WAE Administration

This section contains the following topics:

Manage Users

In WAE 7.0, all users have the administrator role. The following procedure describes how to create and delete users.

Step 1 From the WAE UI, choose **System** > **User Manager**.

Step 2 To add a user, click **+Add User** and fill in all applicable fields.

Step 3 To change a user's password:

a) From the user row, click the pencil icon.
b) Update the password fields.
c) Click **Save**.

Step 4 To delete a user:

a) From the user row, click the trash icon.

Configure Aging

By default, when a circuit, port, node, or link disappears from a network, it is permanently removed and must be rediscovered. To configure how long WAE retains these elements that have disappeared before they are permanently removed from the network, complete the following steps.

Note This is a global option that will be configured for all networks.

Step 1 From the Expert Mode, navigate to **/wae:wae** and click the **nimos** tab.

Step 2 Click **aging**.

Step 3 Enter the number of minutes that WAE must retain the elements (circuit, node, port, or link) in the appropriate fields.

Step 4 Click **Commit**.

wae.conf

`wae.conf` is an XML configuration file that is formally defined by a YANG model, `tailf-ncsconfig.yang`. This YANG file is included in the WAE distribution, as is a commented `wae.conf.example` file.

The `wae.conf` file controls the baseline of the WAE run time. You can change certain configuration parameters in the `wae.conf` file; for example, you can change the default port that WAE runs on (port 8080) to another port.

Whenever you start or reload the WAE daemon, it reads its configuration from `./wae.conf` or *`<waeruntime-directory>`*`/etc/wae.conf`.

