CHAPTER 6

# WSA Control System Command Information

This chapter alphabetically lists and explains the Control System server process. Of particular interest are the **WSA_Dispatcher**, **WSA_Scheduler**, and **WSA_WatchDog** commands.



## Mandatory Support Files

The file **paramlist.dat** must exist in the $WSADIR/tcl/data directory and hold a list of all parameters that WSA is able to configure in the networks under its control. This file is generated automatically and released as part of the default system build. The script $WSADIR/etc/mkpl_dat.sh generates the contents of paramlist.dat and must be run whenever the contents of the database table Param_list change.

# Trace Code and Logfiles

In the WatchDog, Scheduler, and Dispatcher servers, the **-v** flags turn on verbose reporting mode for tracking internal events and errors.

The logfile is in the current working directory from which **CSCOwsa** was invoked, usually **$WSADIR/bin**, and holds all the trace files.

Default **$WSADIR = /opt/CSCOwsa**

---

**Note** If this mode is used, then the file should be purged regularly to conserve disk usage. The time stamp in the logfile for actions by WSA_Dispatcher, WSA_Scheduler, WSA_WatchDog, and WSA_Loader are in Universal Time (UT) for logging purposes.

---

# WSA_Audit

This section gives the description, syntax, and arguments (listed alphabetically) for the **WSA_Audit** command.

## Description

WSA_Audit connects to SV+ networks known to WSA via access points defined in the WSA database. It runs the same discovery probe as WSA_Probe to determine what nodes, slots, cards, ports, trunks, and connections exist in these networks.

Once discovery is complete, WSA_Audit compares the information it has found with that in the WSA database and produces errata records in the WSA database that can be viewed with the WSA Graphical Client.

WSA_Audit can also compare the network with a file produced earlier by WSA_Probe, producing only errata information on stdout.

## Syntax

```
WSA_Audit [-d<database>] [-e] [-i<filename>]
[-L <sev>[:<lev>[<sev>[:<lev>]]] [-n<netname>] [-v] [-x]
```

**Note** If no arguments are supplied, then all networks are audited.

## Arguments (all are optional)

| Option | Description | Default |
|---|---|---|
| **-d**<*database*> | The database to use | **DEFAULT $WSADATABASE** |
| **-e** | List cards, ports, trunks, conns in missing nodes, cards, and ports | |
| **-i**<*file*> | File to use | Use the network |
| **-L** <*sev*>[**:**<*lev*>[<*sev*>[**:**<*lev*>]]] | Log messages at or above both of these severity levels | |
| where: | | |
| <*sev*> | INFO, WARN, ERR, or SEV | |
| <*lev*> | 0 to 10 | |
| | <*sev*>:<*lev*> | INFO:10 |
| | is for **stderr** | |
| | <*sev*>:<*lev*> | same as the first pair <*sev*>:<*lev*> |
| | is for **logfile** | |
| **-n**<*netname*> | Network to audit, rename the FILE net | All in the database |
| **-v** | Verbose mode, also show software build, version, and initialization status | |
| **-x** | Command usage explanation | |

## Auditing Networks

The WSA_Audit process is triggered based on the audit schedule as entered from the WSA Administration GUI. It is imperative that WSA Discovery Server **wsadisc** is running on the SV+ Solaris server so that network discovery can be made and compared.

To run WSA_Audit manually, do the following:

**Step 1** Log in as user **wsa** on the WSA Solaris server.

**Step 2** Ensure that the WSA Discovery Server **wsadisc** is running on the SV+ Solaris server (refer to section Starting and Stopping the WSA Discovery Server, page 5-4).

**Step 3** Execute WSA_Audit as follows (refer to the information earlier in this section for more details):

**WSA_Audit**

Launch the WSA Explorer application and invoke the Audit Log window using the Tools menu to view the audit results.

# WSA_Dispatcher

This section gives the description, syntax, and arguments (listed alphabetically) for the **WSA_Dispatcher** command.

## Description

The function of the WSA_Dispatcher is to send messages from a centralized WSA control system to multiple networks, via network access points.

- The Dispatcher retains a list of all client schedulers that feed commands to it.
- For each client scheduler, a remote procedure call session is set up.
- The Dispatcher holds a list of network access points for all networks to which it sends commands.
- The Dispatcher spawns a child process for each defined Network Access Point that holds all messages for that network in a queue, the process acts as a queue manager.
- For each network, tasks are selected at their due time and sent to a selected network access point.
- When the task is completed, the task is marked ended in the persistent storage of the dispatcher.
- Each client scheduler can ask for a list of completed tasks and their result code. This is then communicated back to the main WSA database.

