_ROOT/envdist/`*envname*`.edf` is used. Only the first 8 characters in the environment name is used.

The exported file is in an internal format, but is portable across all the supported platforms. This exported file contains information about:

- the name and UUID of the active environment
- node definitions
- installed applications and libraries
- environments to which the active environment is connected

FindEnv

The `FindEnv` command designates the specified environment definition in the environment repository or Environment Manager as the current environment, and puts Escript in the environment editing mode.

FindEnv *env_name* [*is_updateable*]

Argument	**Description**
env_name	The name of a simulated environment definition.
is_updateable	A flag that indicates whether or not you are modifying part of the definition of the environment. Set to `1` if you need to change the definition, or `0` if you do not need to change it. Default is `0`.

The `FindEnv` command lets you view and change the definitions of a simulated environment. When you invoke the `FindEnv` command, you are automatically placed in the environment editing mode of `Escript`. For more information on the environment editing mode of `Escript`, see "Configuring Environment Definitions" on page 37. The prompt changes to "`envedit`," and all commands operate on the designated simulated environment definition until you invoke the `Exit` or `Quit` command to return to the Escript agent commands (in the Environment agent).

The `env_name` argument specifies the name of a simulated environment that had previously been created and saved as part of this environment, through a previous use of the `NewEnv` or `ImportEnv` commands.

The `is_updateable` argument specifies whether or not the environment definition is editable. By default, even though you are placed in the environment editing mode of `Escript`, you are only able to view the information about the simulated environment definition. However, if you need to change the information in the simulated environment, you can set the `is_updateable` argument to a value of `1`. Only one Escript or Environment Console session in an environment may modify the definitions of a simulated environment at a given time.

Setting up a simulated environment definition You can set up simulated environment definitions at the Environment agent by invoking the `NewEnv` command, followed by a series of `AddNode` and other environment definition commands. A more common approach is to export an active environment definition at the off-site location by invoking an `ExportEnv` command, transferring the exported file to the environment where you want to do the simulations, and then invoking an `ImportEnv` command to create the new environment definition. You can then map the testing nodes using the `SetEnvNodeForSim` command. (The active environment export file can be quite large and contain a lot of extraneous application information.)

GenerateAlert

The `GenerateAlert` command sends an alert message to the Environment agent.

GenerateAlert *subject_text message_text*

Argument	**Description**
subject_text	A text string that describes the subject of the message.
message_text	A text string which is the text of the alert.

The `GenerateAlert` command sends an alert message to the Environment agent. The alert is then processed in the following ways:

- the alert is written to the environment's log file
- if anyone is using the Environment Console interface, and has enabled the alert window, the alert is displayed
- if any TOOL program is executing and has registered for the AlertFromSystem event in an active event loop, the event is posted to the TOOL code

The `subject_text` and `message_text` arguments are arbitrary text strings that you can use in any way that you wish. If either is to contain embedded spaces, you should surround the text with quotes.

An example use of the `GenerateAlert` command is:

```
escript> GenerateAlert Warning “Shut down in 10 minutes”
```

ImportEnv

The `ImportEnv` command imports the definition of an environment from the specified file.

ImportEnv *file_name*

Argument	Description
file_name	The name of a file created by the `ExportEnv` command.

If the environment represented in the file does not yet exist in the current repository, the environment is created. If an environment by the same time already exists in the repository, you must delete the existing environment definition before importing the new one.

The `ExportEnv` command stores the definition of an environment in a file. You can use this file to transfer the definition of an environment to another iPlanet UDS environment for use as a simulated environment. The exported environment definition can be subsequently loaded with the `ImportEnv` command.

The `file_name` argument is the name of a file that was previously written using the `ExportEnv` command. If you have previously invoked the `UsePortable` command, then give the file name in iPlanet UDS portable file name syntax. If you have previously invoked a `UseLocal`, or have invoked neither of the commands, then use local naming syntax. If no path is given for the name, it is assumed to be in the `FORTE_ROOT/envdist` directory.

The current environment cannot be locked when you invoke the `ImportEnv` command.

ListAppConfig

The `ListAppConfig` command displays the list of applications currently being partitioned in the current environment. This includes applications being configured for simulated environments. `ListAppConfig` is useful as a means to identify who is currently partitioning in the environment if you cannot lock the environments

ListAppConfig

ListDistribs

The `ListDistribs` command lists the application distributions available locally on the node running Escript. This checks for distribution directories starting in `FORTE_ROOT/appdist/`*env_name* on the local node.

ListDistribs

ListEnvs

The `ListEnvs` command lists the names of the simulated environment definitions in the Environment Manager repository. You can then invoke the `FindEnv` command to designate one as the environment definitions to edit using environment editing commands.

ListEnvs

LoadDistrib

The `LoadDistrib` command loads the specified application distribution into the environment repository from the node that is running the Escript utility.

LoadDistrib *application_name compatibility_level*

Argument	Description
application_name	The name of the application to load.
compatibility_level	The compatibility level of the application to load.

The steps for installing an application are described "Install" on page 146. You can use the `LoadDistrib` command to load an application into the environment repository from a local distribution on the node that is currently running Escript. The command automatically makes the loaded Application agent the current Escript agent. For detailed information on installing loaded applications, see "Install" on page 146.

You can also use the `LoadDistrib` command to load an updated version of an application into the environment repository, to be subsequently installed, for example, to fix a bug.

The active environment cannot be locked when you invoke the `LoadDistrib` command.

The application distribution to be loaded is to be found in the `FORTE_ROOT/appdist/`*env_name* directory hierarchy on the node that is running the Escript utility. You can also have any of the executing Node Managers load a distribution that is present on other nodes by invoking the `LoadDistrib` command.

The `application_name` argument specifies the name of the application to be loaded. The `compatibility_level` argument is the compatibility level number of the application to be loaded, prefixed with the letters "cl". These names are exactly as displayed by the `ListDistribs` command. These two arguments are used to find the application in the `appdist` directory tree. The first 8 characters of the `application_name` and the `compatibility_level` is used as the directory name. For example, you could invoke the following command out of environment named "dev" to load a distribution on the node currently running Escript:

```
escript> LoadDistrib MyFirstProject cl2
```

This would look for the application distribution in the following directory:

FORTE_ROOT/appdist/dev/myfirstp/cl2

NewEnv

The `NewEnv` command creates a new simulated environment definition with the specified name.

NewEnv *environment_name*

Argument	Description
environment_name	The name of the new simulated environment definition.

Setting up a simulated environment The `NewEnv` command creates a new environment definition for simulation, and places Escript in the environment editing mode. For more information on the environment editing mode of Escript, see "Configuring Environment Definitions" on page 37. The prompt changes to "envedit," and all commands operate on the new simulated environment definition until you invoke the `Exit` or `Quit` command to return to the Escript agent commands (in the Environment agent).

A more common approach to creating a simulated environment definition is to use the `ExportEnv` command at the remote environment to be simulated, which creates a portable file that can be transferred to other iPlanet UDS environments. You can then import this environment definition as a simulated environment using the `ImportEnv` command.

The `environment_name` argument is the name of the new simulated environment definition. An environment definition by this name cannot already exist in the environment repository.

The active environment cannot be locked when you invoke the `NewEnv` command.

ReleaseAppConfig

The `ReleaseAppConfig` command releases the configuration lock for the named application being configured in the named environment.

ReleaseAppConfig *client_id*

Argument	Description
client_id	The partition ID of the client whose locks you want to drop.

When any developer in the environment enters the Partition Workshop, or uses the Fscript commands related to partitioning, their session with the Environment Manager obtains a configuration lock on the environment. This lock is not exclusive, but does prevent users of Escript and the Environment Console from locking the environment and making changes, which might invalidate the partitioning that is being performed.

If the machines that are running the partitioning terminate abnormally, the configuration lock in the environment can be left locked, even though the partitioning session has terminated. The `ReleaseAppConfig` command can be used in these circumstances to force the Environment Manager to release an orphaned configuration lock.

CAUTION Be careful ***never*** to use this command if the partitioning session is still valid and active.

The `client_id` argument is the full partition ID of the client partition that was running the Partition Workshop and has left the application configuration lock in the repository. You can use the `ListAppConfigs` command to list the applications and environments that currently have configuration locks. The `client_id` values are listed as well.

RemoveEnv

The `RemoveEnv` command removes a simulated environment definition from the environment repository.

RemoveEnv *env_name*

Argument	Description
env_name	The name of the simulated environment that you want to move from the environment repository.

You specify the environment name using the `env_name` argument, which must be the name of an environment definition already in the repository.

You cannot remove the definition for the current active environment. To remove the current active environment, you must destroy the B-tree environment repository for the environment. The repository is located in the `FORTE_ROOT/repos` directory in the installation that first started the Environment Manager for the environment, and is contained in two files: "*env_name*.`btd`" and "*env_name*.`btx`" (the environment name is truncated to 8 characters). Once you remove the environment repository, simply re-starting the Environment Manager with the same environment name recreates the environment, as a new empty environment. You must, of course, repopulate the environment definition with nodes and installations as appropriate. Or you can invoke the `ImportEnv` command to re-import saved definitions.

Shutdown

The `Shutdown` command shuts down the Environment Manager and its agent.

Shutdown [*no_propagate*]

Argument	Description
no_propagate	This flag specifies whether to propagate the shutdown to the subagents of the Environment agent. The values are `0` to propagate (the default) or `1` to not propagate to subagents.

The `Shutdown` command shuts down the Environment Manager (Environment agent). By default, the `Shutdown` request is propagated to all of the subagents of the Environment Manager, which, of course, includes all iPlanet UDS managers and partitions running in the environment. The `Shutdown` command therefore gives a simple way to shutdown all iPlanet UDS processes running in an environment. Obviously, this command should be used carefully.

The optional `no_propagate` argument can be set to `1` if you want to shutdown the Environment Manager only (and its Node Manager), but leave the executing partitions and other Node Managers still executing. The Environment Manager can then be brought back online at a later time, and reconnects to the executing partitions. If you set `no_propagate` to `0`, or leave it unspecified, the `Shutdown` request propagates.

ShutdownSubAgent

The `ShutdownSubAgent` command shuts down the named subagent in the environment.

ShutdownSubAgent *subagent*

Argument	Description
subagent	The name of a subagent to be shut down with its managed object.

If the named subagent does not exist, the `ShutdownSubAgent` command does nothing.

The `ShutdownSubAgent` command performs the same function as the following command sequence in an Escript script:

```
escript> FindSubAgent AutoCompileSvc_cl0
escript> Shutdown
escript> FindParentAgent
```

The difference between this sequence of commands and using the `ShutdownSubAgent` command is that the `Shutdown` command is not invoked on the current agent if the subagent does not exist.

Instruments

EnvironmentLog

The `EnvironmentLog` instrument specifies the name of the file to use when logging important events for the Environment Manager and logging instrument information. The `EnvironmentLog` instrument is a Configuration instrument.

EnvironmentLog *log_file_name*

Argument	Description
log_file_name	Indicates the name of the file to use for logging the Environment Manager events and audit information.

The information that is logged to the environment log includes instrument logging events generated by the `LogTimer` instrument on the active partitions, as well as an audit trail of all important operations performed by the Environment Manager, such as starting Node Managers.

The `log_file_name` argument indicates the name of the log file to use for logging. This file name should be given in one of two ways: relative or absolute. In either case, however, use iPlanet UDS portable file name syntax (UNIX style). If a relative name is given for `log_file_name` (it does not start with a `/`), then the file is given relative to the `FORTE_ROOT/log` directory on the node on which the Environment Manager service is executing. If an absolute path is given in the `log_file_name`, it is an absolute path on the machine on which the Environment Manager service is executing.

If you change the logging file name after the Environment Manager has already started logging to another file, that file is closed, and the new file is opened.

The following example shows how you can set the log file in Escript:

```
escript> UpdateInstrument EnvironmentLog x:/vms/e.log
escript> UpdateInstrument EnvironmentLog /udir/hp/e.log
```

InstrumentLogging

The `InstrumentLogging` instrument sets the automatic logging of instruments to the environment log file. The `InstrumentLogging` instrument is a Configuration instrument.

InstrumentLogging [*is_active*]

Argument	Description
is_active	Indicates whether automatic logging of instrument logging is currently active. Set to `TRUE` to make the logging active or `FALSE` to make it inactive.

The `InstrumentLogging` instrument turns on the automatic logging of active instruments to the environment log file each time the timing interval for any `LogTimer` instrument in any active partition expires. By default, automatic logging into the environment is disabled. The `InstrumentLogging` instrument is used in conjunction with the `LogTimer` instrument, defined on the active partition. For more information on how the process of automatic logging works, see "LogTimer" on page 131.

The `is_active` argument is a boolean value. If `TRUE`, the values of the current set of instruments that are being logged in the Active Partition agent, or any of its subagents, are automatically logged to the environment's log file. If set to `FALSE`, no logging takes place. Note that even if you turn off the InstrumentLogging instrument, the detailed data is still collected, which could be a significant performance drain, so you should disable the LogTimer instrument as well for more efficiency.

InstallTaskCount

The `InstallTaskCount` instrument indicates the number of tasks in the environment that are devoted to installing applications. This Counter instrument is read only.

EventMgr Agent

Parent Agent

Active Partition or RepositoryServer agent

Subagents

None

SystemMonitor Class

EventMgrAgent

States

State	Description
ONLINE	The event manager is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Using the EventMgr Agent

The EventMgr agent manages the EventMgr for a particular active partition. The EventMgr manages receiving and delivering events for an active partition.

Parent and subagents The parent agent for an EventMgr agent is an Active Partition agent. An EventMgr agent has no subagents.

The EventMgr agent has no defined instruments.

States

ONLINE

The event manager for the active partition is running. The event manager is a part of the runtime system, and cannot run unless an active partition is running.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Installed Partition Agent

Parent Agent

Node agent and Partition agent

Subagents

Active Partition agent

SystemMonitor Class

GenericPartitionAgent

States

State	Description
DEGRADED	(Replicated partitions only) Fewer than the specified number of instances of this partition are running.
OFFLINE	No instances of this partition are running.
ONLINE	The minimum number of instances of this partition are running.

Command Summary

Command	**Arguments**	**Environment Console menu**	**Description**
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
ModLoggerRemote	+(*logger_flags*) -(*logger_flags*)	Component > Modify Log Flags	Sets the logger flags for all of the active partitions that are represented by this Installed Partition agent.
SetEnvRemote	*env_variable* *new_value*	Component	Sets the environment variable for all of the active instances of the installed partition managed by the current agent.
Shutdown	kill_executors	Boolean.	Shuts down all active instances of the installed partition represented by the current agent. If the `kill_executors` argument is set to `TRUE` or `1`, any interpreted partitions will shut down their hosting iPlanet UDS executor (ftexec or ftexecd) process as well.
ShutdownSubAgent	*subagent*	none	Shuts down the named subagent and its managed object.
Startup	*argument_list*	Component	Starts one instance of the current installed partition on the current node.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
CanBeActivated	none	Yes	Configuration	Indicates whether the partition is enabled for startup by the management system.
IsCompiled	none	Yes	Boolean	`TRUE` if the installed partition is compiled. Note that it is possible for the value of this instrument to differ between the InstalledPartitionAgent and the ActiveParitionAgent if a partition is started manually.

Programmatic Command Summary

Command	Arguments	Returns	Description
GetLogicalPart	none	Object	Returns the agent for the (logical) Partition that contains the named service object.

Using the Installed Partition Agent

The Installed Partition agent represents partitions that have been installed on particular nodes in the environment.

Ad hoc partition agents In `Escript`, some Installed Partition agents are actually Ad hoc partitions agents, because the partitions represented by the agents have not actually been installed. For information about Ad hoc partition agents, see "Ad hoc partition Agent" on page 133.

Parent and subagents The parent agents to the installed partition are both the Node agents and the Logical Partition agents. However, executing the `FindParentAgent` command while the current agent is an Installed Partition agent navigates to the Node agent. The subagents to an Installed Partition agent are the Active Partition agents, which represent executing instances of the partition.

States

DEGRADED

(Replicated partitions only) At least one instance of the partition—but fewer than the replication count—is running.

OFFLINE

No instances of this partition are running.

ONLINE

The minimum number of instances of this partition are running. If the partition is not replicated, then the minimum number is one.

If the partition is replicated, then the minimum number of running partitions is the replication count. The replication count was specified as part of the properties of the installed partition. To change the replication count, you need to edit the application definition, then reinstall the application. For information about changing the properties of installed partitions, see *iPlanet UDS System Management Guide*.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

ModLoggerRemote

The `ModLoggerRemote` command sets the logger flags for all of the active partitions that are represented by this Installed Partition agent.

ModLoggerRemote +(*logger_flags*) | -(*logger_flags*)

Argument	Description
+(*logger_flags*)	Turn on the logger flag settings given in the parentheses.
-(*logger_flags*)	Turn off the logger flag settings given in the parentheses.

To start logging, invoke the `ModLoggerRemote` command using the '+' followed by a set of logger settings in parentheses. To stop logging, use the '-' followed by a set of logger settings in parentheses.

The logger flag settings in the `ModLoggerRemote` command modify any logger flag settings that were specified for the partition either in the Partition Workshop, the `-fl` startup flag, or by the FORTE_LOGGER_SETUP environment variable.

The modified logger settings are only applied to the first file specified in the original logger settings for the Node Manager (or Environment Manager).

See the `LogMgr` class in the Framework Library online Help for a detailed description of the logger flag syntax.

```
escript> ModLoggerRemote +(trc:os:1:1 cfg:c4:2-3:1)
escript> ModLoggerRemote -(cfg:c4)
```

SetEnvRemote

The `SetEnvRemote` command sets the environment variable for all of the active instances of the installed partition managed by the current agent.

SetEnvRemote *env_variable new_value*

Argument	Description
env_variable	The name of an environment variable to set.
new_value	The new value of the environment variable.

The `SetEnvRemote` command changes the setting of the environment variable in all instances of the active partition managed by the current Installed Partition agent. Within the TOOL code executing in that partition, any subsequent invocation of the `GetEnv` method on the OperatingSystem object gets the new setting.

The `env_variable` argument is the name of an environment variable to set in the process running the active partition, and the `new_value` argument is the value for the environment variable.

On UNIX and VMS nodes, the new setting of the environment variable does not remain beyond the current execution of the partition. On Windows NT, the new setting is stored permanently and is picked up in any client partition or iPlanet UDS application started at a later time, because the value is stored in the registry in Windows NT.

Shutdown

The `Shutdown` command shuts down all active instances of the installed partition represented by the current agent.

Shutdown ***kill_executors***

The `Shutdown` command shuts down the Installed Partition agent. The shutdown request is propagated to all of the subagents of the Installed Partition agent. The most important of these are the Active Partition agents that are running the partition on this node.

If the `kill_executors` argument is set to `TRUE` or `1`, any interpreted partitions will shut down their hosting iPlanet UDS executor (ftexec or ftexecd) process as well.

ShutdownSubAgent

The `ShutdownSubAgent` command shuts down the named active partition.

ShutdownSubAgent *subagent*

Argument	Description
subagent	The name of a subagent to be shut down with its managed object.

If the named subagent does not exist, the `ShutdownSubAgent` command does nothing.

The `ShutdownSubAgent` command performs the same function as the following command sequence in an Escript script:

```
escript> FindSubAgent AutoCompileSvc_cl0_Part1_0x4d7:0x1
escript> Shutdown
escript> FindParentAgent
```

The difference between this sequence of commands and using the `ShutdownSubAgent` command is that the `Shutdown` command is not invoked on the current agent if the subagent does not exist.

Startup

The `Startup` command starts one instance of the current installed partition.

Startup [*argument_list*]

Argument	Description
argument_list	A set of command-line arguments to use in starting the partition. If there are any spaces in the argument list specification, use double quotes.

You can start a single instance of a partition on a specific node by invoking the `Startup` command while the current agent is an installed partition.

The optional `argument_list` argument is passed to the partition as it starts up. You can specify special memory or logger flags for the partition on startup. If specified, these arguments are used in place of the command-line arguments for the partition specified with the `SetArgs` command.

You can use the `Startup` command in the Application and Logical Partition agents to start a number of installed partitions up to the predefined replication count. However, you can invoke the `Startup` command while the current agent is an Installed Partition agent to start a single additional instance of a partition on a node. This is particularly useful for starting additional replicates of failover or load balanced partitions within an application at peak loads.

You can also use the `Startup` command to pre-start iPlanet UDS executors for use by developers performing test runs from the Partition Workshop. For example, to start a partition with special arguments named auction_part1 on a node named Washington, use the following commands:

```
escript> FindActEnv
escript> FindSubAgent Washington
escript> FindSubAgent auction_part1
escript> Startup “-fm ‘(n:4000 x:8000)’ -fl ‘%stdout(trc:user:*)’”
```

Instruments

CanBeActivated

The `CanBeActivated` instrument indicates whether the partition is enabled for startup by the management system. This Configuration instrument is read only.

Programmatic Commands

GetLogicalPart

The `GetLogicalPart` command returns the (logical) Partition agent that is the parent of the Installed Partition agent.

GetLogicalPart

The `GetLogicalPart` command navigates from the Installed Partition agent to the Partition agent that is its parent. This command complements the Escript `FindParentAgent` command, which always navigates from the Installed Partition agent to the Node agent that is its parent.

LoadBalancing Router Agent

Parent Agent

Active Partition agent

Subagents

None

SystemMonitor Class

LoadBalanceRouterAgent

States

State	Description
ONLINE	The load-balanced router is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
ActiveMembers	none	Yes	Counter	The number of currently active replicates to which the managed router is sending messages.
MaxWaitDepth	none	Yes	Counter	The highest recorded number of messages that were waiting for routing at any one time.

Instrument	Argument	Read Only?	Type	Description
MembersInUse	none	Yes	Counter	The number of replicates of the load-balanced service object that are currently processing messages.
MessagesRouted	none	Yes	Counter	The total number of messages that this router has processed.
MessagesWaiting	none	Yes	Counter	The number of messages currently waiting in the router for a member to become available.
PeakMembersInUse	none	Yes	Counter	The maximum number of members that have serviced messages forwarded by the router at any one time

Using the LoadBalancing Router Agent

The LoadBalancing Router agent represents a router for a load-balanced service object.

Parent and subagents The parent agent for the LoadBalancing Router agent is the Active Partition agent for the active partition that contains the router. The LoadBalancing Router has no subagents.

States

ONLINE

The router for a load-balanced service object in the active partition is running.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

ActiveMembers

The `ActiveMembers` instrument indicates the number of currently active replicates of the load-balanced service object to which the managed router is sending messages. This Counter instrument is read only.