The following example shows *`<waeruntime-directory>`*`/etc/wae.conf` contents:

```
<!-- -*- nxml -*- -->
<!-- Example configuration file for wae. -->

<ncs-config xmlns="http://tail-f.com/yang/tailf-ncs-config">

  <!-- WAE can be configured to restrict access for incoming connections -->
  <!-- to the IPC listener sockets. The access check requires that -->
  <!-- connecting clients prove possession of a shared secret. -->
  <ncs-ipc-access-check>
    <enabled>false</enabled>
    <filename>${NCS_DIR}/etc/ncs/ipc_access</filename>
  </ncs-ipc-access-check>

  <!-- Where to look for .fxs and snmp .bin files to load -->

  <load-path>
    <dir>./packages</dir>
    <dir>${NCS_DIR}/etc/ncs</dir>

    <!-- To disable northbound snmp altogether -->
    <!-- comment out the path below -->
```

```
  <dir>${NCS_DIR}/etc/ncs/snmp</dir>
</load-path>

<!-- Plug and play scripting -->
<scripts>
  <dir>./scripts</dir>
  <dir>${NCS_DIR}/scripts</dir>
</scripts>

<state-dir>./state</state-dir>

<notifications>
  <event-streams>

    <!-- This is the builtin stream used by WAE to generate northbound -->
    <!-- notifications whenever the alarm table is changed. -->
    <!-- See tailf-ncs-alarms.yang -->
    <!-- If you are not interested in WAE northbound netconf notifications -->
    <!-- remove this item since it does consume some CPU -->
    <stream>
      <name>wae-alarms</name>
      <description>WAE alarms according to tailf-ncs-alarms.yang</description>
      <replay-support>false</replay-support>
      <builtin-replay-store>
        <enabled>false</enabled>
        <dir>./state</dir>
        <max-size>S10M</max-size>
        <max-files>50</max-files>
      </builtin-replay-store>
    </stream>

    <!-- This is the builtin stream used by WAE to generate northbound -->
    <!-- notifications for internal events. -->
    <!-- See tailf-ncs-devices.yang -->
    <!-- Required for cluster mode. -->
    <stream>
      <name>wae-events</name>
      <description>WAE event according to tailf-ncs-devices.yang</description>
      <replay-support>true</replay-support>
      <builtin-replay-store>
        <enabled>true</enabled>
        <dir>./state</dir>
        <max-size>S10M</max-size>
        <max-files>50</max-files>
      </builtin-replay-store>
    </stream>

    <!-- This is the builtin stream used by WAE to generate northbound -->
    <!-- notifications forwarded from devices. -->
    <!-- See tailf-event-forwarding.yang -->
    <stream>
      <name>device-notifications</name>
      <description>WAE events forwarded from devices</description>
      <replay-support>true</replay-support>
      <builtin-replay-store>
        <enabled>true</enabled>
        <dir>./state</dir>
        <max-size>S10M</max-size>
        <max-files>50</max-files>
      </builtin-replay-store>
    </stream>

    <!-- This is the builtin stream used by WAE to generate northbound -->
    <!-- notifications for plan state transitions. -->
```

```
    <!-- See tailf-ncs-plan.yang -->
    <stream>
      <name>service-state-changes</name>
      <description>Plan state transitions according to
      tailf-ncs-plan.yang</description>
      <replay-support>false</replay-support>
      <builtin-replay-store>
        <enabled>false</enabled>
        <dir>./state</dir>
        <max-size>S10M</max-size>
        <max-files>50</max-files>
      </builtin-replay-store>
    </stream>
    <stream>
      <name>XtcNotifications</name>
      <description>Xtc object change notifications</description>
      <replay-support>false</replay-support>
    </stream>
  </event-streams>
</notifications>

<!-- Where the database (and init XML) files are kept -->
<cdb>
  <db-dir>./ncs-cdb</db-dir>
  <!-- Always bring in the good system defaults -->
  <init-path>
    <dir>${NCS_DIR}/var/ncs/cdb</dir>
  </init-path>
</cdb>


<!--
    These keys are used to encrypt values of the types
    tailf:des3-cbc-encrypted-string and tailf:aes-cfb-128-encrypted-string.
    For a deployment install it is highly recommended to change
    these numbers to something random (done by WAE "system install")
-->
<encrypted-strings>
  <DES3CBC>
    <key1>0123456789abcdef</key1>
    <key2>0123456789abcdef</key2>
    <key3>0123456789abcdef</key3>
    <initVector>0123456789abcdef</initVector>
  </DES3CBC>

  <AESCFB128>
    <key>0123456789abcdef0123456789abcdef</key>
    <initVector>0123456789abcdef0123456789abcdef</initVector>
  </AESCFB128>
</encrypted-strings>


<logs>
  <syslog-config>
    <facility>daemon</facility>
    <udp>
      <enabled>false</enabled>
      <host>syslogsrv.example.com</host>
    </udp>
  </syslog-config>

  <ncs-log>
    <enabled>true</enabled>
    <file>
```

```
    <name>./logs/wae.log</name>
    <enabled>true</enabled>
  </file>
  <syslog>
    <enabled>true</enabled>
  </syslog>
</ncs-log>

<developer-log>
  <enabled>true</enabled>
  <file>
    <name>./logs/devel.log</name>
    <enabled>true</enabled>
  </file>
</developer-log>
<developer-log-level>trace</developer-log-level>

<audit-log>
  <enabled>true</enabled>
  <file>
    <name>./logs/audit.log</name>
    <enabled>true</enabled>
  </file>
</audit-log>

<netconf-log>
  <enabled>true</enabled>
  <file>
    <name>./logs/netconf.log</name>
    <enabled>true</enabled>
  </file>
</netconf-log>

<snmp-log>
  <enabled>true</enabled>
  <file>
    <name>./logs/snmp.log</name>
    <enabled>true</enabled>
  </file>
</snmp-log>

<webui-browser-log>
  <enabled>true</enabled>
  <filename>./logs/webui-browser.log</filename>
</webui-browser-log>

<webui-access-log>
  <enabled>true</enabled>
  <dir>./logs</dir>
</webui-access-log>


 <!-- This log is disabled by default if wae is installed using -->
<!-- the 'system-install' flag. It consumes a lot of CPU power -->
<!-- to have this log turned on, OTOH it is the best tool to -->
<!-- debug must expressions in YANG models -->

<xpath-trace-log>
  <enabled>false</enabled>
  <filename>./logs/xpath.trace</filename>
</xpath-trace-log>

<error-log>
  <enabled>true</enabled>
```

```
    <filename>./logs/wae-err.log</filename>
  </error-log>

</logs>

<ssh>
  <algorithms>
    <mac>hmac-sha1,hmac-sha2-256,hmac-sha2-512</mac>
    <encryption>aes128-ctr,aes192-ctr,aes256-ctr</encryption>
  </algorithms>
</ssh>

<aaa>
  <ssh-server-key-dir>${NCS_DIR}/etc/ncs/ssh</ssh-server-key-dir>

  <!-- Depending on OS - and also depending on user requirements -->
  <!-- the pam service value value must be tuned. -->

  <pam>
    <enabled>true</enabled>
    <service>common-auth</service>
  </pam>
  <external-authentication>
    <enabled>false</enabled>
    <executable>my-test-auth.sh</executable>
  </external-authentication>

  <local-authentication>
    <enabled>true</enabled>
  </local-authentication>

</aaa>

<!-- Hash algorithm used when setting leafs of type ianach:crypt-hash, -->
<!-- e.g. /aaa/authentication/users/user/password -->
<crypt-hash>
  <algorithm>sha-512</algorithm>
</crypt-hash>

<!-- Disable this for performance critical applications, enabling -->
<!-- rollbacks means additional disk IO for each transaction -->
<rollback>
  <enabled>true</enabled>
  <directory>./logs</directory>
  <history-size>50</history-size>
</rollback>


<cli>
  <enabled>true</enabled>

  <!-- Use the builtin SSH server -->
  <ssh>
    <enabled>true</enabled>
    <ip>0.0.0.0</ip>
    <port>2024</port>
  </ssh>

  <prompt1>\u@wae> </prompt1>
  <prompt2>\u@wae% </prompt2>

  <c-prompt1>\u@wae# </c-prompt1>
  <c-prompt2>\u@wae(\m)# </c-prompt2>
```

```
  <show-log-directory>./logs</show-log-directory>
  <show-commit-progress>true</show-commit-progress>
  <suppress-commit-message-context>maapi</suppress-commit-message-context>
  <suppress-commit-message-context>system</suppress-commit-message-context>
</cli>

<webui>
  <absolute-timeout>PT0M</absolute-timeout>
  <idle-timeout>PT30M</idle-timeout>
  <enabled>true</enabled>
  <transport>
    <tcp>
      <enabled>true</enabled>
      <ip>0.0.0.0</ip>
      <port>8080</port>
     <redirect>https://@HOST@:8443</redirect>
    </tcp>
    <ssl>
      <enabled>true</enabled>
      <ip>0.0.0.0</ip>
      <port>8443</port>
      <key-file>${NCS_DIR}/var/ncs/webui/cert/host.key</key-file>
      <cert-file>${NCS_DIR}/var/ncs/webui/cert/host.cert</cert-file>
    </ssl>
  </transport>

  <cgi>
    <enabled>true</enabled>
    <php>
      <enabled>false</enabled>
    </php>
  </cgi>
</webui>

<rest>
  <enabled>true</enabled>
</rest>

<restconf>
  <enabled>true</enabled>
</restconf>

<netconf-north-bound>
  <enabled>true</enabled>

  <transport>
    <ssh>
      <enabled>true</enabled>
      <ip>0.0.0.0</ip>
      <port>2022</port>
    </ssh>
    <tcp>
      <enabled>false</enabled>
      <ip>127.0.0.1</ip>
      <port>2023</port>
    </tcp>
  </transport>
</netconf-north-bound>

<!-- <ha> -->
<!--   <enabled>true</enabled> -->
<!-- </ha> -->

<large-scale>
```

```
    <lsa>
      <!-- Enable Layered Service Architecture, LSA. This requires
           a separate Cisco Smart License.
      -->
      <enabled>true</enabled>
    </lsa>
  </large-scale>

</ncs-config>
```

The default values for many configuration parameters are defined in the YANG file. See wae.conf Configuration Parameters.

LSA Configuration

The basic idea of layered service architecture (LSA) is to split a service into an upper layer and one or several lower level parts. This can be viewed as splitting the service into a customer-facing (CFS) and a resource-facing (RFS) part. The CFS code (upper-level) runs in one (or several) NSO cfs-nodes, and the RFS code (lower-level) runs in one of many NSO rfs-nodes. The rfs-nodes have each a portion of the managed devices mounted in their / devices tree and the cfs-node(s) have the NSO rfs-nodes mounted in their /devices tree.

This section assumes the following:

- You have an understanding of Cisco Network Service Orchestrator (NSO) with regards to deployment, device configuration, and LSA. See the NSO 4.4 documentation for details. Specifically, see the *NSO Getting Started* and the *NSO Layered Service Architecture* guides.
- There are one or more NSO installations (for each LSA resource-facing (RFS) node) on the machine. The device option configuration for each instance is noted in the procedure.