## Syntax

**WSA_Dispatcher** [**-d** <*database*>] [**-L** <*sev*> [**:** <*lev*> [<*sev*>[**:**<*lev*>]]]] [**-v**] [**-x**]

# Arguments (all are optional)

| Option | Description | Default |
|---|---|---|
| **-d**<*database*> | WSA dispatcher database name | wsadisp |
| **-L** <*sev*>[**:**<*lev*>[<*sev*>[**:**<*lev*>]]] | Log messages at or above both of these severity levels | |
| where: | | |
| <*sev*> | INFO, WARN, ERR, or SEV | |
| <*lev*> | 0 to 10 | |
| | <*sev*>:<*lev*> | INFO:10 |
| | is for **stderr** | |
| | <*sev*>:<*lev*> | same as the first pair |
| | is for **logfile** | <*sev*>:<*lev*> |
| **-v** | Verbose mode, also show software build, version, and initialization status | |
| **-x** | Command usage explanation | |

# WSA_Loader

This section gives the description, syntax, and arguments (listed alphabetically) for the **WSA_Loader** command.

## Description

WSA_Loader connects to networks known to WSA via access points defined in the WSA database. It runs the same discovery probe as WSA_Probe to determine what nodes, slots, cards, ports, trunks, and connections exist in these networks.

Once discovery is complete, WSA_Loader populates the WSA database with the information it has found, issuing warnings about invalid information from the network and/or in the WSA database.

An alternative method of loading the WSA database is to get WSA_Loader to read an output file from WSA_Probe.

---

**Note** WSA_Loader loads the initial configurations of the network into the **wsamain** database. After the network is created in the database, the user can use WSA_Audit to compare the network with the database and make proper modifications through WSA_Audit or the GUI. WSA_Loader is not meant for incremental loading.

---

## Syntax

```
WSA_Loader [-a<ip address> -u<username> -p<password>]
[-c [<dd/mm/yyyy>][:][<hh:mm:ss>][,[dd/mm/yyyy][:][<hh:mm:dds>]]
[-d<database>] [-i<file>] [-L<sev>[:<lev>[<sev>[:<lev>]]]] [-n<netname>]
[-R<snmp read>] [-T] [-v] [-W<snmp write>] [-x]
```

# Arguments (all are optional)

| Option | Description | Default |
|---|---|---|
| **-a**<*ip address*> | ip address of the network access point | |
| **-c** [<*dd*/*mm*/*yyyy*>][**:**][<*hh*:*mm*:*ss*>][**,** [<*dd*/*mm*/*yyyy*>][**:**][<*hh*:*mm*:*ss*>]] | Calendar slot into which to load objects | |
| **-d**<*database*> | Main WSA database to use | **DEFAULT** **$WSADATABASE** |
| **-i**<*file*> | File of network information produced by **WSA_Probe** | |
| **-L** <*sev*>[**:**<*lev*>[<*sev*>[**:**<*lev*>]]] | Log messages at or above both of these severity levels | |
| where: | | |
| <*sev*> | INFO, WARN, ERR, or SEV | |
| <*lev*> | 0 to 10 | |
| | <*sev*>:<*lev*> | INFO:10 |
| | is for **stderr** | |
| | <*sev*>:<*lev*> | same as the first pair <*sev*>:<*lev*> |
| | is for **logfile** | |
| **-n**<*netname*> | From the database: which network to load<br>From the file: rename all entries to netname | Default_Network |
| **-p**<*password*> | Command line password at the network access point | |
| **-R**<*snmp read*> | snmp read mode | public |
| **-T** | Create card types for unknown card types | |
| **-u**<*username*> | Command line user name at the network access point | |
| **-v** | Verbose mode, also show software build, version, and initialization status | |

| Option | Description | Default |
|---|---|---|
| **-W**<*snmp write*> | snmp write mode | private |
| **-x** | Command usage explanation | |

# WSA_Probe

This section gives the description, syntax, and arguments (listed alphabetically) for the **WSA_Probe** command.

## Description

WSA_Probe gains access to a network via a nominated node, and runs a discovery probe to determine what nodes, slots, cards, ports, trunks, and connections exist in this network.

Once discovery is complete, WSA_Probe dumps its findings into flat files for import to spread sheets or loading into a WSA main database using the WSA_Loader utility.