MaxWaitDepth

The `MaxWaitDepth` instrument indicates the highest recorded number of messages that were waiting for routing at any one time. This Counter instrument is read only.

MembersInUse

The `MembersInUse` instrument indicates the number of replicates of the load-balanced service object that are currently processing messages. This Counter instrument is read only.

MessagesRouted

The `MessagesRouted` instrument indicates the total number of messages that this router has processed. This Counter instrument is read only.

MessagesWaiting

The `MessagesWaiting` instrument indicates the number of messages currently waiting in the router for a member to become available. This Counter instrument is read only.

PeakMembersInUse

The `PeakMembersInUse` instrument indicates the maximum number of members that have serviced messages forwarded by the router at any one time. This Counter instrument is read only.

Machine Agent

Parent Agent

Node agent

Subagents

Volume agent

SystemMonitor Class

MachineAgent

States

State	Description
ONLINE	The machine is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
ActiveSwapDevices	none	Yes	Counter	System swap devices that are active.
ForteLogSpaceKB	none	Yes	Counter	Kilobytes of disk storage used by `FORTE_ROOT/log`.
ForteRootSpaceKB	none	Yes	Counter	Kilobytes of disk storage used by `FORTE_ROOT`.
ForteSampleIntervalSeconds	*interval*	No	Configuration	Seconds between data refreshes for `ForteRootSpaceKB` and `ForteLogSpaceKB`.
FreeSwapSpaceKB	none	Yes	Counter	Kilobytes of currently unused swap space available to user programs.
FreeSwapSpacePercent	none	Yes	Counter	Percentage of swap space available to user programs.
MMUPageSizeBytes	none	Yes	Counter	System memory management page size.
TotalSwapSpaceKB	none	Yes	Counter	Total kilobytes of swap space available to user programs.

Using the Machine Agent

The Machine agent represents the physical machine on which a Node Manager is running.

If iPlanet UDS retrieves the values of the `ForteLogSpaceKB` and `ForteRootSpaceKB` instruments too often, considerable time and resources will be used. You can set how often iPlanet UDS retrieves these values using the `ForteSampleIntervalSeconds` instrument, which by default specifies that these values be determined every 300 seconds. The `ForteSampleIntervalSeconds` instrument does not affect how frequently the values are logged; this is set by the Active Partition agent's `LogTimer` instrument.

Certain instruments are not available on some platforms. If an instrument value is not available for a Machine agent, the value is set to `-1`.

Parent agents and subagents The parent agent for a Machine agent is a Node agent. A Machine agent has one or more Volume subagents, which represent the storage devices available to the machine.

States

ONLINE

The machine on which the node is running is also running.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

ActiveSwapDevices

The `ActiveSwapDevices` instrument indicates the number of disks that provide virtual memory for swapping. If the swap devices can be set as active or inactive, this instrument counts only the active devices. This Counter instrument is read only.

ForteLogSpaceKB

The `ForteLogSpaceKB` instrument indicates the amount of storage space, in kilobytes, being used by the iPlanet UDS application log files in the `$FORTE_ROOT/log` directory. This Counter instrument is read only.

You can specify how frequently iPlanet UDS determines the value of the `ForteLogSpaceKB` instrument with the `ForteSampleIntervalSeconds` instrument (described in "ForteSampleIntervalSeconds" on page 224).

ForteRootSpaceKB

The `ForteRootSpaceKB` instrument indicates the amount of storage space, in kilobytes, being used by the iPlanet UDS root directory (`$FORTE_ROOT`). This Counter instrument is read only.

You can specify how frequently iPlanet UDS determines the value of the `ForteRootSpaceKB` instrument with the `ForteSampleIntervalSeconds` instrument (described below).

ForteSampleIntervalSeconds

The `ForteSampleIntervalSeconds` instrument specifies, in seconds, how often iPlanet UDS retrieves the values for the `ForteLogSpaceKB` and `ForteRootSpaceKB` instruments. By default, this value is `300 seconds`. The `ForteSampleIntervalSeconds` instrument is a Configuration instrument.

ForteSampleIntervalSeconds *interval*

Argument	Description
interval	Number of seconds between times that iPlanet UDS determines the values for the `ForteLogSpaceKB` and `ForteRootSpaceKB` instruments. By default, this value is `300 seconds`.

FreeSwapSpaceKB

The `FreeSwapSpaceKB` instrument indicates the amount of unused swap space, in kilobytes, that is available to user applications. This value is the difference between the value of the `TotalSwapSpaceKB` instrument and the combined swap space reserved by the operating system and running programs. This Counter instrument is read only.

FreeSwapSpacePercent

The `FreeSwapSpacePercent` instrument indicates the percentage of the swap space on the machine that is unused and available to user applications. This value is a percentage of the `TotalSwapSpaceKB` instrument value. This Counter instrument is read only.

MMUPageSizeBytes

The `MMUPageSizeBytes` instrument indicates the number of bytes that the processor uses as the memory management unit page size. This Counter instrument is read only.

TotalSwapSpaceKB

The `TotalSwapSpaceKB` instrument indicates the total amount of swap space, in kilobytes, available to the machine. This Counter instrument is read only.

Model Node Agent

Parent Agent

Environment agent

Subagents

Node agents

SystemMonitor Class

SystemAgent

States

State	Description
OFFLINE	No nodes belonging to this model node group are running.
ONLINE	At least one node belonging to this model node group is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Using the Model Node Agent

The Model Node agent manages the definition for a model node.

Parent and subagents The parent agent for the Model Node agent is the Environment agent. The subagents for the Model Node agent are Node agents that are active members of the model group. A *model group* is a set of nodes that correspond to the definition of the model node. Usually, these are a set of client nodes.

The Model Node agent has no defined instruments.

States

OFFLINE

No nodes belonging to this model node group are running Node Managers or processes that act as Node Managers, such as Launch Servers or Escript.

ONLINE

At least one node belonging to this model node group is running a Node Manager or a process that acts as a Node Manager, such as a Launch Server or Escript.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

NameService Agent

Parent Agent

Environment agent

Subagents

None

SystemMonitor Class

SystemAgent

States

State	Description
ONLINE	The Name Service is running.

Command Summary

Command	Arguments	Environment Console menu	Description
ConnectEnv	*env_name* *env_location* *user_directory*	Utility > Connect Environment	Connects a target environment to the environment from which the command is invoked.
DisconnectEnv	none	Utility > Disconnect Environment	Separates the current environment from any other environments.
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
ModLoggerRemote	+(*logger_flags*) -(*logger_flags*)	Component > Modify Log Flags	Modify the current logger settings.
NsCd	*directory_name*	Utility > Change Directory	Changes the current name space directory.
NsLs	*directory_name*	Utility > List DIrectory	Lists the contents of a name space directory.
RemoveLostParts	none	Utility > Remove Lost Partitions	Deletes information about partitions that the Environment Manager can no longer access.
ShowAdmin	none	Utility > Show Administration	Shows information about the Name Service.
ShowEnv	*env_name*	Utility > Show Environment	Displays information about an environment or all environments known to this environment.

Command	Arguments	Environment Console menu	Description
ShowPart	*partition_id* | **lost**	Utility > Show Partition	Shows information about partitions known to this environment.
Shutdown	none	Component	Shuts down the NameService agent and the corresponding Name Server.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
DeleteOnCommFailure	**TRUE** | **FALSE**	No	Configuration	Indicates whether the Environment Manager deletes information about partitions that it can no longer access.
EnvSearchPath	*path_spec*	No	Configuration	A list of environments to be used to locate named objects in the name space for a group of connected environments.

Using the NameService Agent

The NameService agent manages the iPlanet UDS Name Service, which runs in the Environment Manager partition. This partition is started by invoking the iPlanet UDS `nodemgr -e` command. The NameService agent is a subagent of the Environment agent.

Parent agent The parent agent for this agent is the Environment agent.

To invoke NameService agent commands, you open the current Environment agent, and then open the NameService subagent. For example:

```
escript> FindActEnv
escript> FindSubAgent NameService
```

iPlanet UDS Name Service The Name Service within an environment brokers requests for distributed services. Most commonly, these requests come about transparently when one partition requests access to a service object in a different partition. The iPlanet UDS Name Service takes care of making the appropriate connection when the requested service object is first referenced.

A TOOL programmer can explicitly use the iPlanet UDS Name Service through methods defined on the `ObjectLocationMgr` class. This class defines two primary methods:

- the `Register` method explicitly registers a service (represented by an anchored object) with the Name Service
- the `Bind` method locates an object that has been registered

For more information, see the description of the ObjectLocationMgr class in the Framework Library online Help.

Connecting iPlanet UDS environments By default, the iPlanet UDS Name Service brokers only names from a single environment. This means that registration and binding of objects includes only objects that are executing within a single iPlanet UDS environment, which is also a single name space. However, you can connect a set of environments using the `ConnectEnv` command of the NameService agent. Once two environments are connected, they becomes part of a broader connected name space. Any number of environments can be connected in this way. There is no defined hierarchical structure to these environments or their connected name space, and a partition in any environment can reference objects in the other connected environments.

You invoke the `ConnectEnv` command from one environment and specify a target environment that is added to the existing name space. Once connected, the environment that invoked the command gains access to the newly added environment, as do all other environments already connected, without needing to invoke any special commands. Likewise, the newly-added environment automatically gains access to all previously connected environments as well.

The connections between environments are persistent, and remain even when the Environment Manager partition, which contains the Name Service and NameService agent, is shut down and brought back online.

Name space directories A name space, single or connected, is organized into two independent directory trees:

- An iPlanet UDS-defined directory tree that manages environment visible service objects that have been defined in the iPlanet UDS development environment.

  This directory tree has a complex internal structure, but you can refer to this directory using the special syntax "`@env_name`", where `env_name` is the name of an environment that is part of the connected name space. The special shorthand "`@`" designates the current active environment. Of course, an environment that has not been connected to other environments through the `ConnectEnv` command cannot get access to other `@env_name` directories in the connected name space. (Note: the actual internal directory path for these directories is `/forte/`*environment_uuid*`/site`, where `environment_uuid` is a long unique identifier of the environment represented by `@env_name`.)

- An explicitly-defined directory tree that resolves references to anchored objects explicitly registered with the Name Service. For more information, see the description of the `ObjectLocationMgr` class in the Framework Library online Help.

  This directory tree starts at the root (`/`) directory of the Name Service registry, and represents all directories and services that are registered in any of the participating environments. The structure of this set of directories is completely arbitrary, and uses UNIX-like syntax for the directory names (a `/` separates directories). The directories are created explicitly when the Register method needs to add them. See"ConnectEnv" on page 234 for details on how this second directory tree is shared when several environments are connected.

Name space search paths The Name Service resolution of references to both service objects (implicitly registered) and anchored objects (explicitly registered) uses a search path to find objects in the name space. The search path lists a set of name-space directory names that are searched, in order, to find registered objects, either through an implicit lookup of service objects, or an explicit lookup using the Bind method of the `ObjectLocationMgr` class.

Search path for implicit bindings For implicit (service object) bindings, the search path is normally specified as a set of directories using the following syntax (no spaces are allowed in the search path):

path [(a)] [: *path* [(a)]...

path is:

(@ | @*environment_name*)

A special (a) option allows you to specify that the service object identified by a specific path should automatically be started if necessary.

You can use an environment variable to specify an environment name. The value for the environment variable is set on first access to the service object, using the value of the environment variable as set on the service object's partition. The syntax is:

${*environment_variable_name*}

For example, a search path could be in the form "`@Env1:@Env2(a)`." This example specifies that when a partition is trying to gain access to a remote service object, it should check first in `Env1`, and then in `Env2` for matching service object registrations. The "`(a)`" after the "`@Env2`" value indicates that the system should attempt to auto-start the service.

Search path for explicit bindings For explicit (anchored object) bindings using the Bind method, the application developer can specify the absolute directory path when resolving anchored object references, in which case the name space search path is not used. If the developer specifies relative path names for objects in the Bind method, however, the relative path name is added to each directory in the environment search path when the Name Service tries to locate an object.

The search path can be specified for both service objects and for an environment. The service object search path takes precedence. For information on specifying a search path for a service object, see *A Guide to the iPlanet UDS Workshops*. To specify a search path for an environment, use the `EnvSearchPath` instrument on the NameService agent (see "EnvSearchPath" on page 244).

You can select whether the Environment Manager automatically deletes information about lost partitions and their named objects by setting the value of the `DeleteOnCommFailure` instrument of the NameService agent to `TRUE` or `FALSE`. The default value is `TRUE`.

If the Environment Manager is shut down normally within the Environment Console or Escript by using the `Shutdown` command on the Environment Manager, all information related to object names and partitions is removed.

DeleteOnCommFailure Value is TRUE

The Environment Manager automatically deletes its information about each lost partition and its named objects.

This option is intended for environments in which all partitions are connected to the Environment Manager using a local LAN and partitions are expected to come up and go down. In this situation, you can usually assume that a communication failure with the Environment Manager means that a partition failed and that the Environment Manager should delete all information associated with the unavailable partition.

If that same partition reconnects to the Environment Manager, the Environment Manager adds information about the partition and its named objects back into its name service database.

With this option, the Environment Manager does not accumulate information about objects that it can no longer access. However, if the communication failure occurred for a reason other than a partition failure, the Environment Manager cannot access named objects in that partition, even though the objects are still available.

DeleteOnCommFailure Value is FALSE

The Environment Manager retains information about lost partitions and their named object until the system manager explicitly requests that all information about lost partitions be deleted.

This option is intended for environments with the following characteristics:

- partitions are widely distributed
- the environment is a production environment
- communication failures with the Environment Manager can occur even though communications between servers and between clients and servers have not been disrupted

With this option, the Environment Manager keeps all information about lost partitions and their named objects.

To illustrate how using the FALSE value works, imagine two partitions, Part1 and Client0. If a communication failure occurs between Part1 and the Environment Manager, iPlanet UDS makes no assumption about the status of Part1 or the ability of other partitions to communicate with Part1. Therefore, if Client0 needs to access a named object in Part1, it can get the addressing information from the name service database and access the named object, even though Part1 cannot currently communicate with the Environment Manager.

The system manager can use the `RemoveLostParts` command on the NameService agent to delete information about lost partitions from the name service database.

RemoveLostParts

To see a list of the lost partitions known to the Environment Manager, specify "`lost`" as a the parameter for the `ShowPart` command on the NameService agent, as shown:

```
escript> ShowPart lost
```

States

ONLINE

The Name Service for the environment is running.

Commands

ConnectEnv

The `ConnectEnv` command connects a target environment to the environment from which the command is invoked.

ConnectEnv {*env_name* | *env_UUID}* *env_location* [*user_directory*]

Argument	Description
env_name	The name of the target environment that is to be connected. Specify either this value or the `env_UUID` value.
env_UUID	The universal unique identifier for the target environment that is to be connected. Specify either this value or the `env_name` value.
env_location	The FORTE_NS_ADDRESS value for the target environment that is to be connected.
user_directory	The directory in the target environment that is to be designated as the root directory for this environment's user-defined name hierarchy. This is required if `env_name` has pre-existing user-defined directories in its name space.

The `ConnectEnv` command merges the name space of the target environment into another environment. After the merge is complete, each environment knows about the other environment, and can access partitions in the other.

An environment cannot simply add itself to a group of connected environments. Instead, an environment in that group must request that another environment be added. Therefore, to add a new environment to a group of connect environments, you:

1. Make your current agent the agent for one of the environments already in the connected group.
2. On that agent, use the `ConnectEnv` command with the name or UUID and location as arguments. Normally, you can use the name; however, if two environments have the same name, you need to specify the UUID to identify which environment to connect.

You need to indicate either:

- the name (`env_name`) of the target environment that is to be merged. This is the environment name defined for an active environment. The following example shows how you can invoke the `ConnectEnv` command using the name:

  ```
  escript > ConnectEnv DocEnv hillary:6000
  ```

- the UUID (`env_UUID`) of the target environment that is to be merged. This is the UUID that is created for a particular active environment. The UUID is a 32-character value, such as `456EFE40-D77C-11D0-A57C-EFD1BD59AA77`. The following example shows how you can invoke the `ConnectEnv` command using the UUID:

```
escript > ConnectEnv 456EFE40-D77C-11D0-A57C-EFD1BD59AA77 hillary:6000
```

  To determine the UUID for an active environment, use the `ShowEnv` or the `ShowAdmin` commands.

You also need to specify the `env_location` value for the environment. This value should be same value as the FORTE_NS_ADDRESS environment value for that environment.

The `ConnectEnv` command merges both the pre-built branches of the name space used for implicit environment-wide service object name resolution, as well as the user-defined branches of the name space. The pre-built branches of the tree are updated in each environment to add the appropriate `@env_name` directory syntax for each environment. One or both of the environments might already be connected to other environments, and therefore know other environments. When these environments are referenced for the first time, the information for the other known environments is also merged into the name spaces for these two connected environments.

The user-defined portions of the directory tree are treated differently. (Note: If no applications in the environment are using the user-defined name space capabilities defined on the `ObjectLocationMgr` class—see the Framework Library online Help for details—then you do not need to worry about the user-defined portions of the tree.)

The `ConnectEnv` command considers the environment from which you are invoking the `ConnectEnv` command to be the "root" of a global name space directory structure for all connected environments. The target environment you are merging (the environment specified in the `env_name` argument) might have a pre-existing user-defined directory tree, so you need to indicate a spot in the global Name Service directory tree structure where the target environment's root directory is to be placed. This placement is specified using the user_directory argument. Once the command is invoked, the target environment's root directory is no longer accessible by its original name, and must be referred to through its new global directory location.

For example, assume that an environment, called littleton, is to be merged into another environment, called bigapple. Assume that littleton has a pre-existing, user-defined directory structure containing the directories `/xlittle` and `/ylittle`. Assume that bigapple has a user-defined directory structure containing `/xbig` and `/ybig`. From an Escript session running in the littleton environment, you can invoke the command:

```
escript> ConnectEnv bigapple nynode:1234 /littledir
```

After this is run, the user-defined directory structure in both environments contains: `/xbig`, `/ybig`, `/littledir`, `/littledir/xlittle` and `/littledir/ylittle`. Note that the original /xlittle and /ylittle directories in the littleton environment can no longer be accessed through those names, even from the littleton environment, and must be accessed through the new global names of `/littledir/xlittle` and `/littledir/ylittle`.

The user-defined directory structure in the name space is not explicitly defined (that is, there is no explicit command to create a directory). Instead, the directories are automatically created whenever a new directory is referenced through the `Register` method on the `ObjectLocationMgr` class.

If you have not used the `Register` method in the target environment, and there are no pre-existing user-defined directories, you do not need to specify the `user_directory` argument.

Updating merge information If you need to change the location information about an environment that has already been merged, you can re-invoke the `ConnectEnv` command, with new values for `env_name` and `env_location`. Once re-invoked, the `ConnectEnv` command fixes any of the discrepancies and removes the old values.

You cannot change the user directory by re-invoking the `ConnectEnv` command, however. In order to change that directory, you should invoke the `DisconnectEnv` command to remove the environment from the connected name space, and then re-invoke the `ConnectEnv` command to correct the directory.

The Environment Manager service for the target environment must be online and accessible for `ConnectEnv` to succeed. However, once merged, the connection need not be always maintained during system operations. The system is resilient to failures between the two environments, and needs access to the other Environment Manager service only for partitions that are only available in the alternate environment.

DisconnectEnv

The `DisconnectEnv` command separates the current environment from any other environments.

DisconnectEnv [*env_name* | *env_UUID*]

Argument	Description
env_name	The name of the target environment that is to be removed from the global name space. Specify either this value or the `env_UUID` value.
env_UUID	The universal unique identifier for the target environment that is to be removed from the global name space. Specify either this value or the `env_name` value.

No argument If you specify no argument, the `DisconnectEnv` command removes the current environment from the global name space of other environments in which it currently belongs (through a previous use of the `ConnectEnv` command). After you invoke the `DisconnectEnv` command, this environment can no longer use services in any other environments.

In certain circumstances, you might need to remove a specific environment from the global name space. For example, if an environment is no longer active or available, you might want to remove the environment from the name space. Normally, you can use the name; however, if two environments have the same name, you need to specify the UUID to identify which environment to remove.

To specify a particular environment, you can indicate either:

- the name (`env_name`) of the target environment that is to be removed. This is the environment name defined for the active environment. The following example shows how you can invoke the `DisconnectEnv` command using the name:

  ```
  escript > DisconnectEnv DocEnv
  ```

- the UUID (`env_UUID`) of the target environment that is to be removed. This is the UUID that is created for a particular active environment. The UUID is a 32-character value, such as `456EFE40-D77C-11D0-A57C-EFD1BD59AA77`. The following example shows how you can invoke the DisconnectEnv command using the UUID:

  ```
  escript > DisconnectEnv 456EFE40-D77C-11D0-A57C-EFD1BD59AA77
  ```

  To determine the UUID for an active environment, use the `ShowEnv` or the `ShowAdmin` commands.

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

ModLoggerRemote

The `ModLoggerRemote` command sets the logger flags.

ModLoggerRemote +(*logger_flags*) | -(*logger_flags*)

Argument	Description
+(*logger_flags*)	Turn on the logger flag settings given in the parentheses.
-(*logger_flags*)	Turn off the logger flag settings given in the parentheses.

To start logging, invoke the `ModLoggerRemote` command using the '+' followed by a set of logger settings in parentheses. To stop logging, use the '-' followed by a set of logger settings in parentheses.

The logger flag settings in the `ModLoggerRemote` command modify any logger flag settings that were specified for the partition either in the Partition Workshop, the `-fl` startup flag, or by the FORTE_LOGGER_SETUP environment variable.

The modified logger settings are only applied to the first file specified in the original logger settings for the Node Manager (or Environment Manager).

See the `LogMgr` class in the Framework Library online Help for a detailed description of the logger flag syntax.

```
escript> ModLoggerRemote +(trc:os:1:1 cfg:c4:2-3:1)
escript> ModLoggerRemote -(cfg:c4)
```

NsCd

The `NsCd` command changes the current name space directory.

NsCd *directory_name*

Argument	Description
directory_name	The name of the directory to make the new current name space directory.

The `NsCd` command changes the current name space directory to the specified directory name. This new directory can then be used by subsequent `NsLs` commands to list the contents of the directory.