LSA Configuration Workflow

This workflow describes the high-level steps to configure LSA in WAE.

Step	For more information, see...
1. Install additional packages for WAE LSA configuration.	Install LSA Packages, on page 8
2. Configure WAE to support LSA.	Configure WAE for LSA, on page 9
3. Bootstrap the RFS model.	Bootstrap the RFS Model for LSA, on page 10

Install LSA Packages

Prior to configuring WAE to support layered service architecture (LSA), you must install additional WAE NSO packages and enable LSA in the ncs.conf file.

Step 1 From `<wae_installation_directory>/packages/lsa`, copy `cisco-wae-rfs` to the run/packages directory on the NSO node.

Step 2 (Optional) If using the lsp-config-nimo for LSP collection, do the following:

a) From `<wae_installation_directory>/packages/lsa`, copy `cisco-wae-lsp-rfs` to the NSO run/packages directory on the NSO node.
b) Copy the required RFS NED to the NSO node. For example:
`<wae_installation_directory>/packages/cisco-wae-lsp-rfs/cisco-wae-lsp-rfs-iosxr`.

Step 3 Ensure that the RFS LSP CONFIG NIMO package is installed. Copy `<wae_installation_directory>/cisco-wae-lsp-config-nimo/cisco-wae-lsp-config-nimo-rfs` to `<wae_run_time_directory>/packages directory on the WAE node.`

Step 4 Enable the LSA feature in the ncs.conf file:

```
<large-scale>
    <lsa>
      <!-- Enable Layered Service Architecture, LSA. This requires
           a separate Cisco Smart License.
      -->
      <enabled>true</enabled>
    </lsa>
  </large-scale>
```

Step 5 Click the **Commit** button.

Step 6 Complete the steps documented in Configure WAE for LSA, on page 9.

Configure WAE for LSA

To configure WAE to support LSA, you must configure certain device options.

This procedure assumes the following:

- You have an understanding of Cisco Network Service Orchestrator (NSO) with regards to deployment, device configuration, and LSA. See the NSO 4.4 documentation for details. Specifically, see the *NSO Getting Started* and the *NSO Layered Service Architecture* guides.
- There are one or more NSO installations (for each LSA resource-facing (RFS) node) on the machine. The device option configuration for each instance is noted in the procedure.

Step 1 In WAE, create an LSA NETCONF device for each LSA RFS node with the following configuration options:

From the Expert mode, navigate to **`/ncs:devices/device/rfs_node`** to easily view options.

- From the device tab:
  - address—IP address of LSA RFS node.
  - port—The netconf-north-bound port configured on the LSA RFS node in ncs.conf. The default is 2022, but each LSA RFS node instance could be configured to have a different port number.
  - use-lsa—Check this box to enable LSA.
- From the device-type tab:
  - netconf— Confirm that check box is checked.
  - ned-id— Select **ned:lsa-netconf.**

- From the ssh tab, click **fetch-host-keys**.
- From the state tab, select **unlocked** under the admin-state drop-down list.
- From the device tab, click **sync-from**.

Step 2 Click the **Commit** button.

Step 3 From the Expert mode, navigate to **`/wae:wae/lsa`** and add all the LSA RFS nodes you just configured in the rfs-node drop-down list.

Step 4 Click the **Commit** button.

Bootstrap the RFS Model for LSA

Bootstrapping the RFS model populates the nodes of the RFS model from available devices. It includes the vendor, operating system, and management-IP information (not LSP-related information). The information can be populated from the RFS node by invoking the `wae-rfs services wae-rfs run-collection` command. This updates the list of nodes (possibly removing non-existent nodes).

Step 1 Invoke the RFS collection from the RFS node:

```
admin@nso# wae-rfs services wae-rfs run-collection
status true
message Wae-rfs Collection done
admin@nso#
```

Step 2 Perform a device sync (`devices sync-from`) to update WAE with the new RFS model and confirm that the RFS model is populated with devices that are present on the LSA RFS node:

```
admin@wae# show running-config devices device rfs1 config wae-rfs:wae-rfs rfs-model nodes node XRv-2
devices device rfs1
 config
  wae-rfs:wae-rfs rfs-model nodes node XRv-2
   ip-manage 192.0.2.24
   platform vendor Cisco
   platform os "IOS XR"
  !
 !
!
```

Note The device model on the NSO instance is assumed to be in-sync. Sync devices (**devices sync-from**) to avoid device sync errors during deployment.

Step 3 (For LSP deployments) Do the following to populate the LSP portion of RFS model:

a) For convenience, you can populate the RFS LSP model from the CFS node through a remote action, but this may result in NED read timeout errors. To avoid the errors, disable the lsp-config-nimo `perform-sync-from` advanced option.

b) Populate the RFS LSP information of the RFS model directly from the RFS node using the following command:
`wae-rfs services lsp-rfs run-collection`

c) Sync devices (**devices sync-from**) again to update WAE.

What to do next

You can perform one of the following:

- LSP Collection Using NSO NEDs
- Multi-Layer Collection

Troubleshoot LSA

Note You can view logs in <*wae_runtime_directory*>/logs.

After LSA configuration is completed and LSP collection (lsp-config-nimo) is invoked, the following occurs:

1. The source network is copied.
2. The LSP collection from the LSA RFS network model is populated.
    1. The LSA RFS model is updated with LSP information (**`wae-rfs services lsp-rfs run-collection`**).
    2. A device sync (**`devices device`** *`<rfs_device_name>`* `sync-from`) updates the WAE device model of the RFS.
    3. The LSP portion of the WAE network model is populated from the RFS model.
    4. The newly populated LSP portion of the WAE network model is committed.
3. The service meta-data for the RFS model is updated (**`re-deploy reconcile`**).