## Syntax

```
WSA_Probe -n<netname> -u<username> -p<password> [-a<ip address>]
[-L<sev>[:<lev>[<sev>:<lev>]]] [-o<filename>][-R<snmp read>] [-v]
[-W<snmp write>] [-x]
```

# Arguments

| Option | Description | Default |
|---|---|---|
| **-a**<*ip address*> | Hostname of machine on which SV+ is running | |
| **-L** <*sev*>[**:**<*lev*>[<*sev*>[**:**<*lev*>]]] | Log messages at or above both of these severity levels | |
| where: | | |
| <*sev*> | INFO, WARN, ERR, or SEV | |
| <*lev*> | 0 to 10 | |
| | <*sev*>:<*lev*> is for **stderr** | INFO:10 |
| | <*sev*>:<*lev*> is for **logfile** | same as the first pair <sev>:<lev> |
| **-n**<*netname*> | What to name the network | Default_Network |
| **-o**<*filename*> | Name of file in which the probe data is dumped. Note: WSA_Probe creates a file containing the data. The name is appended to the file name. | use stdout |
| **-p**<*password*> | SV+ user password | |
| **-R**<*snmp read*> | snmp read community string | public |
| **-u**<*username*> | SV+ user name | |
| **-v** | Verbose mode, also show software build, version, and initialization status | |
| **-W**<*snmp write*> | snmp write mode | private |
| **-x** | Command usage explanation | |

# WSA_Scheduler

This section gives the description, syntax, and arguments (listed alphabetically) for the **WSA_Scheduler** command.

## Description

WSA_Scheduler scans the table of tasks to be actioned, that is **add_conns** and **del_conns** (as created by WSA_Watchdog), and sends them to its WSA_Dispatcher early enough for them to be sent to the networks with the required time accuracy. Periodically it asks its WSA_Dispatcher for notification of ended tasks and marks their status in the WSA database.

---

**Note** The WSA_Dispatcher must be running prior to the WSA_Scheduler initialization or a failure occurs.

---

The sequence of events on it is:

- On initialization, a connection is made with a named <*logical scheduler*> to the dispatcher.
- In normal operation, the Scheduler then sends database objects to the dispatcher or sends updates to objects if they already exist and have changed attributes.
- When the WSA_Scheduler receives notification from the WSA_Dispatcher of a complete task (either success or failure), it records the result in the WSA database and creates an event record.
- Failed tasks are retried after minimal delay, a user defined number <*retry_count*> of times.
- If the maximum retry attempts have failed, then tasks are deferred for retrial at a later time <*now* + *Defer_time*>. The Defer time is user defined. This deferral itself can occur either until success or a user defined maximum number <*defer_count*> of times.

# Syntax

**WSA_Scheduler** [**-.**<*#1*><*#2*>] [**-:**<*secs1*><*secs2*>] [**-c**<*secs*>] [**-C**<*secs*>][**-d**] [**-E**<*secs*>] [**-i**] [**-L** <*sev*>[**:**<*lev*>[<*sev*>[**:**<*lev*>]]][**-m**<*#*>] [**-r**<*secs*>] [**-s**<*+*><*hostnm*>] [**-v**] [**-x**]

# Arguments (all are optional)

| Option | Description | Default |
|---|---|---|
| **-.**<*#1*><*#2*> | *#1*: Maximum number of task retries | DEFAULT 2 |
| | *#2*: Maximum number of task deferrals | DEFAULT 2 |
| **-:**<*secs1*><*secs2*> | *secs1*: Maximum number of task delay seconds for retry | DEFAULT 2 secs |
| | *secs2*: Maximum number of task delay seconds for deferrals | DEFAULT 60 secs |
| **-c**<*secs*> | Interval between task progress checks | DEFAULT 20 |
| **-C**<*secs*> | Seconds to allow tasks to complete | DEFAULT 720 |
| **-d** | dbname_database name | |
| **-E**<*secs*> | secs by which to dispatch tasks early | DEFAULT 20 |
| **-i** | Reconfigure the dispatcher for new access points | |
| **-L** <*sev*>[**:**<*lev*>[<*sev*>[**:**<*lev*>]]] | Log messages at or above both of these severity levels | |
| where: | | |
| <*sev*> | INFO, WARN, ERR, or SEV | |
| <*lev*> | 0 to 10 | |
| | <*sev*>:<*lev*> | INFO:10 |
| | is for **stderr** | |
| | <*sev*>:<*lev*> | same as the first pair <*sev*>:<*lev*> |
| | is for **logfile** | |
| **-m**<*#*> | Maximum number of contiguous message sends | DEFAULT 20 |

| Option | Description | Default |
|---|---|---|
| **-r**<*secs*> | Interval between scanning for tasks | DEFAULT 60 |
| **-s**<*+*><*hostnm*> | Scheduler name to dispatcher | DEFAULT sched1<br>Note: The **+** means re-create the virtual scheduler |
| **-v** | Verbose mode, also show software build, version, and initialization status | |
| **-x** | Command usage explanation | |

# WSA_WatchDog

This section gives the description, syntax, and arguments (listed alphabetically) for the **WSA_WatchDog** command.