The syntax of the `directory_name` argument is a UNIX-like directory name. If the name starts with a `/`, it is considered to be an absolute directory path. If it does not start with a `/`, it is considered to be relative to the current name space directory. A directory_name of ".." indicates to move up a level in the directory tree.

You can use the `NsCd` command to navigate through the explicitly-defined directory tree in the name space, much like you would use the UNIX `cd` command for moving through file directories. You can also specify the special `@env_name` and @ directory names to navigate through the implicitly-defined directory tree for service object resolution.

Example usage of the `NsCd` command:

```
escript> NsCd /a/b/c # An explicitly-defined directory
escript> NsCd ../d # Moves to /a/b/d directory
escript> NsCd /a/b/c # An explicitly-defined directory
escript> NsCd @bigapple # /site dir for bigapple env
escript> NsCd @ # /site dir for this environment
```

NsLs

The `NsLs` command lists the contents of a name space directory.

NsLs [*directory_name*]

Argument	Description
directory_name	The name of the directory whose contents to list.

The `NsLs` command lists the contents of a directory, either other name space directories or services (service objects or anchored objects) that have been registered in that directory.

If you specify the `directory_name` argument, the `NsLs` command lists the contents of the named directory. This directory specification can be an absolute or relative path name, either in the explicitly-defined directory tree of the name space or the explicitly-defined tree. See "NsCd" on page 239 for details on the specification of directory names.

If you do not specify the `directory_name` argument, the current name space directory, as specified by the most recent `NsCd` command, is used.

Examples of the `NsLs` command:

```
escript> NsLs /a/b/c # An explicitly-defined directory
escript> NsLs @ # /site dir for this environment
escript> NsLs 3873-2223-344-232-3-3:0x10070:nnn
```

RemoveLostParts

The `RemoveLostParts` command deletes information about partitions that the Environment Manager can no longer access from the name service database.

RemoveLostParts

The `RemoveLostParts` command is useful when you have set the value of the `DeleteOnCommFailure` instrument of the NameService agent to `FALSE`. Setting the `DeleteOnCommFailure` instrument to `FALSE` means that information about partitions that the Environment Manager can no longer access is not deleted automatically from the name service database. To delete information about these partitions, you need to use the `RemoveLostParts` command.

ShowAdmin

The `ShowAdmin` command shows information about the Name Service.

ShowAdmin

The `ShowAdmin` command lists some basic information about the Name Service for the current environment, including: the current environment name, the current environment UUID, the current default environment Name Service search path, and the current Name Service directory.

ShowEnv

The `ShowEnv` command displays information about an environment or about all environments known to this environment.

ShowEnv [*env_name*]

Argument	Description
env_name	The name of an environment. The default is to show all environments known to this one that have been accessed.

The `ShowEnv` command shows information about a single environment, or about all environments that are directly known to the environment from which you are invoking the `ShowEnv` command. See "ConnectEnv" on page 234 for information about linking environments together.

If the `env_name` argument is specified, information about that environment is shown, which includes the following: the name of the environment, the UUID of the environment, the environment location (the FORTE_NS_ADDRESS of the environment) or the name of an environment through which the other environment is connected.

If you do not specify the `env_name` argument, all environments known to the current environment are shown. The listed environments include only those environments that this environment has already discovered and cached. The `ShowEnv` command does not necessarily show all of the connected environments in the connected name space.

Examples of the `ShowEnv` command follow below:

```
escript> ShowEnv bigapple # Show info on other environments
escript> ShowEnv # Show info on all environments
```

ShowPart

The `ShowPart` command shows information about partitions known to this environment.

ShowPart [*partition_id*]

Argument	Description
partition_id	The ID of a partition that is known to the current environment. The default is to show information about all registered partitions.

The `ShowPart` command displays information about a single partition, or about all partitions that are currently known to this environment. For a partition listed by this command, the following information is displayed: the application name, the communications location for the partition, and the partition ID.

The `partition_id` argument specifies a specific partition for which information is to be displayed. The syntax is:

environment_uuid:part_unique_id:replicate_id

The environment_uuid is the UUID of the environment in which the partition is executing, the part_unique_id is the ID of the partition within the current environment. The replicate_id is a unique ID within replicates. For example, you might see a specific partition_id in an exception message, and can then use the ShowPart command on the NameService agent for more information.

If you do not specify the partition_id argument, all executing partitions known to the current environment are shown. These partitions include all partitions that are executing in the current environment, as well as any partitions executing in other partitions that have been cached in the current environment. This occurs when requests are made that require a path search that spans environments.

Examples of the ShowPart command are:

```
escript> ShowPart # All partitions are shown
escript> ShowPart 3873-2223-344-232-3-3:0x10070:nnn
```

Shutdown

The Shutdown command shuts down the NameService agent and its corresponding Name Server.

Shutdown

Instruments

DeleteOnCommFailure

The DeleteOnCommFailure instrument is a read-write instrument that specifies whether iPlanet UDS automatically deletes information from the name service database for partitions that the Environment Manager can no longer access. The DeleteOnCommFailure instrument is a Configuration instrument.

DeleteOnCommFailure TRUE | FALSE

If this instrument is set to TRUE, then iPlanet UDS automatically deletes information about partitions that the Environment Manager can no longer access from the name service database.

If this instrument is set to `FALSE`, then iPlanet UDS does not delete information about partitions that the Environment Manager can no longer access from the name service database. Instead, the partition is marked "Lost" to indicate that the Environment Manager cannot access it. To delete information about these "lost" partitions from the name service database, use the `RemoveLostParts` command on the NameService agent, which is described in "RemoveLostParts" on page 241.

EnvSearchPath

The `EnvSearchPath` instrument is a read-write instrument that represents the search path for locating named objects in the environment name space. The `EnvSearchPath` instrument is a Configuration instrument.

EnvSearchPath *path_spec*

Argument	Description
path_spec	A list of environments to be used to locate named objects in the name space for a connected environments.

The `EnvSearchPath` instrument defines the environment search path for locating named objects in a connected name space. iPlanet UDS uses this search path to locate an available service object for a service request if no environment search list has been specified for that service object. If a service object has its own defined environment search list, the environment search path defined for the `EnvSearchPath` instrument is not used.

For more information about specifying an environment search list for a service object, see *A Guide to the iPlanet UDS Workshops*.

The environment search path is used for resolving both implicit Name Service lookups (used when looking up service objects) and explicit Name Service lookups (using the `Bind` method on the `ObjectLocationMgr` class). When performing explicit lookups, the path is not used if the Bind method is invoked with an absolute path name for an object name. If a relative path name is specified in the `Bind` method, the relative path is tacked on to each directory in the environment search path to complete the search.

The `path_spec` argument is a set of colon-separated directory specifications, which provides a set of directories in which to search for objects. The directory specifications use the standard syntax for absolute name space directories in the iPlanet UDS-defined or the user-defined portions of the Name Service directory tree. For the details on specifying directory names, see "Name space directories" on page 231.

Normally, the directory names in the environment name space search path are set to the special `@env_name` syntax, for searching through a set of environments to resolve environment-wide implicit service object requests. For example, you could invoke the following command:

```
escript> UpdateInstrument EnvSearchPath @:@bigapple(a):@la
```

In this example, when environment-wide services are requested by any partition executing within the current environment, iPlanet UDS first tries to use services running in the current environment, if available. Next, iPlanet UDS tries to use services running in the bigapple environment (which attempts to auto-start the servers if none are running), followed by services running in the la environment.

The default environment search path is "`@(a)`", which means that iPlanet UDS looks only in the current environment for name resolution.

NOTE You should only set and rely on the environment search path defined by the `EnvSearchPath` instrument in environments where all applications use the same search path. In environments in which applications use different search paths, you should instead use environment search lists specific to each service object. Otherwise, a service object without its own environment search list might inadvertently use services from other environments if a default environment search path that includes multiple environments has been defined by the `EnvSearchPath` instrument.

NativeLangMgr Agent

Parent Agent

Active Partition and RepositoryServer agent

Subagents

None

SystemMonitor Class

NativeLangMgrAgent

States

State	Description
ONLINE	The native language manager is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpAll	none	Utility	Dumps all status information.
DumpCSConv	none	Utility	Dumps the code set conversion information.
DumpLocale	none	Utility	Dumps the currently loaded locale information.
DumpMsgCat	none	Utility	Dumps the currently loaded message catalogs.
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Using the NativeLangMgr Agent

The NativeLangMgr agent represents the Native Language Manager for an active partition. The Native Language Manager manages services for multinational and multilingual functions.

Parent and subagents The parent agent for the NativeLangMgr agent is an Active Partition agent. The NativeLangMgr agent has no subagents.

The NativeLangMgr agent has no defined instruments.

States

ONLINE

The Native Language Manager for the active partition is running. The Native Language Manager is a part of the runtime system, and cannot run unless an active partition is running.

Commands

DumpAll

The `DumpAll` command dumps all status information about the agent, including:

- status information for the agent's managed object
- the current code set conversion information
- currently loaded locale information
- currently loaded message catalogs

DumpAll

DumpCSConv

The `DumpCSConv` command dumps information about the code set conversion.

DumpCSConv

DumpLocale

The `DumpLocale` command dumps information about the currently loaded locale.

DumpLocale

DumpMsgCat

The `DumpMsgCat` command dumps the currently loaded message catalogs.

DumpMsgCat

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Node Agent

Parent Agent

Environment agent or Model Node agent

Subagents

Installed partition agents

SystemMonitor Class

NodeAgent

States

State	Description
ONLINE	A Node Manager is running on the node.
RUNDOWN	The node is shutting down.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
ExecCmdRemote	*opsys_command* *bg_flag* *in_file* *out_file* *err_file*	Utility	Executes the specified operating system command from the Node Manager service that is managed by the current agent.
InstallApp	*application_name* *force_copy*	none	Installs partitions for an application on a node.
ListAppsToInstall	none	none	Lists the names of the applications that need to be installed on the node represented by the current Node agent.
ListDistribs	none	none	Lists the application distributions available in the node managed by the current Node agent.
LoadDistrib	*application_name* *compatibility_level*	Installation	Loads the specified application distribution into the environment from the node represented by the current Node agent.
ModLoggerRemote	+(*logger_flags*) -(*logger_flags*)	Component > Modify Log Flags	Sets the logger flags for the Node Manager that is being managed by the current Node agent.
SetEnvRemote	*env_variable* *new_value*	none	Sets an environment variable value for the Node Manager represented by the current Node agent.

Command	Arguments	Environment Console menu	Description
Shutdown	*no_propagate*	Component	Shuts down the Node Manager (Node agent).
ShutdownSubAgent	*subagent*	none	Shuts down the named subagent and its managed object.
StartInstPart	*partition_name one_more*	none	Starts one instance of an installed partition on the node managed by the current Node agent (Node Manager).

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
Architecture	none	Yes	Configuration	Architecture for the node managed by the current Node agent.
ModelNode	none	Yes	Configuration	Model node name for the node that is managed by the current Node agent.
StartupWaitTime	*interval_in_sec*	No	Configuration	Gets or sets the wait time on the node represented by the current Node agent before it assumes that partitions that it auto-starts have failed.

Programmatic Command Summary

Command	Arguments	Returns	Description
GetPartAgent	none	Object	Returns the Active Partition agent for the node manager partition.
GetRemoteFS	none	Object	Returns a FileSystem object whose current working directory is set to the iPlanet UDS root directory for the Node Manager.

Using the Node Agent

The Node agent manages a node (and is represented by the Node Manager service).

To navigate to a Node agent, you should first set the current agent to the Environment agent, and then use the `FindSubAgent` command, giving the node name.

Parent and subagents The parent agent for the Node agent is the Environment agent. The subagents of the Node agent are the Installed Partition agents for all applications installed on a given node.

States

ONLINE

A Node Manager, or other process that acts as a Node Manager, such as the Launch Server, is running on the node.

RUNDOWN

The node is shutting down.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

ExecCmdRemote

The `ExecCmdRemote` command executes the specified operating system command from the Node Manager service that is managed by the current agent.

ExecCmdRemote *opsys_command* [*bg_flag*] [*in_file*] [*out_file*] [*err_file*]

Argument	Description
opsys_command	Specifies a valid operating system command appropriate to the system on which the Node Manager service is running.
bg_flag	Indicates whether to run the command synchronously or asynchronously with a value of `1`.
in_file	An alternate input file for the operating system command.
out_file	An alternate output file for the operating system command.
err_file	An alternate error file for the operating system command.

The ExecCmdRemote command runs the command specified by the opsys_command argument from the Node Manager that is executing on the node that is being managed by the current Node agent.

You can specify command line arguments to the command by including the command in double quotes. Any references to environment variables must be use the syntax appropriate to the node on which the command executes.

Special syntax for OpenVMS On OpenVMS, if you want OpenVMS to execute the command, you need to specify the characters “`$ `“ (dollar-sign and a space) before the command name so that OpenVMS knows to look for an executable (.com or .exe) file or DCL symbol. If you explicitly specify a path and file extension, OpenVMS tries to execute that particular file in the specified path. You cannot specify both “`$ `“ and a path.

The following example shows how you would use the `ExecCmdRemote` command with the "`$ `" syntax. In this example, the `ExecCmdRemote` command invokes the OpenVMS `SHOW DEFAULT` command, which prints the current directory to the `A.OUT` file. This example then invokes the `ListFile` command to display the contents of the A.OUT file:

```
FindSubAgent VMS_Server_Node
ExecCmdRemote "$ SHOW DEFAULT" "" A.OUT A.OUT
fscript > ListFile A.OUT
>>> BEGIN LISTING <<<
 1>   USER:[TOM]
>>> END LISTING <<<
```

The following example shows how you could use the `ExecCmdRemote` command with a full path name and filename to invoke the iPlanet UDS Corbagen executable. This command must be invoked on one line:

```
ExecCmdRemote "FORTE_ROOT:[INSTALL.BIN.ALPHA]CORBAGEN /CORBA_TYPE=OBB
/IDL_FILE=NEW.IDL"
```

The `bg_flag` argument can be set to `0` to indicate that the command is to be run synchronously until it completes, or to `1` to indicate that the command is to be started in the background. By default, commands are run synchronously.

The `in_file`, `out_file`, and `err_file` arguments can be used to redirect the input, output, or errors for the command.

InstallApp

The `InstallApp` command installs partitions for an application on a node.

InstallApp *application_name* [*reinstall*]

Argument	Description
application_name	The name of an application that contains some partitions that have been enabled on the node. A special value of "`all`" installs all applications.
reinstall	`TRUE` or `FALSE`. If `TRUE`, this command reinstalls the application on the current node even if the application is already installed. The default is `FALSE`.

For Windows client nodes, or for any server nodes that were not running a Node Manager when an application was installed using the `Install` command from the Application agent, you can invoke the `InstallApp` command on the Node agent to install the partitions pending, either a single application or for all applications.

The current environment cannot be locked when you invoke the `InstallApp` command.

The `application_name` argument designates a specific application that contains enabled partitions for this node. For a list of applications in the environment, see the output for the `ShowAgent` on the Environment agent.

The `reinstall` argument specifies that this command should reinstall the application on the current node even if the application is already installed, if the argument is specified as `TRUE`. The default is `FALSE`.

You can also specify a special value of "`all`" for the `application_name` argument in order to install the partitions for all pending applications that the node has not yet installed.

ListAppsToInstall

The `ListAppsToInstall` command lists the names of the applications that need to be installed on the node represented by the current Node agent.

ListAppsToInstall

If a Node Manager was not available when an `Install` command for the Application agent initiated the process of installing an application in an environment, the uninstalled partitions for the application are recorded for later installation. After the Node Manager for that node is brought back online, you can use the `ListAppsToInstall` command to list the uninstalled applications for a node. You can also view the InstallationSteps instrument of the Application agent to see what nodes need further installation for a given application.

After listing the uninstalled applications, you can use the `InstallApp` command on the Node agent to install one (or all) of the pending applications.

ListDistribs

The `ListDistribs` command lists the application distributions available on the node managed by the current Node agent. This checks for distribution directories starting in `FORTE_ROOT/appdist/`*env_name* on the node where the Node Manager is executing.

ListDistribs

LoadDistrib

The `LoadDistrib` command loads the specified application distribution into the environment from the node represented by the current Node agent.

LoadDistrib *application_name compatibility_level*

Argument	Description
application_name	The name of the application to load.
compatibility_level	The compatibility level of the application to load.

The steps for installing an application are described under "Install" on page 146.

The `LoadDistrib` command loads an application into the environment repository from a distribution on the node managed by the current Node agent. The application distribution to be loaded must be in the `FORTE_ROOT/appdist/`*env_name* directory hierarchy on the node represented by the current Node agent.

The active environment cannot be locked when you invoke the `LoadDistrib` command.

The `application_name` argument specifies the name of the application to be loaded. The `compatibility_level` argument is the compatibility level number of the application to be loaded, prefixed with the letters "cl". These names are exactly as displayed in the `ListDistribs` command. These two arguments are used to find the application in the distribution directory tree. The first 8 characters of the `application_name` and the `compatibility_level` is used as the directory name. For example, you could invoke the following command out of environment named "dev" to load a distribution on the node currently running Escript:

```
escript> LoadDistrib MyFirstProject cl2
```

This would look for the application distribution in the following directory:

FORTE_ROOT/appdist/dev/myfirstp/cl2

The `LoadDistrib` command makes the loaded application's agent the current agent. After loading the application you can invoke the `Install` command to actually install the application.

ModLoggerRemote

The `ModLoggerRemote` command sets the logger flags for the Node Manager that is being managed by the current Node agent.

ModLoggerRemote +(*logger_flags*) | -(*logger_flags*)

Argument	Description
+(*logger_flags*)	Turn on the logger flag settings given in the parentheses.
-(*logger_flags*)	Turn off the logger flag settings given in the parentheses.

To start logging, use the '+' followed by a set of logger settings in parentheses. To stop logging, use the '-' followed by a set of logger settings in parentheses.

The `ModLoggerRemote` command changes the log settings in the Node Manager being managed by the current Node agent. To change the logger settings for the Environment Manager, do the following:

1. Execute the `FindActEnv` command.
2. Execute the `FindSubAgent` command to set the node that is running the Environment Manager.
3. Execute the `ModLoggerRemote` command to specify the log settings on the current node.

The settings modify the settings as they were when the Node Manager started execution. The original logger settings are set either with the `-fl` flag on the `nodemgr` command, or from the setting of the FORTE_LOGGER_SETUP environment variable that was set at the time the Node Manager (or Environment Manager) was started.

The modified logger settings are only applied to the first file specified in the original logger settings for the Node Manager (or Environment Manager).

See the `LogMgr` class in the Framework Library online Help for a detailed description of the logger flag syntax.

```
escript> ModLoggerRemote +(trc:os:1:1 cfg:c4:2-3:1)
escript> ModLoggerRemote -(cfg:c4)
```

SetEnvRemote

The `SetEnvRemote` command sets an environment variable value for the Node Manager represented by the current Node agent.

SetEnvRemote *env_variable new_value*

Argument	Description
env_variable	The name of an environment variable to set.
new_value	The new value of the environment variable to set.

The `SetEnvRemote` command changes the setting of the environment variable in the Node Manager that is managed by the current Node agent. To change an environment variable for the Environment Manager:

1. Execute the `FindActEnv` command.
2. Execute the `FindSubAgent` command to specify the node that is running the Environment Manager.
3. Execute the `SetEnvRemote` command to specify the log settings on the current node.

Using the `SetEnvRemote` command modifies the environment variable settings as they were set up when the Node Manager started execution. Because any servers or iPlanet UDS executors that are automatically started by the Node Managers use the current environment variable settings for the Node Manager (or Environment Manager), you can use the `SetEnvRemote` command to set environment variable settings that are picked up in any servers or iPlanet UDS interpreters started subsequently.

On UNIX and VMS nodes, the new setting of the environment variable does not remain beyond the current execution of the Node Manager. On Windows NT, where any execution of the Escript or `econsole` command acts as a Node Manager, the new setting is stored permanently, and is picked up in any client partition or iPlanet UDS application started at a later time.

Shutdown

The `Shutdown` command shuts down the Node Manager (Node agent).

Shutdown [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the shutdown to the subagents of the Node agent. The values are `0` to propagate (the default) or `1` to not propagate to subagents.

The Shutdown command shuts down the Node Manager (Node agent). By default, the shutdown request is propagated to all of the subagents of the Node agent. The most important of these are the Active Partition agents that are running any iPlanet UDS processes on this node. The Shutdown command therefore gives a simple way to shutdown all iPlanet UDS processes running on a node.

The optional no_propagate argument can be set to 1 if you want to shutdown the Node Manager only, but leave the executing partitions running. The Node Manager can then be brought back on-line at a later time, and connects the executing partitions. If you set no_propagate to 0, or leave it unspecified, the shutdown request propagates the shutdown request to the active partitions.

ShutdownSubAgent

The ShutdownSubAgent command shuts down the specified subagent. Usually, the subagents of a Node agent are Installed Partition agents.

ShutdownSubAgent *subagent*

Argument	Description
subagent	The name of a subagent to be shut down with its managed object.

If the named subagent does not exist, the ShutdownSubAgent command does nothing.

The ShutdownSubAgent command performs the same function as the following command sequence in an Escript script:

```
escript> FindSubAgent AutoCompileSvc_cl0_Part1_MIMI
escript> Shutdown
escript> FindParentAgent
```

The difference between this sequence of commands and using the ShutdownSubAgent command is that the Shutdown command is not invoked on the current agent if the subagent does not exist.

StartInstPart

The StartInstPart command starts one instance of an installed partition on the node managed by the current Node agent (Node Manager).

StartInstPart *partition_name [one_more]*

Argument	Description
partition_name	The name of an installed partition on the node.
one_more	Specifies that one partition be started up.

You can start a single instance of a partition on a specific node by invoking the `StartInstPart` command while the current agent is a Node agent. You can do the equivalent operation by using the `Startup` command on the Installed Partition agent (which also allows you to set the command line arguments).

The `partition_name` argument is the name of the Installed Partition that is to be started. Use the `ShowAgent` command for the Application, Node, or Partition agents for the set of names for the partitions.

The `one_more` argument is a boolean flag that specifies whether to start one partition (`TRUE`) or to start additional partitions until the number of running partitions matches the replication count (`FALSE`). If the number of running partitions already matches the replication count, and you do not specify `TRUE` for this argument, no more replicas of this partition are started. The default is `FALSE`.

You can use the `Startup` command in the Application and Logical Partition agents to start server partitions up to their predefined replication count. However, you can invoke the `StartInstPart` command while the current agent is a Node agent in order to start a single additional instance of a partition on a node. This is particularly useful for starting additional replicates of failover or load balanced partitions in an application at peak loads.