The processing required in the above steps can trigger timeouts. Out-of-sync errors in the RFS model and timeout errors in the wae-java-vm.log file often indicate timeout errors. In general, the starting timeout values should be at least 4 seconds per device on each RFS node. If you suspect that timeout times are causing errors, change the following values as you deem necessary:

- **`lsp-config-nimo run-collection`** action timeout—The time allocated by the YANG runtime (YRT) for **`lsp-config-action`** to complete. This value is potentially specified for all actions in the YRT instance in the wae.conf file. You can edit this value: **`networks network`** *`<network_name>`* **`nimo lsp-config-nimo advanced action-timeout`** *`<timeout_value>`*
- **`wae-rfs services lsp-rfs run-collection`** action timeout—Time allocated by the RFS model for the **`lsp-rfs`** action to complete. The value is potentially specified for RFS NSO instances in the ncs.conf file. You can edit this value: **`wae-rfs services lsp-rfs advanced action-timeout`***`<timeout_value>`*

Example: Setting the timeout for 5 minutes in YRT

```
admin@wae# config
Entering configuration mode terminal
Current configuration users:
admin tcp (maapi from 127.0.0.1) on since 2017-08-09 09:46:21 terminal mode
admin@wae(config)# networks network lsp-network nimo lsp-config-nimo advanced action-timeout
 5
admin@wae(config-network-lsp-network)# top
```

```
admin@wae(config)# devices device rfs config wae-rfs:wae-rfs services lsp-rfs advanced
action-timeout 5
admin@wae(config-config)# commit
Commit complete.
admin@wae(config-config)#
```

Understand WAE CLI Logging

WAE has extensive logging functionality. WAE logs to the directory specified in the `wae.conf` file. The following are the most useful log files:

- `wae.log`—WAE daemon log; can be configured to syslog.
- `wae_err.log.1`, `wae_err.log.idx`, `wae_err.log.siz`—If the WAE daemon has a problem, this log contains debug information for support. Display the content with the command **wae --printlog wae_err.log**.
- `audit.log`—Central audit log that covers all northbound interfaces; can be configured to syslog.
- `localhost:8080.access`—All http requests to the daemon. This is an access log for the embedded web server. This file adheres to the Common Log Format, as defined by Apache and others. This log is disabled by default and is not rotated; that is, use **logrotate(8)**.
- `devel.log`—Debug log for troubleshooting user-written code. This log is enabled by default and is not rotated; that is, use **logrotate(8)**. Use this log with the `java-vm` or `python-vm` logs. The user code logs in the `vm` logs and the corresponding library logs in `devel.log`. Disable this log in production systems. Can be configured to syslog.
- `wae-java-vm.log`, `wae-python-vm.log`—Log for code running in Java or Python VMs, such as service applications. Developers writing Java and Python code use this log (in combination with `devel.log`) for debugging.
- `netconf.log`, `snmp.log`—Log for northbound agents; can be configured to syslog.
- `rollback`*`NNNNN`*—All WAE commits generate a corresponding rollback file. You can configure the maximum number of rollback files and file numbering in `wae.conf`.
- `xpath.trace`—XPATH is used in many places, such as XML templates. This log file shows the evaluation of all XPATH expressions. To debug XPATH for a template, use the pipe-target debug in the CLI instead.
- `ned-cisco-ios-xr-pe1.trace`—If device trace is turned on, a trace file is created for each device. The file location is not configured in `wae.conf` but is configured when device trace is turned on, such as in the CLI.

Syslog

Using BSD or IETF syslog format (RFC5424), WAE can syslog to a local or remote syslog server. You can use the `wae.conf` file to choose which logs to save to syslog: `ncs.log`, `devel.log`, `netconf.log`, or `snmp.log`.

The following example shows a common syslog configuration:

```
<syslog-config>
```

```
  <facility>daemon</facility>

  <udp>
     <enabled>false</enabled>
     <host>127.0.0.1</host>
     <port>895</port>
  </udp>

  <syslog-servers>
    <server>
      <host>127.0.0.2</host>
      <version>1</version>
    </server>
    <server>
      <host>127.0.0.3</host>
      <port>7900</port>
      <facility>local4</facility>
      </server>
   </syslog-servers>
  </syslog-config>

<ncs-log>
  <enabled>true</enabled>
  <file>
   <name>./logs/ncs.log</name>
   <enabled>true</enabled>
  </file>
  <syslog>
    <enabled>true</enabled>
  </syslog>
</ncs-log>
```

Syslog Messages and Formats

The following table lists WAE syslog messages and formats.

Symbol	Format String	Comment
DAEMON_DIED	"Daemon ~s died"	An external database daemon closed its control socket.
DAEMON_TIMEOUT	"Daemon ~s timed out"	An external database daemon did not respond to a query.
NO_CALLPOINT	"no registration found for callpoint ~s of type=~s"	ConfD tried to populate an XML tree, but no code had registered under the relevant callpoint.
CDB_DB_LOST	"CDB: lost DB, deleting old config"	CDB found its data schema file but not its data files. CDB recovered by starting from an empty database.
CDB_CONFIG_LOST	"CDB: lost config, deleting DB"	CDB found its data files but not its schema file. CDB recovered by starting from an empty database.

Symbol	Format String	Comment
CDB_UPGRADE_FAILED	"CDB: Upgrade failed: ~s"	Automatic CDB upgrade failed, meaning the data model was changed in a way that is not supported.
CDB_INIT_LOAD	"CDB load: processing file: ~s"	CDB is processing an initialization file.
CDB_OP_INIT	"CDB: Operational DB re-initialized"	The operational database was deleted and reinitialized because of an upgrade or a corrupt file.
CDB_CLIENT_TIMEOUT	"CDB client (~s) timed out, waiting for ~s"	A CDB client failed to answer within the timeout period and was disconnected.
INTERNAL_ERROR	"Internal error: ~s"	A ConfD internal error occurred and should be reported to Cisco technical support.
AAA_LOAD_FAIL	"Failed to load AAA: ~s"	Failed to load the AAA data because the external database is misbehaving or AAA is mounted or populated badly.
EXTAUTH_BAD_RET	"External auth program (user=~s) ret bad output: ~s"	Authentication is external and the external program returned badly formatted data.
BRIDGE_DIED	"confd_aaa_bridge died - ~s"	ConfD is configured to start the confd_aaa_bridge and the C program died.
PHASE0_STARTED	"ConfD phase0 started"	ConfD has started its start phase 0.
PHASE1_STARTED	"ConfD phase1 started"	ConfD has started its start phase 1.
STARTED	"ConfD started vsn: ~s"	ConfD has started.
UPGRADE_INIT_STARTED	"Upgrade init started"	In-service upgrade initialization started.
UPGRADE_INIT_SUCCEEDED	"Upgrade init succeeded"	In-service upgrade initialization succeeded.
UPGRADE_PERFORMED	"Upgrade performed"	In-service upgrade was performed but not yet committed.
UPGRADE_COMMITTED	"Upgrade committed"	In-service upgrade was committed.
UPGRADE_ABORTED	"Upgrade aborted"	In-service upgrade was aborted.
CONSULT_FILE	"Consulting daemon configuration file ~s"	ConfD is reading its configuration file.
STOPPING	"ConfD stopping (~s)"	ConfD is stopping (for example, due to **confd --stop**).
RELOAD	"Reloading daemon configuration"	Initiated daemon configuration reload.
BADCONFIG	"Bad configuration: ~s:~s: ~s"	confd.conf contains bad data.
WRITE_STATE_FILE_FAILED	"Writing state file failed: ~s: ~s (~s)"	Failed to write a state file.