## Description

WSA_WatchDog does the following three things:

- **Object Actioning**: Generates tasks required by WSA database objects for customer connections, that is both **add_cons** and **del_cons**. The tasks are created to be executed at the required action-date and time, as defined by the WSA connection and disconnection request using WSA GUI.
- **Audit Result Actioning**: Scans the latest Audit run of each network and performs the actions required when marked approved. The Audit Results indicate discrepancies between the WSA database and the live network. For objects that are missing in the WSA database but found on the live network, the user can modify the WSA database to reflect the live network by using the Audit Log tool from WSA GUI.
- **Task Debriefing**: Scans tasks that are completed and updates their status.

All of the three previously listed items can either not be done at all, be done once, or be done periodically, and the WSA_WatchDog itself can run continuously or periodically, as defined by the user in the **CSCOwsa** script.

The generation of tasks must de done in advance of the task's required **action-date** and **time** to ensure that the Scheduler can accurately schedule tasks at the appropriate date and time.

## Syntax

```
WSA_WatchDog -u -l [-a <secs>] [-A]
[-c [<dd/mm/yyyy>][:][<hh:mm:ss>][,[dd/mm/yyyy][:][<hh:mm:dds>]] [-d]
[-i<secs> [@<secs>]
[-L <sev>[:<lev>[<sev>[:<lev>]]] [-m<secs> [@<secs>]] [-n] [-s<secs>] [-v]
[-x] [-z<secs>]
```

# Arguments for Object Actioning

| Option | Description | Default |
|---|---|---|
| **-a**<*secs*> | Seconds in advance to prepare tasks<br>Note: Pick up all tasks up to **-a** seconds in advance of required live time. | DEFAULT 1800 (30 mins) |
| **-A** | aid—Identifier of Audit_Result to use | DEFAULT latest |
| **-c** [<*dd*/*mm*/*yyyy*>][**:**][<*hh*:*mm*:*ss*>][**,** [<*dd*/*mm*/*yyyy*>][**:**][<*hh*:*mm*:*ss*>]] | Calendar slot into which to load objects | |
| **-d** | database—Main WSA database to use | **DEFAULT $WSADATABASE** |
| **-i**<*secs*>[**@**<*secs*>] | Interval between task creates | DEFAULT 20*60 (20 mins) |
| **-l** | Lenient mode<br>Note: Allow becoming live when the **cease_date** is expired | |
| **-L** <*sev*>[**:**<*lev*>[<*sev*>[**:**<*lev*>]]]<br>where:<br><*sev*><br><*lev*> | Log messages at or above both of these severity levels<br>INFO, WARN, ERR, or SEV<br>0 to 10 | |
| | <*sev*>**:**<*lev*><br>is for **stderr** | INFO:10 |
| | <*sev*>**:**<*lev*><br>is for **logfile** | same as the first pair <*sev*>:<*lev*> |
| **-n** | network—Network to use | DEFAULT all |
| **-s**<*secs*> | Ignore tasks that are more than <*secs*> seconds late<br>Note: Do not pick up tasks more than <*secs*> seconds old | DEFAULT 64800 (18 hours) |

| Option | Description | Default |
|---|---|---|
| **-u** | Allow becoming live when the authorized flag is not set<br>Note: If this **-u** argument is not supplied, ensure that the connection request is authorized using the connection properties in WSA GUI. | |
| **-v** | Verbose mode, also show software build, version, and initialization status | |
| **-x** | Command usage explanation | |

## Argument for Audit Result Actioning

| Option | Description | Default |
|---|---|---|
| **-m**<*secs*>[**@**<*secs*>] | Interval between mending attempts | DEFAULT 0 (Don't Mend) |

## Argument for Task Debriefing

| Option | Description | Default |
|---|---|---|
| **-z**<*secs*> | Interval between checking for completed tasks | DEFAULT 300 seconds |

**Note** The WSA_WatchDog process for audit result actioning must be invoked separately from the Object Actioning/Task Debriefing WatchDog server.

To invoke a WatchDog server process for audit result action, do the following to mend once every day:

```
WSA_WatchDog -m 86400 -i 0 -z 0
```