You can also use the `StartInstPart` to start iPlanet UDS executors that developers can use when they perform test runs from the Partition Workshop.

Instruments

Architecture

The `Architecture` instrument indicates the architecture for the node managed by the current agent. This Configuration instrument is read only.

ModelNode

The `ModelNode` instrument contains the model node name for the node that is managed by the current Node agent. This Configuration instrument is read only.

The `ModelNode` instrument shows the value of the model node that had been set up using the `SetNodeModel` command in the environment editing commands for `Escript`. If a node is not a member of a model group, then its Node agent does not have the `ModelNode` instrument.

StartupWaitTime

The `StartupWaitTime` instrument gets or sets the wait time on the node represented by the current Node agent before it assumes that partitions that it auto-starts have failed. The `StartupWaitTime` instrument is a Configuration instrument.

StartupWaitTime *interval_in_sec*

Argument	Description
interval_in_sec	The number of seconds to wait after autostart before the Node Manager assumes there was a failure. The default value is `120`.

The `StartupWaitTime` instrument sets the timeout interval for starting up partitions on the current Node agent. When the Node Manager starts a partition, it waits until the partition notifies it that the partition has successfully started. Because this startup is asynchronous, the `StartupWaitTime` instrument sets a timer to wait for the notification. If the notification does not come within a specified interval, iPlanet UDS assumes that the partition failed to start for some reason and raises and exception. The amount of time it waits is maintained in the `StartupWaitTime` instrument.

The `interval_in_sec` argument specifies the number of seconds to wait before the Node Manager assumes that an autostart has failed. By default the value is `120` seconds.

Programmatic Commands

GetPartAgent

The `GetPartAgent` command returns the Active Partition agent for the node manager partition.

GetPartAgent

After you retrieve the Active Partition agent for the Node agent, you can invoke Active Partition agent commands and use instrumentation on the partition containing the Node Manager.

GetRemoteFS

The `GetRemoteFS` command returns a FileSystem object whose current working directory is set to the iPlanet UDS root directory for the Node Manager.

GetRemoteFS

ObjectCache Agent

Parent Agent

RepositorySession agent

Subagents

none

SystemMonitor Class

SystemAgent

States

State	Description
ONLINE	The object cache is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propogate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Class	Description
AvgObjectsCollected	none	Yes	Average	Average number of objects cached that have been volunteered to be reclaimed for each memory reclamation.
CriticalCollects	none	Yes	Counter	Number of times that memory reclamation has indicated a critical memory condition.
ExpirationTime	*ticks*	No	Configuration	Number of "ticks" before a cached object is replaced in the cache by another when space is needed.
NumHits	none	Yes	Counter	Number of times an object was searched for and found.
NumMisses	none	Yes	Counter	Number of times an object was searched for but not found.
NumObjectsCached	none	Yes	Counter	Current number of objects stored in the cache.

Using the ObjectCache Agent

The ObjectCache agent represents the object cache used by the client repository session. This object cache contain objects that have been retrieved from the repository, or that are intended to be written to the repository.

The object cache improves performance by reducing the number of times that the client repository session needs to send messages to the repository server to read or write repository information.

The object cache keeps track of the last time that each object in the cache has been referenced. The `ExpirationTime` instrument specifies how many "ticks" occur after the object's last reference before the object is considered expired. Whenever the runtime system reclaims memory (runs garbage collection), the memory for expired objects is freed.

The number of cached objects in the object cache is proportional to the length of time that an object can be unreferenced in the cache before it expires. You can change the size of the object cache by changing the value of the ExpirationTime instrument, as described in "ExpirationTime" on page 264.

Parent and subagents The parent agent for an ObjectCache agent is a RepositorySession agent.

States

ONLINE

The object cache exists and is in use.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

AvgObjectsCollected

The `AvgObjectsCollected` instrument records the average number of cached objects that have become obsolete and available for memory reclamation. This Average instrument is read only.

CriticalCollects

The `CriticalCollects` instrument counts the number of times that a memory reclamation has indicated a critical memory condition. This Counter instrument is read only.

If this number is non-zero and increasing, increase the partition's memory setting or reduce the value of the `ExpirationTime` instrument to make more memory available to the development environment.

ExpirationTime

The `ExpirationTime` instrument specifies how many "ticks" occur after the object's last reference before the object is considered expired. Whenever the runtime system reclaims memory (runs garbage collection), the memory for expired objects is freed. The `ExpirationTime` instrument is a Configuration instrument.

ExpirationTime [*ticks*]

Argument	Description
ticks	Specifies the number of "ticks" before the cached object is considered replaceable by another object in the cache.

As a general rule, the higher the number of "ticks" before the cached object is considered replaceable, the higher the number of objects in the cache. Therefore, to reduce the size of the cache, reduce the value of the `ExpirationTime` instrument, so that more objects are reclaimed each time the runtime system performs memory reclamation (garbage collection).

NumHits

The `NumHits` instrument counts the number of times an object was searched for and found in the cache. This Counter instrument is read only.

NumMisses

The `NumMisses` instrument counts the number of times an object was searched for but not found in the cache. This Counter instrument is read only.

NumObjectsCached

The `NumObjectsCached` instrument counts the number of objects that are currently stored in the cache. This Counter instrument is read only.

OperatingSystem Agent

Parent Agent

Active Partition or RepositoryServer agent

Subagents

None

SystemMonitor Class

OperatingSystemAgent

States

State	Description
ONLINE	The operating system is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpMemory	none	Utility	Prints the state of garbage-collected memory to Stdout.
DumpMutexes	none	Utility	Writes information about all current mutex locks to the log file.
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
RecoverMemory	none	Utility	Attempts to perform a stable garbage collection.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
ActivePages	none	Yes	Counter	Current number of active pages in the memory manager.
AlarmInstalls	none	Yes	Counter	Number of alarms that have been set up on this operating system.
AllocatedPages	none	Yes	Counter	Current number of allocated pages in the memory heap.

Instrument	Argument	Read Only?	Type	Description
AllocationIncrement	none	Yes	Configuration	Number of pages that the memory manager expands when needed.
AvailablePages	none	Yes	Counter	Current number of available pages in the memory heap.
ContractAtPercent	*percent_value*	No	Configuration	Threshold at which the memory pool should be contracted.
ContractByPercent	*percent_value*	No	Configuration	Percent by which the memory pool should be contracted.
CumulativeRunnableThreads	none	Yes	Counter	Cumulative count of usable operating system threads that are waiting for the processor when iPlanet UDS polled for system activities.
DeadLockCheckInterval	*cycles*	No	Configuration	Specifies how often to check for deadlocks.
DeadThreads	none	Yes	Counter	Number of threads that have completed, and therefore have passed through the dead state.
ExpandAtPercent	*percent_value*	No	Configuration	Threshold at which the memory pool should be expanded.
ExpandByPercent	*percent_value*	No	Configuration	Percent by which the memory pool should be expanded.
FileBytesRead	none	Yes	Counter	Total number of file bytes that have been read using methods on the File class.
FileBytesWritten	none	Yes	Counter	Total number of file bytes that have been written using methods on the File class.

Instrument	Argument	Read Only?	Type	Description
FileCloses	none	Yes	Counter	Number of file close calls that have occurred during the life of the current partition.
FileOpens	none	Yes	Counter	Number of file open calls that have occurred during the life of this partition.
FileReads	none	Yes	Counter	Number of times the current partition has read from files using input methods of the `File` class.
FileSeeks	none	Yes	Counter	Number of times that seek operations were performed on iPlanet UDS controlled files during the life of this partition.
FileWrites	none	Yes	Counter	Number of times the current partition has written to files using output methods of the `File` class.
ForeignThreadAttaches	none	Yes	Counter	Number of threads that have been created by the DCE runtime library and attached to the iPlanet UDS DCE server as running tasks.
ForeignThreadDetaches	none	Yes	Counter	Number of threads that have been created by the DCE runtime library and attached to the iPlanet UDS DCE server as running tasks, then detached when the iPlanet UDS processing completes

Instrument	Argument	Read Only?	Type	Description
InterpreterSwitchingInterval	*number_instructions*	No	Configuration	Specifies how often the TOOL interpreter should switch between tasks that are running as interpreted TOOL code.
MaximumAllocation	*number_pages*	No	Configuration	Maximum number of pages that can be allocated.
MaxOpenFiles	none	Yes	Counter	Maximum number of open files allowed for this partition.
MinimumAllocation	*number_pages*	No	Configuration	Minimum number of pages that are allocated.
OpenFiles	none	Yes	Counter	Number of current open files for this partition.
PeakAllocatedPages	none	Yes	Counter	Maximum number of pages that have been allocated in the memory heap since the partition started.
RunnableThreads	none	Yes	Counter	Number of runnable threads currently in the iPlanet UDS thread manager.
ThreadsForked	none	Yes	Counter	Number of iPlanet UDS and operating system threads that have been created by iPlanet UDS during the life of the current partition.
ThreadSwitches	none	Yes	Counter	Number of iPlanet UDS thread switches that have occurred during the life of this partition.
ThreadSwitchInterval	*time_interval*	No	Configuration	Specifies how often iPlanet UDS switches processing from one thread to another.

Instrument	Argument	Read Only?	Type	Description
ThreadYields	none	Yes	Counter	Number of times during the life of this partition that any iPlanet UDS thread has yielded processing to another thread.
UtilizationPercent	*percent_value*	No	Configuration	Target percentage of the active memory heap that should be allocated to live pages.

Using the OperatingSystem Agent

The OperatingSystem agent is an agent that manages the local operating system services for an active partition. The operating system service provides memory management and other utility functions.

Parent and subagents The parent agent for the OperatingSystem agent is the Active Partition agent, as it runs in all active partitions. There are no subagents to the OperatingSystem agent.

NOTE Instruments of the OperatingSystem agent whose names start with "OSWait" are for iPlanet UDS internal use only, and are not useful for monitoring applications or the iPlanet UDS runtime system. These instruments might change in future iPlanet UDS releases, so do not use them in your applications.

Most of the instruments defined on the OperatingSystem agent are concerned with information and settings used by the memory manager for a partition. These are initially set by the `-fm` startup flag for the partition, although the `SetIntProperty` and `GetIntProperty` methods on the `OperatingSystem` class (described in the Framework Library online Help) also allow a TOOL program to set and get the values.

You can also use the `InterpreterSwitchingInterval` and `ThreadSwitchInterval` instruments to debug parts of your program that rely on the timing of multiple threads to perform correctly. You can use these instruments to change the frequency of switches between threads, which can reveal bugs based on how objects are defined as shared or not, incorrect use of event loops, and dependencies between tasks that can be affected by timing.

States

ONLINE

The operating system underlying the active partition is also running.

Commands

DumpMemory

The `DumpMemory` command writes information about the state of reclaimed (garbage-collected) memory to standard output.

DumpMemory

DumpMutexes

The `DumpMutexes` command writes information about all current mutex locks to the log file.

DumpMutexes

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

RecoverMemory

The `RecoverMemory` command attempts to perform a stable memory reclamation (garbage collection) on the current active partition.

RecoverMemory

Instruments

ActivePages

The `ActivePages` instrument represents the current number of active pages in the memory manager. This Counter instrument is read only.

The `ActivePages` instrument shows the number of pages currently in the memory manager memory heap. The size of a page is typically `1024 bytes`.

The active pages are all the pages in the memory heap, whether they are currently allocated for data or not. The `AllocatedPages` instrument indicates the current number of allocated pages with data references on them.

AlarmInstalls

The `AlarmInstalls` instrument represents the number of alarms that have been set up on the current operating system. The total number of alarms installed include reactivations of continuous alarms. This Counter instrument is read only.

AllocatedPages

The `AllocatedPages` instrument represents the current number of allocated pages in the memory heap. This Counter instrument is read only.

The `AllocatedPages` instrument shows the number of pages currently in the memory manager memory heap, and allocated to data. The size of a page is typically `1024 bytes`.

The allocated pages are the pages in the memory heap that have some amount of live data on them. This value is reset at the end of each memory reclamation. The `ActivePages` instrument indicates the current number of pages in the memory heap, whether they are currently allocated for data or not.

AllocationIncrement

The `AllocationIncrement` instrument is a read-write instrument that represents the number of pages that the memory manager expands when needed. The AllocationIncrement instrument is a Configuration instrument.

AllocationIncrement *number_pages*

Argument	Description
number_pages	The number of pages to increment when memory needs expansion.

The `AllocationIncrement` instrument specifies the incremental number of pages used if the memory manager needs to expand or contract the overall memory pool. This value is normally set by the "`i`" value on the `-fm` startup flag for the partition. The default value is `256`.

The `number_pages` argument specifies the number of pages to use in expanding or contracting memory. The legal values are from `64` to `1,048,576` .

AvailablePages

The `AvailablePages` instrument represents the current number of available pages in the memory heap. This Counter instrument is read only.

The `AvailablePages` instrument shows the number of pages currently in the memory manager memory heap that are available for reuse. The size of a page is typically `1024 bytes`.

The available pages are the pages in the memory heap that are available for use when needed. The value of the `AllocatedPages` instrument plus the value of the `AvailablePages` instrument does not necessarily add up to the number of the `ActivePages` instrument, but is approximately correct.

ContractAtPercent

The `ContractAtPercent` instrument is a read-write instrument that represents the threshold at which the memory pool should be contracted. The `ContractAtPercent` instrument is a Configuration instrument.

ContractAtPercent *percent_value*

Argument	Description
percent_value	The percentage value, from `0` through `100`, which represents when a contraction should occur.

The ContractAtPercent instrument specifies the UtilizationPercent \ that triggers a memory pool contraction. The UtilizationPercent instrument value is calculated after each memory reclamation. If the UtilizationPercent instrument value drops below the ContractAtPercent instrument value, the memory pool is contracted by the value of the ContractByPercent instrument.

The ContractAtPercent instrument value is normally set by the "c" value on the -fm startup flag for a partition. The default value is 20 percent.

The percent_value argument specifies the UtilizationPercent instrument value that triggers a contraction. Valid values are in the range 0 through 100.

ContractByPercent

The ContractByPercent instrument is a read-write instrument that represents the percent by which the memory pool should be contracted. The ContractByPercent instrument is a Configuration instrument.

ContractByPercent *percent_value*

Argument	Description
percent_value	The percentage value, from 0 through 100, by which the memory pool is contracted.

The ContractByPercent instrument specifies the percentage by which the memory pool is contracted when the UtilizationPercent value falls below the ContractAtPercent instrument value.

The ContractByPercent instrument value is normally set by the "s" value on the -fm startup flag for a partition. The default value is 5 percent.

The percent_value argument specifies the percentage by which the memory pool is contracted. Valid values are in the range 0 through 100.

CumulativeRunnableThreads

The CumulativeRunnableThreads instrument represents the cumulative count of usable operating system threads that were waiting for the processor when iPlanet UDS polled for system activities. This Counter instrument is read only.

This instrument can give you an idea of how much work was available when iPlanet UDS polled for additional work.

DeadLockCheckInterval

The `DeadLockCheckInterval` instrument is a read-write instrument that specifies how often to check for deadlocks. You can specify the number of scheduler cycles to occur between the deadlock checks. The `DeadLockCheckInterval` instrument is a Configuration instrument.

DeadLockCheckInterval *cycles*

Argument	Description
cycles	The number of scheduler cycles to occur between the deadlock checks. The default is `1500`. Set this value to `-1` to turn off the checking.

The `cycles` argument specifies the number of scheduler cycles to skip between the deadlock checks. The higher the number, the less frequent the deadlock checks. The default is `1500`. Set this value to `-1` to turn off the checking.

DeadThreads

The `DeadThreads` instrument represents the number of threads that have completed, and therefore have passed through the dead state. This Counter instrument is read only.

This instrument is similar to the `TasksTerminated` instrument of the TaskMgr agent (see "TasksTerminated" on page 335). Although threads do not actually die, their tasks do. The threads themselves are pooled to run other tasks.

ExpandAtPercent

The `ExpandAtPercent` instrument is a read-write instrument that represents the threshold at which the memory pool should be expanded. The `ExpandAtPercent` instrument is a Configuration instrument.

ExpandAtPercent *percent_value*

Argument	Description
percent_value	The percentage value, from `0` through `100`, which represents when an expansion should occur.

The `ExpandAtPercent` instrument specifies the `UtilizationPercent` value that triggers a memory pool expansion. The `UtilizationPercent` instrument value is calculated after each memory reclamation. If the `UtilizationPercent` value rises above the `ExpandAtPercent` value, the memory pool is expanded by the value of the `ExpandByPercent` instrument.

The `ExpandAtPercent` instrument value is normally set by the "`e`" value on the `-fm` startup flag for a partition. The default value is `80 percent`.

The `percent_value` argument specifies the `UtilizationPercent` value that triggers an expansion. Valid values are in the range `0` through `100`.

ExpandByPercent

The `ExpandByPercent` instrument is a read-write instrument that represents the percent by which the memory pool should be expanded. The `ExpandByPercent` instrument is a Configuration instrument.

ExpandByPercent *percent_value*

Argument	Description
percent_value	The percentage value, from `0` through `100`, by which the memory pool is expanded.

The `ExpandByPercent` instrument specifies the percentage by which the memory pool is expanded when the `UtilizationPercent` value rises above the `ExpandAtPercent` value.

The `ExpandByPercent` instrument value is normally set by the "`g`" value on the `-fm` startup flag for a partition. The default value is `10 percent`.

The `percent_value` argument specifies the percentage by which the memory pool is contracted. Valid values are in the range `0` through `100`.

FileBytesRead

The `FileBytesRead` instrument represents the total number of file bytes that have been read using methods on the `File` class. This Counter instrument is read only.

FileBytesWritten

The `FileBytesWritten` instrument represents the total number of file bytes that have been written using methods on the `File` class. This Counter instrument is read only.

FileCloses

The `FileCloses` instrument represents the number of file close calls that have occurred during the life of the current partition. These file close calls correspond to `Close` method calls on objects of the `File` class. This Counter instrument is read only.

FileOpens

The `FileOpens` instrument represents the number of file open calls that have occurred during the life of the current partition. These file open calls correspond to `Open` method calls on objects of the `File` class. This Counter instrument is read only.

FileReads

The `FileReads` instrument represents the number of times the current partition has read from files using input methods of the `File` class. This Counter instrument is read only.

FileSeeks

The `FileSeeks` instrument represents the number of times that seek operations were performed on all iPlanet UDS controlled files during the life of this partition. This Counter instrument is read only.

Seek is an input output function that repositions a file pointer.

FileWrites

The `FileWrites` instrument represents the number of times the current partition has written to files using output methods of the `File` class. This Counter instrument is read only.

ForeignThreadAttaches

The `ForeignThreadAttaches` instrument represents the number of threads that have been created by the DCE runtime library and attached to the iPlanet UDS DCE server as running tasks. These DCE threads are started in response to remote procedure calls (RPCs) invoked by DCE clients or other servers. This Counter instrument is read only.

ForeignThreadDetaches

The `ForeignThreadDetaches` instrument represents the number of threads that have been created by the DCE runtime library and attached to the iPlanet UDS DCE server as running tasks, then detached when the iPlanet UDS processing completes. These DCE threads were started in response to remote procedure calls (RPCs) invoked by DCE clients or other servers. This Counter instrument is read only.

InterpreterSwitchingInterval

The `InterpreterSwitchingInterval` instrument is a read-write instrument that specifies how often the TOOL interpreter should switch between tasks that are running as interpreted TOOL code. The `InterpreterSwitchingInterval` instrument is a Configuration instrument.

You can specify the number of interpreter instructions that the TOOL interpreter should execute before switching to another task. *Interpreter instructions* are the internal representation of TOOL statements, as presented to the interpreter. On average, four internal instructions map to one TOOL statement.

InterpreterSwitchingInterval *number_instructions*

Argument	Description
number_instructions	The number of interpreter instructions executed before an interpreter switches to another task.

Specifying a lower number of internal instructions means that iPlanet UDS switches among tasks more frequently, and the tasks are more concurrent. Conversely, a higher number means that iPlanet UDS switches tasks less frequently.

The `number_instructions` argument specifies the number of interpreter instructions executed before an interpreter switches to another task. The default is `8000`, which maps to approximately 2000 TOOL statements.

MaximumAllocation

The `MaximumAllocation` instrument is a read-write instrument that represents the maximum number of pages that can be allocated. The `MaximumAllocation` instrument is a Configuration instrument.

MaximumAllocation *number_pages*

Argument	Description
number_pages	The maximum number of pages that can be allocated.

The `MaximumAllocation` instrument specifies the absolute maximum number of pages that can be allocated to the memory heap in the current partition. If an attempt is made to expand the heap to allocate more pages than the `MaximumAllocation` instrument allows, the partition gets a fatal error. This value is normally set by the "`x`" value on the `-fm` startup flag for the partition. The default value is `8192`.

The `number_pages` argument specifies the maximum number of pages that are allowed in the memory pool. Valid values are in the range `1024` through `4194304` . The value also must be greater than the value for the `MinimumAllocation` instrument.

MaxOpenFiles

The `MaxOpenFiles` instrument represents the maximum number of open files allowed for this partition. This Counter instrument is read only.

MinimumAllocation

The `MinimumAllocation` instrument is a read-write instrument that represents the minimum number of pages that are allocated. The `MinimumAllocation` instrument is a Configuration instrument.

MinimumAllocation *number_pages*

Argument	Description
number_pages	The minimum number of pages that are allocated.

The `MinimumAllocation` instrument specifies the absolute minimum number of pages that are allocated to the memory heap in the current partition. It also represents the number of pages that are allocated when the partition first starts. If an attempt is made to contract the heap to allocate fewer pages than the `MinimumAllocation` instrument allows, no contraction is done. This value is normally set by the "`n`" value on the `-fm` startup flag for the partition. The default value is `1024`.

The number_pages argument specifies the minimum number of pages that are allowed in the memory pool. Valid values are in the range 1024 through 4194304. The value also must be less than the value for the MaximumAllocation instrument.

OpenFiles

The OpenFiles instrument represents the number of current open files for this partition. This Counter instrument is read only.

PeakAllocatedPages

The PeakAllocatedPages instrument represents the maximum number of pages that have been allocated in the memory heap since the partition started. This Counter instrument is read only.

The PeakAllocatedPages instrument shows the maximum number of pages in the memory heap allocated to data, since the partition started. The size of a page is typically 1024 bytes.