Symbol	Format String	Comment
READ_STATE_FILE_FAILED	"Reading state file failed: ~s: ~s (~s)"	Failed to read a state file.
SSH_SUBSYS_ERR	"ssh protocol subsys - ~s"	Client did not send the **"subsystem\"** command correctly.
SESSION_LIMIT	"Session limit of type '~s' reached, rejected new session request"	Session limit reached; new session request was rejected.
CONFIG_TRANSACTION_LIMIT	"Configuration transaction limit of type '~s' reached, rejected new transaction request"	Configuration transaction limit reached; new transaction request was rejected.
ABORT_CAND_COMMIT	"Aborting candidate commit, request from user, reverting configuration"	Aborting candidate commit due to user request. Reverting the configuration.
ABORT_CAND_COMMIT_TIMER	"Candidate commit timer expired, reverting configuration"	Candidate commit timer expired; reverting configuration.
ABORT_CAND_COMMIT_TERM	"Candidate commit session terminated, reverting configuration"	Candidate commit session terminated; reverting configuration.
ROLLBACK_REMOVE	"Found half created rollback0 file - removing and creating new"	Removing and recreating a rollback0 file that was found only half created.
ROLLBACK_REPAIR	"Found half created rollback0 file - repairing"	Repairing a rollback0 file that was found only half created.
ROLLBACK_FAIL_REPAIR	"Failed to repair rollback files"	Failed to repair a rollback file.
ROLLBACK_FAIL_CREATE	"Error while creating rollback file: ~s: ~s"	An error occurred while creating a rollback file.
ROLLBACK_FAIL_RENAME	"Failed to rename rollback file ~s to ~s: ~s"	Failed to rename a rollback file.
NS_LOAD_ERR	"Failed to process namespace ~s: ~s"	System failed to process a loaded namespace.
NS_LOAD_ERR2	"Failed to process namespaces: ~s"	System failed to process a loaded namespace.
FILE_LOAD_ERR	"Failed to load file ~s: ~s"	System failed to load a file in its load path.
FILE_LOADING	"Loading file ~s"	System is starting to load a file.
SKIP_FILE_LOADING	"Skipping file ~s: ~s"	System skipped a file.
FILE_LOAD	"Loaded file ~s"	System loaded a file.
LISTENER_INFO	"~s to listen for ~s on ~s:~s"	ConfD starts or stops to listen for incoming connections.
NETCONF_HDR_ERR	"Got bad NETCONF TCP header"	The clear text header that indicates users and groups was formatted badly.

Symbol	Format String	Comment
LIB_BAD_VSN	"Got library connect from wrong version (~s, expected ~s)"	An application connecting to ConfD used a library version that does not match the ConfD version (for example, an old version of the client library).
LIB_BAD_SIZES	"Got connect from library with insufficient keypath depth/keys support (~s/ ~s, needs ~s/~s)"	An application connecting to ConfD used a library version that cannot handle the depth and the number of keys used by the data model.
LIB_NO_ACCESS	"Got library connect with failed access check: ~s"	An access check failure occurred when an application connected to ConfD.
SNMP_NOT_A_TRAP	"SNMP gateway: Non-trap received from ~s"	A UDP package was received on the trap receiving port, but it's not an SNMP trap.
SNMP_TRAP_V1	"SNMP gateway: V1 trap received from ~s"	An SNMPv1 trap was received on the trap receiving port, but forwarding v1 traps is not supported.
SNMP_TRAP_NOT_FORWARDED	"SNMP gateway: Can't forward trap from ~s; ~s"	An SNMP trap was not forwarded.
SNMP_TRAP_UNKNOWN_ SENDER	"SNMP gateway: Not forwarding trap from ~s; the sender is not recognized"	An SNMP trap was supposed to be forwarded, but the sender was not listed in confd.conf.
SNMP_TRAP_OPEN_PORT	"SNMP gateway: Can't open trap listening port ~s: ~s"	Could not open the port for listening to SNMP traps.
SNMP_TRAP_NOT_RECOGNIZED	"SNMP gateway: Can't forward trap with OID ~s from ~s; There is no notification with this OID in the loaded models"	An SNMP trap was received on the trap receiving port, but its definition is unknown.
XPATH_EVAL_ERROR1	"XPath evaluation error: ~s for ~s"	An error occurred while evaluating an xpath expression.
XPATH_EVAL_ERROR2	"XPath evaluation error: '~s' resulted in ~s for ~s"	An error occurred while evaluating an xpath expression.
CANDIDATE_BAD_FILE_FORMAT	"Bad format found in candidate db file ~s; resetting candidate"	The candidate database file has a bad format. The candidate database is reset to an empty database.
CANDIDATE_CORRUPT_FILE	"Corrupt candidate db file ~s; resetting candidate"	The candidate database file is corrupt and cannot be read. The candidate database is reset to an empty database.
MISSING_DES3CBC_ SETTINGS	"DES3CBC keys were not found in confd.conf"	DES3CBC keys were not found in confd.conf.