The peak allocated pages are the pages in the memory heap that have some amount of live data on them. The ActivePages instrument indicates the current number of pages in the memory heap, and the AllocatedPages instrument shows the number of pages allocated at the end of the last memory reclamation. If PeakAllocatedPages is always significantly less than ActivePages, you might be able to run with a smaller value for the MaximumAllocation, and conserve on memory. However, because PeakAllocatedPages is measured *after* a memory reclamation, the actual peak is probably higher.

RunnableThreads

The RunnableThreads instrument represents the number of runnable threads currently in the iPlanet UDS thread manager. You can use this value as a measure of how busy the partition is. This Counter instrument is read only.

ThreadsForked

The ThreadsForked instrument represents the number of iPlanet UDS and operating system threads that have been created by iPlanet UDS during the life of the current partition. The ThreadsForked instrument does not count other threads that might have been created by the operating system or DCE. This Counter instrument is read only.

ThreadSwitches

The ThreadSwitches instrument represents the number of iPlanet UDS thread switches that have occurred during the life of the current partition. This Counter instrument is read only.

ThreadSwitchInterval

The `ThreadSwitchInterval` instrument is a read-write instrument that specifies how often iPlanet UDS switches processing from one thread to another. You can specify the number of microseconds that iPlanet UDS allows a thread to run before switching to another runnable thread. The `ThreadSwitchInterval` instrument controls the rate at which iPlanet UDS switches between preemptive iPlanet UDS threads. Preemptive iPlanet UDS threads are threads that interrupt the processing of other threads when those threads yield the processor.

The `ThreadSwitchInterval` does not affect the rate of switching for iPlanet UDS threads that are not preemptive, for example, iPlanet UDS client threads. It also does not affect switching of native threads. The `ThreadSwitchInterval` instrument is a Configuration instrument.

ThreadSwitchInterval *time_interval*

Argument	Description
time_interval	The length of time that iPlanet UDS allows a thread to run before switching to another runnable thread, in microseconds.

The `time_interval` argument specifies the length of time, in microseconds, that iPlanet UDS allows a thread to run before switching to another runnable thread. The `ThreadSwitchInterval` instrument works for iPlanet UDS preemptive threads only. The default value is `20000`. (`20 milliseconds`). An interval less than `10000` is not recommended.

ThreadYields

The `ThreadYields` instrument represents the number of times during the life of the current partition that any iPlanet UDS thread has yielded processing to another thread. This Counter instrument is read only.

UtilizationPercent

The `UtilizationPercent` instrument is a read-write instrument that represents the target percentage of the active memory heap that should be allocated to live pages. The `UtilizationPercent` instrument is a Configuration instrument.

UtilizationPercent *percent_value*

Argument	Description
percent_value	The percentage value, from 30 through 90, which represents the target average for active page utilization.

The UtilizationPercent instrument specifies the target percentage usage of the memory heap, calculated as the ratio of allocated pages to active pages (multiplied by 100). When the calculated value exceeds the target value, a stable memory reclamation is performed. After this memory reclamation has completed, the UtilizationPercent value is recalculated.

If the recalculated value is greater than the ExpandAtPercent instrument value, an expansion is performed. The expansion is equal to the ExpandByPercent value multiplied by the ActivePages value. If the calculated value of UtilizationPercent is less than the ContractAtPercent instrument valued, a contraction is performed. The contraction is equal to the ExpandByPercent value multiplied by the ActivePages value. (This second feature is platform dependent.)

This UtilizationPercent value is normally set by the "u" value on the -fm startup flag for a partition. The default value is 85 percent.

The percent_value argument specifies the target percentage usage of the memory pool. Valid values are in the range 25 through 95.

Partition Agent

Parent Agent

Application agent

Subagents

Installed Partition agent

SystemMonitor class

GenericPartitionAgent

States

State	Description
DEGRADED	At least one installed partition belonging to this partition is DEGRADED, or not all installed partitions that should be running are ONLINE.
OFFLINE	All installed partitions belonging to this partition are OFFLINE.
ONLINE	Minimum installed partitions are ONLINE.

Command Summary

Command	Arguments	Environment Console menu	Description
Assign	*node_name*	none	Assigns the partition represented by the current Partition agent for installation on the specified node.
Disable	*node_name*	none	Disables autostart on a specific node for the partition managed by the current Partition agent.
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Enable	*node_name*	none	Enables autostart for the partition managed by the current Partition agent on a specific node.

Command	Arguments	Environment Console menu	Description
ModLoggerRemote	+(*logger_flags*) -(*logger_flags*)	Component > Modify Log Flags	Sets the logger flags for all of the active partitions that are represented by the current Partition agent.
SetArgs	*node_name* *arguments*	none	Sets the argument string used to startup the partition managed by the current Partition agent on a particular node.
SetCompiled	*node_name* *is_compiled*	none	Turns on or off the compiled server attribute for the partition managed by the current Partition agent on a specified node.
SetEnvRemote	*env_variable* *new_value*	Component	Sets the environment variable for all of the active instances of the partition managed by the current Partition agent.
SetRepCount	*node_name* *replication_count*	none	Sets the replication count on a node for the partition managed by the current Partition agent
Shutdown	*kill_executors*	Component	Shuts down all active instances of the partition represented by the current Partition agent. If the `kill_executors` argument is set to `TRUE` or `1`, any interpreted partitions will shut down their hosting iPlanet UDS executor (ftexec or ftexecd) process as well.
ShutdownSubAgent	*subagent*	none	Shuts down the named subagent and its managed object.

Command	Arguments	Environment Console menu	Description
Startup	none	Component	Starts all server partitions (with all their replicates) for the partition managed by the current Partition agent.
Unassign	*node_name*	none	Dissolves the node assignment of the partition managed by the current Partition agent.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
CanBeActivated	none	Yes	Configuration	Indicates whether the partition is enabled for startup by the management system.

Using the Partition Agent

The Partition agent represents one logical partition within an application across all nodes.

Parent and subagents The parent agent of the Partition agent is the Application agent. The subagents of the Partition agent are the Installed Partition agents, which represent the logical partition as installed on a node.

States

DEGRADED

At least one installed partition belonging to this partition is DEGRADED, or not all enabled installed partitions that should be running are ONLINE.

OFFLINE

All installed partitions belonging to this partition are OFFLINE.

ONLINE

If this Partition agent represents a shared server partition, ONLINE means that all enabled installed partitions are ONLINE.

If this Partition agent represents a private server partition or a client partition, ONLINE means that at least one installed partition is ONLINE.

Commands

Assign

The `Assign` command assigns the partition represented by the current Partition agent to the specified node for installation.

Assign *node_name*

Argument	**Description**
node_name	The name of the node where the partition is to be assigned for installation.

Use the `Assign` command to assign the partition represented by the current Partition agent for installation on a node within the environment. Before invoking the `Assign` command, you must lock the environment by invoking the `LockEnv` command.

The `node_name` argument must be a valid node defined in the environment. The node must have all of the external resource managers, communications protocols, and libraries needed to support the partition.

When an application is partitioned, each partition is designated for future installation and execution on one or more nodes in the environment, based on matching the needed properties of the partition and the actual properties of each node. You can use the `Assign` command to designate additional nodes where a partition can be installed.

If you assign a partition that is not replicated to more than one node, the partition starts on only one of the assigned nodes. You can use the `Enable` command to designate the node on which the partition is automatically started. By assigning the partition for installation on more than one node, you can provide manual failover for a partition that has failed. Manual failover means using the Environment Console or invoking the `StartInstPart` command on the Node agent or the `Startup` command on the Installed Partition agent. You can use these steps to start a partition on any of the nodes to which the partition has been assigned and successfully installed.

Disable

The `Disable` command disables auto-start on a node for the partition managed by the current Partition agent.

Disable *node_name*

Argument	Description
node_name	The node on which the partition is to be disabled.

The `Disable` command removes the auto-start capabilities of a partition on a particular node. You use the `Disable` command when you want to assign a partition to a node, but do not want the server on that node to auto-start when the `Startup` command for the Application agent is invoked. Therefore, you should disable auto-start partitions that you want installed on backup nodes, but that should not run until needed.

Before invoking the `Disable` command, you must lock the environment by invoking the `LockEnv` command.

The `node_name` argument is the name of a node in the environment to which the partition has already been assigned, either as part of the default partitioning, or through the `Assign` command on the Partition agent, or through the `AssignAppComp` command on the Application agent.

A partition within an application can be assigned to one or more nodes within the environment. For non-replicated partitions that are to be placed on servers, only one of the assigned nodes can be designated the node on which the partition is to be automatically started and managed. This is done by the default partitioning, or by invoking the `Enable` command for the Partition agent, or through the `EnableAppComp` command for the Application agent. For replicated partitions, any number of assigned nodes can be enabled using the `Enable` command, and servers are automatically started on all of the enabled nodes.

The `DisableAppComp` command on the Application agent provides equivalent functionality to the `Disable` command on the Partition agent.

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Enable

The `Enable` command enables auto-start for the partition managed by the current Partition agent.

Enable *node_name*

Argument	Description
node_name	The node on which the partition is be enabled.

The `Enable` command turns on the auto-start capabilities of a partition on a particular node. You use the `Enable` command when you want to assign a node for a partition and want the server on that node to auto-start when the `Startup` command for the Application agent is invoked. Normally, this is the default for partitions assigned to a node, so you do not need to explicitly invoke the `Enable` command.

Before invoking the `Enable` command, you must lock the environment by invoking the `LockEnv` command.

The `node_name` argument is the name of a node in the environment to which the partition has already been assigned, either as part of the default partitioning or through the `AssignPart` command.

A partition within an application can be assigned to one or more nodes within the environment. For non-replicated partitions that are to be placed on servers, only one of the assigned nodes can be designated the node on which the partition is to be automatically started and managed. This is done by the default partitioning, or by invoking the `Enable` command on the Partition agent, or the `EnableAppComp` command on the Application agent. For replicated partitions, any number of assigned nodes can be enabled using the `Enable` command, and servers are automatically started on all of the enabled nodes.

The `EnableAppComp` command on the Application agent provides equivalent functionality to the `Enable` command on the Partition agent.

ModLoggerRemote

The `ModLoggerRemote` command sets the logger flags for all of the active partitions that are represented by the current Partition agent.

ModLoggerRemote +(*logger_flags*) | -(*logger_flags*)

Argument	Description
+(*logger_flags*)	Turn on the logger flag settings given in the parentheses.
-(*logger_flags*)	Turn off the logger flag settings given in the parentheses.

To start logging, invoke the `ModLoggerRemote` command using the '+' followed by a set of logger settings in parentheses. To stop logging, use the '-' followed by a set of logger settings in parentheses.

The logger flag settings in the `ModLoggerRemote` command modify any logger flag settings that were specified for the partition, either in the `-fl` startup flag or by the FORTE_LOGGER_SETUP environment variable.

The modified logger settings are only applied to the first file specified in the original logger settings for the Node Manager (or Environment Manager).

See the `LogMgr` class in the Framework Library online Help for a detailed description of the logger flag syntax.

```
escript> ModLoggerRemote +(trc:os:1:1 cfg:c4:2-3:1)
escript> ModLoggerRemote -(cfg:c4)
```

SetArgs

The `SetArgs` command sets the argument string used to startup the partition managed by the current Partition agent on a particular node.

SetArgs *node_name*

Argument	Description
node_name	The name of the node on which the partition is assigned.
arguments	The command line arguments for the partition.

The `SetArgs` command lets you specify specific startup command line flags for a partition on a specific node. The command operates on the partition managed by the current Partition agent, but for one particular node.

Before invoking the `SetArgs` command, you must lock the environment by invoking the `LockEnv` command.

The `node_name` argument specifies one of the nodes to which the partition has been assigned.

The `arguments` argument specifies a set of command line arguments to send to the partition when it first starts. These are given as UNIX style command line arguments on all system, using the '-' to designate the option, and then the value. Since you are usually providing several arguments, you should enclose the set of arguments in double quotes.

On UNIX platforms, you usually need to enclose arguments that contain parentheses in single quotes so that the operating system can parse the flags correctly. The exception to this rule is the `-fm` flag arguments, which do not need to be enclosed in single quotes when there are no spaces in that flag, as shown in the following example.

NOTE	The third escript command below is invoked on one line

```
escript> FindApp Acctg
escript> FindPart AcctMgr
escript> SetArgs myserver '-fl "%stdout(trc:user)"
-fm"(n:4000,x:8000)"'
```

For explanations of the `-fl` and `-fm` flags, see *iPlanet UDS System Management Guide*.

SetCompiled

The `SetCompiled` command sets the compiled attribute for the partition managed by the current Partition agent on a specified node.

SetCompiled *node_name is_compiled*

Argument	Description
node_name	The name of the node on which the partition is assigned.
is_compiled	Set to `TRUE` if the compiled partition is to be used, or `FALSE` if the iPlanet UDS executor is to be used.

When you create a distribution for an application, you can specify which partitions on certain nodes are to be compiled. By default, the compiled setting is preserved when the application is installed in an environment. However, when you troubleshoot problems, you might find it useful to run a server as a standard iPlanet UDS executor partition for a time if there are problems in running the compiled version of the partition. The `SetCompiled` command can change a partition from compiled to standard and back.

Before invoking the `SetCompiled` command, you must lock the environment by invoking the `LockEnv` command.

The `node_name` argument specifies one of the nodes to which the partition has been assigned. When you made the distribution for the application, you must have created the compiled version of this partition for this node type.

The `is_compiled` argument values are `TRUE` for using the compiled version of the server, or `FALSE` for using the standard iPlanet UDS executor version.

SetEnvRemote

The `SetEnvRemote` command sets the environment variable for all of the active instances of the partition managed by the current Partition agent.

SetEnvRemote *env_variable new_value*

Argument	Description
env_variable	The name of an environment variable to set.
new_value	The new value of the environment variable to set.

The `SetEnvRemote` command changes the setting of the specified environment variable in all instances of the active partition managed by the current Partition agent. Within the TOOL code executing in that partition, any subsequent invocation of the `GetEnv` method on the OperatingSystem object gets the new setting.

The `env_variable` argument is the name of an environment variable to set in the process running the active partition, and the `new_value` argument is the value for the environment variable.

Using `SetEnvRemote` modifies the environment variable settings as they were set up when the partition started execution. On UNIX and VMS nodes, the new setting of the environment variable does not remain beyond the current execution of the partition. On Windows NT, the new setting is stored permanently and is picked up in any client partition or iPlanet UDS application started at a later time, because the value is stored in the registry in Windows NT.

SetRepCount

The `SetRepCount` command sets the replication count on a node for the partition managed by the current Partition agent.

SetRepCount *node_name replication_count*

Argument	Description
node_name	The name of the node on which the partition is assigned.
replication_count	The new replication count for autostart of servers for the partition.

When you create a partition that contains a replicated service object, either for failover or for load balancing, you can specify a replication count for the partition. This value specifies the number of replicates of the partition to start when the partition is auto-started with the `Startup` command on the Application agent. You can use the `SetRepCount` command to change the default setting for the replication count for the partition assigned to a specific node.

Before invoking the `SetRepCount` command, you must lock the environment by invoking the `LockEnv` command.

The `node_name` argument specifies one of the nodes to which the partition has been assigned.

The `replication_count` argument specifies the number of replicates of the partition to auto-start on that node when the `Startup` command on the Application agent is executed. You can only set the replication count for partitions that contain replicated service objects.

Shutdown

The `Shutdown` command shuts down all active instances of the partition represented by the current Partition agent.

Shutdown *kill_executors*

If the `kill_executors` argument is set to `TRUE` or `1`, any interpreted partitions will shut down their hosting iPlanet UDS executor (ftexec or ftexecd) process as well.

ShutdownSubAgent

The `ShutdownSubAgent` command shuts down the specified Installed Partition agent.

ShutdownSubAgent *subagent*

Argument	Description
subagent	The name of a subagent to be shut down with its managed object.

If the named subagent does not exist, the `ShutdownSubAgent` command does nothing.

The `ShutdownSubAgent` command performs the same function as the following command sequence in an Escript script:

```
escript> FindSubAgent AutoCompileSvc_cl0_Part1_MIMI
escript> Shutdown
escript> FindParentAgent
```

The difference between this sequence of commands and using the `ShutdownSubAgent` command is that the `Shutdown` command is not invoked on the current agent if the subagent does not exist.

Startup

The `Startup` command starts all server partitions (with all their replicates) for the partition managed by the current Partition agent.

Startup

The `Startup` command directs the Node Managers to start all server partitions for the partition managed by the current Partition agent. For each node that contains an installed and enabled version of the partition, the partition is automatically started. If an installed partition has a partition replication count, that many instances of the partition are started on the node. If some instances of the partition are already running, this command brings up as many new partitions as are necessary to bring it back to its defined level of partitions and replicates. If all partitions are already running at their defined levels, this command does nothing.

You are not required to invoke the `Startup` command to start the servers for an installed application, because the first client attempting to connect to an application triggers the auto-startup of the minimum number of partitions needed for an application to run. However, this can be time consuming, and does not provide for load balancing or failover protection. It is better to start the partitions for an application when you start the runtime system.

You can also invoke the `Startup` command on the Application agent to start all enabled installed partitions in an application up to their replication count, or invoke the `StartInstPart` command on the Node agent to start a single instance of an installed partition on a node.

Unassign

The `Unassign` command removes the node assignment of the partition managed by the current Partition agent.

Unassign *node_name*

Argument	Description
node_name	The name of the node for which the partition is to be unassigned.

The `Unassign` command removes a partition from a future installation on a specific node in the environment.

Before invoking the `Unassign` command, you must lock the environment by invoking the `LockEnv` command.

The `node_name` argument must be a valid node defined in the environment, and must currently have the partition assigned to it.

See "Assign" on page 286 for more information on assigning partitions to nodes.

Instruments

CanBeActivated

The `CanBeActivated` instrument indicates whether the partition represented by the current agent is enabled for startup by the management system. This Configuration instrument is read only.

Process Agent

Parent Agent

Active Partition agent

Subagents

None

SystemMonitor Class

ProcessAgent

States

State	**Description**
ONLINE	The process is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
CachePageFaults	none	Yes	Counter	Page faults that were satisfied by the buffer cache.
ContextSwitches	none	Yes	Counter	Times this process switched off the processor.
CpuUtilizationPercent	none	Yes	Counter	Percentage of the CPU used by this process over last sample.
DiskPageFaults	none	Yes	Counter	Page faults satisfied by the disk.
ElapsedExecutionSeconds	none	Yes	Counter	Seconds since process started.
InputWaits	none	Yes	Counter	Times process blocked waiting for input.
MessagesReceived	none	Yes	Counter	Operating system IPC messages received by this process.
MessagesSent	none	Yes	Counter	Operating system IPC messages sent by this process.
OutputWaits	none	Yes	Counter	Times process blocked waiting to complete output.
ResidentSizeKB	none	Yes	Counter	Resident size of the process in physical memory.

Instrument	Argument	Read Only?	Type	Description
Swaps	none	Yes	Counter	Number of process swaps, when a process was moved to virtual memory.
SystemCpuMicroseconds	none	Yes	Counter	Microseconds, in addition to the seconds specified by `SystemCpuSeconds`, spent executing system calls.
SystemCpuSeconds	none	Yes	Counter	Seconds spent executing system calls for this process.
TotalSizeKB	none	Yes	Counter	Size of the entire process in virtual memory.
UserCpuMicroseconds	none	Yes	Counter	Microseconds, in addition to the seconds specified by `UserCpuSeconds`, spent executing the user application.
UserCpuSeconds	none	Yes	Counter	Seconds spent executing the user application.

Using the Process Agent

The Process agent represents the operating system process in which the partition runs.

To determine the total number of seconds the process has been executed by the processor, add the values of the `System CpuMicroseconds`, `SystemCpuSeconds`, `UserCpuMicroseconds`, and `UserCpuSeconds` instruments.

Certain instruments are not available on some platforms. If an instrument value is not available for a Machine agent, the value is set to `-1`.

Parent agents and subagents The parent agent for a Process agent is an Active Partition agent. A process agent has no subagents.

States

ONLINE

The process underlying the active partition is also running.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	**Description**
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

CachePageFaults

The `CachePageFaults` instrument indicates how many times a page of memory was accessed from the buffer cache when a page fault occurred. This Counter instrument is read only.

If this value gets very high, then the user might be running too many processes concurrently.

ContextSwitches

The `ContextSwitches` instrument indicates how many times the processor has switched from this process to another process. This Counter instrument is read only.

A high value can indicate that too many concurrent processes are running and that many system calls are being executed. To reduce the amount of switching, TOOL programmers can reduce the number of distributed calls.

CpuUtilizationPercent

The `CpuUtilizationPercent` instrument indicates the average percentage of the CPU processing power that was used over the last sample interval. This Counter instrument is read only.

The sample interval is the frequency that this instrument is updated, as set by the `LogTimer` instrument for the active partition. For more information about the `LogTimer` instrument, see *iPlanet UDS System Management Guide*.

DiskPageFaults

The `DiskPageFaults` instrument indicates how many times a page of memory was accessed directly from a mounted volume when a page fault occurred. This Counter instrument is read only.

This instrument can indicate that the machine has insufficient memory for the current work load.

ElapsedExecutionSeconds

The `ElapsedExecutionSeconds` instrument indicates total number of seconds of real time that this process has been running. This Counter instrument is read only.

You can use this instrument to identify processes that are in endless loops.

InputWaits

The `InputWaits` instrument indicates how many times a process blocked to wait for input. This Counter instrument is read only.

MessagesReceived

The `MessagesReceived` instrument indicates the number of interprocess operating system messages received by this process. These messages can be between user processes, or between the operating system and the process. This Counter instrument is read only.

MessagesSent

The `MessagesSent` instrument indicates the number of interprocess operating system messages sent by this process. These messages can be between user processes, or between the operating system and the process. This Counter instrument is read only.

OutputWaits

The `OutputWaits` instrument indicates how many times a process blocked to wait to complete output. This Counter instrument is read only.

ResidentSizeKB

The `ResidentSizeKB` instrument indicates the size of the process, in kilobytes, that is loaded in physical memory. The rest of the process is resident in virtual memory or has not been paged in from the executable. This Counter instrument is read only.

Swaps

The `Swaps` instrument indicates how many times a process was swapped out to virtual memory on a local storage volume. This Counter instrument is read only.

SystemCpuMicroseconds

The `SystemCpuMicroseconds` instrument indicates the number of microseconds—beyond the number of seconds specified by the `SystemCpuSeconds` instrument—that the processor has spent executing system calls for this process. This Counter instrument is read only.

The total number of seconds that the processor has executed system calls for this process is the sum of the `SystemCpuSeconds` and `SystemCpuMicroseconds` instruments.

SystemCpuSeconds

The `SystemCpuSeconds` instrument indicates the number of seconds that the processor has spent executing system calls for this process. This Counter instrument is read only.

The total number of seconds that the processor has executed system calls for this process is the sum of the `SystemCpuSeconds` and `SystemCpuMicroseconds` instruments.

TotalSizeKB

The `TotalSizeKB` instrument indicates the total size of this process in virtual memory. This Counter instrument is read only.

UserCpuMicroseconds

The `UserCpuMicroseconds` instrument indicates the number of microseconds—beyond the number of seconds specified by the `UserCpuSeconds` instrument—that the processor has spent executing the user's application in this process. This Counter instrument is read only.

The total number of seconds that the processor has executed the user's application in this process is the sum of the `UserCpuSeconds` and `UserCpuMicroseconds` instruments.

UserCpuSeconds

The `UserCpuSeconds` instrument indicates the number of seconds that the processor has spent executing the user's application in this process. This Counter instrument is read only.

The total number of seconds that the processor has executed the user's application in this process is the sum of the `UserCpuSeconds` and `UserCpuMicroseconds` instruments.

Repository Agent

Parent Agent

Active Partition or RepositoryServer agent

Subagents

BtreeRepository agent

SystemMonitor Class

SystemAgent agent

States

State	**Description**
ONLINE	The repository is open and in use.
FAULT	The repository switched to read-only mode because of an error.
RUNDOWN	The repository has closed, but the agent has not yet been removed.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Class	Description
BytesRead	none	Yes	Counter	Number of bytes read since the repository was opened.
BytesWritten	none	Yes	Counter	Number of bytes written since the repository was opened.
ObjectsRead	none	Yes	Counter	Number of objects read since the repository was opened.
ObjectsWritten	none	Yes	Counter	Number of objects written since the repository was opened.

Using the Repository Agent

The Repository agent represents a repository managed by the repository server, node manager, client shadow, and so forth.

Parent and subagents The parent agent for a Repository agent is either an Active Partition agent or a RepositoryServer agent when the repository was opened by a repository server. If this repository is a B-tree repository, this agent has a BtreeRepository subagent.

There are typically no Repository agents for a client connected directly to a central repository. There is one Repository agent for a repository server, node manager, or client using a shadow or private repository.

Repository agents have names of the form "**Repository**_*repository_type*:*base_file_name*," where the *repository_type* is one of the following:

bt B-tree repository

ct C-tree repository

The `base_file_name` is the name of the repository files, without an extension.

For example, a B-tree repository whose files are called central.btx and central.btd would have an agent named "Repository_bt:central."

States

ONLINE

The repository is open and in use.

FAULT

The repository switched to read-only mode because of an error. Check the log file for the active partition using the repository to determine what error occurred.

On a central repository, check the .rop file for error and historical information. You probably need to restart the repository server after you fix the problem.

RUNDOWN

The repository has closed, but the agent has not yet been removed from the agent hierarchy.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

BytesRead

The `BytesRead` instrument counts the number of bytes read since the repository was opened. This Counter instrument is read only.

BytesWritten

The `BytesWritten` instrument counts the number of bytes written since the repository was opened. This Counter instrument is read only.

ObjectsRead

The `ObjectsRead` instrument counts the number of objects read since the repository was opened. This Counter instrument is read only.

ObjectsWritten

The `ObjectsWritten` instrument counts the number of objects written since the repository was opened. This Counter instrument is read only.

RepositoryServer Agent

Parent Agent

Ad hoc partition agent

Subagents

BtreeCache agent, Repository agent, and all iPlanet UDS runtime system agents, such as DistObjectMgr agent, TaskMgr, and TransactionMgr agent.

SystemMonitor Class

SystemAgent agent

States

State	Description
ONLINE	The repository server is running.
RUNDOWN	The repository server is shutting down.

Command Summary

Command	Arguments	Environment Console menu	Description
DebugPartition	none	Special	Places this partition under the control of a C++ debugger.
DumpStatus	*no_propagate*	Component	Prints the status of the repository server to Stdout.
FlushLogFiles	none	Component	Flushes all of this partition's log files.
ForceShutdown	none	Utility	Stops the repository server, even if users might still be connected.

Command	Arguments	Environment Console menu	Description
ForceWorkspaceUnreserved	*workspace* password	Utility	Removes the reservation a detached shadow holds on the workspace.
ModLoggerRemote	+(*logger_flags*) –(*logger_flags*)	Component > Modify Log Flags	Sets the logger flags for the active partition. If you are invoking this command within TOOL code, use the `ModLogger` command.
SetEnvRemote	*env_variable new_value*	Component	Sets an environment variable for the active partition. If you are invoking this command within TOOL code, use the `SetEnvVar` command.
Shutdown	none	Component	Shuts down the repository server.
UnlockWorkspace	workspace password	Utility	Frees all locks held on the given workspace.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
CanBeActivated	none	Yes	Configuration	Indicates whether the partition is enabled for startup by the management system.

Instrument	Argument	Read Only?	Type	Description
GlobalLockCount	none	Yes	Counter	Number of global locks currently held on the central repository.
GlobalLocks	none	Yes	SubObject	Information about each of the global locks currently held on the central repository.
InstrumentLogging	*is_active*	No	Configuration	Turns on/off automatic logging of instruments to active partition log file.
LockedWorkspaces	none	Yes	SubObject	Information about workspaces that are currently in use.
LogFile	*log_file_name*	No	Configuration	The name of the file to use when logging instruments for the active partition (compiled partitions only).
LogTimer	*is_active* *interval_in_msec*	No	Timer	Turns on/off and sets interval, in milliseconds, for instrument logging events within the active partition.
ProcessID	none	Yes	Configuration	Contains operating system ID for process running the active partition.
RepositoryName	none	Yes	Configuration	Repository name used to specify the repository when it was opened.
ServiceName	none	Yes	Configuration	Name of the repository server.

Instrument	Argument	Read Only?	Type	Description
UsersConnected	none	Yes	Counter	Number of sessions currently open on the repository.

Using the RepositoryServer Agent

The RepositoryServer agent represents a running repository server. This agent is actually a special variety of Active Partition agent, so many of the commands and instruments are the same as for the Active Partition agent. A RepositoryServer agent has a name with the format "**RpServer_***repository_server_name*."

To start a repository server, you must use the `rpstart` command, as described in *iPlanet UDS System Management Guide*.

Parent and subagents The parent agent for a RepositoryServer agent is an Ad hoc partition agent having a name with the format "**Repository_Server_***node_name*." The subagents of the RepositoryServer agent are a BtreeCache agent, a Repository agent, and the iPlanet UDS runtime agents (DistObjectMgr agent, TaskMgr agent, and so on), as well as almost all user-defined system agents. Use the `ShowAgent` command for a list of subagents to the RepositoryServer agent.

States

ONLINE

The repository server is running. If the repository server is not running, the RepositoryServer agent does not exist in the agent hierarchy.

RUNDOWN

The repository server is shutting down, probably because of a `Shutdown` command on the RepositoryServer agent or on one of the agent's parent agents. When the repository server has completed any processing it needs to shut down, the RepositoryServer agent is removed from the agent hierarchy.

Commands

DebugPartition

The `DebugPartition` command places this partition under the control of the C++ debugger for the compiler installed on that node.

DebugPartition

On UNIX platforms, you need to set the FORTE_JIT_DEBUG environment variable before you can use this command.

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

FlushLogFiles

The `FlushLogFiles` command flushes the buffers for all the log files for this partition.

FlushLogFiles

ForceShutdown

The `ForceShutdown` command stops the repository server, even if users might still be connected.

ForceShutdown

This command is equivalent to the `rpstop -k` command, described in *iPlanet UDS System Management Guide*.

ForceWorkspaceUnreserved

The `ForceWorkspaceUnreserved` command removes the reservation a detached shadow holds on the workspace.

ForceWorkspaceUnreserved *workspace* [*password*]

Argument	Description
workspace	Specifies the workspace to be unreserved.
password	If a workspace password has been set, you need to specify the password.

This command makes a reserved workspace accessible again; however, the detached shadow that holds the reservation will become unusable, so you should use this command only when the detached shadow that has the reservation on the workspace has been corrupted or to release a lock held by a detached shadow.

If you need to release a lock held by an attached shadow, use the `Fscript` command `ForceWorkspaceUnreserved`.

To recover changes made in a detached shadow whose lock is released with this command, you can export the workspace and import the changes into the central repository. If the shadow repository is attached to the central repository when you use the `ForceWorkspaceUnreserved` command, then the changes in this repository are lost.

ModLoggerRemote (ModLogger)

The `ModLoggerRemote` command sets the logger flags for the repository server being managed by the current agent.

ModLoggerRemote +(*logger_flags*) | -(*logger_flags*)

Argument	Description
+(*logger_flags*)	Turn on the logger flag settings given in the parentheses.
-(*logger_flags*)	Turn off the logger flag settings given in the parentheses.

ModLogger `flags`

To start logging, invoke the `ModLoggerRemote` command using the "+" followed by a set of logger settings in parentheses. To stop logging, use the "-" followed by a set of logger settings in parentheses.

The logger flag settings in the `ModLoggerRemote` command modify any logger flag settings that were specified for the partition, either in the `-fl` startup flag or by the FORTE_LOGGER_SETUP environment variable.

The modified logger settings are only applied to the first file specified in the original logger settings for the Node Manager (or Environment Manager).

See the `LogMgr` class in the Framework Library online Help for a detailed description of the logger flag syntax. The following examples illustrate how to use the `ModLoggerRemote` command:

```
escript> ModLoggerRemote +(trc:os:1:1 cfg:c4:2-3:1)
escript> ModLoggerRemote -(cfg:c4)
```

SetEnvRemote (SetEnvVar)

The `SetEnvRemote` command sets the environment variable for the repository server managed by the current agent.

SetEnvRemote *env_variable new_value*

Argument	Description
env_variable	The name of an environment variable to set.
new_value	The new value of the environment variable to set.

SetEnvVar *env_variable new_value*

The `SetEnvRemote` command changes the setting of the environment variable in the repository server managed by the current agent. Within the TOOL code executing in that partition, any subsequent invocation of the `GetEnv` method on the OperatingSystem object gets the new setting.

The `env_variable` argument is the name of an environment variable to set in the process running the repository server, and the `new_value` argument is the value for the environment variable.

On UNIX and VMS nodes, the new setting of the environment variable does not remain beyond the current execution of the partition. On Windows NT, the new setting is stored permanently and is picked up in any client partition or iPlanet UDS application started up at a later time, because the value is stored in the registry in Windows NT.

Shutdown

The `Shutdown` command shuts down the repository server managed by the current RepositoryServer agent.

Shutdown

This command is equivalent to the `rpstop` command.

You cannot use this command if any client repository sessions are running for this central repository. In the case of an emergency, you can use the `ForceShutdown` command, described in "ForceShutdown" on page 309, to stop the repository server even if client repository sessions might be running.

UnlockWorkspace

The `UnlockWorkspace` command frees all locks held on the given workspace.

UnlockWorkspace *workspace* [*password*]

Argument	Description
workspace	Specifies the workspace whose locks are to be released.
password	If a workspace password has been set, you need to specify the password.

The repository software normally removes locks, so you rarely need this command. However, in certain error conditions, it is necessary to explicitly unlock workspaces. You must be ***very*** careful when using the `UnlockWorkspace` command. You can lose work in your repository if you use the `UnlockWorkspace` command incorrectly.

This command is equivalent to the Fscript `UnlockWorkspace` command, described in *Fscript Reference Guide*.

Instruments

CanBeActivated

The `CanBeActivated` instrument indicates whether the partition is enabled for startup by the management system. This Configuration instrument is read only.

GlobalLockCount

The `GlobalLockCount` instrument shows the number of global locks that are currently held in the central repository. This value is typically `0`, unless users are updating or integrating their workspaces. This Counter instrument is read only.

GlobalLocks

The `GlobalLocks` instrument contains information about each of the global locks currently held on the central repository. This SubObject instrument is read only.

A workspace holds a global lock when a user updates or integrates the workspace.

The `GlobalLocks` instrument contains an array of Compound instruments. These instruments contain read-only Configuration instruments that show the workspace that was locked, the type of lock held, and the node that holds the lock.

InstrumentLogging

The `InstrumentLogging` instrument turns on or off the automatic logging of instruments to the repository server's log file. The `InstrumentLogging` instrument is a Configuration instrument.

InstrumentLogging [*is_active*]

Argument	Description
is_active	Indicates whether automatic logging of instrument logging is currently active. Set to the string `TRUE` to make the logging active or `FALSE` to make it inactive.

The `InstrumentLogging` instrument turns on the automatic logging of active instruments to the repository server's log file each time the timing interval for the `LogTimer` instrument in the active partition expires. By default, automatic logging is disabled for repository servers. The `InstrumentLogging` instrument is used in conjunction with the `LogTimer` instrument, also defined on the repository server. See "LogTimer" on page 131 for more information on how the process of automatic logging works.

The `is_active` argument is a boolean value set to `FALSE` by default. If `is_active` is `TRUE`, then the values of the current set of instruments being logged in the RepositoryServer agent, or any of its subagents, are automatically logged to the repository server's log file. If `is_active` is `FALSE`, no logging takes place. Note that even if you turn off the `InstrumentLogging` instrument, the detailed data is still collected, which could be a significant performance drain. Therefore, you need to disable the `LogTimer` instrument as well (unless you want to log to the environment log file).

LockedWorkspaces

The `LockedWorkspaces` instrument contains information about each of the workspaces that are currently locked in the central repository. This SubObject instrument is read only.

The `LockedWorkspaces` instrument is a SubObject instrument that contains an array of Compound instruments. These instruments contain read-only Configuration instruments that show the workspace that was locked, the type of lock held, and the node that holds the lock.

LogFile

The `LogFile` instrument indicates the name of the file to use when logging events for the repository server. The `LogFile` instrument is a Configuration instrument.

LogFile *log_file_name*

Argument	Description
log_file_name	Indicates the name of the file to use for logging the repository server logging events.

The `LogFile` instrument specifies the name of the log file to use for logging instrument events for the RepositoryServer agent. The information that is logged includes instrument logging events, as well as an audit trail of all important operations performed by the repository server, such as stopping. The RepositoryServer agent log file is independent of the log files specified using the `-fl` flag for logging messages generated by an application or the iPlanet UDS runtime system.

The `log_file_name` argument indicates the name of the log file to use for logging. This file name should be given in one of two ways: relative or absolute. In either case, however, it uses iPlanet UDS portable file name syntax (UNIX style). If a relative name is given for `log_file_name` (it does not start with a `/`), then the file is given relative to the `FORTE_ROOT/log` directory on the node on which the repository server is executing. If an absolute path is given in the `log_file_name`, it is an absolute path on the machine on which the repository server is executing.

If you change the logging file name after the repository server has already started logging to another file, that file is closed, and the new file is opened.

The following example shows how to set the `LogFile` Instrument in `Escript`:

```
escript> UpdateInstrument LogFile vdir:/onvms/ap.log
escript> UpdateInstrument LogFile /udir/sparc/ap.log
```

LogTimer

The `LogTimer` instrument turns on or off logging events and sets the interval for instrument logging events within the repository server. The `LogTimer` instrument is a Configuration instrument.

LogTimer [“*is_active interval_in_msec* ”]

Argument	**Description**
is_active	Indicates whether the timer is currently active. Set to the string “`TRUE`” to make the timer active or “`FALSE`” to make it inactive. The default value is `FALSE`.
interval_in_msec	The number of milliseconds between log timer events. The default value is `300000`.

The `LogTimer` instrument sets the timer interval and active status for instrument logging events in the repository server. By default, no instruments are on the instrument logging list for the repository server, so you must explicitly turn them on.

To turn on the `LogTimer` instrument, set the `is_active` argument to the string `TRUE` and the `interval_in_msec` to the interval, in milliseconds, between timer logging events. If you use the Escript `UpdateInstrument` command, you must specify quotes around the pair of arguments. For example, the following command turns the log timer on for a 10 second logging interval:

```
escript> UpdateInstrument LogTimer “TRUE 10000”
```

To turn off the `LogTimer` instrument, set the `is_active` argument to the string `FALSE`. You must also set the value for the `interval_in_msec` argument to some value, but it will not matter because the timer is not active.

Specifying instruments to log and log files After you have turned on the `LogTimer` instrument, and have invoked a `SetInstrumentLogging` command on one or more instruments in the repository server or any of its subagents, the current values of these instruments are collected whenever the `LogTimer` timing interval expires. However, by default, the values of the instruments are still not automatically logged to any files. You can request that the instruments be automatically logged, either to the environment or the repository server log (or both), by turning on the InstrumentLogging instrument on either of those agents. Once the InstrumentLogging instrument has been enabled, the logging of instruments occurs at each expiration of the timing interval.

The values of the instruments are also posted with the LogInstruments event on the repository server and Environment agents. See *Programming with System Agents* for information on how to process this event programmatically.

ProcessID

The `ProcessID` instrument contains the operating system ID for the process running the repository server. This Configuration instrument is read only.

The ProcessID instrument is system-specific, and can be used to help troubleshoot problems using other system management tools.

RepositoryName

The `RepositoryName` instrument contains the repository name used to specify the repository when it was opened. This Configuration instrument is read only.

The value of the `RepositoryName` instrument is the same as the value specified by either the `-fr` flag or the FORTE_REPOSNAME setting for the `rpstart` command. For example, the value of the `RepositoryName` instrument for a B-tree repository named “central” might be “bt:central.”

ServiceName

The `ServiceName` instrument contains the service name for the repository server. This Configuration instrument is read only.

The value of the `RepositoryName` instrument is the same as the value specified by the `-n` flag of the `rpstart` command.

UsersConnected

The `UsersConnected` instrument contains the number of sessions currently open on the repository. This Configuration instrument is read only.

The value of this instrument is generally equal to the number of items in the `LockedWorkspaces` instrument.

RepositoryServerInfo Agent

Parent Agent

Environment agent

Subagents

None

SystemMonitor Class

SystemAgent agent

States

State	Description
ONLINE	The repository server is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the repository server represented by this agent to Stdout.
Shutdown	none	Component	Stops the repository server represented by this agent.

Instrument Summary

Instrument	Argument	Read Only?	Class	Description
NodeName	none	Yes	Configuration	Node on which the repository server is running.
RepositoryName	none	Yes	Configuration	Repository name used to specify the repository when it was opened.
StartTime	none	Yes	Configuration	Date and time that the repository server was last started.

Using the RepositoryServerInfo Agent

The RepositoryServerInfo agent is a place holder for information about a repository server that is running in the current environment. There is one RepositoryServerInfo agent for each repository server.

To find the RepositoryServerInfo agent in the Environment Console Active Environment window, open the Application View. Each RepositoryServerInfo agent's name contains the name of the repository server in the format "**RpServerInfo_***repository_server_name*." For example, the RepositoryServerInfo agent for the PrinceRepository repository server is "RpServerInfo_PrinceRepository." The RepositoryServer agent is described in "RepositoryServer Agent" on page 304.

This agent has instruments that provide information about the node on which the repository server is running, the name of the repository, and the time that the repository server started.

Parent and subagents The parent agent for a RepositoryServerInfo agent is the Environment agent.

States

ONLINE

The repository server is running. When the repository server shuts down, the RepositoryServerInfo agent is removed from the agent hierarchy.

Commands

DumpStatus

The `DumpStatus` command prints the status of the repository server represented by the current RepositoryServerInfo agent to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Shutdown

The `Shutdown` command shuts down the repository server represented by the current RepositoryServerInfo agent.

Shutdown

This command is equivalent to the `rpstop` command.

You cannot use this command if any client repository sessions are running for this central repository. In the case of an emergency, you can use the `ForceShutdown` command on the RepositoryServer agent, as described in "ForceShutdown" on page 309, to stop the repository server even if client repository sessions might be running.

Instruments

NodeName

The `NodeName` instrument indicates the node on which the repository server is running. This Configuration instrument is read only.

iPlanet UDS System Management Guide describes how you can use the node name to locate the RepositoryServer agent for this repository server.

RepositoryName

The `RepositoryName` instrument indicates the repository name used to specify the repository when it was opened. This Configuration instrument is read only.

The value of this instrument is the same as the *repository_name* value specified for the `-fr` flag or the FORTE_REPOSNAME environment variable, for example, "bt:banking."

StartTime

The `StartTime` instrument indicates the date and time that the repository server was last started. This Configuration instrument is read only.

RepositorySession Agent

Parent Agent

Active Partition agent

Subagents

ObjectCache agent

SystemMonitor Class

SystemAgent agent

States

State	Description
ONLINE	The repository session is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Class	Description
RepositoryName	none	Yes	Configuration	Repository name used to specify the repository when it was opened.
Type	none	Yes	Configuration	Type of repository session.
Workspace	none	Yes	Configuration	Workspace currently used by the session.

Using the RepositorySession Agent

The RepositorySession agent represents a client repository session running on a node with a running Node Manager. Client nodes that do not have a running node manager do not have corresponding RepositorySession agents that are accessible using the Environment Console or `Escript`.

Client applications that access the development repository typically have two RepositorySession agents: one that represents a client session on the development repository, and one that represents a client session on the environment repository.

For interpreted partitions, the RepositorySession agent for the image repository appears as a subagent of the iPlanet UDS executor Active Partition agent instead of as a subagent of the Installed Partition agent.

Parent and subagents The parent agent for a RepositorySession agent is an Active Partition agent. RepositorySession agents have ObjectCache subagents.

States

ONLINE

The repository session is running and in use. When the repository session is shut down, the RepositorySession agent is removed from the agent hierarchy.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

RepositoryName

The `RepositoryName` instrument indicates the repository name used to specify the repository when it was opened. This Configuration instrument is read only.

The value of this instrument is the same as the `repository_name` value specified for the `-fr` flag or the FORTE_REPOSNAME environment variable, for example, “bt:banking” or “PrinceRepository.”

Type

The `Type` instrument indicates the type of repository session that this agent represents. This Configuration instrument is read only.

Possible types of repository sessions are:

- Private

  Private is the type for private repositories, image repositories, environment repositories, and node repositories.

- Attached shadow
- Detached shadow
- Direct connection to the central repository

For more information about these types of repository sessions, see *A Guide to the iPlanet UDS Workshops* and *iPlanet UDS System Management Guide*.