Symbol	Format String	Comment
MISSING_AESCFB128_ SETTINGS	"AESCFB128 keys were not found in confd.conf"	AESCFB128 keys were not found in confd.conf.
SNMP_MIB_LOADING	"Loading MIB: ~s"	The SNMP agent is loading a MIB file.
SNMP_CANT_LOAD_MIB	"Can't load MIB file: ~s"	The SNMP agent failed to load a MIB file.
SNMP_WRITE_STATE_ FILE_FAILED	"Write state file failed: ~s: ~s"	Failed to write the SNMP agent state file.
SNMP_READ_STATE_ FILE_FAILED	"Read state file failed: ~s: ~s"	Failed to read the SNMP agent state file.
SNMP_REQUIRES_CDB	"Can't start SNMP. CDB is not enabled"	CDB must be enabled before the SNMP agent can start.
FXS_MISMATCH	"Fxs mismatch, slave is not allowed"	A slave connected to a master with different fxs files.
TOKEN_MISMATCH	"Token mismatch, slave is not allowed"	A slave connected to a master with a bad authorization token.
HA_SLAVE_KILLED	"Slave ~s killed due to no ticks"	A slave node did not produce its ticks.
HA_DUPLICATE_NODEID	"Nodeid ~s already exists"	A slave arrived with a node ID that already exists.
HA_FAILED_CONNECT	"Failed to connect to master: ~s"	An attempted library to become a slave call failed because the slave could not connect to the master.
HA_BAD_VSN	"Incompatible HA version (~s, expected ~s), slave is not allowed"	A slave connected to a master with an incompatible HA protocol version.
NETCONF	"~s"	NETCONF traffic log message.
DEVEL_WEBUI	"~s"	Developer web UI log message.
DEVEL_AAA	"~s"	Developer AAA log message.
DEVEL_CAPI	"~s"	Developer C API log message.
DEVEL_CDB	"~s"	Developer CDB log message.
DEVEL_CONFD	"~s"	Developer ConfD log message.
DEVEL_SNMPGW	"~s"	Developer SNMP gateway log message.
DEVEL_SNMPA	"~s"	Developer SNMP agent log message.
NOTIFICATION_REPLAY_ STORE_FAILURE	"~_s"	A failure occurred in the built-in notification replay store.
EVENT_SOCKET_TIMEOUT	"Event notification subscriber with bitmask ~s timed out, waiting for ~s"	An event notification subscriber did not reply within the configured timeout period.

Symbol	Format String	Comment
EVENT_SOCKET_WRITE_BLOCK	"~s"	Write on an event socket was blocked for too long.
COMMIT_UN_SYNCED_DEV	"Committed data towards device ~s which is out of sync"	Data was committed toward a device with a bad or unknown sync state.
NCS_SNMP_INIT_ERR	"Failed to locate snmp_init.xml in loadpath ~s"	Failed to locate snmp_init.xml in the load path.
NCS_JAVA_VM_START	"Starting the NCS Java VM"	Starting the NCS Java VM.
NCS_JAVA_VM_FAIL	"The NCS Java VM ~s"	An NCS Java VM failure or timeout occurred.
NCS_PACKAGE_SYNTAX_ERROR	"Failed to load NCS package: ~s; syntax error in package file"	Syntax error in package file.
NCS_PACKAGE_DUPLICATE	"Failed to load duplicate NCS package ~s: (~s)"	Duplicate package found.
NCS_PACKAGE_COPYING	"Copying NCS package from ~s to ~s"	A package was copied from the load path to a private directory.
NCS_PACKAGE_UPGRADE_ ABORTED	"NCS package upgrade failed with reason '~s'"	The CDB upgrade was aborted, implying that the CDB is untouched. However, the package state changed.
NCS_PACKAGE_BAD_ NCS_VERSION	"Failed to load NCS package: ~s; requires NCS version ~s"	Bad NCS version for the package.
NCS_PACKAGE_BAD_ DEPENDENCY	"Failed to load NCS package: ~s; required package ~s of version ~s is not present (found ~s)"	Bad NCS package dependency.
NCS_PACKAGE_CIRCULAR_ DEPENDENCY	"Failed to load NCS package: ~s; circular dependency found"	Circular NCS package dependency.
CLI_CMD	"CLI '~s'"	User executed a CLI command.
CLI_DENIED	"CLI denied '~s'"	Due to permissions, a user was denied from executing a CLI command.
BAD_LOCAL_PASS	"Provided bad password"	A locally configured user provided a bad password.
NO_SUCH_LOCAL_USER	"no such local user"	A non existing local user tried to log in.
PAM_LOGIN_FAILED	"pam phase ~s failed to login through PAM: ~s"	A user failed to log in through PAM.
PAM_NO_LOGIN	"failed to login through PAM: ~s"	A user failed to log in through PAM.

Symbol	Format String	Comment
EXT_LOGIN	"Logged in over ~s using externalauth, member of groups: ~s~s"	An externally authenticated user logged in.
EXT_NO_LOGIN	"failed to login using externalauth: ~s"	External authentication failed for a user.
GROUP_ASSIGN	"assigned to groups: ~s"	A user was assigned to a set of groups.
GROUP_NO_ASSIGN	"Not assigned to any groups - all access is denied"	A user was logged in but was not assigned to any groups.
MAAPI_LOGOUT	"Logged out from maapi ctx=~s (~s)"	A management agent API (MAAPI) user was logged out.
SSH_LOGIN	"logged in over ssh from ~s with authmeth:~s"	A user logged into ConfD's built-in SSH server.
SSH_LOGOUT	"Logged out ssh <~s> user"	A user was logged out from ConfD's built-in SSH server.
SSH_NO_LOGIN	"Failed to login over ssh: ~s"	A user failed to log in to ConfD's built-in SSH server.
NOAAA_CLI_LOGIN	"logged in from the CLI with aaa disabled"	A user used the --noaaa flag to confd_cli.
WEB_LOGIN	"logged in through Web UI from ~s"	A user logged in through the web UI.
WEB_LOGOUT	"logged out from Web UI"	A web UI user logged out.
WEB_CMD	"WebUI cmd '~s'"	A user executed a web UI command.
WEB_ACTION	"WebUI action '~s'"	A user executed a web UI action.
WEB_COMMIT	"WebUI commit ~s"	A user performed a web UI commit.
SNMP_AUTHENTICATION_FAIL	"ESDNMP authentication failed: ~s"	An SNMP authentication failed.
LOGIN_REJECTED	"~s"	Authentication for a user was rejected by application callback.
COMMIT_INFO	"commit ~s"	Information about configuration changes committed to the running data store.
CLI_CMD_DONE	"CLI done"	CLI command finished successfully.
CLI_CMD_ABORTED	"CLI aborted"	CLI command aborted.
NCS_DEVICE_OUT_OF_SYNC	"NCS device-out-of-sync Device '~s' Info '~s'"	A check-sync action reported out-of-sync for a device.
NCS_SERVICE_OUT_OF_SYNC	"NCS service-out-ofsync Service '~s' Info '~s'"	A check-sync action reported out-of-sync for a service.
NCS_PYTHON_VM_START	"Starting the NCS Python VM"	Starting the NCS Python VM.