Workspace

The `Workspace` instrument indicates the name of the workspace currently opened by this repository session. This Configuration instrument is read only.

TaskMgr Agent

Parent Agent

Active Partition or RepositoryServer agent

Subagents

None

SystemMonitor Class

TaskMgrAgent

States

State	Description
ONLINE	The task manager is running.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
CommunicationResumes	none	Yes	Counter	Number of times that tasks have resumed after waiting for the arrival of a message or packet.

Instrument	Argument	Read Only?	Type	Description
CommunicationSuspends	none	Yes	Counter	Number of times that tasks have been suspended to wait for a message or packet to be sent or to arrive.
ConditionVariableCreates	none	Yes	Counter	Number of condition variables that have been created by internal iPlanet UDS processing.
ConditionVariableDestroys	none	Yes	Counter	Number of discarded condition variables and their associated mutex locks.
CurrentTaskCount	none	Yes	Counter	Number of tasks that currently exist in the iPlanet UDS system.
DebugResumes	none	Yes	Counter	Number of times tasks have resumed after waiting for a user to continue during a debugging session.
DebugSuspends	none	Yes	Counter	Number of times tasks have been suspended to wait for a user to continue during a debugging session.
DurableLockBlocks	none	Yes	Counter	Total number of times that tasks have been suspended to wait for available durable locks during internal iPlanet UDS processing.

Instrument	Argument	Read Only?	Type	Description
DurableLockCancels	none	Yes	Counter	Number of times that iPlanet UDS has canceled waits for available durable locks during internal iPlanet UDS processing.
DurableLockCreates	none	Yes	Counter	Total number of durable locks that have been created by internal iPlanet UDS processing.
DurableLockFrees	none	Yes	Counter	Number of times that one or more levels of durable locks have been freed during internal iPlanet UDS processing.
DurableLockLocks	none	Yes	Counter	Total number of durable locks that have been locked by internal iPlanet UDS processing.
DurableLockUnlocks	none	Yes	Counter	Total number of durable locks that have been unlocked by internal iPlanet UDS processing.
EventResumes	none	Yes	Counter	Number of times tasks have resumed after waiting for an event.
EventSuspends	none	Yes	Counter	Number of times tasks have been suspended to wait for an event.

Instrument	Argument	Read Only?	Type	Description
LockResumes	none	Yes	Counter	Number of times tasks have resumed after waiting for an available durable lock.
LockSuspends	none	Yes	Counter	Number of times tasks have been suspended to wait for an available durable lock.
MissedResumes	none	Yes	Counter	Number of times tasks did not resume processing after being passed information about a condition.
MutexCreates	none	Yes	Counter	Number of mutex locks that have been created by the task manager.
MutexDestroys	none	Yes	Counter	Number of mutex locks that have been discarded.
MutexLocks	none	Yes	Counter	Number of Task Manager-level mutex locks that have been locked during this process.
MutexUnlocks	none	Yes	Counter	Total number of Task Manager-level mutex locks that have been unlocked during this process.
PlayBackResumes	none	Yes	Counter	Number of times tasks have resumed after waiting for a user to start the playback operation in Autotester.

Instrument	Argument	Read Only?	Type	Description
PlayBackSuspends	none	Yes	Counter	Number of times tasks have been suspended to wait for a user to start the playback operation in Autotester.
PopVisitors	none	Yes	Counter	Number of tasks that started in one partition, started processing remotely in the current partition (visiting), then completed processing in the current partition and returned to the original partition.
PushVisitors	none	Yes	Counter	Number of tasks that started in the current partition, started processing remotely in another partition (visiting), then started processing remotely back to the current partition (where it originally started).
RemoteResponseResumes	none	Yes	Counter	Number of times tasks have resumed after waiting for a response from a TOOL remote method call.
RemoteResponseSuspends	none	Yes	Counter	Number of times tasks have been suspended to wait for a response from a TOOL remote method call.

Instrument	Argument	Read Only?	Type	Description
TaskCancels	none	Yes	Counter	Number of times the `SetCancel` method of the `TaskHandle` class has been called.
TaskCreates	none	Yes	Counter	Number of tasks that have been created.
TaskKills	none	Yes	Counter	Number of tasks that have been killed by the iPlanet UDS system.
TasksTerminated	none	Yes	Counter	Number of tasks that have terminated. These tasks have completed processing, been killed, or been cancelled.
TimeOutResumes	none	Yes	Counter	Number of times tasks have resumed after waiting for an installed timer to time out.
TimeOutSuspends	none	Yes	Counter	Number of times tasks have been suspended to wait for an installed timer to time out.
TransactionCompletionResumes	none	Yes	Counter	Number of times tasks have resumed after waiting for a transaction to complete by being either committed or aborted.

Instrument	Argument	Read Only?	Type	Description
TransactionCompletionSuspends	none	Yes	Counter	Number of times tasks have been suspended to wait for a transaction to complete by being either committed or aborted.
TransactionLockResumes	none	Yes	Counter	Number of times tasks have resumed after waiting for an available transaction lock.
TransactionLockSuspends	none	Yes	Counter	Number of times tasks have been suspended to wait for an available transaction lock.

Using the TaskMgr Agent

The TaskMgr agent is associated with the task management services of an active partition. This monitors creation and destruction of tasks for the iPlanet UDS executor. It also manages the shared locks and mutexes for an active partition.

Parent and subagents The parent agent for the TaskMgr agent is an Active Partition agent. There are no subagents of the TaskMgr agent.

States

ONLINE

The task management services of the active partition are running. The task management services are part of the runtime system, and cannot run unless the active partition is running.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

CommunicationResumes

The `CommunicationResumes` instrument represents the number of times that tasks have resumed after waiting for the arrival of a message or packet. This Counter instrument is read only.

CommunicationSuspends

The `CommunicationSuspends` instrument represents the number of times that tasks have been suspended to wait for a message or packet to be sent or to arrive. This Counter instrument is read only.

ConditionVariableCreates

The `ConditionVariableCreates` instrument represents the number of condition variables that have been created by internal iPlanet UDS processing. Each condition variable has an associated mutex lock. This Counter instrument is read only.

ConditionVariableDestroys

The `ConditionVariableDestroys` instrument represents the number of discarded condition variables and their associated mutex locks. This Counter instrument is read only.

CurrentTaskCount

The `CurrentTaskCount` instrument represents the number of tasks that currently exist in the iPlanet UDS system. This Counter instrument is read only.

DebugResumes

The `DebugResumes` instrument represents the number of times tasks have resumed after waiting for a user to continue during a debugging session. This Counter instrument is read only.

DebugSuspends

The `DebugSuspends` instrument represents the number of times tasks have been suspended to wait for a user to continue during a debugging session. This Counter instrument is read only.

DurableLockBlocks

The `DurableLockBlocks` instrument represents the total number of times that tasks have been suspended to wait for available durable locks during internal iPlanet UDS processing. This Counter instrument is read only.

DurableLockCancels

The `DurableLockCancels` instrument represents the number of times that iPlanet UDS has canceled waits for available durable locks during internal iPlanet UDS processing. This Counter instrument is read only.

DurableLockCreates

The `DurableLockCreates` instrument represents the total number of durable locks that have been created by internal iPlanet UDS processing. This Counter instrument is read only.

DurableLockFrees

The `DurableLockFrees` instrument represents the number of times that one or more levels of durable locks have been freed during internal iPlanet UDS processing. This Counter instrument is read only.

DurableLockLocks

The `DurableLockLocks` instrument represents the total number of durable locks that have been locked by internal iPlanet UDS processing. This Counter instrument is read only.

DurableLockUnlocks

The `DurableLockUnlocks` instrument represents the total number of durable locks that have been unlocked by internal iPlanet UDS processing. This Counter instrument is read only.

EventResumes

The `EventResumes` instrument represents the number of times tasks have resumed after waiting for an event. This Counter instrument is read only.

EventSuspends

The `EventSuspends` instrument represents the number of times tasks have been suspended to wait for an event. This Counter instrument is read only.

LockResumes

The `LockResumes` instrument represents the number of times tasks have resumed after waiting for an available durable lock. This Counter instrument is read only.

LockSuspends

The `LockSuspends` instrument represents the number of times tasks have been suspended to wait for an available durable lock. This Counter instrument is read only.

MissedResumes

The `MissedResumes` instrument represents the number of times tasks did not resume processing after being passed information about a condition. This Counter instrument is read only.

A task might not resume processing when it receives the information because the task might already be running (not suspended), or the task might be suspended and waiting for a different condition.

MutexCreates

The `MutexCreates` instrument represents the number of mutex locks that have been created by the Task Manager. This Counter instrument is read only.

The MutexCreates instrument does not count some mutexes created during internal iPlanet UDS processing.

MutexDestroys

The `MutexDestroys` instrument represents the number of mutex locks that have been discarded. This Counter instrument is read only.

MutexLocks

The `MutexLocks` instrument represents the total number of Task Manager-level mutex locks that have been locked during this process. This Counter instrument is read only.

MutexUnlocks

The `MutexUnlocks` instrument represents the total number of Task Manager-level mutex locks that have been unlocked during this process. This Counter instrument is read only.

PlayBackResumes

The `PlayBackResumes` instrument represents the number of times tasks have resumed after waiting for a user to start the playback operation in Autotester. This Counter instrument is read only.

PlayBackSuspends

The `PlayBackSuspends` instrument represents the number of times tasks have been suspended to wait for a user to start the playback operation in Autotester. This Counter instrument is read only.

PopVisitors

The `PopVisitors` instrument represents the number of tasks that started in one partition, started processing remotely in the current partition (visiting), then completed processing in the current partition and returned to the originating partition. This Counter instrument is read only.

PushVisitors

The `PushVisitors` instrument represents the number of tasks that started in the current partition, started processing remotely in another partition (visiting), then started processing remotely back to the current partition (the originating partition). This Counter instrument is read only.

RemoteResponseResumes

The `RemoteResponseResumes` instrument represents the number of times tasks have resumed after waiting for a response from a TOOL remote method call. This Counter instrument is read only.

RemoteResponseSuspends

The `RemoteResponseSuspends` instrument represents the number of times tasks have been suspended to wait for a response from a TOOL remote method call. This Counter instrument is read only.

TaskCancels

The `TaskCancels` instrument represents the number of times the `SetCancel` method of the `TaskHandle` class has been called. This Counter instrument is read only.

TaskCreates

The `TaskCreates` instrument represents the number of tasks that have been created. This Counter instrument is read only.

The `TaskCreates` instrument counts both the tasks started by TOOL `start task` statements and those started during internal iPlanet UDS processing.

TaskKills

The `TaskKills` instrument represents the number of tasks that have been killed by the iPlanet UDS system. This Counter instrument is read only.

TasksTerminated

The `TasksTerminated` instrument represents the number of tasks that have terminated. These tasks have completed processing, been killed, or been cancelled. This Counter instrument is read only.

TimeOutResumes

The `TimeOutResumes` instrument represents the number of times tasks have resumed after waiting for an installed timer to time out. This Counter instrument is read only.

TimeOutSuspends

The `TimeOutSuspends` instrument represents the number of times tasks have been suspended to wait for an installed timer to time out. This Counter instrument is read only.

TransactionCompletionResumes

The `TransactionCompletionResumes` instrument represents the number of times tasks have resumed after waiting for a transaction to complete by being either committed or aborted. This Counter instrument is read only.

TransactionCompletionSuspends

The `TransactionCompletionSuspends` instrument represents the number of times tasks have been suspended to wait for a transaction to complete by being either committed or aborted. This Counter instrument is read only.

TransactionLockResumes

The `TransactionLockResumes` instrument represents the number of times tasks have resumed after waiting for an available transaction lock. This Counter instrument is read only.

TransactionLockSuspends

The `TransactionLockSuspends` instrument represents the number of times tasks have been suspended to wait for an available transaction lock. This Counter instrument is read only.

TransactionMgr Agent

Parent Agent

Active Partition or RepositoryServer agent

Subagents

None

SystemMonitor Class

TransactionMgrAgent

States

State	**Description**
ONLINE	The transaction manager is running.

Command Summary

Command	**Arguments**	**Environment Console menu**	**Description**
DumpAll	none	Utility	Prints out all information related to transactions.

Command	Arguments	Environment Console menu	Description
DumpLock	none	Utility	Prints out information related to transaction locks.
DumpLog	none	Utility	Prints out information related to the transaction log.
DumpStatus	*no_propagate*	Component	Prints the status of the managed object to Stdout.
DumpTM	none	Utility	Prints information about the status of running transactions.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Class	Description
Abort_ByPartitionLost	none	Yes	Counter	Number of transactions aborted by the system due to a communications failure.
Abort_BySystem	none	Yes	Counter	Number of transactions aborted by the system due to an error other than a communications failure.
Abort_Independent	none	Yes	Counter	Number of independent transactions that have been aborted.
Abort_Nested	none	Yes	Counter	Number of nested transactions that have been aborted.
Active_Locks	none	Yes	Counter	Number of currently active locks.
Active_ReadOnlyPartitions	none	Yes	Counter	Number of partitions that are marked “read-only.”
Active_ResourceManagers	none	Yes	Counter	Number of currently active registered resource managers.
Active_Transactions	none	Yes	Counter	Number of currently active transactions.

Instrument	Argument	Read Only?	Class	Description
Begin_Auto	none	Yes	Counter	Number of transactions that iPlanet UDS has started automatically so that it could interact with a database.
Begin_Dependent	none	Yes	Counter	Number of dependent transactions that have been started.
Begin_Independent	none	Yes	Counter	Number of independent transactions that have been started.
Begin_Nested	none	Yes	Counter	Number of nested transactions that have been started.
Begin_RemoteIndependent	none	Yes	Counter	Number of independent transactions that have migrated to this partition.
Begin_RemoteNested	none	Yes	Counter	Number of nested transactions that have migrated to this partition.
Commit_Asynchronous	none	Yes	Counter	Number of asynchronous tasks that participated in transactions and ended and committed their parts of the transactions.
Commit_Dependent	none	Yes	Counter	Number of dependent transactions that have committed.
Commit_Independent	none	Yes	Counter	Number of independent transactions that have committed.
Commit_Nested	none	Yes	Counter	Number of nested transactions that have committed.
Commit_Wait	none	Yes	Counter	Number of times transactions had to wait to commit until all asynchronous participants had commited.

Instrument	Argument	Read Only?	Class	Description
Foreign_Associate	none	Yes	Counter	Number of associations between iPlanet UDS transactions and external transactions, such as those managed by Encina or OpenTP1.
Foreign_Disassociate	none	Yes	Counter	Number of dissociations between iPlanet UDS transactions and external transactions, such as those managed by Encina or OpenTP1.
Lock_Convert	none	Yes	Counter	Number of transactional lock promotions from read to write.
Lock_DeadLock	none	Yes	Counter	Number of times requests for transactional locks encountered deadlock.
Lock_Exclusive	none	Yes	Counter	Number of exclusive (write) locks that have been acquired.
Lock_ExplicitUnlock	none	Yes	Counter	Number of times the iPlanet UDS runtime system explicitly requested that a transactional lock be released.
Lock_Shared	none	Yes	Counter	Number of shared (read) locks that have been acquired.
Lock_Wait	none	Yes	Counter	Number of times a request for a lock had to wait for the transactional object to become available.
Log_ByTask	none	Yes	Counter	Number of times transactional objects have been logged before being modified in TOOL.
Log_ForRemote	none	Yes	Counter	Number of times transactional objects were logged before they were passed as parameters to remote partitions.
Remote_AbortIndependent	none	Yes	Counter	Number of messages received from other partitions that request that independent transactions be aborted.

Instrument	Argument	Read Only?	Class	Description
Remote_AbortNested	none	Yes	Counter	Number of messages received from other partitions that request that nested transactions be aborted.
Remote_CommitIndependent	none	Yes	Counter	Number of messages received from other partitions that request that an independent transaction be committed.
Remote_CommitNested	none	Yes	Counter	Number of messages received from other partitions that request that a nested transaction be committed.
Remote_PrepareToCommit	none	Yes	Counter	Number of messages received from other partitions that request that a transaction be prepared to commit.
Remote_ReceiveAsynchronousMethod	none	Yes	Counter	Number of asynchronous transactional messages received from other partitions.
Remote_ReceiveReply	none	Yes	Counter	Number of transactional synchronous or asynchronous replies received from other partitions.
Remote_ReceiveSynchronousMethod	none	Yes	Counter	Number of remote synchronous transactional messages received.
Remote_SendAsynchronousMethod	none	Yes	Counter	Number of asynchronous transaction messages that this partition has sent to other partitions.
Remote_SendReply	none	Yes	Counter	Number of transactional synchronous or asynchronous replies sent by this partition to other partitions.
Remote_SendSynchronousMethod	none	Yes	Counter	Number of synchronous transactional messages sent by this partition to other partitions.
Task_Detach	none	Yes	Counter	Number of times the iPlanet UDS runtime system detached a task from a transaction.

Instrument	Argument	Read Only?	Class	Description
Task_DetachPermanent	none	Yes	Counter	Number of times the iPlanet UDS runtime system permanently detached a task from a transaction.
Task_Join	none	Yes	Counter	Number of times a task joined as a participant in an existing transaction.
Task_StartTransactional	none	Yes	Counter	Number of times an asynchronous task was started as part of a transaction.

Using the TransactionMgr Agent

The TransactionMgr agent represents the Transaction Manager for an active partition. The Transaction Manager monitors the state of transactions across partitions.

Parent and subagents The parent agent for the TransactionMgr agent is an Active Partition agent. The TransactionMgr agent has no subagents.

States

ONLINE

The Transaction Manager for the active partition is running. The Transaction Manager is part of the runtime system, and cannot run unless the active partition is running.

Commands

DumpAll

The `DumpAll` command prints out all information related to transactions for this partition. `DumpAll` and `DumpStatus` print out the same information.

DumpAll

DumpLock

The `DumpLock` command prints out information related to transaction locks for this partition.

DumpLock

DumpLog

The `DumpLog` command prints out information related to the transaction log for this partition.

DumpLog

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout. `DumpAll` and `DumpStatus` print out the same information.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

DumpTM

The `DumpTM` command prints information about the status of running transactions for this partition.

DumpTM

Instruments

Abort_ByPartitionLost

The `Abort_ByPartitionLost` instrument represents the number of transactions aborted by the system due to a communications failure. This Counter instrument is read only.

Abort_BySystem

The `Abort_BySystem` instrument represents the number of transactions aborted by the system due to an error other than a communications failure. This Counter instrument is read only.

Abort_Independent

The `Abort_Independent` instrument represents the number of independent transactions that have been aborted. This Counter instrument is read only.

Abort_Nested

The `Abort_Nested` instrument represents the number of nested transactions that have been aborted. This Counter instrument is read only.

Active_Locks

The `Active_Locks` instrument represents the number of currently active locks. This Counter instrument is read only.

Active_ReadOnlyPartitions

The `Active_ReadOnlyPartitions` instrument represents the number of partitions that are marked “read-only.” When a partition is marked “read-only,” transactions that are started in other partitions do not abort if communication failures occur between the other partitions and the “read-only” partition. This Counter instrument is read only.

Active_ResourceManagers

The `Active_ResourceManagers` instrument represents the number of currently active registered resource managers. This Counter instrument is read only.

Active_Transactions

The `Active_Transactions` instrument represents the number of currently active transactions. This Counter instrument is read only.

Begin_Auto

The `Begin_Auto` instrument represents the number of transactions that iPlanet UDS has started automatically so that it could interact with a database. This Counter instrument is read only.

Begin_Dependent

The `Begin_Dependent` instrument represents the number of dependent transactions that have been started. This Counter instrument is read only.

Begin_Independent

The `Begin_Independent` instrument represents the number of independent transactions that have been started. This Counter instrument is read only.

Begin_Nested

The `Begin_Nested` instrument represents the number of nested transactions that have been started. This Counter instrument is read only.

Begin_RemoteIndependent

The `Begin_RemoteIndependent` instrument represents the number of independent transactions that have migrated to this partition. In other words, this instrument counts the independent transactions that have started on this partition as parts of transactions started on other partitions. This Counter instrument is read only.

Begin_RemoteNested

The `Begin_RemoteNested` instrument represents the number of nested transactions that have migrated to this partition. In other words, this instrument counts the nested transactions that have started on this partition as parts of transactions started on other partitions. This Counter instrument is read only.

Commit_Asynchronous

The `Commit_Asynchronous` instrument represents the number of asynchronous tasks that participated in transactions and ended and committed their parts of the transactions. These tasks were started using the `start task` statement with the `(transaction=dependent)` clause. This Counter instrument is read only.

Commit_Dependent

The `Commit_Dependent` instrument represents the number of dependent transactions that have committed. This Counter instrument is read only.

Commit_Independent

The `Commit_Independent` instrument represents the number of independent transactions that have committed. This Counter instrument is read only.

Commit_Nested

The `Commit_Nested` instrument represents the number of nested transactions that have committed. This Counter instrument is read only.

Commit_Wait

The `Commit_Wait` instrument represents the number of times transactions had to wait to commit until all of their asynchronous participants had committed. Asynchronous participants can include asynchronous tasks that are nested or dependent parts of a particular transaction. This Counter instrument is read only.

Foreign_Associate

The `Foreign_Associate` instrument represents the number of times external transactions, such as those managed by Encina or OpenTP1, are associated with iPlanet UDS transactions when the external client uses transactional remote procedure calls (RPCs) to call iPlanet UDS services. This Counter instrument is read only.

Foreign_Disassociate

The `Foreign_Disassociate` instrument represents the number of times iPlanet UDS transactions have disassociated from external transactions, such as those managed by Encina or OpenTP1, as transactional remote procedure calls (RPCs) return from iPlanet UDS services. This Counter instrument is read only.

Lock_Convert

The `Lock_Convert` instrument represents the number of times transactional locks were promoted from read to write. This Counter instrument is read only.

Lock_DeadLock

The `Lock_DeadLock` instrument represents the number of times requests for a transactional lock or lock promotion encountered deadlock. This Counter instrument is read only.

The value of this instrument should be zero. Any other value indicates a design error in a running application.

Lock_Exclusive

The `Lock_Exclusive` instrument represents the number of exclusive (write) transactional locks that have been acquired. This Counter instrument is read only.

Lock_ExplicitUnlock

The `Lock_ExplicitUnlock` instrument represents the number of times the iPlanet UDS runtime system explicitly requested that a transactional lock be released. This Counter instrument is read only.

Lock_Shared

The `Lock_Shared` instrument represents the number of shared (read) transactional locks that have been acquired. This Counter instrument is read only.