Symbol	Format String	Comment
NCS_PYTHON_VM_FAIL	"The NCS Python VM ~s"	The NCS Python VM failed or timed out.
NCS_SET_PLATFORM_ DATA_ERRORS	"NCS Device '~s' failed to set platform data Info '~s'"	The device failed to set the platform operational data at connect.
NCS_SMART_LICENSING_ START	"Starting the NCS Smart Licensing Java VM"	Starting the NCS Smart Licensing Java VM.
NCS_SMART_LICENSING_FAIL	"The NCS Smart Licensing Java VM ~s"	The NCS Smart Licensing Java VM failed or timed out.
NCS_SMART_LICENSING_ GLOBAL_NOTIFICATION	"Smart Licensing Global Notification: ~s"	Smart Licensing global notification.
NCS_SMART_LICENSING_ ENTITLEMENT_NOTIFICATION	"Smart Licensing Entitlement Notification: ~s"	Smart Licensing entitlement notification.
NCS_SMART_LICENSING_ EVALUATION_COUNTDOWN	"Smart Licensing evaluation time remaining: ~s"	Smart Licensing evaluation time remaining.
DEVEL_SLS	"~s"	Developer Smart Licensing API log message.
JSONRPC_REQUEST	"JSON-RPC: '~s' with JSON params ~s"	JSON-RPC method requested.
DEVEL_ECONFD	"~s"	Developer econfd API log message.
CDB_FATAL_ERROR	"fatal error in CDB: ~s"	CDB encountered an unrecoverable error.
LOGGING_STARTED	"Daemon logging started"	Logging subsystem started.
LOGGING_SHUTDOWN	"Daemon logging terminating, reason: ~s"	Logging subsystem terminated.
REOPEN_LOGS	"Logging subsystem, reopening log files"	Logging subsystem reopened log files.
OPEN_LOGFILE	"Logging subsystem, opening log file '~s' for ~s"	Indicate target file for certain type of logging.
LOGGING_STARTED_TO	"Writing ~s log to ~s"	Write logs for a subsystem to a specific file.
LOGGING_DEST_CHANGED	"Changing destination of ~s log to ~s"	The target log file will change to another file.
LOGGING_STATUS_CHANGED	"~s ~s log"	Notify a change of logging status (enabled/disabled) for a subsystem.
ERRLOG_SIZE_CHANGED	"Changing size of error log (~s) to ~s (was ~s)"	Notify a change of log size for an error log.
CGI_REQUEST	"CGI: '~s' script with method ~s"	CGI script requested.
MMAP_SCHEMA_FAIL	"Failed to setup the shared memory schema"	Failed to set up the shared memory schema.

Symbol	Format String	Comment
KICKER_MISSING_SCHEMA	"Failed to load kicker schema"	Failed to load the kicker schema.
JSONRPC_REQUEST_ IDLE_TIMEOUT	"Stopping session due to idle timeout: ~s"	JSON-RPC idle timeout.
JSONRPC_REQUEST_ ABSOLUTE_TIMEOUT	"Stopping session due to absolute timeout: ~s"	JSON-RPC absolute timeout.

Database Locking

This section explains the different locks that exist in WAE and how they interact.

Global Locks

The WAE management backplane keeps a lock on the data store: *running*. This lock is known as the global lock and provides a mechanism to grant exclusive access to the data store. The global lock is the only lock that can explicitly be taken through a northbound agent—for example, by the NETCONF <lock> operation—or by calling Maapi.lock().

A global lock can be taken for the entire data store, or it can be a partial lock (for a subset of the data model). Partial locks are exposed through NETCONF and MAAPI.

An agent can request a global lock to ensure that it has exclusive write access. When an agent holds a global lock, no one else can write to that data store. This behavior is enforced by the transaction engine. A global lock on running is granted to an agent if there are no other lock holders (including partial locks), and if all data providers approve the lock request. Each data provider (CDB or external data provider) has its lock() callback invoked to refuse or accept the lock. The output of **ncs --status** includes the lock status.

Transaction Locks

A northbound agent starts a user session towards the WAE management backplane. Each user session can then start multiple transactions. A transaction is either read/write or read-only.

The transaction engine has its internal locks toward the running data store. These transaction locks exist to serialize configuration updates toward the data store and are separate from global locks.

When a northbound agent wants to update the running data store with a new configuration, it implicitly grabs and releases the transactional lock. The transaction engine manages the lock as it moves through the transaction state machine. No API exposes the transactional lock to the northbound agent.

When the transaction engine wants to take a lock for a transaction (for example, when entering the validate state), it first checks that no other transaction has the lock. It then checks that no user session has a global lock on that data store. Finally, it invokes each data provider with a transLock() callback.

Northbound Agents and Global Locks

In contrast to implicit transactional locks, some northbound agents expose explicit access to global locks. The management API exposes global locks by providing Maapi.lock() and Maapi.unlock() methods (and the

corresponding Maapi.lockPartial() Maapi.unlockPartial() for partial locking). Once a user session is established (or attached to), these functions can be called.

In the CLI, global locks are taken when entering different configure modes, as follows:

- **config exclusive**—Takes the running data store global lock.
- **config terminal**—Does not grab any locks.

The CLI keeps the global lock until the configure mode is exited.

The Expert Mode behaves in the same way as the CLI: it has edit tabs called **Edit private** and **Edit exclusive**, which correspond to the CLI modes described above.

The NETCONF agent translates the <lock> operation into a request for a global lock for the requested data store. Partial locks are also exposed through the partial-lock rpc.

External Data Providers and CDB

An external data provider is not required to implement the lock() and unlock() callbacks. WAE never tries to initiate the transLock() state transition toward a data provider while a global lock is taken. The reason for a data provider to implement the locking callbacks is if someone else can write to the data provider's database.

CDB ignores the lock() and unlock() callbacks (because the data provider interface is the only write interface towards it).

CDB has its own internal locks on the database. The running data store has a single write lock and multiple read locks. It is not possible to grab the write lock on a data store while there are active read locks on it. The locks in CDB exist to ensure that a reader always gets a consistent view of the data. (Confusion occurs if another user deletes configuration nodes in between calls to getNext() on YANG list entries.)