Lock_Wait

The `Lock_Wait` instrument represents the number of times a request for a transactional lock had to wait for the transactional object to become available. This Counter instrument is read only.

Log_ByTask

The `Log_ByTask` instrument represents the number of times transactional objects have been logged before being modified in TOOL. This Counter instrument is read only.

Log_ForRemote

The `Log_ForRemote` instrument represents the number of times transactional objects were logged before they were passed as parameters to remote partitions. This Counter instrument is read only.

Remote_AbortIndependent

The `Remote_AbortIndependent` instrument represents the number of messages received from other partitions that request that independent transactions be aborted. This Counter instrument is read only.

Remote_AbortNested

The `Remote_AbortNested` instrument represents the number of messages received from other partitions that request that nested transactions be aborted. This Counter instrument is read only.

Remote_CommitIndependent

The `Remote_CommitIndependent` instrument represents the number of messages received from other partitions that request that an independent transaction be committed. This Counter instrument is read only.

Remote_CommitNested

The `Remote_CommitNested` instrument represents the number of messages received from other partitions that request that a nested transaction be committed. This Counter instrument is read only.

Remote_PrepareToCommit

The `Remote_PrepareToCommit` instrument represents the number of messages received from other partitions that request that a transaction be prepared to commit. This Counter instrument is read only.

Remote_ReceiveAsynchronousMethod

The `Remote_ReceiveAsynchronousMethod` instrument represents the number of asynchronous transactional messages received from other partitions. This Counter instrument is read only.

Remote_ReceiveReply

The `Remote_ReceiveReply` instrument represents the number of transactional synchronous or asynchronous replies received from other partitions. These replies are sent in response to a transactional message that has been received. This Counter instrument is read only.

Remote_ReceiveSynchronousMethod

The `Remote_ReceiveSynchronousMethod` instrument represents the number of remote synchronous transactional messages received. This Counter instrument is read only.

Remote_SendAsynchronousMethod

The `Remote_SendAsynchronousMethod` instrument represents the number of asynchronous transaction messages that this partition has sent to other partitions. This Counter instrument is read only.

Remote_SendReply

The `Remote_SendReply` instrument represents the number of transactional synchronous or asynchronous replies sent by this partition to other partitions. These replies are sent in response to a transactional message that has been received. This Counter instrument is read only.

Remote_SendSynchronousMethod

The `Remote_SendSynchronousMethod` instrument represents the number of synchronous transactional messages sent by this partition to other partitions. This Counter instrument is read only.

Task_Detach

The `Task_Detach` instrument represents the number of times the iPlanet UDS runtime system has detached a task from a transaction. This Counter instrument is read only.

Task_DetachPermanent

The `Task_DetachPermanent` instrument represents the number of times the iPlanet UDS runtime system has permanently detached a task from a transaction. This Counter instrument is read only.

Task_Join

The `Task_Join` instrument represents the number of times a task joined as a participant in an existing transaction using the `TransactionHandle.Join` method. This Counter instrument is read only.

Task_StartTransactional

The `Task_StartTransactional` instrument represents the number of times an asynchronous task was started using the `start task` statement and the `transaction` clause to indicate that the task is part of a transaction. This Counter instrument is read only.

Volume Agent

Parent Agent

Machine agent

Subagents

None

SystemMonitor Class

VolumeAgent

States

State	Description
ONLINE	The volume is available.

Command Summary

Command	Arguments	Environment Console menu	Description
DumpStatus	*no_propagate*	Component	Prints the status of the repository server represented by this agent to Stdout.
Shutdown	none	Component	Not available.

Instrument Summary

Instrument	Argument	Read Only?	Type	Description
FileCount	none	Yes	Counter	Number of files on the volume.
FileSystemType	none	Yes	Configuration	Describes the type of file system.
FreeFileCount	none	Yes	Counter	Number of unused files on the volume.
FreeFilesPercent	none	Yes	Counter	Percentage of predefined files that are unused.
FreeSizeKB	none	Yes	Counter	Amount of storage, in kilobytes, currently available on volume.
FreeSpacePercent	none	Yes	Counter	Percentage of storage that is currently available.
IsCDRomFileSystem	none	Yes	Counter	Indicates whether the volume is a CD-ROM drive.
IsReadOnlyFileSystem	none	Yes	Counter	Indicates whether the volume is read only.
IsRemoteFileSystem	none	Yes	Counter	Indicates whether the volume is mounted remotely.

Instrument	Argument	Read Only?	Type	Description
MaximumNameLength	none	Yes	Counter	Maximum file name length permitted on this volume.
TotalSizeKB	none	Yes	Counter	Total storage, in kilobytes that this volume holds.

Using the Volume Agent

The Volume agent represents a storage device that is available to the machine.

Certain instruments are not available on some platforms. If an instrument value is not available for a Volume agent, the value is set to `-1`.

Parent agents and subagents The parent agent for a Volume agent is a Machine agent. Volume agents have no subagents.

States

ONLINE

The volume on the machine is available. Volumes that are not available do not have associated agents, and therefore do not appear in the agent hierarchy at all.

Commands

DumpStatus

The `DumpStatus` command prints the status of the managed object to Stdout.

DumpStatus [*no_propagate*]

Argument	Description
no_propagate	This flag indicates whether to propagate the `DumpStatus` request to the subagents. The default is `0`, which dumps information about subagents as well. If this argument is set to `1`, only information for this agent is dumped.

Instruments

FileCount

The `FileCount` instrument indicates the number of files on the volume. This Counter instrument is read only.

FileSystemType

The `FileSystemType` instrument indicates the type of file system. This Configuration instrument is read only.

FreeFileCount

The `FreeFileCount` instrument indicates how many files defined for the volume are unused. This instrument typically indicates how many more files you can create on this volume. This Counter instrument is read only.

This instrument is useful only for file structures that preset the number of files for the volume, such as UNIX and Windows NT.

If this instrument is not applicable for a volume, the value is set to `-1`.

FreeFilesPercent

The `FreeFilesPercent` instrument indicates the percentage of files defined for the volume that are unused. This instrument is useful only for file structures that preset the number of files for the volume, such as UNIX. This Counter instrument is read only.

If this instrument is not applicable for a volume, the value is set to `-1`.

FreeSizeKB

The `FreeSizeKB` instrument indicates the amount of storage, in kilobytes, that is available on the volume. This Counter instrument is read only.

FreeSpacePercent

The `FreeSpacePercent` instrument indicates the percentage of the total space on the volume that is available. This Counter instrument is read only.

IsCDRomFileSystem

The `IsCDRomFileSystem` instrument indicates whether the volume is a CD-ROM device (`1`) or not (`0`). This Counter instrument is read only.

IsReadOnlyFileSystem

The `IsReadOnlyFileSystem` instrument indicates whether the volume is read only (`1`) or not (`0`). This Counter instrument is read only.

IsRemoteFileSystem

The `IsRemoteFileSystem` instrument indicates whether the volume is remotely-mounted file system (`1`) or not (`0`). This Counter instrument is read only.

MaximumNameLength

The `MaximumNameLength` instrument indicates the maximum name length supported on this volume. This Counter instrument is read only.

TotalSizeKB

The `TotalSizeKB` instrument indicates the total storage, in kilobytes, that this volume can hold. This Counter instrument is read only.

Appendix A

All Escript and System Agent Commands

This appendix contains a list of all general Escript commands and iPlanet UDS system agent commands that can be accessed using the Escript utility.

All General Escript and System Agent Commands

This section contains a list of all commands that can be accessed by the Escript utility and used to manage and monitor the iPlanet UDS runtime system and deployed iPlanet UDS applications.

A dot (•) preceding the command name means that Escript must be in environment edit mode before you can invoke the command. For information about the environment edit mode, see "Configuring Environment Definitions" on page 37.

Command	**Agent**	**Description**	**See:**
• Add3GLProj *project_name*	any	Adds the specified restricted 3GL project to the list of those supported by the current node.	page 59
• AddCommProtocol *protocol_name*	any	Adds the specified communications protocol to the current node.	page 60

Command	Agent	Description	See:
• AddExternalRM *resource_manager_name resource_manager_type*	any	Adds the specified external resource manager name to the current node, and sets the resource manager type to the value specified.	page 62
• AddNode *node_name* [*existing_node_name* | *template_node_name*]	any	Adds a node with the specified name to the current environment definition.	page 63
AddPath *directory_name* [;*directory_name*...]	any	Adds the specified directories to the current search path.	page 66
Assign *node_name*	Partition	Assigns the partition represented by the current Partition agent for installation on the specified node.	page 286
AssignAppComp *node_name component_name*	Application	Assigns the specified application component in the current application for installation on the specified node.	page 142
Cd *directory_name*	any	Changes the current working directory.	page 67
CollectMem	any	Runs memory management on Escript.	page 68
CommentOff	any	Turns off writing of script file commands and output to standard output.	page 68
CommentOn	any	Writes script file commands and output to standard output.	page 68
Commit	any	Saves all changes to the environment repository.	page 69
ConnectEnv *env_name env_location* [*user_directory*]	NameService	Connects a target environment to the environment from which the command is issued.	page 234

Command	Agent	Description	See:
DebugPartition	Active Partition	Places this partition under the control of a C++ debugger.	page 126
	RepositoryServer	Places this partition under the control of a C++ debugger.	page 309
Disable *node_name*	Partition	Disables autostart on a node for the partition managed by the current Partition agent.	page 287
DisableAppComp *node_name partition_name*	Application	Disables startup of the specified partition on the specified node in the current application.	page 144
DisconnectEnv	NameService	Separates the current environment from any other environments.	page 237
DumpAll	NativeLangMgr	Dumps all status information.	page 247
	TransactionMgr	Prints out all information related to transactions.	page 341
DumpCSConv	NativeLangMgr	Dumps the code set conversion information.	page 247
DumpLocale	NativeLangMgr	Dumps the currently loaded locale information.	page 247
DumpLock	TransactionMgr	Prints out information related to transaction locks.	page 342
DumpLog	TransactionMgr	Prints out information related to the transaction log.	page 342
DumpMemory	OperatingSystem	Prints the state of garbage-collected memory to Stdout.	page 271
DumpMsgCat	NativeLangMgr	Dumps the currently loaded message catalogs.	page 247

Command	**Agent**	**Description**	**See:**
DumpMutexes	OperatingSystem	Writes information about all current mutex locks to the log file.	page 271
DumpStatus [*no_propagate*]	any	Writes detailed troubleshooting information to log file for current agent.	page 69
DumpTM	TransactionMgr	Prints information about the status of running transactions.	page 342
EditEnv	any	Start environment editing mode with the active environment definition the current environment definition.	page 70
Enable *node_name*	Partition	Enables autostart for the partition managed by the current Partition agent.	page 288
EnableAppComp *node_name partition_name*	Application	Enables startup for the specified partition on the specified node, in the current application.	page 145
ExecCmd *opsys_command* [*bg_flag*] [*in_file*] [*out_file*] [*err_file*]	any	Executes the specified operating system command.	page 71
ExecCmdRemote *opsys_command* [*bg_flag*] [*in_file*] [*out_file*] [*err_file*]	Node	Executes the specified operating system command from the Node Manager service that is managed by the current agent.	page 252
Exit	any	Exits Escript, prompting you to save if there are outstanding changes to the environment.	page 72
ExitIfNoEnv	any	Sets Escript to exit when it loses contact with an active Environment Manager.	page 72

Command	Agent	Description	See:
ExportEnv [*file_name*] [*environment_name*]	Environment	Exports an environment definition into a specified file.	page 198
FindActEnv	any	Makes active Environment agent the current agent.	page 73
FindEnv *env_name* [*is_updateable*]	Environment	Designates the specified environment definition as the current environment and puts Escript in environment editing mode.	page 199
• FindNode *node_name*	any	Designates the specified node within the current environment definition as the current node.	page 74
FindParentAgent	any	Makes the parent agent the current agent.	page 74
FindSavedAgent [*agent_tag*]	any	Makes the referenced saved agent the current agent.	page 75
FindSubAgent *agent_name*	any	Makes a subagent the current agent.	page 76
FlushLogFiles	Active Partition	Flushes all of this partition's log files.	page 127
	RepositoryServer	Flushes all of the repository server's log files.	page 309
ForceShutdown	RepositoryServer	Stops the repository server, even if users might still be connected.	page 309
ForceWorkspaceUnreserved *workspace* [*password*]	RepositoryServer	Removes the reservation a detached shadow holds on the workspace.	page 309
GenerateAlert *subject_text* *message_text*	Environment	Sends an alert message to the Environment agent.	page 200
Help [*command_name* | *match_string*]	any	Lists help for commands.	page 77

Command	**Agent**	**Description**	**See:**
ImportEnv *file_name*	Environment	Imports the definition of an environment from the specified file.	page 201
Include *file_name*	any	Executes the commands in a specified script file.	page 77
Install	Application	Installs the current application into the environment.	page 146
InstallApp *application_name* [*reinstall*]	Node	Installs partitions for an application on a node.	page 253
ListAppConfig	Environment	Displays the list of applications currently being partitioned in the current environment.	page 201
ListAppsToInstall	Node	Lists the names of the applications that need to be installed on the node represented by the current Node agent.	page 254
ListDistribs	Environment	Lists the application distributions available locally on the node running Escript.	page 202
	Node	Lists the application distributions available in the node managed by the current Node agent.	page 254
ListEnvs	Environment	Lists the names of the environments in the Environment Manager repository.	page 202
ListFile *file_name*	any	Lists the contents of the specified file onto standard output.	page 78
ListSavedAgents	any	Lists agents in the pool of saved agents.	page 78

Command	Agent	Description	See:
LoadDistrib *application_name compatibility_level*	Environment	Loads the specified application distribution into the environment repository from the node that is running the Escript utility.	page 202
	Node	Loads the specified application distribution into the environment from the node represented by the current Node agent.	page 255
LockEnv [*wait_flag*]	any	Obtains an exclusive lock on the environment until the next `Commit` or `UnlockEnv` command.	page 79
Ls [*directory_name*]	any	Lists the files in a directory.	page 80
ModLogger +(*logger_flags*) | -( )	any	Modifies the current logger flag settings for Escript.	page 81

Command	Agent	Description	See:
ModLoggerRemote +(*logger_flags*) | -(*logger_flags*)	Active Partition	Sets the logger flags for the active partition. If you are invoking this command within TOOL code, use the `ModLogger` command.	page 127
	Installed Partition	Sets the logger flags for all of the active partitions that are represented by this Installed Partition agent.	page 213
	NameService	Modify the current logger settings.	page 238
	Node	Sets the logger flags for the Node Manager that is being managed by the current Node agent.	page 255
	Partition	Sets the logger flags for all of the active partitions that are represented by the current Partition agent.	page 289
	RepositoryServer	Sets the logger flags for the active partition. If you are invoking this command within TOOL code, use the `ModLogger` command.	page 310
Mv *file1_name file2_name*	any	Renames a file in the local file system.	page 82
NewEnv *environment_name*	Environment	Creates a new simulated environment with the specified name.	page 203
NsCd *directory_name*	NameService	Changes the current name space directory.	page 239
NsLs [*directory_name*]	NameService	Lists the contents of a name space directory.	page 240
PrintStmtQueue	DBSession	Displays the statement cache.	page 181

Command	Agent	Description	See:
Pwd	any	Prints the name of the current working directory.	page 82
Quit	any	Exits Escript, prompting you to save if there are outstanding changes to the environment definition.	page 82
Reconnect	DBSession	Reconnects a database session to the database.	page 181
RecoverMemory	OperatingSystem	Attempts to perform a stable memory reclamation.	page 272
RefreshEnv	any	Forces an immediate refresh of the information about the current environment from the Environment Manager.	page 83
ReleaseAppConfig *client_id*	Environment	Forces a release of the configuration lock for the named application being configured in the named environment.	page 204
ReleaseLock	Application	Releases any installation locks on the current application.	page 146
• Remove3GLProj *project_name*	any	Removes a 3GL project from the list of those supported by the current node in the environment definition.	page 83
• RemoveCommProtocol *protocol_name*	any	Removes the specified communication protocol from the current node in the environment definition.	page 84
RemoveEnv *env_name*	Environment	Removes a simulated environment from the repository.	page 204

Command	Agent	Description	See:
• RemoveExternalRM *resource_manager_name*	any	Removes the specified external resource manager from the current node in the environment definition.	page 84
RemoveLostParts	NameService	Deletes information about partitions that the Environment Manager can no longer access.	page 241
• RemoveNode *node_name*	any	Removes the specified node from the environment definition.	page 85
Rm *file_name*	any	Removes a file in the local file system.	page 86
SaveAgent *tag_name*	any	Adds current agent to pool of saved (tagged) agents.	page 86
Script *file_name*	any	Captures Escript commands and writes them into a specified script file.	page 87
SetAppCompCompiled *node_name compiled_flag component_name*	Application	Declares whether a partition (or library) is to be used in compiled or iPlanet UDS executor form.	page 146
SetArgs *node_name arguments*	Partition	Sets the argument string used to startup the partition managed by the current Partition agent on a particular node.	page 289
SetCompiled *node_name is_compiled*	Partition	Turns on or off the compiled server attribute for the partition managed by the current Partition agent on a specified node.	page 291
• SetEnvForSim [*environment_name*]	any	Specifies the name of the environment definition to be simulated by the active environment.	page 87

Command	Agent	Description	See:
• SetEnvPrefNode [*node_name*]	any	Sets the preferred node for servers for the current environment definition.	page 88
SetEnvRemote *env_variable new_value*	Active Partition	Sets an environment variable for the active partition. If you are invoking this command within TOOL code, use the `SetEnvVar` command.	page 128
	Installed Partition	Sets the environment variable for all of the active instances of the installed partition managed by the current agent.	page 214
	Node	Sets an environment variable value for the Node Manager represented by the current Node agent.	page 256
	Partition	Sets the environment variable for all of the active instances of the partition managed by the current Partition agent.	page 291
	RepositoryServer	Sets an environment variable for the active partition. If you are invoking this command within TOOL code, use the `SetEnvVar` command.	page 311

Command	Agent	Description	See:
SetInstrumentLogging *instrument_name is_logged*	any	A toggle that sets the IsLogged attribute of an instrument. This determines if the instrument is on the list of instrument values that are regularly logged, assuming you have set a log timer and enabled logging to an active partition log file.	page 89
• SetNodeClient *client_flag*	any	Specifies the current node's client property.	page 90
• SetNodeForSim *node_name*	any	Sets the node to use in simulating the current node.	page 91
• SetNodeModel *model_flag*	any	Sets the current node's model property.	page 92
• SetPassword *old_password new_password*	any	Replaces the current environment password, needed to start Escript or the Environment Console, with the new password.	page 93
SetPath *directory_name* [;*directory_name*...]	any	Sets the directory search path used by any of the commands that take a file name as an argument.	page 94
SetRepCount *node_name replication_count*	Partition	Sets the replication count on a node for the partition managed by the current Partition agent	page 292
• SetSimForNode *simulation_flag*	any	Sets the current node's simulation property.	page 95
ShowAdmin	NameService	Shows information about the Name Service.	page 241
ShowAgent	any	Shows parent agent and subagents of current agent.	page 96

Command	Agent	Description	See:
• ShowEnv	any	Shows details of the environment definition.	page 96
ShowEnv [*env_name*]	NameService	Displays information about an environment or all environments known to this environment.	page 241
ShowInstrument *instrument_name*	any	Refreshes the instrument value for any instrument.	page 96
• ShowNode	any	Shows details of the current node.	page 97
ShowPart [*partition_id*]	NameService	Shows information about partitions known to this environment.	page 242
ShowPath	any	Shows the current search path.	page 97
ShowSubAgent *subagent_name*	any	Shows information about the subagent without changing current agent.	page 98

Command	Agent	Description	See:
Shutdown [*kill_executors*]	Active Partition	Shuts down the active partition represented by this agent.	page 128
	Application	Shuts down all server partitions running in the application on all nodes.	page 147
	Environment	Shuts down the Environment Manager and its agent.	page 205
	Installed Partition	Shuts down all active instances of the installed partition represented by the current agent.	page 215
	NameService	Shuts down the NameService agent and the corresponding Name Server.	page 243
	Node	Shuts down the Node Manager represented by this agent.	page 257
	Partition	Shuts down all active instances of the partition represented by the current Partition agent.	page 293
	RepositoryServer	Shuts down the repository server.	page 312
	RepositoryServer Info	Shuts down the repository server represented by this agent.	page 319

Command	Agent	Description	See:
ShutdownSubAgent *subagent*	Application	Shuts down the named subagent and its managed object.	page 148
	Environment	Shuts down the named subagent and its managed object.	page 206
	Installed Partition	Shuts down the named subagent and its managed object.	page 215
	Node	Shuts down the named subagent and its managed object.	page 258
	Partition	Shuts down the named subagent and its managed object.	page 293
StartInstPart *partition_name* [*one_more*]	Node	Starts one instance of an installed partition on the node managed by the current Node agent (Node Manager).	page 258
Startup	Application	Starts all server partitions (with all their replicates) of the current application.	page 149
Startup [*argument_list*]	Installed Partition	Starts one instance of the current installed partition on the current node.	page 216
Startup	Partition	Starts all server partitions (with all their replicates) for the partition managed by the current Partition agent.	page 294
Step	any	Allows you to step through the commands interactively before they are executed.	page 98
Unassign *node_name*	Partition	Dissolves the node assignment of the partition managed by the current Partition agent.	page 294

Command	Agent	Description	See:
UnassignAppComp *node_name component_name*	Application	Removes the assignment of an application component from a node.	page 149
Uninstall	Application	Removes the definition of the current application from the environment.	page 150
UnlockEnv (no save)	any	Unlocks the exclusive lock on the environment, and aborts any changes made since the last `LockEnv` command in this session.	page 99
UnlockWorkspace *workspace* [*password*]	RepositoryServer	Frees all locks held on the given workspace.	page 312
UpdateInstrument *instrument_name* [*instrument_data*...]	any	Used to set the value of an instrument that is not read only.	page 99
UseLocal	any	Sets Escript to expect file names to be specified in local operating system format, rather than iPlanet UDS portable file name format.	page 100
UsePortable	any	Sets Escript to expect file names to be specified in iPlanet UDS portable file name format, rather than local operating system format.	page 101
WaitForEnvMgr [*number_seconds*]	any	Forces scripts to wait for the Environment Manager to start before continuing execution.	page 102
WhichFile *file_name*	any	Searches through the directories in the current directory search path to locate the first directory in which the specified file exists.	page 103

Index