During a transaction transLock() takes a CDB read lock toward the transaction's data store and writeStart() tries to release the read lock and grab the write lock instead. A CDB external reader client implicitly takes a CDB read lock between Cdb.startSession() and Cdb.endSession(). This means that while a CDB client is reading, a transaction cannot pass through writeStart(). Conversely, a CDB reader cannot start while a transaction is in between writeStart() and commit() or abort().

The operational store in CDB does not have any locks; WAE's transaction engine can only read from it. CDB client writes are atomic per write operation.

Lock Impact on User Sessions

When a session tries to modify a data store that is locked, it fails. For example, the CLI might print:

```
admin@wae(config)# commit
Aborted: the configuration database is locked
```

Because some locks are short-lived (such as a CDB read lock), WAE is configured by default to retry the failed operation for a configurable length of time. If the data store remains locked after this time, the operation fails.

To configure the retry timeout, set the `/ncs-config/commit-retry-timeout` value in `wae.conf`.

Security

WAE requires privileges to perform certain tasks. Depending on the target system, the following tasks might require root privileges:

- Binding to privileged ports. The `wae.conf` configuration file specifies which port numbers WAE should bind(2) to. If a port number is lower than 1024, WAE usually requires root privileges unless the target operating system allows WAE to bind to these ports as a non-root user.
- If PAM is used for authentication, the program installed as `$NCS_DIR/lib/ncs/priv/pam/epam` acts as a PAM client. Depending on the local PAM configuration, this program might require root privileges. If PAM is configured to read the local `passwd` file, the program must either run as root, or be setuid root. If the local PAM configuration instructs WAE to run for example pam_radius_auth, root privileges might not be required, depending on the local PAM installation.
- If the CLI is used to create CLI commands that run executables, modify the permissions of the `$NCS_DIR/lib/ncs/priv/ncs/cmdptywrapper` program.

To run an executable as root or as a specific user, make cmdptywrapper setuid root:

```
# chown root cmdptywrapper
# chmod u+s cmdptywrapper
```

Failing that, all programs are executed as the user running the WAE daemon. If that user is root, you need not perform the chmod operations above.

Failing that, all programs are executed as the user running the confd daemon. If that user is root, you need not perform the preceding chmod operations.

For executables that run via actions, modify the permissions of the `$NCS_DIR/lib/ncs/priv/ncs/cmdwrapper` program:

```
# chown root cmdwrapper
# chmod u+s cmdwrapper
```

WAE can be instructed to terminate NETCONF over clear text TCP, which is useful for debugging (NETCONF traffic can be captured and analyzed) and when providing a local proprietary transport mechanism other than SSH. Clear text TCP termination is not authenticated; the clear text client simply tells WAE which user the session should run as. The assumption is that authentication is already done by an external entity, such as an SSH server. If clear text TCP is enabled, WAE must bind to localhost (127.0.0.1) for these connections.

Client libraries connect to WAE. For example, the CDB API is TCP-based and a CDB client connects to WAE. WAE learns which address to use for these connections through the `wae.conf` parameters `/ncs-config/ncs-ipc-address/ip` (the default address is 127.0.0.1) and `/ncs-config/ncs-ipcaddress/port` (the default port is 4565).

WAE multiplexes different kinds of connections on the same socket (IP and port combination). The following programs connect on the socket:

- Remote commands, such as **ncs --reload**.
- CDB clients.

- External database API clients.
- Management agent API (MAAPI) clients.
- The ncs_cli program.

By default, the preceding programs are considered trusted. MAAPI clients and the ncs_cli authenticate users before connecting to WAE. CDB clients and external database API clients are considered trusted and do not have to authenticate.

Because the ncs-ipc-address socket allows full, unauthenticated access to the system, it is important to ensure that the socket is not accessible from untrusted networks. You can also restrict access to the ncs-ipc-address socket by means of an access check. See Restrict Access to the IPC Port, on page 24.

Restrict Access to the IPC Port

By default, clients connecting to the IPC port are considered trusted; no authentication is required. To prevent remote access, WAE relies on the use of 127.0.0.1 for `/ncs-config/ncs-ipc-address/ip`. However, you can restrict access to the IPC port by configuring an access check.

To enable the access check, set the `wae.conf` element `/ncs-config/ncs-ipc-accesscheck/enabled` to **true**, and specify a filename for `/ncs-config/ncs-ipc-accesscheck/filename`. The file should contain a shared secret (a random-character string). Clients connecting to the IPC port must provide a challenge handshake before they are granted access to WAE functions.

Note The access permissions on this file must be restricted via OS file permissions, such that the file can only be read by the WAE daemon and client processes that are allowed to connect to the IPC port. For example, if both the daemon and the clients run as root, the file can be owned by root and have only "read by owner" permission (mode 0400). Another possibility is to create a group that only the daemon and the clients belong to, set the group ID of the file to that group, and have only "read by group" permission (mode 040).

To provide the secret to the client libraries and instruct them to use the access check handshake, set the environment variable NCS_IPC_ACCESS_FILE to the full path name of the file that contains the secret. This is sufficient for all clients mentioned above; there is no need to change the application code to enable this check.

Note The access check must be either enabled or disabled for both the daemon and the clients. For example, if the `wae.conf` element `/ncsconfig/ncs-ipc-access-check/enabled` is not set to **true**, but clients are started with the environment variable NCS_IPC_ACCESS_FILE pointing to a file with a secret, the client connections fail.

Clear WAE Operational Data

To clear WAE operational data from the database, you must delete the model, l1-model, from the respective NIMO network models. Then, delete the device tree. If your NIMO network model has layouts, delete those layouts from the NIMO network models.

The following example commands show how to clear operational data from the as64002 network model and device tree:

```
delete networks network as64002 model
delete networks network as64002 layouts
delete networks network as64002 l1-model
delete devices device *
commit
```

Back Up and Restore the WAE Configuration

With the YANG run-time framework, you can easily back up and restore the WAE configuration. We recommend that you back up the WAE configuration before starting any collection (that is, before any operational data is populated).

- To back up a WAE configuration:

  ```
  admin@wae% save /home/wae/wae-backup.cfg
  ```

  The preceding command backs up both the configuration data and the operational data. To back up only the configuration data, you must clear the operational data from the database as described in Clear WAE Operational Data, on page 24. Be careful before clearing operational data in a production environment, because all operational data is deleted.

- To restore a WAE configuration:

  ```
  [wae@wae ~]$ ncs_load -l -m -F j wae-backup.cfg
  ```